Handling Ontology Change:
Survey and Proposal for a Future Research Direction

Giorgos Flouris, Dimitris Plexousakis

Institute of Computer Science, FO.R.T.H.
P.O. Box 1385, GR 71110, Heraklion, Crete, Greece
{fgeo, dp}@ics.forth.gr

Abstract

One of the crucial tasks towards the realization of the vision of the Semantic Web is the efficient encoding of human knowledge in ontologies. The proper maintenance of these, usually large, structures and, in particular, the adaptation of ontologies to new knowledge (ontology change) is one of the most challenging problems in the Semantic Web research. Due to the uncontrolled and decentralized nature of the Semantic Web, ontology change is a complicated and multifaceted task; this has led to the emergence of several different, but closely related, research areas that deal with the problem. In this report, we attempt to draw a clear line between these research areas, by underlying their relations, interactions and overlaps. Our work uncovers a certain gap in the current research and proposes a supplementary research direction, based on belief revision. We argue that our approach uncovers an interesting new dimension to the problem that is likely to find important applications in the future.

Introduction

Ontologies, originally introduced by Aristotle, have found several applications in the area of Knowledge Representation (KR) and in the Semantic Web [11], because they provide a means to define the basic terms and relations that comprise the vocabulary of a certain domain of interest [75]. The importance of ontologies in current Artificial Intelligence (AI) research is emphasized by the interest shown by both the research and the enterprise community to various problems related to ontologies and ontology manipulation [84].

One of the most important such problems is the problem of *ontology change*, which refers to the problem of modifying a given ontology in response to a certain need. In this work, we use the term in a broad sense, covering any type of change, including changes to the ontology in response to external events, changes dictated by the ontology engineer, changes forced by the need to translate the ontology in a different language or using different terminology and so on.

Several reasons for changing an ontology have been identified in the literature. An ontology, just like any structure holding information regarding a domain of interest, may need to change simply because the domain of interest has changed [101]; but even if we assume a static world (domain), which is a rather unrealistic assumption for most applications, we may need to change the perspective under which the domain is viewed [92], or we may discover a problem in the original conceptualization of the domain; we might also wish to incorporate additional functionality, according to a change in users' needs [48]. Furthermore, new information, which was previously unknown, classified or otherwise unavailable may become available or different features of the domain may become important [56]. Moreover, ontology development is becoming more and more a collaborative and parallelized process, whose subproducts (parts of the ontology) need to be combined to produce the final ontology [71];

this process would require changes in all the "subontologies" to reach a consistent final state; but even then, the so-called "final" state is rarely final, as ontology development is usually an ongoing process [56]. There are also reasons related to the distributed nature of the Semantic Web: ontologies are usually depending on other ontologies, over which the knowledge engineer may have no control; if the remote ontology is changed for any of the above reasons, the depended ontology might also need to be changed to reflect possible changes in terminology or representation [56]. In other cases, a certain agent, service or application may need to use an ontology whose terminology or representation is different from the one it can understand [30], so he needs to perform some kind of translation (change) in the imported ontology; or we may need to merge or integrate information from two or more ontologies in order to produce a more appropriate one [97].

The problem of ontology change is far from trivial. Several philosophical problems related to the adaptation of knowledge in general have been identified in the research area of *belief change* (also known as *belief revision*) [41], [42], [67]; these are obviously applicable to knowledge represented in ontologies as well. The large size of modern day ontologies makes this problem even more complicated [84]. But it's not just that: the Semantic Web is characterized by decentralization, heterogeneity and lack of central control or authority. This is both a blessing and a curse; these features have greatly contributed to the success of the web but they have also introduced several new, challenging and interesting problems, which didn't exist in traditional AI.

As far as ontology change is concerned, one such problem is the lack of control on who uses a certain ontology once it has been published; subtle changes in an ontology may have unforeseeable effects in depended applications, services, data and ontologies [100]. The ontology designer cannot know who uses which part of his ontology and for what purpose, so he cannot predict the effects of the change. The same holds in the opposite direction: if an ontology is depending on other ontologies, there is no way for the ontology designer to control when and how these ontologies will change. These facts raise the need to support and maintain different interoperable versions of the same ontology [56], [70], a problem greatly interwoven with ontology change [69]. On the other hand, heterogeneity leads to the absence of any standard terminology for any given domain which may cause problems when an agent, service or application uses information from two different ontologies [30]. As ontologies often cover overlapping domains from differing viewpoints and with differing terminology, some kind of translation may be necessary in many practical applications.

In order to cope with the many different aspects of the problem, several related research areas have emerged, each dealing with a different facet of the problem of ontology change, such as ontology alignment, evolution, mapping, merging, articulation, versioning etc. These areas are greatly interlinked; as a result, several works and systems deal with more than one of these topics causing a certain confusion to a newcomer.

In this work, we perform a short literature review on the different types of ontology change (with emphasis on ontology evolution) in an attempt to draw a fine line between these closely related research fields and study their relation and interaction. Our study critically reviews the field and uncovers what we consider to be a weakness of the currently used approach. Based on this fact, we propose an alternative, more formal, logic-based method to address certain aspects of ontology change, which is inspired and greatly influenced by previous research on the area of belief change. We argue that this logic-based approach complements the currently used approaches and could be used as a supplementary tool for ontology maintenance and change.

Types of Ontology Change

Basic Definitions

Lately, the term ontology has come to refer to a wide range of formal representations, including taxonomies, hierarchical terminology vocabularies or detailed logical theories

describing a domain [92]. For this reason, a precise definition of the term is rather difficult. One definition that is commonly used in the literature was given by Gruber in [47] who defined an *ontology* to be a *specification of a shared conceptualization of a domain*.

A more formal, algebraic, approach, identifies an ontology as a pair (S,A), where S is the *vocabulary* (or *signature*) of the ontology (being modeled by some mathematical structure, such as a poset, a lattice or an unstructured set) and A the *set of ontological axioms*, which specify the intended interpretation of the vocabulary in a given domain of discourse [65].

As already outlined, we use the term ontology change in a broad sense to cover all aspects of ontology modification, as well as the problems that are indirectly related to the change operation such as the maintenance of different versions of an ontology or the translation of information in a common terminology. In short, we will use the term ontology change to refer to the problem of *deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of the above modifications and the management of their effects in depending data, services, applications, agents or other elements*.

The operation of "changing" in the above definition may (but need not) involve keeping a copy of the original ontology. The "need" to change the ontology may take several different forms, including, but not limited to, the discovery of new information (which could be some instance data, another ontology or an observation), a change in the focus or the viewpoint of the conceptualization, information received by some external source, a change in the domain, communication needs between heterogeneous sources of information, the combination of information from different ontologies and so on.

The above definition covers several closely related research areas which are studied separately in the literature. In this work, we identify 9 such areas, namely *ontology alignment, mapping, morphism, articulation, translation, evolution, merging, integration and versioning*. Each of these areas covers a small part of the complex problem of ontology change from a different view or perspective, or for different application needs. All these areas are greatly interlinked and several papers, works and systems deal with more than one of these subproblems. In the following subsections we will attempt to precisely define the boundaries of each area and uncover their relations and differences.

Ontology Alignment, Mapping, Morphism, Articulation and Translation

Works related to these areas try to mitigate the problems caused by the heterogeneity of the Semantic Web. The general motivation for these works is that different sources of information generally use different terminology, different representation languages and different syntax to refer to the same or similar concepts. A nice list of use cases where this heterogeneity may cause problems can be found in [30].

The solution that is generally proposed for the above problem consists of a set of "translation rules" of some kind that will allow us to nullify the differences in terminology or syntax. To put it simply, the goal of the whole process is to make two ontologies refer to the same entities with the same name and to different entities with different names. For example, an ontology alignment algorithm should be able to identify that two concepts (classes) RESEARCHER and RESEARCH_STAFF_MEMBER that appear in two different ontologies refer to the same real-world concept, i.e. the class of all researchers. It should also be able to differentiate between two different uses of the entity CHAIR, as it could refer to the class of chairs (as a furniture) in one ontology and to the people forming a Workshop's Chair in another ontology.

Therefore, these areas basically deal with the same problem (i.e., how to deal with heterogeneous information), but the output of the methods (i.e., the result that is produced towards the solution of the problem) is different in each case. The different research fields are identified based on the type of "translation rules" that is produced as the output. Due to the close relationship between these topics, sometimes the term ontology alignment (in [30]) or

ontology mapping (in [65]) is used to refer collectively to all of these areas. In the following, we will provide a definition of the above fields and outline their differences and similarities. Most of the material and the definitions for this subsection are taken from [30], [65].

The term *ontology mapping* can be defined as *the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected*. The result of an ontology mapping algorithm is a collection of functions (morphisms) of ontological signatures. We could differentiate between two different types of mappings, the *total ontology mapping* and the *partial ontology mapping*. A total ontology mapping maps the whole source ontology (say $O_1=(S_1,A_1)$) to the target ontology O_2 while a partial ontology mapping maps a subontology of O_1 (say $O_1'=(S_1',A_1')$, with $S_1'\subseteq S_1$ and $A_1'\subseteq A_1$) to O_2.

The above definition restricts the mappings to ontological signatures. A more ambitious and interesting approach would be to create mappings that deal with both the signatures and the axioms of the ontologies. The term *ontology morphism* refers to that approach, i.e., *the development of a collection of functions (morphisms) that relate both ontological signatures and axioms*. Notice that ontology morphism, unlike the other fields discussed in this subsection, is not restricted to the ontology vocabulary only, but covers the ontological axioms as well.

In both ontology mapping and ontology morphism, we try to relate the two ontologies via functions. Alternatively, the two ontologies could be related in a more general fashion, namely by means of a relation. *The task of finding relationships between entities belonging to two different ontologies is called ontology alignment*. So the result of ontology alignment is a binary relationship between the entities of the two ontologies. This approach is more liberal, allowing greater flexibility, so it is more commonly used in practice.

A binary relationship could be decomposed into a pair of total functions from a common intermediate source; therefore, the alignment of two ontologies (say O_1, O_2) could be described by means of a pair of ontology mappings, from a common intermediate ontology (say O_0). We use the term *ontology articulation* to refer to *the process of determining the intermediate ontology (O_0) and the two mappings with the initial ontologies (O_1, O_2)*.

Finally, the term *ontology translation* is used in the literature to describe two different things. Under one understanding, ontology translation refers to *the process of changing the formal representation of the ontology from one language to another* (say from OWL to RDF or from Ontolingua to Prolog). This changes the syntactic form of the axioms, but not the vocabulary of the ontology. Works related to ontology translation (under this understanding) leave the vocabulary of the ontology unaffected, dealing with the ontological axioms only. Under the second understanding, ontology translation refers to *a translation of the vocabulary*, in a manner similar to that of ontology mapping. The difference between ontology mapping and ontology translation is that the former specifies the function(s) that relate the two ontologies' vocabularies, while the latter applies this (these) function(s) to actually implement the mapping.

As was made clear in the above definitions, the above research areas try (with a few exceptions) to relate the vocabularies of two ontologies. This problem is far from trivial; methods that are commonly used for this purpose include studying the taxonomic or mereological structure of the entities, evaluating name similarities (in effect, studying the string similarity of the entities' names, under a certain metric) and so on. Other methods use a thesaurus to study the linguistic similarities of terms (names), use semantic approaches, or determine the similarity based on the instances of each entity. The final similarity evaluation may also be affected by the evaluation of the similarity of the entity's neighborhood. In real systems, a combination of some of these approaches with some kind of human intervention usually works best. A detailed classification and description of the above methods can be found in [30].

Two of the most important systems that deal with heterogeneity are PROMPT (originally called SMART) [94], [95], [96] and Chimaera [84]. In [32], the term *ontology matching* is used to refer to an ontology mapping algorithm based on the linguistic properties of terms, using a thesaurus based on WordNet [87]. Some thoughts on the issue of heterogeneity in the context of the SHOE language can also be found in [56], [57]. An interesting general-purpose approach to the problem of translation is described in [18]. A much more extensive list of systems and works related to these research areas can be found in [15], [30], [65].

Unfortunately, heterogeneity resolution in ontologies still relies on human intervention; however, the process has to be automatic in order to be practical [65]. In this direction, advances in the field of natural language processing will probably help researchers gain a better understanding on the processes behind automatic heterogeneity resolution [65].

But why do we study these fields in a survey related to ontology change? The result of an ontology mapping algorithm (for example) is a mapping between the vocabularies of two ontologies instead of a modified ontology. Why does ontology mapping constitute a type of ontology change? The answer to these questions is twofold: first of all, heterogeneity resolution constitutes a prerequisite for successful ontology change, as it makes no sense to try to change an ontology in response to a new information, unless they are both formulated using the same terminology, language and syntax. So, it makes practical sense to study these fields along with the problem of ontology change. But it's not just that. We argue that ontology mapping (and the related fields) are not simply "closely related" to ontology change for "practical" reasons; on the contrary, they fall under the definition of ontology change, in the wide sense of the term that we use in this work.

Indeed, consider two agents with heterogeneous ontologies that need to communicate and an ontology mapping algorithm that provides a mapping allowing this communication. In this particular example, the driving force behind ontology mapping is the need for communication. The mapping produced does not directly modify any ontology; however, it allows each agent to "change" the other agent's ontology to fit his own terminology, language and syntax. So the "change" in this particular case is made on-the-fly by each agent during each message exchange and it is trivial, if the mapping is given. In this sense, we could consider ontology mapping as an ontology change algorithm that provides us with a "method" to change an ontology (but does not perform the change itself), in response to a need, which in this particular case is the need for communication between agents with heterogeneous ontologies. Thus, ontology mapping is a part of ontology change in a very real sense. Similar arguments can be made for the other fields discussed in this subsection.

Ontology Evolution and Versioning

In some works, ontology versioning is considered a stronger variant of ontology evolution [49]. Under that viewpoint, ontology evolution is concerned with the ability to change the ontology without losing data or negating the validity of the ontology, while ontology versioning should additionally allow access to different variants of the ontology. Thus, while ontology evolution is concerned with the validity of the newest version, ontology versioning additionally deals with the validity, interoperability and management of all previous versions, including the current (newest) one. This viewpoint is influenced by related research on database schema evolution and versioning. An excellent survey on this issue, studying the differences and similarities of the two fields can be found in [92], where ontologies are compared with database schemas outlining their differences and these differences' impact with respect to (ontology and database schema) evolution and versioning.

Under this understanding however, ontology evolution and versioning become indistinguishable [92]; due to the distributed and decentralized nature of the Semantic Web, multiple versions of ontologies are bound to exist and must be supported [92]. Furthermore, ontologies, as well as dependent elements, are likely to be owned by different parties or spread across different servers; as a result, some parties may be unprepared to change and

others may even be opposed to it [56]. All these facts force us to maintain and support different versions of ontologies, making ontology evolution (under this understanding) useless in practice.

In this work, we will use the term *ontology evolution to refer to the process of modifying an ontology in response to a certain change in the domain or its conceptualization*. On the other hand, *ontology versioning refers to the ability to handle evolving ontologies by creating and managing different variants of it* [69].

Ontology evolution could be considered as the "purest" type of ontology change, in the sense that it deals with the changes themselves. This is a very important problem, as the effectiveness of an ontology based application heavily depends on the quality of the conceptualization of the domain by the underlying ontology [101], which is directly affected by the ability of an evolution algorithm to properly adapt the ontology to new needs.

On the other hand, ontology evolution cannot be a stand-alone process; the extensive web of interrelationships that is usually formed around an ontology, forces us to consider the issue of propagating the changes to dependent elements [80]. This tight coupling has caused ontology evolution algorithms to deal with these issues as well. For example, in [100], ontology evolution is defined as the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, as well as the consistent management and propagation of these changes to dependent elements. This definition covers both ontology evolution and versioning. Under this understanding, ontology versioning and change propagation are considered internal parts of the process of ontology evolution.

In our work, we differentiate between ontology evolution and versioning, by restricting ontology evolution to the process of modifying an ontology while maintaining its validity and ontology versioning to the process of managing different versions of an evolving ontology, maintaining interoperability between versions and providing access to each version as required by the accessing element (data, service, application or other ontology).

Ontology evolution is an important field of ontology-related research, as ontologies often model dynamic environments [100]. As already stated, an ontology is, according to [47], a specification of a shared conceptualization of a domain. Thus, ontology evolution (change) may be caused by either a change in the domain, a change in the conceptualization or a change in the specification [69]. The third type of change (change in the specification) refers to a change in the way the conceptualization is formally recorded, i.e., a change in the representation language. This type of change is dealt with in the field of ontology translation, studied in the previous subsection. Our definition of ontology evolution covers the first two types of change (changes in the domain and changes in the conceptualization).

Both types of changes are not rare. The conceptualization of the domain may change for several reasons, including a new observation or measurement, a change in the viewpoint or usage of the ontology, newly-gained access to information that was previously unknown, classified or otherwise unavailable and so on. The domain itself may also change, as the real world itself is not static but evolves over time. More examples of changes can be found in [69], [92].

In order to cope with the complexity of the problem, six phases of ontology evolution have been identified, occurring in a cyclic loop [100]. Initially, we have the *change capturing* phase, where the changes to be performed are identified. Three types of change discovery have been identified: structure-driven, usage-driven and data-driven [49]. Once the changes have been determined, they have to be properly represented in a suitable format during the *change representation* phase. The third phase is the *semantics of change* phase, in which the effects of the change(s) to the ontology itself are identified; during this phase, possible problems that might be caused in the ontology by that change are also determined and resolved. The *change implementation* phase follows, where the changes are physically applied to the ontology, the ontology engineer is informed on the changes and the performed changes are logged. These changes need to be propagated to dependent elements; this is the role of the

change propagation phase. Finally, the *change validation* phase allows the ontology engineer to review the changes and possibly undo them, if desired. This phase may uncover possible problems with the ontology, thus initiating new changes that need to be performed; in this case, we need to start over by applying the change capturing phase of a new evolution process, closing the cyclic loop.

We will try to illustrate the role of the six phases using an example. Suppose that, for some reason, we decide to remove a concept A from our ontology. During the change capturing phase we identify the need to remove A and initiate the six-phase process of ontology evolution.

During the change representation change we determine the kind of change that must be performed in order to remove A. There are two major types of changes, namely elementary and composite changes [100] (also called atomic and complex in [102]). Elementary changes represent simple, fine-grained changes; composite changes represent more coarse-grained changes and can be replaced by a series of elementary changes. However, it is not generally appropriate to use a series of elementary changes to replace a composite change, as this might cause undesirable side-effects [100]. The proper level of granularity should be identified at each case. Examples of elementary changes are the addition and deletion of elements (concepts, properties etc) from the ontology. In our particular example, a simple "Delete_Concept" operation should be enough to perform the required change. This is not always the case though. For example, we might have wished to move the concept A to some other point in the concept hierarchy. This is represented by a composite change. There is no general consensus in the literature on the type and number of composite changes that are necessary. In [100], 12 different composite changes are identified; in [92], 22 such operations are listed; in [102] however, the authors mention that they have identified 120 different interesting composite operations and that the list is still growing! In fact, the number of definable composite operations can only be limited by setting a granularity threshold on the operations considered; if we allow unlimited granularity, we will be able to define more and more operations of coarser and coarser granularity, limited only by our imagination [71]. Thus, creating a complete list of composite operations is not possible, but, fortunately, it is not necessary either, since a composite operation can always be defined as a series of elementary operations [71] (at least approximately).

Once the required change is identified (i.e., "Delete_Concept" for the concept A in our example), we can proceed to the next step of the evolution process, which is the "semantics of change" phase; at this point, we should identify any problems that will be caused when the decided change is actually implemented, thus guaranteeing the validity of the ontology at the end of the process. In our example, we need (among other things) to re-classify the instances of concept A by determining what to do with those instances; for example, we could delete them or re-classify them to one of A's superconcepts. In [100], the authors suggest that the final decision is made indirectly by the ontology engineer, through the selection of certain pre-determined "evolution strategies", which indicate the appropriate action in such cases. Other approaches are also possible (see [49]).

During the implementation phase, the changes identified in the two previous phases are actually implemented in the ontology, using an appropriate tool, like, for example, the KAON API [100]. Such a tool should have transactional properties, based on the ACID model, i.e., guaranteeing Atomicity, Consistency, Isolation and Durability of changes. It should also present the changes to the ontology engineer for final verification and keep a log of the implemented changes [49].

The change propagation phase should ensure that all induced changes will be propagated to the interested parties. In [80], two different methods to address the problem are compared, namely push-based and pull-based approaches. Under a push-based approach, the changes are propagated to the depended ontologies as they happen; in a pull-based approach, the propagation is initiated only after the explicit request of each of the depended ontologies. In both [80] and [100] the authors choose to use the former approach (push-based).

Alternatively, one could avoid this step altogether, by using an ontology versioning algorithm [70], allowing the interested parties to work with the original version of the ontology and update to the newer version at their own pace, if at all. This alternative is considered more realistic for practical purposes, given the decentralized and distributed nature of the Semantic Web [56].

Finally, the change validation phase should allow the ontology engineer to review and possibly undo the changes performed, or initiate a new sequence of changes to further improve the conceptualization of the domain as represented by the ontology.

The above model for ontology evolution allows us to ignore heterogeneity problems. Obviously, any ontology evolution method will collapse in the face of heterogeneous information, unless coupled with an algorithm like those discussed in the previous subsection. However, heterogeneity is not an issue under the proposed model, because ontology evolution algorithms assume human participation in the process; the ontology engineer identifies the change to be performed during the change representation phase, so it can be reasonably assumed that the change will be represented in a suitable terminology.

Once the actual change has been performed, ontology versioning algorithms come into play. Ontology versioning typically involves keeping both the old and the new version of the ontology and takes into account identification issues (i.e., how to identify the different versions of the ontology), the relation between different versions (i.e., a tree of versions resulting from the various ontology modifications) as well as some compatibility information (i.e., information regarding the compatibility of the various ontology versions).

Several non-trivial problems are associated with this task. For example, any ontology versioning algorithm should be based on some type of identification mechanism to differentiate between various versions of an ontology. This task is not as easy as it may seem; for example, it is not clear when two ontologies constitute different "versions". When a comment changes, does this constitute a new version? When a concept specification changes, but the new specification is semantically equivalent to the original one, does this constitute a new version? This and similar problems are dealt with in [56] and [70].

Another desirable property of an ontology versioning mechanism includes the ability to allow transparent access to different versions of the ontology, by automatically relating versions with data sources, applications and other depended elements [69]. Other issues involved is the so-called "packaging of changes" [70] as well as the different types of compatibility and how these are identified [69].

Another related problem is the introduction of a certain "version relation" between ontological elements (such as classes) in different versions of the ontology and the properties that such a relation should have. This relation is called a change specification in [69] and its role is to make the relationship between different versions of ontological elements explicit. Using this relation, one can identify the changes that any given element went through between different versions; in addition, a version relation should include certain meta-data regarding these changes [70]. Similar considerations led to the definition of "migration specifications" [106], which associate concepts between different versions of an ontology after a change has been performed.

As an aid to the task of ontology versioning, certain tools have been developed which automatically identify the differences between ontology versions. PROMPTDIFF [93] uses certain heuristics to compare different versions of ontologies and outline their differences, by producing a "structural diff" between them; OntoView [70] contains a tool similar to PROMPTDIFF, whose output is a certain "ontology of changes", i.e., an ontology that stores the types of changes performed, thus providing a kind of "mapping" between versions. [71] contains an excellent survey on the different ways that can be used to represent a change, as well as the relation and possible interactions between such representations; furthermore, a standard "ontology of changes" is proposed, containing both basic (i.e., elementary) and complex (i.e., composite) operations. A method to identify compatibility between ontology versions is presented in [56], [57] where the SHOE language [78] is used to make backward

compatibility between versions explicit in a machine-readable format, allowing a computer agent to determine compatibility between versions. This is an indirect approach to the problem of ontology versioning, as it allows the computer agent to determine autonomously which version to use, as opposed to [69], [70], where a more direct and centralized path is taken.

Ontology Integration and Merging

In short, both ontology integration and ontology merging refer to the development of a new ontology based on the information found in two or more source ontologies. However, the exact meaning of each term is not clear in the literature, as they are often used interchangeably. This situation has led to a certain confusion regarding the exact boundaries of each field.

In [94], [95] ontology merging is defined as the process of creating a new, coherent ontology that includes information from two or more source ontologies. In these papers, ontology merging is implicitly assumed to include the process of resolving any possible heterogeneities between the merged ontologies, even though this is not apparent in their definition. Under that understanding, the difference between ontology merging and alignment is that ontology merging results in the creation of a new ontology, while in ontology alignment the merged ontologies persist, with links established between them. A similar use of the term can be found in [84], while, in [56], the same research area is described using the term ontology integration. According to [76], ontology merging amounts to making sure that different agents use the same terms in identical ways (in a manner similar to ontology alignment). In [65] ontology integration is defined as the process of combining ontologies to build new ones, but whose respective vocabularies are usually not interpreted in the same domain of discourse.

In this work, we will define the terms along the lines of [97], which was an attempt to disambiguate between different uses of the term ontology integration. Three different uses of the term were identified in that paper. The first refers to the *composition (via reuse) of ontologies covering loosely related domains (subjects)*; this is mainly used when building a new ontology that covers all these subjects. The term *ontology integration* has been reserved for this process. The second use of the word refers to the *combination of ontologies covering highly overlapping or identical fields*; this process is used to unify ontologies that contain information about the same subject into one large (and hopefully more accurate) ontology. The term *ontology merging* was attached to this interpretation. Finally, the third use of the term integration refers to the development of an application that uses one or more ontologies; the more appropriate term *ontology use* was reserved for this process. In our work, we are interested in the first two understandings of the term, namely ontology integration and merging.

There are certain subtle differences between the processes of integration and merging. Ontology integration is mainly applied when the main concern is the reuse of other ontologies. The domain of discourse of the new, resulting ontology is usually more general than the domain of the source ontologies. Integration often places the different ontologies in different modules that comprise the resulting ontology. On the other hand, in ontology merging, the focus is on creating an ontology that combines information on a given topic from different sources. In this case, the information from the source ontologies is greatly intermingled, so it's difficult to identify the part(s) of the final ontology that resulted from each source ontology. A more extensive discussion on the above subject can be found in [97].

It is a common practice in the literature (see for example [15], [56], [94] or [97]), to consider heterogeneity resolution to be an internal part of ontology merging or integration. This is a reasonable choice, because in most cases the combined ontologies come from different sources, thus they are usually heterogeneous in terms of vocabulary, syntax, representation etc. Therefore, the task of relating the heterogeneous elements of the source ontologies constitutes a major part of the task of ontology merging (or integration). This is

mostly true in merging, where the domain of discourse is (almost) identical. This has led to even more confusion on the exact meaning of the terms, as several works consider ontology merging (or integration) and alignment to be variations of the same problem (see for example [76], [94]).

However, it should be clear that simply resolving the heterogeneity issues between two ontologies is not enough for successful integration (or merging); recall that different ontologies may encode different viewpoints regarding the real world, thus several conceptual differences are bound to exist, even if the same terminology is used. This is reminiscent of how beliefs held by different people are often different (and in some cases contradictory), even if a common terminology is agreed upon. Similarly, modeling conventions and choices may be different; one example of modeling choice that often depends on personal taste or convention is whether to model a certain distinction between similar elements by introducing separate classes or by introducing a qualifying attribute relation in one class [18]. Such modeling differences need to be taken into account when selecting what to keep from each ontology during the process. Reckless inclusion of ontology elements from the source ontologies (even when homogeneous) is likely to lead to a problematic, invalid or inconsistent ontology. For this reason we argue that ontology merging (and integration) are not simply a variation of ontology alignment.

According to [18], the process of merging can be broken down in five different steps. During the first step, we identify the semantic overlap between the source ontologies; during the second, we devise ways (transformations) that will bring the sources into mutual agreement in terms of terminology, representation etc. In the third step, we apply these transformations, so we can now take the union of the sources, which is the fourth step of the process. The final step consists of evaluating the resulting ontology for consistency, uniformity, redundancy etc; this evaluation might force us to repeat some or all of the above steps. We believe that the same generic steps are necessary for integration as well, but they should be adapted to the different properties of the process. The authors of [18] claim that their translation tool is a necessary part of any merging algorithm, as it facilitates the design and implementation of the transformation used in the merging process (second and third step, respectively).

An extensive list of works related to ontology merging can be found in [15]. The main tools for ontology merging are PROMPT [94], [95] (originally called SMART [96]) and Chimaera [84]. These tools use a semi-automatic approach focused on suggesting how elements from the source ontologies should be merged in the resulting ontology. The final choice relies on the ontology engineer. Some ideas on ontology merging (called integration there) in the context of SHOE can be found in [56]; however, their work is focused on the part of merging that deals with heterogeneity resolution. An interesting theoretical approach to ontology merging can be found in [9], while in [95] some interesting connections of object-oriented programming with the problem of ontology merging are uncovered. The FCA-MERGE algorithm [103] performs ontology integration in a very efficient way, but is based on certain strong assumptions. A more detailed list of tools and systems related to the problem can be found in [97]. It should be emphasized that all the currently available tools use manual or semi-automatic methods to perform merging (or integration), guiding the user through the different steps of the process via some intuitive interface. The processes behind ontology merging and integration are not yet well-understood: in [97], it is claimed that "ontology merging is currently more of an art".

Even though the issue of evaluating ontology merging techniques is still open in AI [103], certain comparison attempts have been made. In [75], the authors perform a comparison between PROMPT [94] and Chimaera [84] in the context of bioinformatics. In [94], the same two tools are compared with the generic Protégé-2000. Furthermore, [84] compares the efficiency of ontology merging with a simple plain-text editor, merging with the Ontolingua editor and merging with the specialized tool Chimaera, which is described in the same paper.

Discussion on the Current Research Direction

Critique of Existing Ontology Evolution Tools

The rest of this work focuses solely on the problem of changing an ontology in the face of new information (ontology evolution). We assume that heterogeneity is not an issue, or that it has been somehow resolved using an ontology alignment or related algorithm. We also ignore the issue of propagating the changes actively or passively (via an ontology versioning algorithm). This does not mean that we consider these problems less important or easier to address; however, they are irrelevant to our main research focus, which is the problem of identifying the changes that should be performed to an ontology when new information becomes available.

The above assumptions move the focus of our attention to the problem of ontology evolution instead of the whole ontology change problem. Our approach could also be useful in ontology merging and integration, to the part that goes beyond alignment, namely to the part of incompatibility resolution between the (homogenized) source ontologies.

The current state-of-the-art in ontology evolution, as well as a list of existing tools that help the process can be found in [49]. Some of these tools are simple ontology editors, while others provide more specialized features to the user. In some cases, the user can define some kind of pre-defined "evolution strategies" [100] that control how changes will be made, thus allowing the tool to perform some of the required changes automatically. Other tools allow collaborative edits, i.e., several users can work simultaneously on the same ontology [28]. In other cases, features related to ontology versioning, undo/redo operations and other helpful utilities are included in the tools [28].

Despite these nice features, there is a certain part of ontology evolution that is inadequately handled by current tools [80]. The contribution of our work lies mainly on this problematic part, which resides in the second and third phase of ontology evolution, namely the change representation and the semantics of change phase [100]. Before these phases, the need for a change is identified; during these phases, we represent the change in a suitable format and determine what modifications must be made to the ontology in response to this need as well as the indirect effects of these modifications to the validity, consistency and quality of the information stored in the ontology; if such problems are detected, further changes will be required to restore the ontology to an "acceptable" state. Once all these changes have been determined, they can be applied to the ontology, propagated to dependent elements and validated (during the fourth, fifth and sixth phase respectively).

Unfortunately, current state-of-the-art approaches simply avoid the difficulty of dealing with these two phases (second and third) by letting the user handle them. To the authors' knowledge, there is no tool acting like a "black box", i.e., a tool that receives the update and the ontology and returns the updated ontology in the output, without user participation. At best, there is some kind of pre-defined "evolution strategy" [100] that controls how changes will be made or a declarative specification of the required changes using a suitable language [101]. In short, current work on ontology evolution resorts to ontology editors or other, more specialized tools whose aim is to *help* users perform the change(s) manually rather than *performing* the change(s) automatically [49].

We believe that this approach will prove itself insufficient in the long run. Even though such tools and the features they include (change propagation, transactional properties of changes, undo/redo operations, collaborative edits etc) are undoubtedly useful, they need to be coupled with methods that allow the system to handle the whole cycle of ontology evolution by itself, without human intervention. In this respect, we believe that current research is misguided. In the rest of this section we will provide reasons for this claim and propose a direction for future research, based on the research area of belief change, that will, hopefully, lead to a solution to this problem.

One problem with the current approaches is that they require a varying level of human intervention in order to work properly. Some researchers consider this a necessary feature of

the ontology evolution process [48], [49]. Indeed, any given change can be resolved in several ways, so it is difficult for a computer system to decide on the best way to resolve it because the user requirements may be differing, even for the same change [101]. Despite that problem, we believe that this is not a practical approach to be taken, because it is unrealistic to rely on human beings for ontology evolution [100]. First of all, the human user that intervenes in the process should be an ontology engineer and have certain knowledge on the domain. Very few people can be both domain and ontology experts. But even for these specialized experts, it is very hard to perform ontology evolution manually [48], [100]. So, in domains where changes occur often, it is simply not practical to rely on humans.

Furthermore, it is unrealistic to believe that an ontology engineer will always be nearby whenever an ontology needs to change. For example, a robot may include an ontology as part of its knowledge, as well as sensors that provide information on the surrounding world; if the robot needs to behave autonomously, it should be able to handle any new information received from its sensors, including information that contradicts its current (ontological) knowledge, as well as to update its knowledge, if necessary.

A human-based evolution approach will, in many cases, perform better, in the sense of rationality, than any computer system devisable. However, there are exceptions to this rule. Different ontology engineers may have different views on how a certain change should be implemented [100]. These views may be affected by commonsense knowledge, personal preferences or ideas, subjective opinions on the domain and so on. This means that there is no single "correct" way of updating an ontology. A computer-based evolution could, at least, guarantee determinism, objectivity and reproducibility of the results, even though some people may disagree on how a change was implemented. But then, is there a consensus on the effects of a given change even among humans? Do we revise our beliefs in the same way? If two ontology engineers are presented with the same ontology and the same change (or sequence of changes), is it certain that the final result (updated ontology) will be the same? We doubt that even humans can agree on the best way to perform a given change.

Another source of problems for manual ontology evolution is the complexity of modern day ontologies: such complex ontologies are usually developed by several ontology engineers. A change in one part of the ontology might have unintended effects in other parts of the ontology [101]. The person who made the change may be unaware of the full content of his change's effects, as he doesn't know all the parts of the ontology. In fact, no single ontology engineer may be in position to determine the full effects of a change, unlike a well-designed computer system.

The above points uncover the need for automatic ontology evolution; computer-based ontology evolution is not only *necessary* for many applications, it is also *desirable* in certain contexts. A human supervision by specialized experts should be highly welcome and it should be encouraged whenever possible; however, the system should be able to work even without it. Human intervention should constitute an optional feature guaranteeing the quality and effectiveness of the ontology evolution process; but it should not be a necessary one.

Another problem with the current research is related to the representation of changes (second phase of ontology evolution). In tools that are simple ontology editors, there is usually little or no support for any kind of complex changes to the ontology; the user can simply delete and add ontological elements, or, at best, move or copy them [49]. In more specialized tools for ontology evolution, there is a pre-defined set of atomic and/or complex operations that are supported, providing a greater flexibility to the user. For each such operation, there is an associated procedure that handles the change as well as the effects of the change (semantics of change phase); this procedure can, in some cases, be parameterized to cover different needs. Unfortunately, there is no guarantee that the provided parameterization is enough to cover any possible need of the knowledge engineer. Unforeseeable needs may require unforeseeable reactions to a given change.

Furthermore, as explained in a previous section, there is no limit on the number of complex operations that can be considered; even if we restrict ourselves to the most common

types, there is usually a large number of them (in [102], for example, the authors claim that 120 such operations have been defined and the list is growing). This results in too many different operations (complex changes) to consider, making it impossible to define a specific process to deal with each one; a unifying approach is necessary to cover all cases.

The problem becomes even more complicated due to the fact that not all different types of change are readily available at design-time. New needs may require new operations. For operations that are not in the supported list, the ontology engineer should choose a sequence of two or more simpler (more elementary) operations of different granularity. Unfortunately, such a choice will undoubtedly affect the quality of the change, leading to unforeseeable problems [100]. For example, deleting and then re-adding a concept in a different position in the hierarchy is not the same as moving the concept; in the former case, the instances of the concept will have to be removed or re-instantiated to a different concept; in the latter, no changes to the individuals will be required. In this simple example, the effects of the splitting of the operation in two more elementary operations may be easy to determine and/or undo; in more complex cases, this is far from trivial.

In any circumstance, the above process cannot be performed without human intervention; it is impossible for a computer system to handle automatically a change that is not in its list of supported operations, even if the unknown operation can be perfectly broken down in more elementary, supported operations. How would the system know which operations to use in order to handle an operation it does not know? How would it evaluate the effects of its choice on the final result? How could it even know the desired result of the needed (unknown) operation in order to compare it with the result of the sequence of operations that substitute the unknown operation? If the system knew the semantics of the unknown operation, this operation would not be unknown in the first place! It could be handled normally and no substitution would be necessary at all. Such a problem is very likely to occur, as the creation of a complete list of composite operations is not possible [71]. It is related to the lack of a formal, unifying approach to handle the introduction of new information to an ontology.

In current approaches, a change request is an explicit statement of the changes to be performed upon the ontology; however, this request must be determined by the knowledge engineer in response to a more abstract need (e.g., an observation). Thus, current systems do not determine the actual changes to be made upon the ontology when faced with a need for a change; they simply help the user determine them and feed them to the system for implementation. Those systems that perform some of the tasks of evolution themselves (by suggesting the best way to resolve the change, or by doing some of the simpler changes themselves) have been developed using certain heuristics based on the experience and expertise of the researchers (their creators) and ontology engineers; in effect, they attempt to emulate human behavior [103]. They are not theoretically founded and their theoretical properties remain unspecified. Moreover, no formal methods of evaluation have been determined; as a result, there is no formal characterization of which result is "best" given an ontology and a change, so we can't compare two ontology evolution algorithms in terms of effectiveness or rationality.

Given the current state of the research, such issues have been neglected, because the decisions on the changes to be made are left upon humans. This way, empirical decisions based on the expertise of the person who performs the change are taken at the time when evolution is done: whenever the ontology engineer is faced with a change, he decides on his alternatives and selects the "best" one, which is then fed to the system for implementation. All these decisions are based on his expertise on the subject, not on a formal, step-by-step, exhaustive method of evaluation.

However, to develop fully automatic ontology evolution algorithms, several questions need to have a definite, formal answer. For example, how could one track down all the alternative ways to address a given change, using a formal and exhaustive process? How can one guarantee that there are no further alternatives for a given change, except the ones found?

How can a computer system decide on the "best" of the different alternatives? Most importantly, what is the definition of "best" in this context? What are the features or properties that make a certain result of a change "better" than another? Are there any properties that should be satisfied by a "good" ontology evolution algorithm?

Unfortunately, resolving the above issues in a general manner is not easy using the current research direction because each type of change is treated differently, using a stand-alone, specialized process. Unless a more formal path is taken, the ontology evolution research is doomed to never find answers to these questions. This again leads us to the need of developing a unifying, formal method of representing the change, evaluating the situation and choosing how the required change will be resolved. The above concerns are also applicable, to a certain extent, to all types of ontology change that require human intervention and supervision, mostly so for ontology merging and integration.

A More Formal Approach

In order to deal with the deficiencies of the currently used model of evolution, we propose a different, logic-based approach, heavily influenced by the related field of belief revision (or belief change) [41]. Our approach deals with the gap in the current ontology evolution research that was identified in the previous subsection, namely the manual or semi-automatic way of dealing with the second and third phase of ontology evolution. In this respect, our approach could be considered complementary to the current ontology evolution research.

The focus of belief change is on determining the most rational ways of dealing with changes on knowledge, as well as on the development of algorithms that update Knowledge Bases (KBs) automatically. This is exactly what the current ontology evolution research is missing: the development of formal results related to the "proper" way to perform the required changes and algorithms implementing such results in an automatic, effective and efficient way.

Unfortunately, works related to belief change are not directly applicable to ontologies, as they are based on different knowledge representation languages, which are not used in ontologies. So our approach consists in studying both fields in an attempt to identify common grounds and connections between them; this will hopefully allow the migration of certain techniques used in belief change into ontology evolution. Moreover, we can use the ideas expressed in works related to belief change as an inspiration for expressing similar ideas in the context of ontologies and the Semantic Web.

In order to do that, we will have to use a different view of an ontology, based on its logical properties. A formal semantics and terminology for ontologies will be required; this will be provided by *Description Logic* (DL) languages [7], a family of knowledge representation languages that will arguably play a very important role in the development of ontologies and the Semantic Web [8]. DLs provide a well-founded formal semantics, as well as enough diversity to cover very different needs; these desirable properties have triggered a significant amount of research on the subject, allowing a good understanding on the properties of this family of languages.

In the rest of this work, we will assume that ontological knowledge is expressed using some kind of DL; despite this assumption, our general ideas are transferable to other formalisms for ontology representation as well. We provide some initial thoughts on the feasibility of applying belief change techniques to ontology evolution. We perform a preliminary study on the results of this application and study its advantages and disadvantages with respect to the current model of ontology evolution. This work is not intended to provide concrete solutions to the problem of fully automatic ontology evolution; rather, we aim at providing some general guidelines towards a research direction that will ultimately give us the desired solutions. Before getting into the details of our approach, we will perform a short literature review on the main tools that will be required, namely DLs and belief revision.

Related Work

Description Logics

The term Description Logics (or DLs for short) refers to a family of knowledge representation languages, heavily used in the Semantic Web [7]. Knowledge in DLs is structured in a formal, yet intuitive way. The basic "building blocks" used to represent knowledge in DLs are *classes*, *roles* and *individuals*. Classes represent concepts, i.e., groups of objects which share some common property; examples of classes are PENGUIN, MALE, HUMAN etc. Classes can also be viewed as unary relationships. Roles, on the other hand, represent certain relations between objects, such as HAS_CHILD, MARRIED_TO, LARGER_THAN etc. Roles are binary relationships. Finally, individuals represent the objects themselves, which are used to populate classes and roles; examples of individuals are BOB, JOHN, PENGO etc. Individuals can be viewed as constant symbols. The allowable names of classes, roles and individuals form a (usually infinite) set, called the *namespace* of the DL.

These basic elements are combined to form DL formulas using *operators* and *connectives*. Operators allow the formation of more complex real-world constructions, called *terms*, expressible using more "primitive" elements (i.e., classes, roles and individuals). For example, we could represent the concept of a male human who has at least one human child, using the term:

$$\text{HUMAN} \sqcap \text{MALE} \sqcap \exists \text{HAS\_CHILD}.\text{HUMAN}$$

The above term conceptualizes the concept of a father; if this concept is important for our purposes, we may wish to give it a separate class name, i.e., creating the class FATHER. To express the fact that FATHER is exactly the same concept as the one described above we would need an *axiom*; axioms are formed using some connective. In this particular case, the needed axiom is:

$$\text{FATHER} \equiv \text{HUMAN} \sqcap \text{MALE} \sqcap \exists \text{HAS\_CHILD}.\text{HUMAN} \qquad (1)$$

Knowledge in DLs is represented using such axioms. Thus, a *Description Logic Knowledge Base (DL KB) is simply a set of DL axioms expressing knowledge regarding a domain of interest.*

There are several different operators and several different connectives that are allowed in a DL, each with its own syntax, arity and semantics (for example, $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$, $(\leq_n)$, $(\geq_n)$, $\{...\}$, $^-$). There are also 0-ary operators (constants), like $\bot$, $\top$. Similarly, there are several different connectives, like $\sqsubseteq$, $\equiv$, $\sqsubset$, disj(.,.), Trans(.). One interesting feature of the family of DLs stems from this variety: each particular DL allows for only a subset of all these proposed operators and connectives. This way, one can trade expressive power for computational efficiency and vice-versa, selecting the appropriate blend of connectives and operators for the application at hand. Some particular combinations have been considered more useful for practical applications, so they have been explicitly studied [7], but it is possible that new combinations (i.e., new DLs), as well as new operators and connectives will be proposed in the future.

Another factor that further increases this diversity is the variety of axiom types allowed in a DL. For example the axiom (1) in the example above can be viewed as a definition of the concept FATHER; in other words the class FATHER can be viewed as an abbreviation for HUMAN $\sqcap$ MALE $\sqcap$ $\exists$HAS_CHILD.HUMAN. Such axioms are called *definitorial*. But there are other axiom types as well:

$$\text{HUMAN} \sqsubseteq \forall \text{HAS\_CHILD}.\text{HUMAN} \qquad (2)$$
$$\text{MARRIED\_TO} \equiv \text{MARRIED\_TO}^- \qquad (3)$$
$$\text{MARRIED\_TO} (\text{BOB}, \text{MARY}) \qquad (4)$$

Axiom (2) has the form of an integrity constraint, stating that all humans' children must be humans as well. It is also *cyclic*, in the sense that the same class (HUMAN) is used in both the left and right hand side of the axiom. In some cases, no axiom is cyclic by itself, but a cycle occurs in a sequence of axioms; in this case we have a cyclic set of axioms. Certain DLs overrule cyclic axioms or cyclic sets of axioms; other DLs allow only for definitorial axioms,

like (1). Axiom (3) is also cyclic and deals with roles; it states that the role MARRIED_TO is symmetric. There are DLs where roles are considered second-class objects, so no axioms involving role terms are allowed. Axiom (4) deals with individuals, stating that the individual BOB is married to the individual MARY. Axioms containing assertions about individuals (also called *assertional axioms*) form the *Abox*; axioms dealing with concept and roles only (also called *terminological axioms*) form the *Tbox*. In certain contexts, the Tbox is viewed as the schema of a DL KB, expressing certain facts and relations regarding the real world, while the Abox forms the data, i.e., the part of the KB dealing with the individual objects in the domain of interest.

In our context, the strict distinction between Tbox and Abox is not useful. In ontologies, schema and data are often intermingled in such a way that it is hard to distinguish where the ontology ends and its instances begin [92]. For this reason, we will not use this distinction; in the rest of this report, we will assume that an ontology may (but need not) be populated. Thus, the term *ontology will refer to a DL KB*, i.e., a set of Abox and/or Tbox axioms expressing knowledge regarding a domain of interest and the terms ontology and DL KB will be used interchangeably.

The strongest feature of DLs is that they are equipped with formal, logic-based semantics, unlike their predecessors, i.e., semantic networks and frames [8]. The formal semantics of DLs is defined using *interpretations* I which consist of a non-empty set Δ^I and a function $\cdot^I$ which maps every concept A to a set $A^I \subseteq \Delta^I$, every role R to a relation $R^I \subseteq \Delta^I \times \Delta^I$ and every individual X to an object $X^I \in \Delta^I$. This mapping is extended to terms using the semantics of each operator; for example $(A \sqcap B)^I = A^I \cap B^I$. The semantics of each connective determines whether an axiom is *satisfied* by a given interpretation; for example the axiom $A \sqcap B \sqsubseteq C$ is satisfied by the interpretation I iff $(A \sqcap B)^I \subseteq C^I$. The semantics of each connective and operator are usually straightforward; the interested reader is referred to [7] for a full account.

The above semantics can be easily generalized to sets of axioms: a set of axioms is *satisfied* by an interpretation I iff all the axioms in the set are satisfied by I. A set of axioms P *implies* a set of axioms Q (denoted by $P \vDash Q$) iff all interpretations that satisfy P also satisfy Q. A set of axioms is *inconsistent* iff there is no interpretation satisfying it. Notice that inconsistent sets of axioms trivially imply all axioms; for this reason, inconsistent axioms (and sets of axioms) are often considered to carry no knowledge, as no meaningful inference can be made from them. For the same reason, inconsistent DL KBs are highly undesirable, so their study is very important for our purposes: any ontology change algorithm should first and foremost guarantee that the result of the change is not an inconsistent set of DL axioms.

The semantics described above allows us to conclude certain facts from any given DL KB. For example, the combination of axioms (3) and (4) above should allow us to conclude that the individual MARY is married to the individual BOB, i.e., the axiom:

$$\text{MARRIED\_TO (MARY, BOB)} \qquad (5)$$

This intuitively expected result should be (and is) deducible from the above semantics in a formal manner. Such kind of inference capabilities are provided by DLs and are necessary for any kind of formal treatment of DL KBs. Other, closely related, types of inference services are possible as well; these are irrelevant for our purposes, so we refer the reader to [7] for a full account. There has been a considerable amount of work dealing with the issue of reasoning in DLs, resulting in several algorithms and complexity results for several DLs (see, for example [6], [17], [25], [59], [60], [64], [79], [99] and [107] for a collection of these results).

The formal semantics accompanying DLs has also allowed the development of several formal results relating DLs with other formalisms, such as modal logics [7] or fragments of First-Order Logic [7], [14]. It has also allowed the development of certain DL extensions, like temporal DLs [5] or nonmonotonic DLs [26].

Another formalism closely related to DLs is the Web Ontology Language [22], known as OWL. OWL is a knowledge representation formalism for ontologies that is expected to

play an important role in the future of the Semantic Web, as it has become a W3C Recommendation. OWL comes in three "flavors" (or "species"), namely OWL Full, OWL DL and OWL Lite, with varying degree of expressive power and reasoning complexity. In OWL, knowledge is represented using an RDF-like [88] syntax. OWL contains several features allowing the representation of complex relationships between classes, roles and individuals in a pattern very similar to the one used in DLs; this close relationship was verified in [58], where OWL DL and OWL Lite (with some secondary features removed) were shown equivalent to the DLs $SHOIN^+(D)$ and $SHIF^+(D)$ respectively. On the other hand, OWL Full provides a more complete integration with RDF, containing features not normally allowed in DLs; furthermore, its inference problem is undecidable [58].

As already stated, we will assume that ontologies are represented using some type of DL; in effect, an ontology is simply a DL KB, so the two terms will be used interchangeably. This assumption prohibits the use of some ontological features which are not available in DLs, like certain meta-logical annotation features stating the author, version etc of the ontology or the feature of including an ontology in another (ontology reuse). However, these restrictions are not important for our purposes; as already stated, our focus is on determining the changes that need to be made in an ontology in response to a certain need (phases 2 and 3 of ontology evolution). Updating meta-logical features, like the author, for example, is a trivial operation in this respect, even though certain types of change (like the change of the version number for example) may induce complications in other phases of the ontology change process. Regarding the ontology reuse feature, we will assume that there is a preprocessing phase in which the included ontology (or ontologies) is (are) physically copied to the including ontology; for our purposes, this is the same as keeping them separate, even though the two approaches are not generally equivalent.

Unless otherwise specified, we will not assume any special properties for the underlying DL, nor any specific operators and/or connectives; in effect, we assume that ontologies are represented in some generic, unspecified DL. We will not even assume that the DL at hand is one of the DLs that have been studied in the literature; this allows our results to be useful in a variety of current and/or future DLs.

Belief Change

One's beliefs are not generally static, but evolve over time. This may be because of several reasons: new, previously unknown, classified, or otherwise unavailable information may have become known; a new observation or experiment may reveal a new fact; or the actual world (domain of interest) may change, as domains are in general not static themselves. In all the above cases, the beliefs held should be somehow adapted to the new information. Alternatively, the new information could be ignored.

The above thoughts are valid for any structure (i.e., KB) holding information (beliefs, facts, rules etc) regarding a domain of interest. The research area of *belief change* deals with this problem: *the adaptation of a KB to new information*. While knowledge representation deals with static knowledge and how this can be represented, belief change deals with the dynamic aspects of knowledge, i.e., how this knowledge evolves.

Several philosophical and practical issues related to belief change can be identified. A detailed description of such issues is outside the scope of this work; we will briefly touch the main tradeoffs and problems that have been dealt with in the field and point to certain references where the various issues are analyzed in more detail. The purpose of this short review is to show the diversity of the problem, as well as the variety of approaches that have been considered. The amount of diverse work performed in the area, as well as the maturity of many approaches, allow us to hope that several belief change techniques, ideas or intuitions will prove useful in the field of ontology change.

Before any practical approach on the subject of knowledge changing, one should decide on the formalism(s) that will be considered for the representation of knowledge. Most approaches in the literature deal with a family of logics, using certain quite general

assumptions that are satisfied by classical logics like Propositional Logic (PL) or First-Order Logic (FOL) [16], [29], [85]. Examples of such works are [1], [2], [3], [13], [23], [43], [46], [82], [90]. Others are restricted to the most studied logic in the area, namely PL ([19], [20], [68], [104]). There are few exceptions that deal with non-classical logics, most notably [12], dealing with nonmonotonic theories and [31], which focuses on the database paradigm. Unfortunately, we are not aware of any belief change approaches dealing with DLs or other related languages; however, there are some works which try to apply existing belief change approaches to DLs, namely [34], [35], [37] (by the authors), as well as [66], [76], [86].

The selection of the underlying language is not the only issue involved; more general questions related to the representation of knowledge should be resolved even before deciding which logic to use. For example, should we consider the knowledge being represented in the form of a *belief base* or *belief set*? Belief bases are small, finite sets of expressions of the underlying language; belief sets are large, infinite structures, closed under logical consequence. In other words, belief sets contain explicitly all the knowledge deducible from the KB, while belief bases contain some of the information explicitly, the rest being deducible via the standard reasoning (inference) mechanism of the underlying language.

As far as knowledge representation is concerned, the two views are the same, as we can generally calculate the logical consequences of a belief base in order to reveal the full knowledge of a KB, whenever necessary. But in belief change things are different. When changes are performed upon a belief base, we temporarily have to ignore the logical consequences of the base; in effect, there is a clear distinction between knowledge stored explicitly (which can be changed directly) and knowledge stored implicitly (which cannot be changed, but is indirectly affected by the changes in the explicit knowledge). In the belief set approach, all the information is stored explicitly, so there is no distinction between implicit and explicit knowledge. In the belief base approach, our options for change are limited to the belief base itself, so the changes are generally more coarse-grained; this limits our options for "proper" belief change. On the other hand, belief sets are infinite structures, so it is not possible to deal with them directly in practical applications. Some thoughts on the connection between the two approaches can be found in [39], [55]; in [54] a thorough knowledge-level discussion and motivation on belief base operations is presented; a comparison of the approaches can be found in [52], [53].

A related philosophical consideration studies how our knowledge *should* be represented. There are two viewpoints here: *foundational* theories and *coherence* theories. Under the foundational viewpoint, each piece of our knowledge serves as a justification for other beliefs; our knowledge is like a pyramid, in which "every belief rests on stable and secure foundations whose identity and security does not derive from the upper stories or sections" [42]. This viewpoint is closer to the belief base approach. On the other hand, under the coherence theory, no justification is required for our beliefs; a belief is justified by how well it fits with the rest of the beliefs, in forming a coherent and mutually supporting set; thus, every piece of knowledge helps directly or indirectly to support any other. In this sense, knowledge is like a raft, "every plank of which helps directly or indirectly to keep all the others in place, and no plank of which would retain its status with no help from the others" [42]. Coherence theories match with the use of belief sets as a proper knowledge representation format. A more detailed study on the above issues can be found in [42].

Another issue deals with the nature of the new information; in most cases, the new information is something that must be *added* to our knowledge. In other cases, the new information represents something which is wrong with our beliefs and should be *retracted*. In [1] three different types of belief change were identified. The simplest one is *expansion*, which refers to the reckless addition of information to our knowledge, without taking any special provisions to ensure the quality (i.e., consistency) of the KB after the addition. This is not an interesting operation, because, if the new information contradicts the currently held beliefs then the result will be an inconsistent (and thus useless) KB. Furthermore, it is trivial to perform an expansion operation.

The second operation is *revision*, which is the most useful operation in terms of practical applicability. Revision is similar to expansion, with the important difference that the result should be a consistent set of beliefs. In order to achieve this, one may, in some cases, need to abandon some of the currently held beliefs while adding the new information. The tough part of revision is how to select the beliefs that should be abandoned.

Contraction is, according to [41], the most fundamental operation and, consequently, the most important operation for theoretical purposes. Contraction is used when one wishes to consistently remove information from a KB. This may be necessary when a piece of information becomes unreliable and we would like to stop believing it. A contraction operation should remove the unreliable information from both explicit and implicit knowledge; simply retracting the information from explicit knowledge may not be enough, as it may re-emerge as a consequence of the remaining beliefs. Thus, a contraction operator may be forced to also remove beliefs which on first look seem unrelated to the retracted piece of knowledge.

The above operations deal with a static world; in effect, the world itself does not change, but our perception of the world does. Thus, revision and contraction are used when some new information about the real world has been disclosed, forcing us to change our conceptualization of the world in order to represent it in a (hopefully) more accurate manner. But this is not the only change possible, because the real world might change as well; in this case, the KB should be adapted to the new reality. The semantics of this kind of change is different, forcing us to introduce a new pair of operations, namely *update* and *erasure* [67]. Update is similar to revision (it refers to addition of information) while erasure is similar to contraction (it refers to removal of information); however, they both apply when the world dynamically changes.

One example may help clarify the situation: suppose that you have three independent switches A, B and C and the information that exactly one of A, B or C is on. Our knowledge could be represented by the PL KB:

$$(A \land \neg B \land \neg C) \lor (\neg A \land B \land \neg C) \lor (\neg A \land \neg B \land C)$$

Now, suppose that, through observation, we notice that A is on. The proper reaction to this kind of information is to assume that B and C are off, as this coincides with both the fact that exactly one switch is on (old KB) and with the fact that A is on (change). So the new KB should be:

$$A \land \neg B \land \neg C$$

In this particular case, we choose, among the three possible states of the world, the one that agrees with the new information. This is the case in a revision operation.

On the other hand, if we had sent a robot into the room with the order to turn A on, then we have no reason to assume that A was on before the robot was sent in. The robot changes the real world by switching A on, so A should be on in the new KB, but that's all we know. So we should assume that at the end of the robot's actions, either A alone, or A and B, or A and C are on. The new KB should be:

$$(A \land \neg B \land \neg C) \lor (A \land B \land \neg C) \lor (A \land \neg B \land C)$$

We conclude that the expected result is different than in the case of revision; we don't choose among the possible worlds, rather we alter each of the possible worlds so as to coincide with the new fact. This is the case in an update operation.

Notice that in the example above, the original KB and the information that initiated the change (A) are formally represented in the same way; however, the desired result of the change is different. For this reason, the introduction of the new operations is necessary in order to capture the different semantics that the same change can take. For a thorough discussion on the differences between the various operators, see [67].

The format and semantics of the new information may also vary. Under the standard approach, "new information" is a single new expression of the underlying logic, which is received and accommodated to our knowledge in an autonomous, independent manner. In other approaches, the new information can be a whole set of expressions and may have

package or *choice* semantics [40]. This semantics was introduced for contraction: under package semantics, all expressions in the set must be retracted from the KB; under choice semantics, it is sufficient that at least one of them is retracted. In other cases, belief change is considered to be a continuous process, so each change is related to previous (and future) changes. Under this viewpoint, the change is not a one-off, standalone process, but a set of changes occurring in a sequence. This type of belief change has different semantics and is treated in the field of *iterated belief change* [21], [63], [77], [89]. In other cases, the new information may be a whole new KB; in this case, we are dealing with *belief merging* [72], [74] which is the analogous of ontology merging for KBs.

Once the above technicalities have been dealt with, one has to decide how the change itself should be performed. In general, there are two approaches here: *postulation* or *explicit construction* [82]. Under the postulation approach one seeks to formulate a number of formal conditions in the form of certain postulates that a belief change algorithm should satisfy in the given context. Under the explicit construction approach, one seeks certain explicit algorithms or constructions leading to algorithms.

In fact, the two approaches are not rivalrous but complementary. In the process of developing an explicit construction, one may identify certain desired conditions for this construction (or algorithm), something like a "request specification", that could lead to a postulation. On the other hand, postulation may produce better results if its initial findings are checked with an explicit construction that satisfies (or does not satisfy) the postulates; this construction may help in determining the applicability and rationality of the proposed postulation and lead to refinements and improvements.

Both methods have been used in belief change and both have given interesting results. Explicit belief change algorithms are presented in [19], [20], [23], [27], [104]; constructions that lead to families of algorithms are given in [1], [2], [3], [13], [39], [43], [46], [63], [68], [90], [105]; possible postulations of the process are provided in [1], [23], [44], [54], [63], [67], [68]. Such works have been developed in a parallel and complementary manner [82]; this is also apparent by the overlap in the above lists of works. On the other hand, current research on ontology change uses only the explicit construction method. One interesting side-effect of our approach is that it allows the development of a postulation method for ontology evolution.

Most postulation approaches try to resolve certain philosophical issues for which, in general, there is no unique answer because such an answer depends on the application at hand. The more practical approaches (explicit constructions) usually take a certain stance on each of these issues, thus determining the properties of the algorithm or construction. We have already seen several such issues, mostly related to the representation of the change and knowledge; in the following we will briefly discuss those related to the actual implementation of the change, which is the main focus of belief change.

One such issue is related to the acceptance of the new information. It is usually assumed that the new information (revision, contraction or other) is accepted unconditionally, so the resulting KB should contain the new information (or should not contain it, depending on the operation at hand). This implies a complete reliance to the incoming data, according to the *Principle of Primacy of New Information* [19]. This principle coincides with common intuition, because the new information generally reflects a newer and more accurate view of the domain. Alternatively, one could review the new data under a critical eye and possibly reject it, partially or totally; this is the case in the so-called non-prioritized belief change [50]. This is most useful in an agent communication context, or when the information may be obtained by unreliable sources.

Some researchers argue that the result of a change should not be affected by the syntactical representation of the KB or the change (*Principle of Irrelevance of Syntax* [19]). Normally, the principle applies in both the KB and the change; however, it is generally possible that it could be applied in the KB only or in the change only. This principle generally fails for belief bases (under the foundational viewpoint), because logically equivalent bases

may be formed using completely different sets of axioms, implying different justifications and, thus, different ways of adapting the KB to the new data [54]. A stronger version of this principle may be of use in the foundational case. For belief sets (and coherence theories) though, this principle is generally valid [1], [19], [41].

As already argued, inconsistent KBs (under classical logic) carry no interesting information and, consequently, should be avoided. For this reason, it is generally accepted that the result of a change should be a consistent KB, according to the *Principle of Consistency Maintenance* [19]. This is not an issue for nonmonotonic and paraconsistent KBs, in which the underlying logic itself has an internal mechanism to deal with inconsistencies [4], but for classical logic the principle is valid (see also [73] for a recently proposed alternative approach). The only thing that remains to be settled is the exact meaning of the term "consistency"; in the belief change literature, the meaning of the word is clear, denoting a KB which implies a proposition that is tautologically false. For ontology change however, the term "consistency" has been used (others would say abused) to denote several different things; we will address this issue in a later section.

Two more basic principles have been proposed in the literature: the *Principle of Fairness* guarantees determinism and reproducibility of the result of a change, while the *Principle of Adequacy of Representation* guarantees that the new (modified) KB will be represented using the same underlying formalism as the old one [19]. These principles are necessary for technical reasons, so they are often implicitly used in the literature without explicit introduction.

Undoubtedly, the most important principle related to the way a change is implemented is the *Principle of Minimal Change* [68], which can be found by several names in the literature, like the *Principle of Persistence of Prior Knowledge* [19] or the *Principle of Conservation* [42]. This principle states that the new KB should be as "close" as possible to the original KB; in other words, from all the possible change results that satisfy the other principles, one should choose the one that retains "the most information" from the old KB.

To the authors' knowledge, there is no debate on whether this principle should be valid for belief change operations; this is taken for granted. However, there is no consensus on how this principle should be formally expressed. For such a formulation, the exact meaning of "closeness of KBs" needs to be defined, as well as how the "loss of information" can be measured. There have been several proposals on metrics that count information loss in several ways, being used in different algorithms or representation results (for example [2], [19], [20], [24], [39], [43], [46], [68], [82], [104]); there are also different postulations that capture this principle in a different manner [1], [54] and long debates on the pros and cons of each postulation [41], [52], [83].

One of the reasons for this diversity is that logical considerations alone do not tell us what has to be given up in a certain change [41]; extra-logical information needs to be considered when it comes to choosing. The way this information is structured and used determines the stance of the approach towards the Principle of Minimal Change. In fact, the formulation of this principle is in the core of each belief change algorithm or postulation system, being the very essence of belief change and determining the properties of any given approach to a large extent. This notion also provides interesting connections between belief revision and several related research areas, like defeasible inference, counterfactual conditionals and others (see [81]).

The above considerations form only a partial list of the issues that have been discussed in the belief change literature. This analysis shows that the determination of the change(s) to be made in response to some new data is a complex and multifaceted issue and that several considerations need to be taken into account before choosing the modifications to be made upon a KB. The same considerations hold for any type of knowledge change, including ontology change. For this reason, we argue that current ontology change approaches handle the problem in an inadequate manner, as they are not even considering most of the above issues in the determination of their algorithms.

In this respect, it is interesting to note that, in the belief change literature, there is no human involved in the process of change, unlike the ontology change literature. All the approaches in belief change deal with the problem of changing the knowledge in a fully automatic manner. In fact, to the authors' knowledge, the option of using a human in the loop of belief change was never even considered as an option, despite the complexity of the problem. This fact forms an additional argument in favor of the use of belief change techniques in ontology evolution, as such techniques will arguably lead to automatic methods for dealing with this problem too.

Our Proposition

As already stated, our approach deals with the second and third phase of ontology evolution; therefore, our method should determine, first, how changes should be represented and, second, what modifications should be made upon the ontology in response to a given change. The ultimate objective of our approach is the exclusion of the human user from the loop of ontology evolution. As stated in the introduction, the focus of this work is not to provide answers and solutions, but to set stable foundations upon which future work towards this objective may be based.

Representation of Ontologies

Before representing the changes, one should first decide on the representation of the ontologies themselves. Most ontology editors (such as [84], [94]) use a graphical interface to represent and manipulate ontologies [49]. This allows the user to actually view the changes performed through an intuitive interface. This graph-based viewpoint regarding the ontology representation is pervasive in most ontology evolution algorithms, as it affects the decisions on how each change should be implemented.

Graphical representations are extremely useful for visualizing the way that the domain conceptualization was implemented in an ontology. They also help novice users and domain experts get acquainted with the field and understand the conceptualization, by hiding much of the semantic and syntactic complexity of the ontology behind intuitive interfaces and simple visual metaphors [93]. They allow one to view the concept hierarchy at a glance, see the roles that connect concepts etc. However, they are often not expressive enough for complex applications, because certain complex DL axioms cannot be easily expressed using a graph; for example, one cannot easily express in a graph the axiom:

$$A \sqcap B \sqsubseteq \forall R.\exists S.(A \sqcup \neg D) \sqcap C$$

Similarly, axioms using complex connectives, such as $\sqsubset$, $\not\sqsubseteq$ etc cannot be directly represented. This might be a problem for certain DLs that allow this kind of axioms.

Current ontology evolution approaches are based on the structure of the ontology in a graph-based representation, which has shifted the focus of the relevant research to concepts, roles, individuals and how they are structured in the ontology graph; in effect, most existing work on ontology evolution builds on frame-like or object models [48]. Arbitrary axioms are often not considered part of an ontology and little or no attention is paid to them in ontology change [95]. This results in the loss of much of the expressiveness of the underlying language, disallowing, for example, the definition of constraints like the above.

For knowledge engineers and ontology experts, an algebraic representation provides a more concise and formal representation of the conceptualization, has a cleaner semantics and allows easier formal manipulation than the graph-based approach. In fact, a combination of the approaches usually works best, as it allows us to use the best of both worlds.

Under the algebraic approach, the knowledge of the ontology is stored as a pair (S,A), where S is the vocabulary (or signature) containing information on the elements appearing in the ontology (concepts, roles, individuals) and A is a set of ontological axioms [65]. The vocabulary may be a single unstructured set containing all the concepts, roles and individuals relevant to the ontology, or it may have some structure denoting, for example, the concept

hierarchy. The set of ontological axioms may contain an arbitrary number of axioms; in our case, these should be axioms of the DL used to represent the ontology.

In this work, we will use a simplification of the algebraic approach, by dropping the signature structure and *representing an ontology as a set of DL axioms, under a given, predefined DL* (DL KB). This way, our approach focuses on axioms, following the axiom-centered ontology model [48], ignoring the signature of the ontology.

This model is based on a very general logical framework introduced by Tarski in 1928. Under that framework a *logic is a pair <L,Cn>*, where L is a set of expressions or propositions of the underlying language and Cn is a function mapping sets of propositions to sets of propositions (*consequence operation*). The intuitive meaning of Cn is that a set X implies exactly the propositions contained in Cn(X). It is assumed that Cn satisfies three intuitive properties that allow it to behave in a rational manner. More specifically, Cn is constrained to satisfy the following relations:

For all X⊆L, X⊆Cn(X)   (*inclusion*)

For all X⊆L, Cn(X)=Cn(Cn(X))   (*iteration*)

For all X, Y⊆L, if X⊆Y then Cn(X)⊆Cn(Y)   (*monotony*)

Using this operator one can easily define an *inference relation* between sets as follows: X⊢Y ⇔ Y⊆Cn(X). It can be shown that the inference relation thus defined satisfies the following properties:

For all X, Y⊆L, if X⊆Y, then Y⊢X   (*reflexivity*)

For all X, Y, Z⊆L, if X⊢Y and X∪Y⊢Z then X⊢Z   (*transitivity*)

For all X, Y, Z⊆L, if X⊢Y, then X∪Z⊢Y   (*weakening*)

The opposite can also be shown: given a relation ⊢ which satisfies reflexivity, transitivity and weakening, the function Cn(X)={y∈L | X⊢{y}} is a consequence operation (i.e., it satisfies inclusion, iteration and monotony). Thus, there is a tight correspondence between consequence operations and inference relations, making them interdefinable. This way, a logic can also be defined as a pair <L,⊢> (see also [39], [82]).

It can be easily shown that all monotonic DLs can be engulfed by the above framework, by setting the set L to contain exactly the axioms that said DL allows for and ⊢ to be the inference relation of the underlying DL under its standard model-theoretic semantics [7]. Under this formalization, a DL is represented by a pair <L,Cn> (equivalently: <L,⊢>) and an ontology is represented by a single set O⊆L, where L is the set of expressions of the underlying knowledge representation DL <L,Cn>.

Since the graphical structure of the ontology can be completely determined by a set of axioms, we believe that viewing the ontology as a set of axioms provides a more general method of ontological representation, allowing the easy switch to the graphical representation (with some possible loss of expressiveness) if necessary. Most importantly, this viewpoint of DLs (and ontologies) is necessary in order to have a "common ground" under which classical logic and DLs can be both described. This facilitates the task of migrating belief change techniques (mostly based on classical logic) to the DL framework. It is also simpler and more straightforward than the classical algebraic approach, and, as we will see, it allows us to solve one of the main problems in current ontology evolution research that we described before, namely the lack of a unifying method of treating the different kinds of changes.

The main disadvantage of this model is that we lose the information normally stored in the signature of the ontology. This is not as major a problem as it seems, because most of the information in S can be represented using axioms as well. For example, if S is a poset representing a certain hierarchy between concepts, then the hierarchy information can be expressed in the form of axioms in the set O using the subconcept connective of DLs (⊑). Things become more complicated if S is an unstructured set containing the elements relevant to the ontology; more discussion on this issue will appear at a later section.

Here is a simple example showing how a simple ontology can be represented in graphical, algebraic and logical format.

ONTOLOGY: BIRD_ONTOLOGY

BIRD⊑FEATHERS   (all birds have feathers)
BIRD⊑FLY   (all birds fly)
PENGUIN⊑BIRD   (all penguins are birds)
PENGUIN⊑∀EAT.FISH   (penguins eat only fish)
BIRD (TWEETY)   (TWEETY is a bird)

        The logical representation of the ontology is a set containing the above axioms. This information can also be represented in a graph as follows:
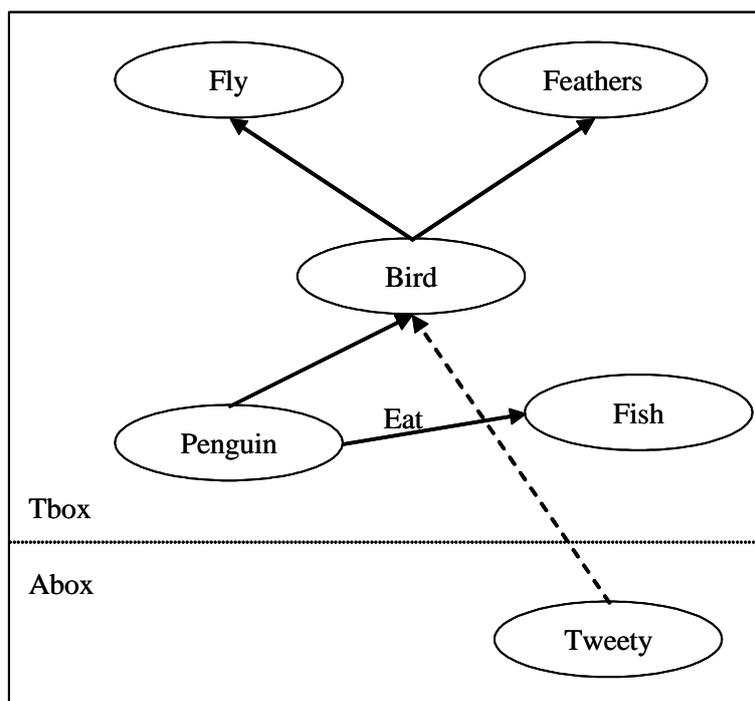
        The algebraic representation is composed of two structures: the signature S and the axioms A. The set of axioms A contains all the above axioms (as in the logical representation); the set S contains all the elements that are relevant to the ontology; one such S could be:

S={BIRD, FEATHERS, FLY, PENGUIN, FISH, EAT, TWEETY}

Change Representation Phase

The great number of different complex operations that can be defined for ontology evolution [102], each with its own process to handle it, along with the need for automatic ontology evolution, forces us to drop the current viewpoint regarding the representation of the change. In effect, we need a method that will allow us to represent all possible changes in a uniform way. This need has already been identified in the literature, as the desire for a new declarative language for the specification of a change request [49].

        We argue that this language need not be different from current ontological languages. In fact, we propose the use of the same abstraction that we proposed for ontology representation, i.e., to *represent the change as a set of DL axioms* that should be true "at the end of the day". The underlying DL should be the same as the one used for the representation of the ontology. This way, if <L,Cn> is the logic used for ontology representation, a change can be any set X⊆L.

        This is a rather naïve approach, as it ignores the cases when one needs to remove knowledge from the ontology; in this case, the information provided (i.e., the set of DL

axioms) is a set of facts that *should not be implied* by the resulting ontology. This approach also ignores the fundamental philosophical difference between a change that refers to a change in the conceptualization (i.e., revision and contraction in the belief change terminology) and a change that refers to a change in the real world (i.e., update and erasure in the belief change terminology).

To cover these needs, we propose the introduction of four different ontology evolution operations, namely *ontology revision*, *ontology contraction*, *ontology update* and *ontology erasure*. We could also introduce ontology expansion, but the operation of expansion is rather trivial, so we ignore it to keep the model simple. In all the above cases, the change is represented using a set of DL axioms; however, this set of DL axioms has different semantics in each operation. This causes the splitting of the ontology evolution research in four different areas, each dealing with a different type of change (operation). This is similar to the belief change field, where belief change is split in four different, but closely related, subfields, namely revision, contraction, update and erasure. It is also a great example of how the intuitions expressed in belief change can be applied to ontology evolution, saving us from potential pitfalls.

More specifically, we *define ontology revision as the process of incorporating in an ontology some piece of knowledge regarding a static world*. Similarly, *we define ontology contraction as the process of retracting from an ontology some piece of knowledge regarding a static world*. Analogously, *ontology update is defined as the process of incorporating in an ontology some piece of knowledge regarding a dynamic world*, while *ontology erasure is defined as the process of retracting from an ontology some piece of knowledge regarding a dynamic world*.

Operation	Type of Change (Addition/Deletion)	State of the World (Static/Dynamic)
Ontology Revision	Addition	Static
Ontology Contraction	Deletion	Static
Ontology Update	Addition	Dynamic
Ontology Erasure	Deletion	Dynamic

Table 1: Ontology Evolution Operations

Table 1 shows the properties of each of the above operations. When the type of change is an addition (revision, update), we have the incorporation of new knowledge in the ontology; in effect, the set of DL axioms of the change should be implied by the resulting ontology. When the type of change is a deletion (contraction, erasure), the set of DL axioms should not be a part of the resulting ontology, neither explicitly, nor implicitly (i.e., deducible by the ontology). When the state of the world is static (revision, contraction), we have some new observation, experiment etc regarding a world that has not changed; in effect, our knowledge is enhanced with new data, allowing us to improve the conceptualization of the world. When the world is dynamic (update, erasure), there is nothing wrong with our conceptualization, but the world itself is evolving; therefore, our conceptualization of the world should adapt to the new reality.

Representing changes at the level of axioms provides a very fine-grained approach to the problem [48]. It also saves us from the need to define complex operators for each type of change imaginable, since all the changes can be now expressed using one of the four operations above and a set of DL axioms; if the change cannot be expressed using a set of DL axioms, then the change is beyond the expressive power of the underlying DL, so we can't express it in the ontology anyway. In such a case, the problem lies on the representation language chosen, not on our approach.

In the ontology BIRD_ONTOLOGY above, if we lose our faith in the belief that all birds can fly, then we would like to remove the IsA relationship BIRD⊑FLY. In this case, we should contract with the set {BIRD⊑FLY}. If, on the other hand, we observe that penguins actually

don't fly and that TWEETY is not a penguin then we would like to revise with the set {PENGUIN⊑¬FLY, ¬PENGUIN(TWEETY)}.

Now, let us present a more concrete example that will allow us to show the differences between all four types of ontology evolution. Consider a box containing several objects, some of which are chess pieces; all chess pieces are either black or white, but all black pieces are made of plastic and all white pieces are made of wood. We also know that there is at least one chess piece in the box, call it KING. The above facts are represented in an ontology as follows:

ONTOLOGY: CHESS_PIECES

CHESS_PIECE ⊑ BLACK ⊔ WHITE	(all chess pieces are black or white)
BLACK ⊓ CHESS_PIECE ⊑ PLASTIC	(all black pieces are plastic)
WHITE ⊓ CHESS_PIECE ⊑ WOODEN	(all white pieces are wooden)
disj(BLACK,WHITE)	(nothing can be both black and white)
disj(PLASTIC,WOODEN)	(nothing can be both plastic and wooden)
CHESS_PIECE(KING)	(KING is a chess piece)

Let us view some possible changes that can be performed in this ontology and their intended effects:

- *Revision*: suppose that, through observation, we learn that KING is black. The proper change in this case should be a revision with the set {BLACK(KING)}. This would normally require no further changes to the ontology, except from the addition of the new fact: BLACK(KING); in effect, the new observation allowed us to refine our knowledge regarding the chess piece KING. Let us name this new ontology CHESS_PIECES_NEW.

- *Contraction*: suppose that, starting from the revised ontology (CHESS_PIECES_NEW), we lose our faith in the belief that KING is black. This is a contraction operation, which, once again, should be represented using the expression {BLACK(KING)}. This contraction can be easily accommodated by simply removing the above statement from the ontology.

- *Update*: suppose now that we have the original ontology (CHESS_PIECES) and paint KING black; then the change is an update. This update should be represented using the same set: {BLACK(KING)}. But the result of the change is very different this time: if the piece was originally white, then it would be wooden. But now it is painted black; thus, there will exist one chess piece that is both black and wooden. This analysis shows that the rule BLACK ⊓ CHESS_PIECE ⊑ PLASTIC should no longer be valid after the update.

- *Erasure*: suppose that we send a robot in the box with the following order: "if KING is black, paint it with an arbitrary color; otherwise, do nothing". Thus, if the color of KING is not black, then no action will be taken. If its color is black, then the robot may choose to paint it any color, including red, green, or even repaint it black. Thus, all we can assume by the robot's action is that the belief that the color of KING is black should be removed; this is an erasure operation, with the set {BLACK(KING)}. We can't assume that KING has (or does not have) any particular color, as we don't know the color that the robot selected to paint it, so update is not applicable here. This action may invalidate some of the statements of our ontology; indeed, we may now have a red chess piece, or a chess piece that is both white and plastic. Thus, the expressions CHESS_PIECE ⊑ BLACK ⊔ WHITE and WHITE ⊓ CHESS_PIECE ⊑ WOODEN are both invalidated.

If we had fed the above example to one of the current ontology editors, we would have to take the literal description of the change (for example: "KING was painted black") and then decide on the complex and/or elementary changes required; in this particular case we should remove one IsA (BLACK ⊓ CHESS_PIECE ⊑ PLASTIC) and add one assertion statement (BLACK(KING)). Alternatively, we could have chosen to weaken, instead of removing, BLACK ⊓ CHESS_PIECE ⊑ PLASTIC, by replacing it with the following IsA: BLACK ⊓ CHESS_PIECE ⊓ ¬{KING} ⊑ PLASTIC. The latter statement asserts that if something is a black chess piece, but not the particular chess piece KING, then it is plastic. Such a statement would be more accurate in this situation. And there are other possibilities too.

Therefore, even in this simple example, there is a substantial amount of effort required by the ontology engineer in order to accommodate the change. The ontology engineer might have failed to notice the IsA removal required; such a failure would not lead to any inconsistencies, so the system could not have noticed the problem and it could take a long time before the ontology engineer noticed the problem himself. Even if he had noticed the problem in the first place, he could have chosen to remove the IsA altogether, failing to notice the second, more accurate alternative of weakening it.

In the alternative approach that we propose, the ontology engineer would need to decide on the operation required and to represent the change (in this particular case: "update with {BLACK(KING)}"). So, it would only need to feed the system with the input ontology and order it to update this ontology with the above expression. The system should then (somehow) perform the update in a proper and automatic way. The problem of the system deciding on the changes to be performed is very difficult, but there are several belief change techniques that could be of use here. We will deal with this part of the problem in the next subsections. The important thing with these techniques is that they act in a formal and deterministic manner, eliminating the possibility of mistakes like the ones described above.

In effect, what we are proposing is an extra layer in the system, that will perform the actions currently handled by humans (phases 2 and 3). Once the problem of deciding the changes to be made is solved, the system could then (automatically) forward the results to the next phase for the implementation of these changes; from this point on, the process can be automatically handled by existing systems. For this reason, our approach actually supplements the existing approaches.

Of course, even in our approach, there is some user interaction that is required; however, this could be easily avoided if necessary. For example, a robot can reasonably assume that all information it receives through its sensors are revisions; sensors should have inbuilt logic that allows them to translate observations (like the observation that a particular chess piece is black) to logical statements (i.e., {BLACK(KING)}). Once this is accomplished, the whole process can be fully automated. But even if this is not the case, the human interaction required is substantially less than the one required in standard ontology evolution approaches.

Open Versus Closed Vocabulary Assumption

Unfortunately, there is one important deficiency in our method. One common type of operation that occurs in ontologies is the addition or deletion of concepts, roles and individuals. In the standard algebraic approach, this amounts to changing the vocabulary S of the ontology. However, the vocabulary has been dropped in our logical framework, so such kinds of changes cannot be directly expressed.

This problem stems from the fact that DLs do not contain any constructors allowing us to express the fact that "concept PENGUIN exists in the ontology". Therefore, no set of DL axioms could express the change "remove concept PENGUIN" or "add concept PENGUIN". Notice that such facts cannot be expressed in the ontology either, so this is actually a problem of the representation language.

We will propose two methods to deal with this deficiency. These methods use the concepts of *Closed Vocabulary Assumption* (*CVA*) and *Open Vocabulary Assumption* (*OVA*), as well as the *Existence Assertion Operator* (hereby denoted by %), originally proposed (under slightly different names) in [33], [35].

The existence assertion operator is a modal operator that can be used to enhance the expressive power of DLs, allowing the expression of facts like the above. The symbol % is a DL connective, followed by one element from the DL namespace to form an axiom. The intuitive meaning of such an axiom is that said element "exists" in the ontology; for example, the axiom %PENGUIN explicitly states that the concept PENGUIN exists in the ontology and is useful for our conceptualization. This amounts to saying that PENGUIN is part of the ontology's vocabulary. This is a direct improvement to the expressive power of the DL and is

applicable to all DLs. The addition of this operator to DLs eliminates the need for a special signature structure in ontologies.

In order to make this operator useful, we must equip it with formal semantics constraining how it can be used in the reasoning processes of the underlying DL; these formal semantics have appeared in [33] and will not be repeated here. Informally, two conditions are necessary in order to capture the semantics of the % operator. Firstly, any set of axioms should imply the existence of all namespace elements that appear in the set and should not imply the existence of any element that does not appear in the set. Secondly, an existence assertion axiom of the form %A provides no information about A itself, except from the fact that it is relevant to the ontology; thus, we cannot infer any non-trivial statements from %A. Therefore, a set of axioms of the form {%A, %B, ...} should only imply trivial statements, like A≡A, B⊑B and so on. These ideas are consistent with our intuition behind the introduction of the existence assertion operator.

In the above version of the existence assertion operator, a statement of the form %A does not indicate whether A is a concept, role or individual, so one could consider refining the operator in order to allow the discrimination between elements that are concepts ($\%_C$), roles ($\%_R$) and individuals ($\%_I$), in effect replacing % with $\%_C$, $\%_R$, $\%_I$. Under this viewpoint, $\%_C A$ would mean that A "exists" (as part of the ontology's vocabulary) and is a concept, $\%_R A$ would mean that A "exists" and is a role and $\%_I A$ would mean that A "exists" and is an individual. Unfortunately, it is quite hard to generalize the semantics of the single operator to capture the semantics of the more complex case with the three operators.

Indeed, the semantics of the additional operators need to capture more facts. One should make sure that something is of one type only; for example, $\%_C A$ should not imply $\%_R A$ and vice-versa. Furthermore, one should be able to correctly determine the type of a certain namespace element in an axiom; for example the axiom $\forall R.A \equiv B$ implies that R is a role, while A and B are concepts, i.e., it implies that: $\%_R R$, $\%_C A$, $\%_C B$. To correctly determine this fact, one must take into account the semantics of the DL operator $\forall$ and connective $\equiv$. The same holds for all operators and connectives admitted by the DL at hand. Finally, the two conditions of the single operator (%) should be generalized to apply to all three operators ($\%_C$, $\%_R$, $\%_I$).

The introduction of these additional operators may be an overkill for most applications. We could reasonably assume that the namespace of a DL is split into three disjoint sets, which allow us to tell in an unambiguous manner whether a certain namespace element is a class, role or individual. This eliminates the need for three different modal operators and allows us to use the former approach, with the single operator, which is much simpler and has much cleaner semantics. For this reason, we will only use the simple approach in this work.

The use of the existence assertion operator is necessary for the introduction of the Closed Vocabulary Assumption (CVA). Under CVA it is assumed that concepts, roles or individuals that are not used by an ontology (i.e., they appear nowhere in the ontology) are not relevant to the ontology. In other words, an ontology needs and uses exactly the namespace elements that explicitly appear in it. This assumption is usually desirable and is often made implicitly in the literature (using the ontology signature). For example, a University ontology uses concepts like STUDENT, COURSE etc, but does not use concepts like PENGUIN or BIRD.

The Open Vocabulary Assumption (OVA) is the exact opposite: under this assumption, all kinds of bizarre elements that the DL namespace allows us to define actually exist in the ontology. For most of them, no information may be available at the time, but this does not mean that they are not part of (or relevant to) our ontology. For example, a University ontology contains no information about birds, yet the concept BIRD is part of the ontology, as well as any other concept imaginable.

Under the standard terminology, an ontology under OVA corresponds to an ontology whose signature contains all the elements in the DL namespace; an ontology under CVA corresponds to an ontology whose signature contains exactly the elements that explicitly

appear in some axiom of the ontology. Such elements may appear in "standard" DL axioms (like $A \sqsubseteq B$) or in existence assertion axioms (like %A).

The assumption (CVA or OVA) chosen for the ontology has several consequences in both reasoning and ontology change. It also affects the definition $<L,Cn>$ of the underlying DL. Under the CVA, the existence assertion operator plays a central role in reasoning: a namespace element A exists in an ontology (is part of its vocabulary) iff %A is a consequence of the ontology. If A does not appear in any axiom (including axioms of the form %A), then %A is not a consequence of the ontology (by the semantics of %), thus A is not considered a part of (or relevant to) the ontology.

This fact has several strange side-effects; for example the axiom $A \sqsubseteq A$ is no longer a tautology. This is true because $A \sqsubseteq A$ implies %A, so it implies that A exists in the ontology; but A does not exist in an ontology that does not contain A in any of its axioms, so $A \sqsubseteq A$ is not implied by any ontology (so it is not a tautology). If we accept $A \sqsubseteq A$ as a tautology, then every ontology would imply $A \sqsubseteq A$, which in turn implies that A exists in the ontology; this way, BIRD would exist in the University ontology, voiding the founding principle of the CVA. Under the standard viewpoint, this corresponds to the inability to reason about elements that don't appear in the ontology signature. For similar reasons, $A \sqsubseteq B \sqcup C$ is not a consequence of $A \sqsubseteq B$ alone; however, $A \sqsubseteq B \sqcup C$ is a consequence of the set $\{A \sqsubseteq B, \%C\}$. If we assume CVA, we must enhance the set L of the underlying DL by adding all the existence assertion axioms that can be defined; furthermore, the consequence operator Cn must be changed according to the guidelines presented above to reflect the intuitions behind the CVA.

Under the OVA, the existence assertion operator is no longer useful and provides no real improvement to the expressive power of the underlying DL. Under OVA, all namespace elements exist in all ontologies; thus, all axioms of the form %A are actually tautologies. The bizarre reasoning behavior of the CVA no longer occurs; under OVA, $A \sqsubseteq A$ is a tautology and $A \sqsubseteq B \sqcup C$ is implied by $A \sqsubseteq B$. Thus, under OVA, no changes are required in the formal representation of the DL as an $<L,Cn>$ pair. This behavior is similar to the behavior of an ontology whose signature contains all the elements of the underlying DL namespace.

Regarding ontology change, only the CVA can capture changes of the form "add concept A to the ontology" or "remove concept A from the ontology". For OVA, such statements don't make much sense, because A will be a part of the ontology anyway. For CVA, such changes do make sense; they are also possible and can be expressed using the existence assertion operator: to remove a concept A from an ontology, we need to contract with $\{\%A\}$; to add A, we need to revise with $\{\%A\}$. Therefore, the deficiency that motivated the introduction of OVA and CVA can be dealt with in one of the following two ways:

- We choose to use the CVA and enhance the underlying DL with the existence assertion operator; this way, the information normally stored in the signature structure can be fully expressed in the form of DL axioms, so the problematic changes are possible and are cleanly included in our framework. This approach solves the problem by enhancing the underlying language.

- We choose to use the OVA and keep the underlying DL as-is (no existence assertion axioms are allowed); this way the problematic changes are not possible, because they simply make no sense in the underlying language. This solves the problem by making it void, i.e., by making this kind of changes impossible, because their formulation and effects are not expressible in the underlying DL.

Notice that most approaches in the literature use a kind of hybrid approach on the subject. The use of signatures implicitly forces the use of the CVA; yet, the existence assertion operator is not introduced, because the use of the signature structure eliminates the need for this operator. Addition and deletion of namespace elements is allowed (using the signature structure implicitly); however, clean, formal semantics on what such an action means is missing. As is common practice in the relevant research, only informal, intuitive descriptions of the process are given.

The introduction of the existence assertion operator and the use of the logical representation of ontologies, allow the connection of the process with Tarski's model and provide the basis for the definition of its formal semantics. Furthermore, there is no reference to the consequences of CVA in reasoning, assuming standard reasoning. This approach contains the "a posteriori" assumption that the signature structure contains all the elements necessary for the deduction at hand. This fact directly contradicts the initial assumption of closed vocabulary. Our proposition provides the semantics required for a formal treatment of this part of the problem of evolution.

Semantics of Change Phase

The semantics of change phase (phase 3) is probably the most crucial phase of ontology evolution; during that phase the (direct or indirect) changes caused by a given change request are determined. In the following, we will describe a framework that will allow us to use belief change techniques and intuitions to address this problem.

In the rest of this discussion, it is irrelevant whether we use the OVA or the CVA, as long as the assumption chosen has been correctly reflected in the <L,Cn> definition of the representation language (DL). We will not generally force any particular properties on the underlying DL, unless otherwise specified. As described above, ontologies will be expressed as sets of expressions of the underlying language (O⊆L); changes will also be represented in the same way (X⊆L). Again, this section will not provide concrete answers to the problem, but it will provide an introductory framework that will hopefully motivate future work.

Defining an Ontology Evolution Algorithm

Before proceeding, it is useful to provide a formal definition of what constitutes an *ontology evolution algorithm* (or operation). Our approach follows the path taken in [48], which also coincides with the standard approach used in belief change; under this approach, an ontology evolution algorithm is a function, mapping an ontology and a change to an ontology. Therefore, for a given underlying logic <L,Cn>, an ontology evolution algorithm can be formally defined as a function $OE:P(L) \times P(L) \rightarrow P(L)$.

We have defined four different types of evolution operations, namely revision, contraction, update and erasure; therefore, a complete ontology evolution system should support all four, by implementing four ontology evolution functions, one per operation. All four of them should have the same signature $(P(L) \times P(L) \rightarrow P(L))$, but the different semantics of the four operations does not allow one to use the same function for all. An alternative would be to add a certain flag on the change that indicates its semantics, i.e., the type of operation required. This changes the signature of the ontology evolution function (because its domain should include the flag) and it allows the integration of all four functions into one. In this work, we will use the former approach, because it avoids the further definition of a "flag". The operation involved in each case should be clear from the context.

The above definition does not constrain an ontology evolution algorithm in any way; any function with the proper range and domain can be so classified. This means that the definition engulfs algorithms (i.e., functions) which exhibit properties undesirable for the purpose of ontology evolution. In the following, we will address some of the properties which affect the rationality of the result of the ontology evolution algorithm. These issues have already been addressed in the belief change literature, but are now revisited under the prism of ontology evolution. As we will see, in most cases, the considerations that have appeared in the belief change literature are valid in the context of ontology evolution as well.

The Semantics of the Set of Changes

Under the viewpoint described in previous sections, a change request is represented using a set of propositions (i.e., axioms) of the underlying DL. In belief change, the change is usually represented using a single proposition. Our approach is, of course, more general than the standard belief change option; the question is, is this generalization appropriate or necessary

for ontology evolution? We argue that the properties of the representation languages commonly used in ontologies (such as DLs) make such an option necessary.

Most belief change approaches assume that the underlying logic contains the usual operators of classical logic, like $\wedge$, $\vee$, $\neg$, $\rightarrow$ etc and include classical tautological implication. Furthermore, sets of expressions have conjunctive semantics, so the set {A, ¬B} is actually equivalent to {A∧¬B} (but see [73] for an alternative semantics). This way, any finite set can be equivalently represented by a singular set, containing the conjunction of the finite set's propositions. These assumptions are taken for granted in most works of belief change, an indicative list being [1], [13], [23], [46], [82], [90]. Under these assumptions, generalizing the change to be a set of expressions (instead of a single one) is not a real improvement over the standard case, complicating the problem without adding any interesting generality to it. This is true because changes which are expressible *only* via infinite sets of expressions are not interesting in practice (because they cannot be represented in a finite machine), while the changes expressible via finite sets of expressions are also expressible by a single expression.

However, the above assumptions fail for DLs; in many DLs, the conjunction of axioms is not possible, as the axioms have equational semantics (see [16] for details on equational logic). For example, how could you express in a single axiom the set {A⊑B, R⊑S} in the ALH DL, where A, B are concepts and R, S are roles? It follows that, for many DLs, there are facts which are expressible by a set of axioms, yet non-expressible by any single axiom. In this case the proposed generalization does make sense, as the new information (i.e., change) might not be expressible using a single axiom. For this reason, we believe it would be unnecessarily restrictive to constraint the change to be a single axiom only, as this does not take full advantage of the expressive power of the underlying DL.

Assuming that the change is a set of axioms does not address the problem of the semantics of this set of axioms. There are different ways that a set of propositions can be interpreted (e.g., [73]). In this work, we assume the standard semantics for a set, i.e., that a set represents a group of axioms all of which should be satisfied (conjunctive semantics); this is the standard option for sets of axioms in DLs, as well as in most knowledge representation languages.

For belief change (and for ontology evolution), there are even more facets to this problem: a set of expressions may represent an iterated change ([21], [77], [89]); or it may represent an entire new KB, in which case we are dealing with belief merging ([72]); in the case of a contraction operation, a set of expressions may have package or choice semantics ([40]); alternatively, we may adopt the standard semantics, under which a set of expressions represents the conjunction of a group of beliefs that should be used to perform a certain operation (revision, contraction, update or erasure) upon the underlying KB. All these issues are equally applicable for ontology evolution and should be resolved before addressing the problem of determining the effects of the change, because the expected result of a certain change differs, depending on the semantics adopted.

The proper semantics chosen for each change is obviously depending on the application at hand. In current ontology evolution research, only the standard option is considered; the case of merging is dealt with in the fields of ontology merging and integration (under a different viewpoint) and the other options have not been considered. We believe that the standard option is the most suitable as an initial approach to ontology evolution, given the immaturity of the field; however, future research on the subject should address the other options as well, as these alternative semantics are useful in certain cases.

Foundations and Coherence Theories (Belief Bases and Belief Sets)

As was argued in previous sections, there is a fundamental philosophical choice to be made in belief change regarding the representation of the knowledge, i.e., whether the explicitly represented knowledge serves as a justification for our beliefs (a belief base under the foundational semantics) or whether it simply forms a manageable representation of an infinite structure (a belief set under the coherence semantics).

This choice is very important, greatly affecting the ontology evolution (and belief change) algorithms considered. However, it is orthogonal to the representation language used, because it is related to the knowledge level [91]. The choice of the viewpoint to employ depends on philosophical intuition, personal preference and on the intended use (application) of the KB (ontology in our context). Therefore, all the arguments, ideas and results discussed in the belief change literature (see, for example, [42], [52], [53]) are equally applicable here.

In current ontology evolution literature, the foundational theory is usually implicitly assumed for an ontology; no arguments are made in favor of this option, it's just followed without explicit mentioning or justification. In fact, the more practical character of ontology evolution as opposed to general belief change and the need for efficient ontology evolution algorithms favor this option. However, such a choice should not be taken lightly; only future research can verify that this option is indeed more appropriate for the development of rational ontology evolution algorithms. For more details on the pros and cons of each of the alternatives (in the belief change context), refer to [42].

Postulation Methods Versus Explicit Constructions

To the authors' knowledge, all current approaches for ontology evolution use the explicit construction approach, in effect providing specific (manual or semi-automatic) approaches to deal with the problem. This is partly justified by the more practical nature of ontology evolution, as opposed to general belief revision. However, it is generally acceptable that both approaches should be developed in parallel, one benefiting from the results of the other [82]; this is the standard methodology in belief change and most other areas of science, so we can't see why it should not be applicable here as well.

In this respect, we believe there is a certain gap in current research, partly caused by the lack of efficient formalizations of the processes behind ontology evolution and partly by the focus of current research on the technical and practical issues related to ontology evolution. One of the goals of this work is the development of an adequate formalization for ontology evolution, which will hopefully lead to certain postulation approaches. In a following section, we will describe our work on the application of the AGM theory [1] to ontology evolution, which uses the proposed framework to introduce a certain postulation for ontology contraction [35], [36], inspired by the AGM theory of contraction.

Principle of Primacy of New Information

This principle has a special importance in the context of ontology evolution. The distributed nature of the Semantic Web and the many diverse uses of ontologies in agent-based systems or other autonomous environments, does not allow us to accept the new information in an unconditional manner in all applications. In such cases, non-prioritized belief change [50] may prove useful, under which it is possible that part (or all) of the change is rejected (i.e., ignored), because, for example, the source may be untrustworthy. We believe that, in the context of the Semantic Web, the choice of the approach to be taken regarding this issue should be made on a per-application basis.

In current ontology evolution algorithms, the user selects the change to be made, which is then accepted unconditionally by the system. So the principle is generally valid [48]. However, the active participation of the user in the change process allows him to select only a part of the change for application in the ontology. So, we could say that the principle is valid for the system itself, even though it can be implicitly invalidated by the user, if he chooses to modify an external change request before applying it to the system.

Principle of Irrelevance of Syntax

In the ontological context, the validity of this principle determines whether the syntactical formulation of the ontology or the change affects the result of ontology evolution. Normally, semantic, rather than syntactic, considerations should be the driving force behind ontology evolution; in this sense the principle is valid in our context. Notice however that under the foundational viewpoint, which is usually implicitly assumed in ontologies, a stronger version

of this principle may be more appropriate, like in standard belief change. In any case, we believe that the validity of this principle does not depend on the representation language itself, but on the nature of the change and the KB (i.e., the application); it is also affected by our choice regarding the knowledge model to use (coherence or foundational).

Consistency, Inconsistency, Coherence and Incoherence

The issue of ontology consistence (i.e., the Principle of Consistency Maintenance in the terminology of belief change) is very important for ontology evolution in general [48] and, in particular, for the semantics of change phase; according to [49] and [101], the main goal of this phase is to ensure that the result of the change is a consistent ontology. The problem with this statement is that it is not clear what exactly a "consistent ontology" is; the term "consistent" (and "inconsistent") is an overused term which has caused a certain amount of confusion in the literature.

For example, in [9], an ontology is called consistent iff it has at least one model; under the DL terminology, this means that there should be at least one interpretation that satisfies all the (DL) axioms of the ontology. In [80], a consistent ontology is one that satisfies the set of invariants defined in the ontology model and all its used entities are defined. The definition is extended for sets of ontologies which have some kind of dependency and for replicated ontologies; these issues are outside the scope of our discussion. According to [49], consistent is an ontology which conforms to its "consistency model". Under this viewpoint, there are various notions of "consistency", each with its own "consistency model". In [48], a similar use of the term is adopted, by defining three types of consistency, namely structural, logical and user-defined, which are based on certain "consistency conditions", depending on the underlying ontology model. Under [100], "inconsistency" may be semantical or syntactical. In [101], certain conditions that should be satisfied by any consistent ontology are presented.

All these different notions are not compatible with each other and may cause a certain amount of confusion. In this report, we will use two different terms to describe the various understandings of the term consistency (and inconsistency). *An ontology will be called inconsistent iff there is no interpretation satisfying all the DL axioms in the ontology; it will be called incoherent iff it does not satisfy certain predefined constraints or invariants related to efficient ontology design.* These constraints describe the ontology's "consistency model", in the terminology of [49]. Two popular sources of incoherence in this sense is the existence of concepts which are unsatisfiable, or the existence of roles which have no predefined range or domain. Inconsistency, in our sense, corresponds to "logical inconsistency" in the terminology of [48].

Notice that an inconsistent ontology implies a logical contradiction, so any DL formula is inferable by such an ontology. An incoherent ontology implies bad design which could cause problems in the long run. Of course, the exact definition of "bad design" depends on the ontology at hand. We will discuss the differences between inconsistency and incoherence in more detail later in this section.

There are several causes for inconsistencies (and incoherencies) in an ontology; in [62], four different scenarios which may cause such behavior in ontologies in the Semantic Web are identified. It is also argued that, due to the chaotic nature of the Semantic Web, such problems are unavoidable [62]. For this reason, several methods have been developed in the literature in order to deal with inconsistencies (and incoherences).

In [61], [62] for example, the development of a non-standard inference relation is proposed and the desired properties of such a relation are described. This inference relation allows meaningful answers to queries, even when the underlying ontology is inconsistent; thus, such a relation allows us to ignore the issue of inconsistency, or postpone the resolution of an inconsistency to a later time. This is similar to the approaches used in paraconsistent and nonmonotonic reasoning to deal with the explosive nature of classical entailment in the presence of contradictions. In [57], it is proposed that inconsistencies are avoided altogether by using a language (like SHOE) that disallows them.

In [98], ontologies with unsatisfiable concepts are called incoherent and the process of identification and elimination of such concepts is called "debugging". Such ontologies are considered to contain "logical contradictions" and a debugging algorithm for such ontologies is proposed; this algorithm identifies the problematic concepts and provides clever algorithms for tracking down the sources of this problematic behavior. The merging tool Chimaera [84] has certain diagnostic tools allowing one to check the ontology for several types of incoherencies (called inconsistencies in that paper). The same holds for other merging tools as well, like PROMPT [94]. Classical DL reasoners can also be used to detect inconsistencies or incoherencies in an ontology, but provide little support for resolving them [86].

We argue that ontology evolution need only concern with inconsistencies and ignore incoherences. Incoherences are probably signs of bad design but nothing more; tools that identify incoherences and provide methods for resolving them (like [98]) are clearly desirable, but such a process has nothing to do with ontology evolution, being more related to ontology design. Therefore, such modeling problems should not be automatically corrected by an ontology evolution algorithm; instead they should be simply pointed out to allow other tools (or the knowledge engineer) to handle them.

Inconsistent ontologies are undesirable in knowledge representation. An inconsistent ontology implies anything, so it will return "yes" to all queries under a classical reasoner; this "explosive" behavior is problematic [62], regardless of the application at hand, because such an ontology does not really imply anything about the domain. On the other hand, incoherent ontologies are not necessarily undesirable, even though in many cases an incoherent ontology is a potential source of contradiction that would lead to an inconsistent ontology.

We will explain our arguments with certain examples. Consider the well-known penguin example, in which the concept PENGUIN is defined as both a flying and a non-flying animal:

<div align="center">ONTOLOGY: PENGUIN_ONTOLOGY</div>

BIRD⊑FLY   (birds fly)
PENGUIN⊑BIRD   (penguins are birds)
PENGUIN⊑¬FLY   (penguins don't fly)

In several contexts (e.g., [49]) the PENGUIN_ONTOLOGY would be considered inconsistent; under our terminology, this ontology is only incoherent, but not inconsistent. It is incoherent because the concept PENGUIN is necessarily an empty concept under the above conceptualization and empty (i.e., unsatisfiable) concepts are usually undesirable, being excluded from the ontology's "consistency model"; for this reason, ontologies that don't satisfy this requirement are incoherent in most contexts. The ontology is consistent though, because there are interpretations in which all the above DL axioms are satisfied. The problem with this ontology is that, once we add an assertion stating that a certain individual belongs to the concept PENGUIN (for example PENGUIN(PENGO)), the ontology becomes inconsistent; however, no inconsistency arises until such an assertion is made. This is the source of the problem and the reason that unsatisfiable concepts are usually undesirable: an unsatisfiable concept is, in a sense, a potential source for contradiction when we start populating the classes.

However, unsatisfiable concepts are not necessarily problematic. Let us change the above ontology as follows: replace BIRD with HORSE, FLY with ¬HORNED and PENGUIN with UNICORN. The resulting ontology is:

<div align="center">ONTOLOGY: UNICORN_ONTOLOGY</div>

HORSE⊑¬HORNED   (horses don't have horns)
UNICORN⊑HORSE   (unicorns are horses)
UNICORN⊑HORNED   (unicorns have horns)

Performing the same analysis as before, we reach the conclusion that a unicorn both has and does not have horns. This is again an incoherent ontology, but not an inconsistent one. The concept UNICORN is unsatisfiable. However, could one characterize this behavior as "problematic"? Probably not. To the authors' knowledge there are no unicorns; in effect the

concept UNICORN *should* be an empty one. It could even be the intention of the ontology engineer for the concept UNICORN to be unsatisfiable, to denote exactly the fact that unicorns do not exist.

Therefore, the problematic behavior of the PENGUIN_ONTOLOGY stems from our common knowledge that penguins in fact do exist, so an unsatisfiable concept by the name of PENGUIN is problematic; this argumentation collapses in the UNICORN_ONTOLOGY, exactly because our background knowledge tells us that unicorns do not exist. The important point here is that the two ontologies are structurally identical (see also the graph below), so a computer-based system cannot discriminate between the two. Only a human, using a vast amount of background knowledge, can detect the difference between these two cases.
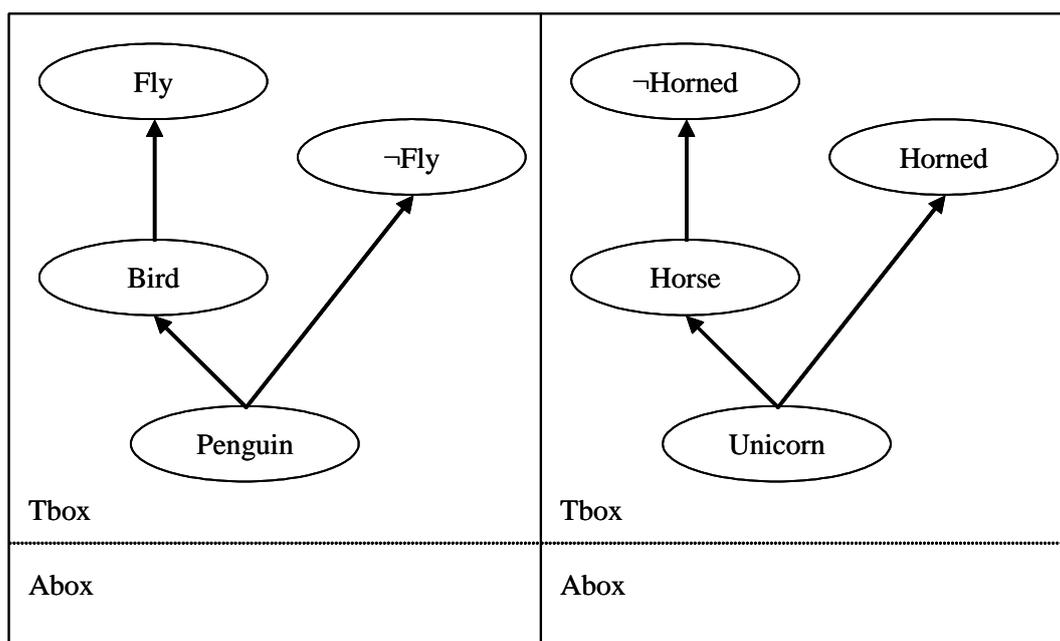
Now consider an ontology with the facts:

BIRD⊑FLY   (birds fly)
PENGUIN⊑BIRD   (penguins are birds)

This ontology is both coherent and consistent. Assume that we feed the above ontology to an ontology evolution algorithm and ask it to revise it with the fact:

PENGUIN⊑¬FLY   (penguins don't fly)

Simply adding the new fact in the ontology would lead to incoherency (but not inconsistency). What should the algorithm do in this case? In many contexts, it is proposed that the algorithm should detect the incoherency and take an action against it, for example removing or weakening the axiom BIRD⊑FLY [62]. But, is that really what we want to do? Should we give up our belief that birds fly because of the fact that such a belief makes the class PENGUIN unsatisfiable? If you feel that the answer is positive, then consider the same revision performed in the unicorn example: should we give up our belief that horses do not have horns, just because someone came up with an imaginary concept called UNICORN? We believe not; at least not until some new knowledge asserts that the concept PENGUIN (or UNICORN) is in fact non-empty, in which case an action against this problematic behavior (inconsistency) is justified. But no such action is justified against an incoherence, because no computer system could tell the difference between a really problematic incoherence (i.e., PENGUIN) and a non-problematic "incoherence" (i.e., UNICORN). Similar arguments can be made for other forms of incoherencies, like the existence of roles with no predefined range

and/or domain [80], classes with only one subclass [100], undefined entities [80] or the explicit storage of redundant statements (i.e., axioms) [48].

The above analysis shows that incoherences are not directly related to the problem of ontology evolution. Spotting unsatisfiable concepts (as well as other types of incoherencies) is undoubtedly an important task, because incoherences usually denote a problematic, or suboptimal ontology design [98]. If an incoherence is identified, it should be pointed out, warning the knowledge engineer of a potential design problem and allowing the continuous refinement of the ontology [100]; for this reason, tools that detect incoherences are highly desirable. However, the development of such tools is outside the scope of ontology evolution research, being more related to the area of ontology design and maintenance.

This leads to another very important difference between incoherent and inconsistent ontologies: inconsistent ontologies are *always* undesirable, so the sources of inconsistency should be tracked down and eliminated during ontology evolution, ontology merging or any other process that created them; incoherent ontologies are *usually* undesirable, so the sources of incoherency should be pointed out for further processing that will determine whether they are indeed undesirable or not; yet, such an action need not be taken immediately, because an incoherent ontology (unlike an inconsistent one) can still provide meaningful answers to a user.

There is yet another type of "inconsistency" that is addressed in the ontology evolution literature: the term "structural consistency" refers to ontologies which are formulated according to their representation language, i.e., using no features, constructions or axioms which are not included in the language [48]. This issue corresponds to the Principle of Adequacy of Representation in the belief change terminology, i.e., to the requirement that the new, evolved ontology (respectively KB) must be formulated using the same representational language as the original ontology (respectively KB).

We argue that this problem has nothing to do with inconsistency or incoherency; an ontology that does not conform to the requirements of the underlying representation language is simply ill-formed. It is the same as using the axiom $\forall A.B \sqsubseteq C$, for A a concept term: such an axiom does not denote an inconsistency; it's just not part of the underlying DL. Regardless of terminology, an ontology evolution algorithm should always make sure that the resulting ontology is properly expressed according to the rules of the underlying representational language. This is a most basic prerequisite for any kind of formal or informal treatment of the problem. This basic requirement has already been incorporated in the definition of an ontology evolution function (i.e., algorithm) and we will always implicitly assume that it holds, without further mentioning.

Principle of Fairness

The principle of fairness is a very important, but underestimated principle. It guarantees the determinism and reproducibility of an ontology evolution (or belief change) algorithm. It is important because non-determinism may lead to complications, such as the inability to predict the result of a change or the inability to "undo" a certain change. Current ontology evolution algorithms, being human-driven, are non-deterministic: one cannot predict the sequence of changes that a user will initiate in response to a certain need (i.e., a complex change), because not all users have the same views on how a certain change should be implemented [100]. On the other hand, the principle of fairness is an inherent property of our definition of an ontology evolution algorithm, because a function is by definition deterministic. This is also one of the advantages of our methodology: automatic ontology evolution guarantees that the result of a change will be reproducible and reversible.

Principle of Minimal Change

As already stated, this is the most important issue to resolve before developing a rational belief change algorithm. Any given change can be resolved in several ways [48] and the system (belief change algorithm) should choose one; this should be the one that minimizes a certain undesired quantity (e.g., information loss), or maximizes a certain desired quantity

(e.g., reliance in the resulting KB). There are several ways to count these quantities, each resulting from a different metric and leading to a different implementation of the principle; such an implementation underlies every work on belief change, determining the algorithm's properties and behavior when faced with a change.

This is also true in the ontology evolution world: according to [48], the result of an ontology evolution operation should be an ontology that satisfies the user's requirement for a change (i.e., the Principle of Primacy of New Information) and it is at the same time a consistent ontology (i.e., satisfies the Principle of Consistency Maintenance); since there are several different ways to resolve a particular inconsistency (just like in belief change), one must choose the one that modifies the original ontology in a minimal manner [48]. This is the Principle of Minimal Impact, in the terminology of [48] (i.e., the Principle of Minimal Change in the belief change terminology). Notice the closeness between the considerations expressed in [48] and the ones that have been expressed in the context of belief change (using a slightly different terminology) in [41]; this is another argument in favor of the connection between the two research areas.

Under our proposition, the basic "block of knowledge" in ontologies is the axioms. Therefore, the loss of information could be counted in terms of the number and importance of the axioms that need to be removed from the ontology in order for the new information to be accommodated. Alternatively, the loss of information could be counted in model-theoretic terms (via some kind of distance metric between the interpretations satisfying the original and the modified ontology) through some specially designed distance metric between ontologies or via certain conditions (postulates) that identify acceptable and non-acceptable modifications. All the above methods have been successfully used in the belief change context (see [1], [20], [43], [46], [54], [68] for some examples).

Current approaches to ontology evolution focus on the structure of the ontology in a graph-based representation, so there is no similar "basic block of knowledge". The concepts, roles and individuals are the building blocks in this case, as well as the connections (relations) between these objects. Since axioms are usually not considered to be parts of an ontology [95], the graphical representation of the ontology could prove useful for determining the distance between the original and the modified ontology in a formal manner. Under this viewpoint, specially designed distances between graphs could prove interesting.

Current ontology evolution tools avoid the difficulty of defining such a distance through the use of semi-automatic evolution methods: they usually provide the knowledge engineer with certain alternatives allowing him to select the one that minimizes the "information loss". Since the selection is made by the user, he is responsible for minimizing the information loss resulting from the change. For example, when a concept is removed, its former instances could be either removed or reclassified; it is up to the user to determine the best alternative, either directly, or indirectly, via some kind of evolution strategy [100]. Of course, there is no guarantee that these are the only alternatives, since there is no formal study showing that. Also, the system should allow hybrid approaches, for example removing some of the instances and reclassifying the others. In any case, there is no formal study (or metric) measuring the information loss of each alternative; it is up to the user to select the less plausible facts for removal from the ontology.

In [48], a method based on the notion of confidence is sketched, but not validated. The authors prefer to use a simpler approach, using the notion of "structural connection" between axioms and counting information loss as being equal to the number of removed axioms. The alternative method (based on confidence) would exploit extra-logical information regarding our confidence on each axiom, allowing a computer system to detect and remove the less plausible axioms automatically. The notion of confidence is similar to the belief change notion of epistemic entrenchment [43]. Weakening a contradictory ontology using any of the above methods allows us to restore its consistency with minimal loss of information. However, the loss of information is counted differently in each case giving different meaning to the term "minimal"; thus, the result of the change would also be different in each case.

Some Simple Examples

In this section we will attempt to sum up the most important features and ideas presented above by showing the practical feasibility of the approach through certain simple, but intuitive examples. Of course, the examples will be addressed in an informal manner, as our proposition does not provide an explicit algorithm for ontology evolution, but a set of general suggestions on how the intuitions and results developed in the belief change literature can be exploited to produce working ontology evolution algorithms. We will use both logical and graphical representations to illustrate these examples.

Let us consider a very simple ontology, asserting that elephants cannot fly:

ONTOLOGY: DUMBO_ONTOLOGY

ELEPHANT$\sqsubseteq\neg$FLY   (elephants don't fly)

Now, reading the story by Helen Aberson Mayer and Harold Perl regarding Dumbo, the little flying elephant, we learn that there is a certain individual, DUMBO, which is an elephant. So we revise our beliefs with the fact:

ELEPHANT(DUMBO)   (DUMBO is an elephant)

This can be easily accomplished by adding the above axiom to our original ontology, because this would result to no contradiction. It is generally acknowledged in the belief change literature that revision should be the same as expansion when no contradiction occurs [1], [41]. For similar reasons, the problem of ontology revision is interesting only when the new information contradicts the old. This case occurs when, during our reading of the tale on Dumbo, we find out that Dumbo can actually fly. Our ontology should be revised with a set containing the following fact:

FLY(DUMBO)   (DUMBO can fly)

Simply adding this fact to our ontology would result to an inconsistency, because a certain individual (DUMBO) would be asserted to be a member of two concepts which are by definition disjoint (FLY and $\neg$FLY). The question is, how do you deal with this problem? In other words, how do you revise the ontology:

{ELEPHANT$\sqsubseteq\neg$FLY, ELEPHANT(DUMBO)}

in the face of the new data:

{FLY(DUMBO)}

One possible approach is based on the fact that the above DL axioms can be translated to FOL [7]; following this translation, we could use one of the numerous belief change algorithms that deal with revision in classical logic and then translate the result back to a DL axiom. This approach is rather naïve, for two reasons; first, even though the majority of DLs are fragments of FOL, there are several ones which go beyond FOL expressiveness [14]; second, even if the underlying DL is a fragment of FOL, there is no guarantee that the result of the revision will be expressible in this DL. If the result of the applied algorithm is something not expressible in the underlying DL, we would violate one of the most important principles of ontology evolution (and belief change), namely the Principle of Adequacy of Representation. For these reasons, we would like to use an algorithm that will be applicable to DL axioms directly, without the need for any intermediate translation.

As already mentioned, we would like the system to be able to decide, given the new information that DUMBO can fly, how to accommodate this fact, autonomously, without human intervention. We believe that this is a plausible goal, because in standard belief change the problem has been successfully addressed without human intervention. The additional difficulty in this approach (with respect to standard approaches to ontology evolution) is that the new data simply informs us that DUMBO can fly, but this gives us no clue as to how to include this information to our ontology. In standard ontology evolution approaches the change includes information that determines how the change should be resolved because the changes represent operations such as "Remove_Concept", "Add_IsA", "Move_SubTree" etc; this greatly simplifies the problem, but relies on human intervention to determine the type and order of these operations in response to the new data.

Following the general principles of belief revision, the proper approach towards implementing this revision is to find a weakening of the original ontology:

$$\{\text{ELEPHANT} \sqsubseteq \neg\text{FLY}, \text{ELEPHANT}(\text{DUMBO})\}$$

such that the addition of the revising expression to this weakened ontology will not lead to an inconsistency. This weakening should be "minimal" in the sense of the Principle of Minimal Change; what remains to be fixed is the definition of the term "minimal".

A human agent could in this case determine that, since the information came from a fairy tale, we should not treat it as a standard, but as an exception. Thus, we have no reason to eliminate our belief that elephants don't fly; instead, we should add an exception to this belief stating that all elephants but DUMBO cannot fly. This conclusion is based on common sense (e.g., fairy tales are not reliable) and on extra-logical information (e.g., reliability of each of our beliefs). An alternative approach would be to weaken (or remove) the fact that DUMBO is an elephant, because the resulting contradiction caused us to simply lose our faith in what we read in the fairy tale.

In any case, this is something that the system should be able to decide upon. To allow this, we could for example equip our axioms with a certain entrenchment [43] or confidence [48] ordering, which would determine the less reliable facts of the ontology. This option is not as simple as it may seem, because this ordering should be also extended to implicit axioms (i.e., consequences of the ontology) in a proper way (see [2], [43] for details). Alternatively, we could consider all axioms equally reliable (using no additional information) and use a certain algorithm that weakens the whole ontology uniformly, until the desired level of weakening is reached; examples of belief change algorithms that implement this idea are: [1], [20], [46]. Of course, the above works are not directly applicable to DLs, as the underlying assumptions are generally different; however, the intuitions behind the algorithms are valid in this context as well.

Let us assume that the system chooses to weaken the axiom ELEPHANT $\sqsubseteq \neg$FLY. One extreme approach would be to eliminate the axiom altogether; we consider this a special case of weakening, in which the axiom is replaced by the empty set. However, as explained above, a better option would be to treat DUMBO as an exception. Under this viewpoint, we would like to replace the above axiom with the axiom: ELEPHANT $\sqcap \neg\{$DUMBO$\} \sqsubseteq \neg$FLY. This would result to the following (revised) ontology:

ONTOLOGY: DUMBO_ONTOLOGY_REVISED

ELEPHANT $\sqcap \neg\{$DUMBO$\} \sqsubseteq \neg$FLY	(all elephants, but DUMBO, don't fly)
ELEPHANT(DUMBO)	(DUMBO is an elephant)
FLY(DUMBO)	(DUMBO can fly)

This is a consistent ontology (Principle of Consistency Maintenance); the revision has resulted in minimal loss of information (Principle of Minimal Change) and has accommodated the revision (Principle of Primacy of New Information). We see that this result is therefore compatible with the most important general principles of the dynamics of change, as studied in belief change. There is one question that remains unanswered though: how could the system select the proper weakening of the ontology? Assuming that the axiom(s) to be weakened have been selected using one of the above methods, how would the system decide on the exact weakened version of those axioms?

This task could be accomplished either in model-theoretic terms, or using a certain syntactic or semantic scheme that iteratively weakens the selected axiom(s) until the desired result has been reached. Both approaches have been used in the belief change literature (e.g., [1], [10], [19], [68], [82], [105]); the problem has been addressed in the DL context as well (in [48]); the scheme used for reasoning under inconsistency in [61], [62] could also be adapted to address the same problem.

In the following figure we will show how this revision can be performed using current ontology evolution systems. Notice the number of operations upon the ontology graph that should be initiated in order to perform this (seemingly simple) change in the ontology: first (phase 1), we should introduce three new concepts ({DUMBO}, $\neg\{$DUMBO$\}$,

ELEPHANT⊓¬{DUMBO}); following that, we should add the obvious relationships between these concepts (phase 2), by asserting that the individual DUMBO is an instance of {DUMBO} and connecting the concept ELEPHANT⊓¬{DUMBO} via IsA relationships to its immediate parents (ELEPHANT and ¬{DUMBO}); then, we should add two more obvious IsA relationships between {DUMBO} and ELEPHANT, as well as between {DUMBO} and FLY (phase 3); next, the IsA between ELEPHANT and ¬FLY should be removed and replaced by the weaker IsA between ELEPHANT⊓¬{DUMBO} and ¬FLY (phase 4); finally, we should remove the redundant membership assertions between DUMBO and ELEPHANT and between DUMBO and FLY (phase 5), as these are direct consequences of other relationships in the ontology (eliminating redundant relationships is considered necessary in several ontology evolution works, like [48], [100]). All the above modifications are shown, step-by-step, in the figure in the next page.

Notice that in certain contexts, the existence of a single subconcept of a given concept (as is the case with ¬FLY and ¬{DUMBO} in our example) is considered suboptimal behavior [100], [101], which should initiate further changes and refinements in the concept hierarchy. We don't address this issue here because we believe that this problem could be considered a mild type of incoherency; as argued before, dealing with incoherencies is part of ontology design (not ontology evolution), thus outside the scope of this work.

The above considerations are equally true in the ontology merging context. Indeed, consider the following ontologies, which are homogeneous and they use the same terminology and naming conventions, the same underlying DL language etc:

<div align="center">ONTOLOGY: TWEETY_ONTOLOGY1</div>

BIRD⊑FEATHERS	(all birds have feathers)
FEATHERS(TWEETY)	(TWEETY has feathers)
BIRD(TWEETY)	(TWEETY is a bird)
¬FLY(TWEETY)	(TWEETY can't fly)

<div align="center">ONTOLOGY: TWEETY_ONTOLOGY2</div>

BIRD⊑FLY	(birds fly)
BIRD(TWEETY)	(TWEETY is a bird)
FEATHERS(TWEETY)	(TWEETY has feathers)

Both ontologies are consistent (and coherent); however, if one tries to merge them by simply taking the union of the axioms in each, the result would be an inconsistent ontology, because the individual TWEETY would be asserted to be both a flying and a non-flying individual. To address this problem, one could choose to weaken one or both ontologies before taking the union; the dynamics of this weakening are generally similar to the standard case (revision). However, in the context of merging, it makes sense to consider the axioms that appear in both ontologies as more reliable [74]; also, the Principle of Primacy of New Information is not applicable here. In short, the general considerations (regarding ontology evolution) that appeared in previous sections make sense and can be applied in (some part of) ontology merging, with the necessary adaptations to address the slightly different nature of the problem. Notice however that, in ontology merging, resolving any possible heterogeneities between the source ontologies is an important issue, whose resolution should precede any attempt of weakening the ontologies. Therefore, the general principles of our work are applicable in the part of merging that resolves the resulting inconsistencies.
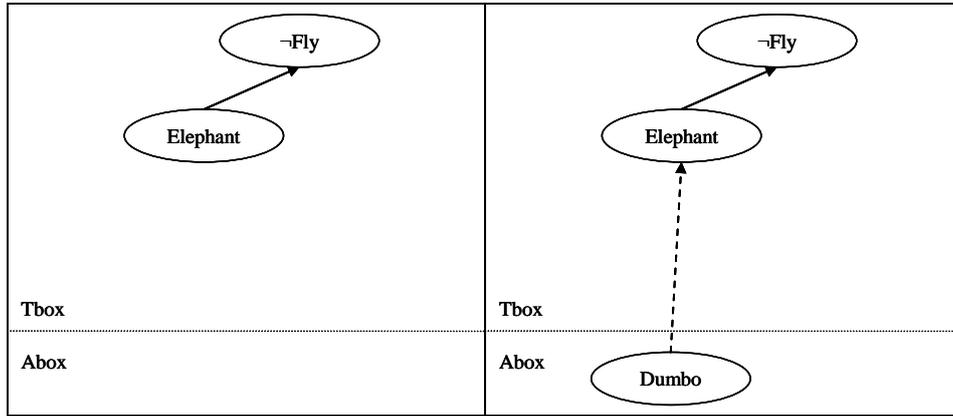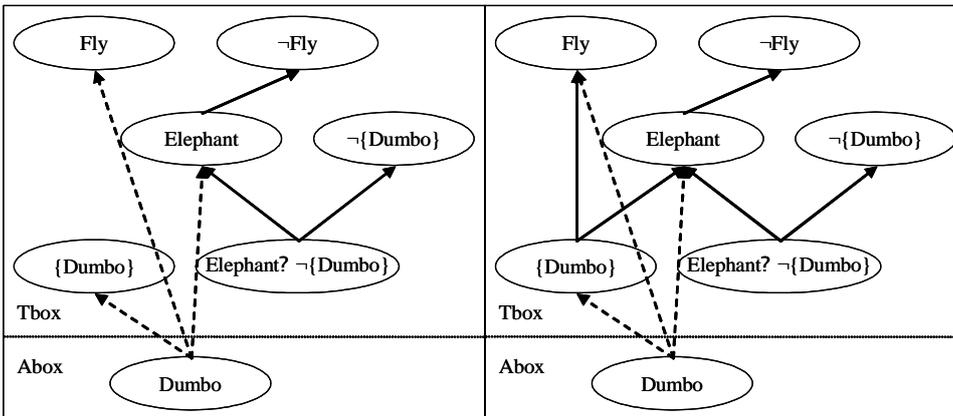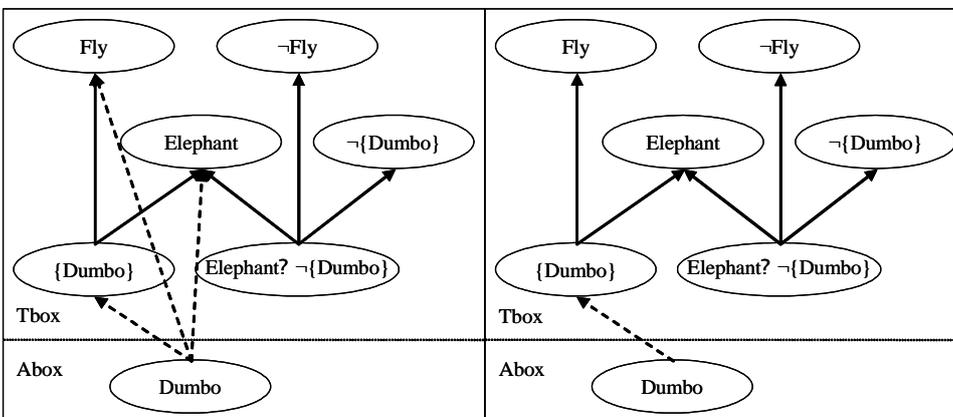
Original Ontology

First Revision
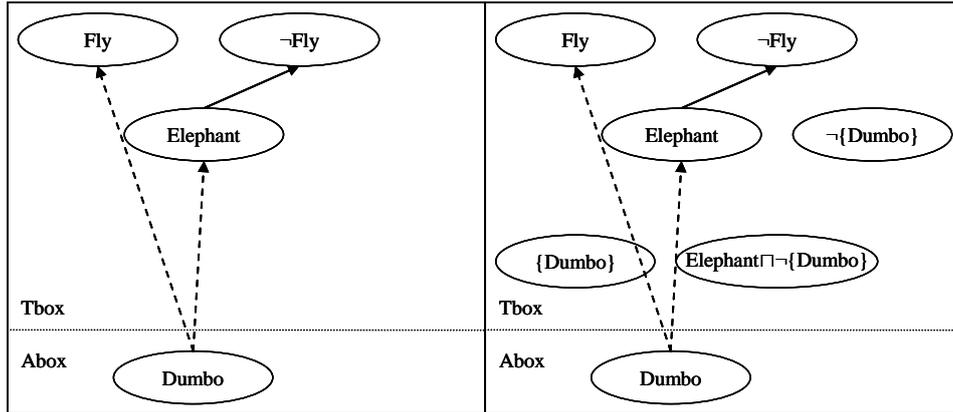
Second Revision: Phase 0 (inconsistent)

Second Revision: Phase 1

Second Revision: Phase 2

Second Revision: Phase 3

Second Revision: Phase 4

Second Revision: Phase 5 (final state)

Discussion and Comparison of the Approaches

We believe that our approach will be helpful in resolving some of the deficiencies of the proposed ontology evolution methods. As already mentioned, this approach is complementary (and not rivalrous) to the current state-of-the-art techniques, allowing them to resolve certain issues that are currently improperly handled, the most important one being the manual or semi-automatic way of handling the changes.

In current ontology evolution, ontology change is considered a one-off process; as such, its manual handling is feasible (even though not practical), as the ontology engineer can take the time required to perform the needed changes. Our method is addressed to applications in which changes occur often, so the ontology engineer cannot handle the burden of updating the ontology manually; it is also applicable in cases where it is difficult, impossible or undesirable for an ontology engineer to handle the change himself (for example in autonomous robots or software agents, in large ontologies developed by a group of engineers or in time-critical applications). In such cases, user-driven ontology evolution is not applicable, so a framework allowing automatic ontology evolution is highly desirable.

Apart from applicability, we consider our method superior because it provides clean semantics, allowing a formal treatment to the problem of ontology evolution; this kind of formality is generally missing in standard approaches. It allows us to use the postulation approach to address the general problems of ontology evolution under a theoretical viewpoint (also missing in the current state-of-the-art). In this respect, the issues addressed in the belief change literature may prove extremely useful towards the formalization and postulation of the process of ontology evolution; several examples of this pattern have been mentioned in previous sections.

The formality of our approach also allows us to abstract from the knowledge representation formalism used to represent the ontology, using only the properties necessary to establish our desired results; on the other hand, existing ontology evolution methods heavily depend on the exact properties of the underlying ontology language [49], so they are not easily transferable to other formalisms. Finally, this formality led to the development of a novel formal distinction between the CVA and OVA, a distinction that has been neglected in the relevant literature, as the issues involved in this distinction are dealt with implicitly and in a rather blurred and inconsistent manner.

Standard methods deal with each different atomic or composite operation straightforwardly. There is a certain matching, assigning each type of change (i.e., operation) with a certain function that describes the actions taken to address the given change. The price to pay for this simplicity is scalability: one new process (function) must be defined for each new type of operation devised. Unfortunately, there is a great (possibly infinite) number of different operations to consider [71], [102], so the lack of scalability is a real bottleneck for future research. On the contrary, under our viewpoint, there are only four different operations to consider (namely, revision, contraction, update and erasure); these engulf all the types of operations that one may wish to perform upon an ontology, even though there is no exact matching between these four operations and the (atomic or composite) operations used in the standard approach.

The reason for the lack of a mapping between these two sets of operations lies on a fundamental difference underlying their definition: the two approaches reflect a different viewpoint on how the changes should be interpreted and handled. Our approach is "observation-centered": a new observation reflects a certain need for ontology evolution. The ontology engineer (or some automatic sensor or similar device) should identify the type of observation i.e., whether it changed the real world or not and whether it added knowledge or added uncertainty by casting doubt on some existing knowledge (in which case we are dealing with removal of knowledge). Addressing these two issues is enough to determine the type of operation involved; the exact operand of the change (i.e., the set of propositions representing the change) should also be determined, manually or automatically. Using these two facts, the

change is then fed into the system which should identify the actual modifications to perform upon the ontology to address the change (observation) and perform these changes automatically.

The standard approach is "change-centered": we are not interested in the observation itself that initiated the change, rather we are interested on the actual changes that should be physically performed upon the ontology in response to this observation. In this case, the ontology engineer identifies the observation and decides (manually or with the aid of the system) on the actual changes that should be performed upon the ontology; these are ultimately fed to the system for implementation. This gives more control to the ontology engineer, making the process more straightforward, but it also adds a significant overhead to the process of ontology evolution, making it more time-consuming and error-prone in several cases [101].

This analysis shows that our approach adds an extra layer of abstraction to ontology evolution: the changes to be performed upon the ontology are decided by the system, not by the ontology engineer. This allows the ontology engineer to deal with high-level observations only, leaving the low-level modifications that should be performed upon the ontology in response to these observations to be determined and handled by the system.

Another important innovation associated with our approach is that it focuses on axioms, rather than on the structural connection between concepts, roles and individuals in a representation graph, as is usually the case in the standard approaches. This axiom-based viewpoint allows us to use the full power of the underlying logical formalism; more importantly, it provides cleaner and more formal semantics, compared to the standard approaches, which build on frame-like or object models [48].

The above observations show that our approach provides the ontology evolution research field with tools, ideas and features that are currently unavailable. This method, coupled with the exciting features provided by current ontology evolution tools, such as undo/redo capabilities, visual representation, collaborative editing, atomicity and transactional nature of changes, change propagation, verification features, inconsistency and incoherency detection tools, ontology repair capabilities etc, will provide interesting solutions to problems currently faced by ontology-based applications when modifications in the underlying ontology are required.

Specific Applications of Belief Change to Ontology Evolution

There are certain works that apply belief change techniques to ontology evolution, such as [66], [76], [86]; these works were developed independently, but they closely follow our line of thought as described in the previous sections. We will briefly present these approaches, as they are exemplary of the research direction we are proposing regarding ontology evolution. They can be viewed as applications of our viewpoint for certain belief change techniques.

One such work, at a preliminary stage, is [66], where the authors propose the use of the AGM theory [1] for ontology evolution. Only some informal ideas regarding the connection of the AGM theory with ontology evolution are provided. The focus lies on the operations of contraction and revision and, following the lead of the AGM postulates [1], certain properties that should hold in a rational contraction and revision operation are presented. However, these properties are not directly applicable to many DLs, for the same reasons that the AGM theory is not directly applicable to such DLs (see [35]), so this is a problem that remains to be solved by the authors. We will discuss this problem in more detail at a later point.

In [76], a similar path is taken. The focus is on a certain preference ordering that is inspired by previous work on epistemic entrenchment [43] in the belief change literature. The interrelationship between the Abox and the Tbox of the DL KB in the context of ontology evolution is studied and the distinction between the two parts of the KB is taken seriously. Only revision is considered, studying how new knowledge can be incorporated in a DL KB without introducing inconsistencies. Unlike our approach, the new knowledge (change) is a single DL axiom. The DL considered is ALU. The effects of this work on ontology merging

are briefly discussed, as well as the differences of this problem with classical belief merging [72], [74]. The authors also study ontology revision in a sense slightly different than the one used in our work: under their understanding, the term ontology revision refers to the addition of an Abox assertion to the DL KB.

An interesting extension of a belief change algorithm appears in [86], where the authors attempt to recast the maxi-adjustment algorithm [10], originally introduced for propositional knowledge integration, to the context of DLs. This algorithm allows the elimination of any inconsistencies that could arise in a stratified KB after its expansion with a new proposition, which, in turn, allows the development of a revision algorithm. The authors propose a basic and a refined algorithm which apply the ideas of maxi-adjustment to DLs. Unfortunately, for their method to be applicable, the disjunction of DL axioms needs to be definable, which is not possible in standard DLs. For this reason, the authors provide the necessary definitions and semantics for the introduction of disjunctive DLs, a certain DL extension for which axiom disjunction is allowed. This, however, constitutes a departure from the classical DL model, limiting the applicability of this approach.

In the rest of this section, we will describe a certain approach that we developed which follows the ideas and intuitions described in previous sections; this approach has been presented in [33], [34], [35], [36], [37], [38] and can be seen as an application of the above framework. The method is based on one of the most important works on belief change, namely the AGM theory [1], and studies the feasibility of its application to DLs and OWL, two families of languages commonly used for the representation of ontologies. The problems encountered during this work are exemplary of the complications that arise during the migration of belief change techniques to ontology evolution.

In these works, only the problem of ontology contraction is addressed; ontology revision is also handled [36], but research on revision is currently at a preliminary stage. Inspired by the notion of rational contractions as was formally defined by Alchourron, Gärdenfors and Makinson (AGM for short) in their seminal paper ([1]), we attempted to apply the same notion to DLs and OWL and determined whether it is possible to recast the original formulation of the AGM postulates so as to be applicable to the above logics.

AGM Theory

The AGM theory [1] is undoubtedly the most influential approach in the field of belief change. The authors of [1] attempted to formalize the field of belief change by addressing the general problem of finding the properties that a rational belief change operator should satisfy. In effect, they used the postulation method, as opposed to an explicit construction which was the general trend at the time. They dealt with two important operations (revision, contraction) and one trivial one (expansion), providing one set of postulates (the AGM postulates) for each of revision and contraction.

AGM made several, quite general assumptions when formulating their theory. They followed Tarski's viewpoint on the definition of a logic, so a logic, in the AGM theory, is a pair $<L,Cn>$. This assumption provides the link between their theory and the world of ontologies, as ontological representation languages (like DLs and OWL) can also be viewed as $<L,Cn>$ pairs, as we explained before. However, AGM made additional assumptions as well: they assumed that the underlying logic is closed under the standard operators of PL ($\neg$, $\wedge$, $\vee$, $\rightarrow$ etc) and that the consequence operator includes classical tautological implication, is compact and satisfies the rule of introduction of disjunctions in the premises (i.e., $Cn(X\cup(Cn(Y)\cap Cn(Z)))=Cn(X\cup Y)\cap Cn(X\cup Z)$, for all X, Y, Z$\subseteq$L).

Regarding the operation of contraction, AGM assumed that a KB is a set of propositions of the underlying logic (say K$\subseteq$L) which is closed under logical consequence (i.e., K=Cn(K)), also called a *theory*. This means that AGM use belief sets, i.e., they adopt the coherence model. The KB can be contracted or revised with any single expression x$\in$L of the logic. Thus, the operations of contraction and revision can be formalized as functions mapping the pair (K, x) to a new KB K′ (denoted by K′=K−x, K′=K+x respectively). Notice

that this formalization is very similar to our formalization for an ontology evolution operator. The above assumptions allow any binary operator to be a "contraction" or "revision" operator, which, of course, should not be the case; for this reason, AGM introduced several restrictions (in the form of rationality postulates) on the result of such operators.

Regarding contraction, AGM restricted the result to be a theory itself (see postulate (K−1) below). As already stated, contraction is an operation that is used to remove knowledge from a KB; thus, the result should not contain any new, previously unknown, information (K−2). Moreover, contraction is supposed to return a new KB such that the contracted expression is no longer believed or implied (K−4). Finally, the result should be syntax-independent (K−5) and should remove as little information from the KB as possible, in accordance with the Principle of Minimal Change (K−6); such a removal of information should be made only when necessary (K−3). The above intuitions were formalized in the *basic AGM postulates for contraction*, which are the following:

(K−1)   K−x is a theory   (*closure*)

(K−2)   K−x$\subseteq$K   (*inclusion*)

(K−3)   If x$\notin$Cn(K), then K−x=K   (*vacuity*)

(K−4)   If x$\notin$Cn($\varnothing$), then x$\notin$Cn(K−x)   (*success*)

(K−5)   If Cn(x)=Cn(y), then K−x=K−y   (*preservation*)

(K−6)   K$\subseteq$Cn((K−x)$\cup${x})   (*recovery*)

Likewise, for revision, the result should be a theory (K+1); the new information should be included in the result, because we are dealing with revision (K+2). The result should be consistent (K+4) and should be independent of the exact syntactical formulation of the revision (K+5). If the new information contradicts the KB, then some information should be removed to avoid contradictions; but information should be removed only when necessary, and only as little as possible to guarantee the consistency of the result (see (K+3) and (K+6) below). The *basic AGM postulates for revision* are the following:

(K+1)   K+x is a theory

(K+2)   x$\in$K+x

(K+3)   If $\neg$x$\notin$Cn(K), then K+x=Cn(K$\cup${x})

(K+4)   If $\neg$x$\notin$Cn($\varnothing$), then K+x is consistent

(K+5)   If Cn(x)=Cn(y), then K+x=K+y

(K+6)   (K+x)$\cap$K = K−($\neg$x)

In addition to the above basic postulates, AGM proposed two, quite powerful, *supplementary* postulates [AGM85] for each operation, which concern composite belief change. The supplementary postulates for contraction are:

(K−7)   (K−x)$\cap$(K−y)$\subseteq$K−(x$\wedge$y)

(K−8)   K−(x$\wedge$y)$\subseteq$K−x, provided that x$\notin$K−(x$\wedge$y)

The supplementary postulates for revision are:

(K+7)   K+(x$\wedge$y)$\subseteq$Cn((K+x)$\cup$y)

(K+8)   Cn((K+x)$\cup${y})$\subseteq$K+(x$\wedge$y), provided that $\neg$y$\notin$K+x

These postulates formally express the AGM viewpoint on some of the philosophical debates that we described in previous sections, in effect describing the AGM notion of "rationality" for a contraction and revision operator. It must be emphasized that there is a whole class of operators that satisfies the AGM postulates. In fact, this is the role of postulation [41]: further refinement of such operators should be the subject of explicit constructions and more applied research. The AGM postulates determined a class of rational operators; the issue of selecting one of them for use in a particular application is greatly depending on the application itself, among other things.

The AGM model is still one of the dominating models in belief change research. The key reason for its importance lies in the wide acceptance that it received by fellow

researchers. The only postulate that has been seriously debated is the contraction postulate of recovery (K−6), as some works [39], [54] state that (K−6) is counter-intuitive, while others state that it forces a contraction operator to remove too little information from the KB [51]. However, it is generally acceptable that the recovery postulate cannot be dropped unless replaced by some other constraint that would somehow express the Principle of Minimal Change. For a thorough discussion on this issue refer to [83].

Despite these objections, the postulates greatly influenced future postulation attempts as well as explicit constructions. It would not be an overstatement to say that this work set the foundations for future research on the field of belief change providing a theoretical basis upon which deeper results could be developed. Most subsequent works praised, criticized, commented, used, extended or provided alternative characterizations of the AGM model making it the most influential approach on belief change.

The line of research that followed the publication of the AGM theory gave several interesting results. For example, several intuitively appealing explicit constructions for families of contraction or revision operators (like partial meet selection functions [1], epistemic entrenchment [43], safe contraction [2] and systems of spheres [46]) have been shown to produce exactly the class of operators that satisfies the AGM postulates for contraction and revision. The fact that these intuitively appealing constructions produce exactly the same operators as the AGM postulation approach forms an additional argument in favor of the importance of the AGM postulates as a determinant for rational revision and contraction operators.

Other results related to the AGM theory deal with the connection of the AGM postulates with other conditions or informal notions that have appeared in the literature; this study was made by AGM themselves in [1], [41], [82]. Alternative formulations of the postulates in the propositional setting were proposed in [68]. All these results form another argument in favor of the AGM model.

A detailed account of such results is outside the scope of this work; however, we will describe one particularly interesting result, which is related to the connection between the operations of revision and contraction. This connection is described by the Levi and Harper identities [41]. The Levi identity has the following formulation:

$$K+x=Cn((K−\neg x)\cup\{x\})$$

The intuitive idea behind this identity is that, in order to revise a KB K with x, one needs to remove any possible source of contradiction (by contracting with $\neg x$) and then expand the result with x. This viewpoint makes revision a two-stage process: initially everything that implies $\neg x$ is removed from the KB; then the KB is expanded with x. The first step guarantees that no inconsistencies will arise during the second step. This identity allows us to produce a revision operator from a given contraction operator.

The Harper identity is actually equivalent to the AGM postulate (K+6). Its formulation is given from (K+6) by replacing x with $\neg x$ and vice-versa:

$$K−x=(K+\neg x)\cap K$$

The intuitive idea is the following: if we revise a KB K with $\neg x$, the revision operator will make sure that $\neg x$ is a consequence of the new KB; since the new KB is consistent it cannot imply both x and $\neg x$ at the same time. Therefore, the revision by $\neg x$ will, as a side-effect, eliminate from the original KB any occurrences of x, as well as any propositions that imply x. If we keep from $K+\neg x$ the part that originally belonged to K, we will get a part of K that does not imply x. This process allows us to get a contraction operator from a revision operator.

These identities show that the two operations are interdefinable. Their importance stems from their relation with the AGM theory: if a contraction operator satisfies the basic postulates for contraction, then the revision operator that the Levi identity generates satisfies the basic postulates for revision. The dual result holds for the Harper identity as well and similar results can be shown for the supplementary postulates. Therefore, the problem of defining a rational (in the AGM sense) revision operator is actually a different facet of the problem of defining a rational contraction operator.

The Generalization of the AGM Theory

It is easy to notice that the intuitions behind the development of the AGM theory are independent of the actual language used to represent the knowledge. This supports our belief that the concept of rationality of a belief change operator is independent of the underlying knowledge representation scheme. Since the AGM theory formally encodes common intuition about the properties that a rational belief change operator should satisfy, we believe that it would be of interest to apply the AGM model in any given logic in order to determine the rationality of a given revision or contraction operator.

The key problem with this approach is that the formulation of the AGM postulates themselves uses the assumptions made by AGM in the formulation of their theory. Thus, the AGM postulates do depend on the underlying language, placing the work mainly in the context of PL and FOL and disallowing its direct application to several languages, including DLs which are the focus of our work. Indeed, a DL is not necessarily closed under the usual operators ($\neg, \wedge, \vee, \rightarrow$, etc), because DL axioms are of equational form (e.g., A⊓B⊑C), so, for example, the negation of a DL axiom cannot be defined in general. The same holds for OWL, as well as for many other families of knowledge representation languages [33].

This issue is reminiscent of the connection between the problem of belief change with ontology evolution: the differences are not on the underlying intuitions, but on the representation languages and formalisms used. For the AGM theory, since the problem actually lies on the formulation of this theory instead of its underlying intuition, we feel that it makes sense to recast it in a setting general enough to contain DLs.

This work focuses on the AGM theory of contraction. Contraction was chosen for our initial approach because, according to AGM, it is the most fundamental among the three belief change operators [1], [41]. The theoretical importance of contraction has also been accepted by most researchers, even though revision is more often used in practical applications. Our research on the generalization of the revision postulates is currently at a preliminary stage.

Our approach constitutes a generalization because it only makes one single assumption: that the underlying logic is described by an <L,Cn> pair, in Tarski's sense [35]. As explained before, this general framework engulfs DLs (and OWL), as well as most of the languages that have been used for knowledge representation (a notable exception being nonmonotonic logics [4]). It also engulfs all the logics that have been considered in the original AGM framework.

Following that, the definition of the contraction (and revision) operator was extended to include cases where the contracted expression i`s a set of propositions instead of a single one [35]. This set should have choice semantics [40]. The motivation for this generalization has been explained in previous sections. We also made another generalization, allowing the KB to be any set of propositions (not necessarily a theory). This generalization is purely technical; we still adopt AGM's coherence model and we consider K to be a belief set, but for the definition of the contraction (and revision) function, the input can also be a set which is non-closed under logical consequence. If one feels uncomfortable with the notion of a KB being a non-closed set, he could consider that the KB is closed under logical consequence as a pre-processing step before contraction (or revision).

Having set our general framework, the next step of our generalization was to reformulate the basic AGM postulates of contraction in such a way as to be applicable to all logics in our more general framework, while preserving the original intuition that led to each postulate's definition. The resulting postulates can be found in the following list, where the naming and numbering of each postulate corresponds to the original AGM naming and numbering [35]:

(K−1) Cn(K−X)=K−X   (*closure*)

(K−2) K−X⊆Cn(K)   (*inclusion*)

(K−3) If X⊄Cn(K), then K−X=Cn(K)   (*vacuity*)

(K−4) If X⊄Cn(∅), then X⊄Cn(K−X)   (*success*)

(K−5)   If Cn(X)=Cn(Y), then K−X=K−Y   (*preservation*)

(K−6)   K⊆Cn((K−X)∪X)   (*recovery*)

It is easy to show that the above postulates coincide with the original ones in the presence of the AGM assumptions. Unfortunately, a major problem appeared soon after the reformulation of the postulates: not all logics in our wide framework can admit a contraction operator that satisfies the (generalized) AGM postulates. On the other hand, there is always an operator satisfying (K−1)-(K−5) [35]; this implies that it is the debated recovery postulate (K−6) that causes this problem.

Following this observation, the focus shifted on defining the conditions under which a logic admits a contraction operator satisfying all 6 postulates; such logics were called *AGM-compliant* logics. Three different necessary and sufficient conditions for a logic to be AGM-compliant were formulated, based on the notions of *decomposability*, *cuts* and *max-cuts* (see [35] for details).

These initial results had several interesting side-effects. Firstly, they were formulated in a very general fashion, allowing them to be applied to any logic that follows Tarski's model (not just DLs). Furthermore, this class of logics was shown to have a strong connection with lattice theory [45]. In [33], an equivalence relation was defined between logics. This relation has two important properties: first, the class of complete lattices, modulo the standard equivalence relation on lattices (see [45]) is isomorphic to the class of logics under Tarski's model, modulo this equivalence relation; second, equivalent logics have the same status as far as AGM-compliance is concerned (see [33]). The above results combined show that, as far as AGM-compliance is concerned, any logic can be equivalently represented as a complete lattice and vice-versa. In other words, AGM-compliance is a property exclusively related to the structure of the (complete) lattice that represents the logic. This allows us to use the richness of lattice-related results to our study of AGM-compliant logics.

Following that, we attempted to apply the AGM theory in the context of a foundational theory. It is a well-known result from the literature that the AGM postulates are not applicable to belief bases under the foundational model [39], mainly because of the recovery postulate (again). However, this is true under the AGM assumptions, so we chose to determine whether this fact remains true under our more general assumptions. The AGM postulates were once again reformulated to capture the intuition behind the foundational model. Using an approach similar to the one used for AGM-compliance, we determined two necessary and sufficient conditions for a logic to be *base-AGM-compliant*, based on the notions of *base decomposability* and *base cuts*. These are very strong properties satisfied only by certain logics outside the AGM framework [33]; for more details on the migration of the AGM theory to the foundational model see [33], [34], [35].

Given the theoretical foundations set by this work, we tried to determine whether the AGM theory is applicable to DLs; in other words, we tried to identify the DLs which are AGM-compliant. This way, we can identify the ontology representation languages in which the AGM theory of contraction can be applied. Initially, it was shown that all but some trivial DLs under the CVA model are non-AGM-compliant [35]. For this reason, the CVA model was abandoned for the purposes of AGM-compliance and the focus shifted on the OVA model, under which things are more promising.

Indeed, in [36], [37] it was shown that if a DL allows us to define a certain transformation, then this DL is AGM-compliant (see [37] for details). This transformation is definable under very generic conditions and its existence depends on the operators, connectives and axioms allowed by the logic. Certain negative results were also presented, showing that some logics commonly used in ontologies, including OWL DL and OWL Lite, are non-AGM-compliant. A detailed list of such logics can be found in [36].

Finally, this study uncovered certain heuristics regarding the AGM-compliance of DLs; for example, it seems that the existence of role operators *usually* guarantees AGM-compliance, while the absence of such operators in the presence of role hierarchies *usually* results to non-AGM-compliance. Also, role operators sometimes allow the definition of an

axiom's negation in a DL; for AGM-compliant DLs with negation, it is possible to define a revision operator via a contraction operator, using the Levi identity. A more detailed account of such facts can be found in [36].

Evaluation of AGM-compliance for Ontology Evolution

There is still a long way to go before fully determining the connection between the AGM theory and DLs. It should be emphasized that AGM-compliance is a property that simply guarantees the existence of a contraction operator that satisfies the (generalized) AGM postulates in the DL under question [35]. The extent to which the richness of results related to the AGM theory can be applied to AGM-compliant DLs still remains to be determined. Furthermore, the connection of AGM-compliance with the operation of revision, the Levi identity or the AGM-related representation results [41] is still unexplored.

On the other hand, our approach indicates that important theories from the belief change literature can be migrated, at least partially, to the world of ontologies. Therefore, it's not only the intuitions of the belief change research that can be used in our quest for ontology evolution algorithms; certain theories themselves could also prove helpful. Moreover, the application of the AGM theory in this context showed that our ontological framework is suitable not only for capturing the peculiarities of the ontology representation languages and the needs of the related applications, but also for allowing the application of belief change theories to the problem of ontology evolution.

Conclusion and Future Work

Ontologies have recently found several applications in the Semantic Web, as well as in other areas of AI. Unfortunately, the majority of research studies in the area of ontology engineering have focused on construction issues [48], with considerably less attention being given to the problem of changing an ontology in response to a certain need; in effect, ontologies have been so far mainly treated as being static [49]. However, there are several reasons why an ontology should change, the most important being changes in the domain, or changes in the conceptualization of the domain. We have used the term ontology change to refer to any type of change required in an ontology in response to a certain need.

This work can be divided in two parts: in the first part, we performed a short, critical literature review covering all the diverse types of ontology change. This allowed us to fix a certain terminology in an area that is characterized by the use of underspecified terms which are used with different meanings by different authors. We also reviewed the current state-of-the-art in each field related to the broad area of ontology change.

In the second part, we restricted our attention to the problem of ontology evolution and identified certain deficiencies of the currently used methods. We argued that research in ontology evolution is still at a preliminary, immature stage and outlined the most important problems currently faced in the field. We also explained why many of the practical applications of ontology evolution will require a different, more formal, computer-based approach to the problem.

Towards this aim, we proposed the use of old tricks to address this new problem. More specifically, we have presented a certain logic-based framework which allows the problem of ontology evolution to be formulated in terms of the more general problem of belief change [41], which has been extensively studied in the literature over the past 20 years. The proposed framework is not based on any additional assumptions except the standard assumptions made in the ontology evolution field, thus our reformulation is sound in the sense that it does not alter the properties of the problem in any way. It also allows the ideas and intuitions appearing in the rich belief change literature to be applied, with only minor modifications, to the ontology evolution problem. We believe that our approach complements the currently used methods and could be used as a supplementary tool for ontology maintenance and evolution.

As an application of our method, we have sketched a certain attempt ([35], [36]) to reformulate the AGM theory [1] in the context of DLs [7], one of the leading formalisms for ontological representation [8], as well as in the context of OWL [22]. This attempt is exemplary of the research path we are proposing and shows the main problems that are likely to be encountered during the migration of belief change theories to the context of ontology evolution.

Our proposition uncovers a different viewpoint on the problem of ontology evolution. We have only scratched the surface of the relation between ontology evolution and belief change; much work needs to be done on this issue, both in theoretical and in practical grounds. The application of specific belief change algorithms or postulations in the context of ontology evolution could prove interesting and uncover useful approaches to this problem. The proposed migration of the AGM theory in the ontology evolution context is not complete either, as only the contraction operator was considered; future work should address the problem of revision as well.

References

[1] C. Alchourron, P. Gärdenfors, D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. Journal of Symbolic Logic, 50:510-530, 1985.

[2] C. Alchourron, D. Makinson. On the Logic of Theory Change: Safe Contraction. Studia Logica 44:405-422, 1985.

[3] C. Alchourron, D. Makinson. Maps Between Some Different Kinds of Contraction Function: the Finite Case. Studia Logica 45:187-198, 1986.

[4] G. Antoniou. Nonmonotonic Reasoning. The MIT Press, 1997.

[5] A. Artale, E. Franconi, M. Mosurovic, F. Wolter, M. Zakharyaschev. The DLR_{US} Temporal Description Logic. In Proceedings of the 2001 Description Logic Workshop (DL-01), pp. 96-105, 2001.

[6] F. Baader, U. Sattler. An Overview of Tableau Algorithms for Description Logics. Studia Logica, 69, pp. 5-40, 2001.

[7] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (eds). The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002.

[8] F. Baader, I. Horrocks, U. Sattler. Description Logics as Ontology Languages for the Semantic Web. In D. Hutter, W. Stephan, (eds). Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 2003.

[9] T. Bench-Capon, G. Malcolm. Formalizing Ontologies and Their Relations. In Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA-99), pp. 250-259, 1999.

[10] S. Benferhat, S. Kaci, D. Le Berre, M. Williams. Weakening Conflicting Information for Iterated Revision and Knowledge Integration. Artificial Intelligence, 153:339-371, 2004.

[11] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. Scientific American, 284(5):34-43, 2001.

[12] D. Billington, G. Antoniou, G. Governatori, M. Maher. Revising Nonmonotonic Theories: The Case of Defeasible Logic. In Proceedings of the 22nd German Conference on Artificial Intelligence (KI-99), Lecture Notes in Artificial Intelligence (LNAI), Volume 1701/1999, Springer, 1999.

[13] A. Bochman. Entrenchment versus Dependence: Coherence and Foundations in Belief Change. Journal of Logic, Language and Computation, 11(1):3-27, 2002.

[14] A. Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. Artificial Intelligence, 82, pp. 353-367, 1996.

[15] J. de Bruijn, F. Martin-Recuerda, D. Manov, M. Ehrig. D4.2.1: State of the Art Survey on Ontology Merging and Aligning. 2004. Available on the Web (last visited September, 2005):
http://www.aifb.uni-karlsruhe.de/WBS/meh/publications/debruijn04state.pdf

[16] S.N. Burris. Logic For Mathematics And Computer Science. Prentice Hall, New Jersey, 1998.

[17] D. Calvanese. Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity. In Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96), pp. 303-307, 1996.

[18] H. Chalupsky. OntoMorph: A Translation System for Symbolic Knowledge. In Proceedings of the 7th International Conference on Knowledge Representation and Reasoning (KR-00), 2000.

[19] M. Dalal. Investigations Into a Theory of Knowledge Base Revision: Preliminary Report. In Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88), pp. 475-479, 1988.

[20] M. Dalal. Updates in Propositional Databases. Technical Report, DCS-TR-222, Department of Computer Science, Rutgers University, 1988.

[21] A. Darwiche, J. Pearl. On the Logic of Iterated Belief Revision. Artificial Intelligence, 89(1-2):1-29, 1997.

[22] D. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuiness, P. Patel-Schneider, L.A. Stein. OWL Web Ontology Language Reference. W3C Recommendation, 2005. Available on the Web (last visited September, 2005):
http://www.w3.org/TR/owl-ref/

[23] J.P. Delgrande. A Consistency-Based Approach for Belief Change. Artificial Intelligence, 151(1-2):1-41, 2003.

[24] J.P. Delgrande. A Minimal Modelling for Successful Knowledge Base Revision. In Frontiers of Belief Revision, M.A. Williams, H. Rott (eds.), Applied Logic Series, Volume 22, Series Editors: D. Gabbay, J. Barwise, Kluwer Academic Publishers, 2001.

[25] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf . Reasoning in description logics. In G. Brewka (ed). Foundations of Knowledge Representation, pp. 191-236. CSLI-Publications, 1996.

[26] F.M. Donini, D. Nardi, R. Rosati. Autoepistemic Description Logics. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), pp. 136-141, 1997.

[27] J. Doyle. A Truth Maintenance System. Artificial Intelligence, 12(3):231-272, 1979.

[28] A.J. Duineveld, R. Stoter, M.R. Weiden, B. Kenepa, V.R. Benjamins. WonderTools? A Comparative Study of Ontological Engineering Tools. International Journal of Human-Computer Studies, 52(6):1111-1133, 2000.

[29] H.B. Enderton. A Mathematical Introduction to Logic. Academic Press, New York, 1972.

[30] J. Euzenat, T. Le Bach, J. Barrasa, P. Bouquet, J. de Bo, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckeschmidt, P. Shvaiko, S. Tessaris, S. van Acker, I. Zaihrayeu. D2.2.3: State of the Art on Ontology Alignment. 2004. Available on the Web (last visited September, 2005):
http://www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf

[31] R. Fagin, J.D. Ullman, M.Y. Vardi. On the Semantics of Updates in Databases. In Proceedings of the 2nd SIGACT-SIGMOD Symposium on Principles of Database Systems, pp. 352-365, 1983.

[32] A. Ferrara. Methods and Techniques for Ontology Matching and Evolution in Open Distributed Systems. In Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAISE-04), 2004.

[33] G. Flouris, D. Plexousakis, G. Antoniou. AGM Postulates in Arbitrary Logics: Initial Results and Applications. Technical Report FORTH-ICS/TR-336, April 2004.

[34] G. Flouris. Belief Change in Arbitrary Logics. In Proceedings of the 3rd Hellenic Data Management Symposium (HDMS-04), PhD Presentations Session, 2004.

[35] G. Flouris, D. Plexousakis, G. Antoniou. Generalizing the AGM Postulates: Preliminary Results and Applications. In Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR-04), 2004.

[36] G. Flouris, D. Plexousakis, G. Antoniou. On Applying the AGM Theory to DLs and OWL. In Proceedings of the 4th International Semantic Web Conference (ISWC-05), 2005.

[37] G. Flouris, D. Plexousakis, G. Antoniou. Updating Description Logics Using the AGM Theory. In Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning, 2005.

[38] G. Flouris, D. Plexousakis, G. Antoniou. Updating DLs Using the AGM Theory: A Preliminary Study. In Proceedings of the 2005 Description Logic Workshop (DL-05), 2005.

[39] A. Fuhrmann. Theory Contraction Through Base Contraction. Journal of Philosophical Logic, 20:175-203, 1991.

[40] A. Fuhrmann, S.O. Hansson. A Survey of Multiple Contractions. Journal of Logic, Language and Information, 3:39-76, 1994.

[41] P. Gärdenfors. Belief Revision: An Introduction. In P. Gärdenfors (ed), Belief Revision, pp. 1-20, Cambridge University Press, 1992.

[42] P. Gärdenfors. The Dynamics of Belief Systems: Foundations vs Coherence Theories. Revue Internationale de Philosophie 44, pp. 24-46. Reprinted in C. Bicchieri, M. L. Dalla Chiara (eds). Knowledge, Belief and Strategic Interaction, pp. 377-396, Cambridge University Press, 1992.

[43] P. Gärdenfors, D. Makinson. Revisions of Knowledge Systems Using Epistemic Entrenchment. In Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge (TARK-88), pp. 83-95, 1988.

[44] J. Giesl, I. Neumann. The Semantics of Rational Contractions. Technical Report 29/94, Universität Karlsruhe, Germany, 1994.

[45] G. Gratzer. Lattice Theory: First Concepts And Distributive Lattices. W. H. Freeman and co. San Francisco, 1971.

[46] A. Grove. Two Modellings for Theory Change. Journal of Philosophical Logic, 17:157-170, 1988.

[47] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993.

[48] P. Haase, L. Stojanovic. Consistent Evolution of OWL Ontologies. In Proceedings of the 2nd European Semantic Web Conference (ESWC-05), 2005.

[49] P. Haase, Y. Sure. D3.1.1.b State of the Art on Ontology Evolution. 2004. Available on the Web (last visited September, 2005): http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/SEKT-D3.1.1.b.pdf

[50] S.O. Hansson. A Survey of Non-Prioritized Belief Revision. Erkenntnis, 50:413-427, 1999.

[51] S. O. Hansson. A Textbook Of Belief Dynamics. Kluwer Academic Publishers, 1999.

[52] S.O. Hansson. In Defense of Base Contraction. Synthese 91:239-245, 1992.

[53] S.O. Hansson. Revision of Belief Sets and Belief Bases. Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 3, pp. 17-75, 1998.

[54] S.O. Hansson. Knowledge-level Analysis of Belief Base Operations. Artificial Intelligence, 82:215-235, 1996.

[55] S.O. Hansson. Theory Contraction and Base Contraction Unified. Journal of Symbolic Logic, 58(2):602-625, 1993.

[56] J. Heflin, J. Hendler, S. Luke. Coping with Changing Ontologies in a Distributed Environment. In Proceedings of the Workshop on Ontology Management of the 16th

National Conference on Artificial Intelligence (AAAI-99), WS-99-13, AAAI Press, pp. 74-79, 1999.

[57] J. Heflin, J. Hendler. Dynamic Ontologies on the Web. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00), pp. 443-449, 2000.

[58] I. Horrocks, P. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. Journal of Web Semantics, 1(4):345-357, 2004.

[59] I. Horrocks, U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), pp. 199-204, 2001.

[60] I. Horrocks, U. Sattler, S. Tobies. Practical Reasoning for Very Expressive Description Logics. In H. Ganzinger, D. McAllester, A. Voronkov, (eds). Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR-99), Lecture Notes in Artificial Intelligence (LNAI), Volume 1705/1999, pp. 161-180, Springer-Verlag, 1999.

[61] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, G. Visser. Reasoning with Inconsistent Ontologies. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[62] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, G. Visser. Reasoning with Inconsistent Ontologies: Framework and Prototype. SEKT Deliverable D3.4.1, 2005. Available on the Web (last visited September, 2005):
http://wasp.cs.vu.nl/~huang/papers/sekt341.pdf

[63] A. Hunter, J. Delgrande. Iterated Belief Change: A Transition System Approach. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[64] U. Hustadt, R.A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In Automated Deduction in Classical and Non-Classical Logics, Lecture Notes in Artificial Intelligence (LNAI), Volume 1761/2000, pp. 191-205, Springer, 2000.

[65] Y. Kalfoglou, M. Schorlemmer. Ontology Mapping: the State of the Art. Knowledge Engineering Review, 18(1), pp. 1-31, 2003.

[66] S.H. Kang, S.K. Lau. Ontology Revision Using the Concept of Belief Revision. In Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES-04), part III, pp. 8-15, 2004.

[67] H. Katsuno, A.O. Mendelzon. On the Difference Between Updating a Knowledge Base and Revising It. Technical Report on Knowledge Representation and Reasoning, University of Toronto, Canada, KRR-TR-90-6, 1990.

[68] H. Katsuno, A.O. Mendelzon. Propositional Knowledge Base Revision and Minimal Change. Technical Report on Knowledge Representation and Reasoning, University of Toronto, Canada, KRR-TR-90-3, 1990.

[69] M. Klein, D. Fensel. Ontology Versioning on the Semantic Web. In Proceedings of the International Semantic Web Working Symposium (SWWS), pp. 75-91, 2001.

[70] M. Klein, D. Fensel, A. Kiryakov, D. Ognyanov. Ontology Versioning and Change Detection on the Web. In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02), 2002.

[71] M. Klein, N.F. Noy. A Component-Based Framework for Ontology Evolution. In Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems, CEUR-WS, vol. 71, 2003.

[72] S. Konieczny. On the Difference Between Merging Knowledge Bases and Combining Them. In Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-00), pp. 135-144, 2000.

[73] S. Konieczny, J. Lang, P. Marquis. Reasoning Under Inconsistency: The Forgotten Connective. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[74] S. Konieczny, R. Pino-Perez. Propositional Belief Base Merging or How to Merge Beliefs/Goals Coming from Several Sources and Some Links with Social Choice Theory. European Journal of Operational Research, 160(3):785-802, 2005.

[75] P. Lambrix, A. Edberg. Evaluation of Ontology Merging Tools in Bioinformatics. In Proceedings of the 8th Pacific Symposium on Biocomputing, pp. 589-600, 2003.

[76] K. Lee, T. Meyer. A Classification of Ontology Modification. In Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI-04), pp. 248-258, 2004.

[77] P. Liberatore. The Complexity of Iterated Belief Revision. In Proceedings of the 6th International Conference on Database Theory (ICDT-97), pp. 276-290, 1997.

[78] S. Luke, L. Spector, D. Rager, J. Hendler. Ontology-based Web Agents. In Proceedings of the 1st International Conference on Autonomous Agents, pp. 59-66, 1997.

[79] C. Lutz, U. Sattler. Mary Likes All Cats. In Proceedings of the 2000 International Workshop in Description Logics (DL-00), pp. 213-226, 2000.

[80] A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz. An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. In Proceedings of the 12th International World Wide Web Conference (WWW-03), 2003.

[81] D. Makinson. Five Faces of Minimality. Studia Logica, 52:339-379, 1993.

[82] D. Makinson. How to Give It Up: A Survey of Some Formal Aspects of the Logic of Theory Change. Synthese 62, pp. 347-363, 1985.

[83] D. Makinson. On the Status of the Postulate of Recovery in the Logic of Theory Change. Journal of Philosophical Logic 16:383-394, 1987.

[84] D. McGuiness, R. Fikes, J. Rice, S. Wilder. An Environment for Merging and Testing Large Ontologies. In Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-00), Technical Report KSL-00-16, Knowledge Systems Laboratory, Stanford University, 2000.

[85] E. Mendelson. Introduction to Mathematical Logic. Wadsworth and Brooks/Cole Advanced Books and Software, Pacific Grove, California, 1987.

[86] T. Meyer, K. Lee, R. Booth. Knowledge Integration for Description Logics. In Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning, 2005.

[87] G.A. Miller. WordNet: A Lexical Database for English. Communications of the ACM (CACM), 38(11):39-41, 1995.

[88] E. Miller, R. Swick, D. Brickley. Resource Description Framework (RDF) / W3C Semantic Web Activity, 2004. Available on the Web (last visited September, 2005): http://www.w3.org/RDF/

[89] A.C. Nayak. Iterated Belief Change Based on Epistemic Entrenchment. Erkenntnis, 41:353-390, 1994.

[90] B. Nebel. A Knowledge Level Analysis of Belief Revision. In Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR-89), pp. 301-311, 1989.

[91] A. Newell. The Knowledge Level. Artificial Intelligence, 18(1), 1982.

[92] N.F. Noy, M. Klein. Ontology Evolution: Not the Same as Schema Evolution. Knowledge and Information Systems, 6(4), pp. 428-440. Available as SMI technical report SMI-2002-0926, 2004.

[93] N.F. Noy, S. Kunnatur, M. Klein, M.A. Musen. Tracking Changes During Ontology Evolution. In Proceedings of the 3rd International Conference on the Semantic Web (ISWC-04), published as Lecture Notes in Computer Science (LNCS), Volume 3298/2004, pp. 259-273, S.A. McIlraith, D. Plexousakis, F. van Harmelen (eds), Springer-Verlag, 2004.

[94] N.F. Noy, M.A. Musen. Algorithm and Tool for Automated Ontology Merging and Alignment. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831, 2000.

[95] N.F. Noy, M.A. Musen. An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support. In Proceedings of the Workshop on Ontology Management at 16th National Conference on Artificial Intelligence (AAAI-99). Available as SMI technical report SMI-1999-0799, 1999.

[96] N.F. Noy, M.A. Musen. SMART: Automated Support for Ontology Merging and Alignment. In Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management. Available as SMI technical report SMI-1999-0813, 1999.

[97] H.S. Pinto, A. Gomez-Perez, J.P. Martins. Some Issues on Ontology Integration. In Proceedings of the Workshop on Ontologies and Problem-Solving Methods (KRR5) at 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.

[98] S. Schlobach, R. Cornet. Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.

[99] M. Schmidt-Schaub, G. Smolka. Attributive Concept Descriptions with Complements. Artificial Intelligence, 48:1-26, 1991.

[100] L. Stojanovic, A. Maedche, B. Motik, N. Stojanovic. User-driven Ontology Evolution Management. In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02), published as Lecture Notes in Computer Science (LNCS), Volume 2473/2002, pp. 285-300, Springer-Verlag, 2002.

[101] L. Stojanovic, A. Maedche, N. Stojanovic, R. Studer. Ontology Evolution as Reconfiguration-Design Problem Solving. In Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP-03), pp. 162-171, 2003.

[102] H. Stuckenschmidt, M. Klein. Integrity and Change in Modular Ontologies. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.

[103] G. Stumme, A. Maedche. Ontology Merging for Federated Ontologies on the Semantic Web. In E. Franconi, K. Barker, D. Calvanese (eds). Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII-01), Lecture Notes in Artificial Intelligence (LNAI), Springer, 2002.

[104] A. Weber. Updating Propositional Formulas. In Proceedings of the 1st International Conference on Expert Database Systems (EDS-86), pp. 487-500, 1986.

[105] M.A. Williams. Anytime Belief Revision. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), 1997.

[106] Z. Zhang, L. Zhang, C.X. Lin, Y. Zhao, Y. Yu. Data Migration for Ontology Evolution. In Poster Proceedings of the 2nd International Semantic Web Conference (ISWC-03), 2003.

[107] E. Zolin. Navigator on Description Logics Complexity. Web Page (last visited September, 2005):
http://www.cs.man.ac.uk/~ezolin/logic/complexity.html