Representing Definitional Changes in Legal Ontologies

Szymon Klarman and Marc Bron

University of Amsterdam
Leibniz Center for Law
{sklarman, mbron}@science.uva.nl

Abstract. A definitional change is one of the common ways in which legal ontologies can be amended. A revision of the meaning of a term in a model of legislation entails different classification of individuals, and thus affects knowledge about the domain and possibly the set of normative consequences implied by the model. In the paper we propose and discuss a description logic based representation that allows modeling and switching between different definitional variants of concepts used in an ontology.

1 Introduction

An ontology represents a shared conceptualization, by means of which a certain domain is comprehended, and provides a semantic basis for reasoning about objects in the domain. One of the central constructs employed in ontologies is concept definition. A definition determines all sufficient and necessary conditions that an individual has to satisfy in order to be classified as an instance of a particular concept.

A serious challenge for ontology design and maintenance, broadly discussed in the recent literature (e.g. [EK04], [HH00], [HSV04]), concerns handling dynamic aspects of represented knowledge. A special case of that problem, which is in the focus of this paper, is how to approach changes that affect definitions of terms covered by an ontology.

In some domains, such as science or law, introducing definitional changes is a relatively common practice. For instance, new scientific theories lead to revisions of concepts used differently within the previously accepted body of knowledge, whereas pieces of legislation impose new interpretations on terms defined in older legislative documents. Such changes can clearly have far-reaching implications both from epistemic and pragmatic perspective. A new concept definition entails a different classification of objects in the domain, which, by the entrenchment of terms in the ontology, can effect in a deep restructuring of the domain's representation. Consequently, reasoning over the revised ontology is unlikely to result in the original set of conclusions. For instance, definitional changes in law

can bring about new normative consequences and thus directly affect rights and obligations of citizens[1].

Interestingly, however, there is a significant asymmetry in the status granted to outdated definitions in different domains. Parts of scientific knowledge that are replaced with more recent results are simply considered false and, hence, deprived of any substantial value. It is always the most recent version of an ontology that is applied to provide reliable explanations and predictions in science. Conversely, in law there is a justified interest in keeping track of older versions. A legal case can be properly assessed only according to law that is applicable at the time the case takes place. Since it is often necessary in the process to consider and analyze past cases, previous ontologies of law have to stay available. We conclude that the sort of revision that a legal ontology is submitted to in face of a definitional change is therefore not that of a plain update but rather of versioning, and hence requires a more cautious and sophisticated approach.

In this paper we want to address the problem of handling definitional changes in legal ontologies and propose a description logic based representation, which provides a simple, but in many respects convenient ontology versioning technique. The representation consists of a set of TBox and ABox axiom schemes, which implemented in an ontology, allow for reasoning about a domain in selected points of time strictly according to concept definitions valid at that time. In the next section we review related work and formulate basic requirements for an ontology versioning formalism. In the following part we present the main result of the work and close the paper with conclusions and discussion of the proposed solution.

2 Background

2.1 Related Work

Representing change and handling different variants of ontologies is the subject of research on ontology evolution and versioning.

In ontology evolution the focus lies on implementing a change in an existing ontology, while maintaining consistency. For example, deleting a concept from an ontology requires deletion of all semantic links to this concept. Until all these have been removed, the ontology remains inconsistent. However, finding all the links can be difficult in even a small ontology and so revising becomes a laborious task. Systems that support knowledge engineers in updating ontologies and finding and resolving inconsistencies are e.g. KAON [Sto04] and evOWLution [HSV04].

In ontology versioning, changes are handled by creating new versions of an ontology. Besides recording changes and relations between different versions, it is useful to know if an ontology is backward compatible with the previous

[1] For an illustrative example see discussion on the consequences of changing the definition of "Dependent" in "Working Families Tax Relief Act" accepted by the Congress of USA in October 2004 [BIB04].

version. When an application depends on a certain ontology, of which a new version becomes available, knowing if the version can also be used is essential for proper functioning of the application. A system that implements such a versioning system is e.g. SHOE [HH00].

Efforts at combining both strategies in a single system are described in [KN04].

In the case of dynamic domains, such as law, whose past representations should stay accessible to reasoning services, for every change a different version of the ontology has to be maintained. Issues of special importance here are that compatibility between different versions has to be guaranteed, that there should be a quick way of switching between ontologies and that redundancy of representation should be reduced to a minimum.

Such an approach, by which multiple changes can be represented in a single ontology is described in [EK04], where a timestamping technique is used. An ontology is described by a graph where each node (concept) and edge (property) is supplied with a timestamp indicating its validity.

The three proposed ways of implementation are:

Meta Ontology: an ontology that stores different versions of an ontology with their timestamps and deals with the sturctural and temporal relationships between the different ontologies.

Standard Extension: extending a representation language with the capability to represent time and change.

Using Standard: using standard techniques provided by a representation language to timestamp concepts; versionInfo tag or isDefinedBy tag.

Each of the three techniques are reported to have drawbacks when implemented in OWL. One of them is that an ontology that incorporates timestamps would not support reasoning if the time dimension is not considered in the right way. Basically, a reasoning algorithm should only infer conclusions that are valid for the current time point. Another reported drawback is that OWL does not guarantee that temporal constraints are correctly interpreted, e.g. that a relation can only exist between two concepts that are both valid at the same time.

Furthermore, in a timestamping approach care must be taken that no update blowup is caused. If, for example, a concept A has a relation to concept B and the latter concept is updated to a new definition B', then A has to be updated as well to A', so that the relation of A' points to B'. Now any concept with a relation to A also has to be updated.

Alternatively, one can handle definitional changes by resorting to rules in order to classify only relevant individuals at particular points in time. In a rule language like SWRL definitional variants of "block" can be expressed like:

Rule 1: $\text{square}(?x) \wedge \text{wood}(?x) \wedge \text{time}(t_1) \wedge \text{valid\_at}(?x, t_1) \Rightarrow \text{block}(?x)$.
Rule 2: $\text{rectangle}(?x) \wedge \text{metal}(?x) \wedge \text{time}(t_2) \wedge \text{valid\_at}(?x, t_2) \Rightarrow \text{block}(?x)$.

Such an approach would certainly tackle some of the discussed problems, however, applying a rule formalism in ontology modeling should never be preferred over standard ontology languages because of the risk of undecidability

and the need for employing an additional rule engine for obtaining standard reasoning services (here classification tasks).

2.2 Formalism Requirements

Taking all the above into account leads to the following requirements for a representation formalism:

- Preventing many versions with only small changes.
- Relying on standard languages and reasoning tools and keeping the architecture possibly homogenous.
- Enabling the possibility of switching between versions of the ontology.
- Preventing an update blowup.
- Maintaining compatibility to maximum possible extent.

The solution proposed in this paper can be used to represent concepts that change (dynamic concepts) in a single ontology, using OWL DL while preserving the possibility of using a standard reasoner.

Intuitively the formalism can be explained as follows (see figure 1): a generic concept C is defined in terms of the subsumed concepts: C_1, C_2 or C_3, representing its different definitional variants. Each variant is scoped, to be only valid within a certain time period. Similar temporal limits are imposed on individuals in the domain.

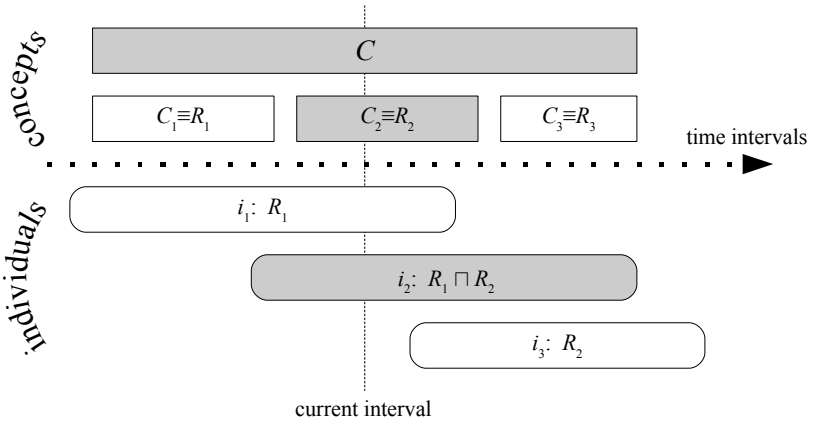

Figure 1.

Respective concepts and individuals are activated by specifying which point in the time line is the current or valid time. Only those entities, for which the current time falls within their scope are considered. Consequently, the generic concept receives the meaning of its appropriate variant and covers only the relevant and existing individuals.

By selecting the current time point only the version of the ontology that is valid at that point is satisfiable.

2.3 Preliminaries

The representation requires a description logic language as expressive as $\mathcal{SHOIN}$. The expressiveness of $\mathcal{SHOIN}$ is directly supported by OWL-DL, what makes implementing the solution in OWL ontologies practically straightforward[2]. In the presentation we will use some of the standard description logic constructors, which along their OWL counterparts and underlying semantics are summarized in the Appendix.

Vocabulary of a DL knowledge base consists of a set of concept names N_C, role names N_R and individual names N_I. The semantics is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty domain of individuals and $\cdot^{\mathcal{I}}$ is a function interpreting the vocabulary in terms of domain objects. The semantics of complex concepts and roles is defined inductively on the set of DL construction rules.

A concept definition in DL is an expression $C \equiv D$, where C and D are two concepts and $C \in N_C$. A finite set of concept definitions with unique left hand side concept names forms a terminology (TBox). The definition is satisfied in $\mathcal{I}$ provided that $C^{\mathcal{I}} = D^{\mathcal{I}}$. We can further distinguish between atomic concepts that occur on the left hand side of some TBox axiom (defined concepts) and ones that are used only on the right hand side of definitions (primitive concepts). Since $\mathcal{SHOIN}$ language accounts also for nominals (i.e. enumerations of individuals), which might be equally used in definitions, we will cover them with an extended notion of primitive concepts.

A base interpretation $\mathcal{I}$ for a TBox is one that interprets only its primitive concepts. An interpretation $\mathcal{J}$ is an extension of $\mathcal{I}$, if it interprets also defined concepts, while agreeing with $\mathcal{I}$ as to the domain of interpretation and to interpretation of primitive concepts. A terminology is definitorial if its base interpretation determines a unique extension [BN03].

An ABox consists of a finite set of assertions of one of the following forms: $C(a), r(a,b), \neg r(a,b)$, where $a, b \in N_I$, $C \in N_C$ and $r \in N_R$. Assertions in ABox are naturally understood as pieces of information about the represented world, expressed by means of selected vocabulary.

In the further presentation we will refer to an arbitral definitorial knowledge base $\mathcal{K}(\mathcal{T}, \mathcal{A})$ with a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. Formally, we shall develop a formalism for representing a a concept C by means of its definitional variants and a simple mechanism for selecting a valid time $i \in \{1, \ldots, n\}$, such that the respective version $\mathcal{K}^i$ of $\mathcal{K}$ has the same base interpretation as $\mathcal{K}$ but possibly a different extension $\mathcal{J}_i$, yielding $C^{\mathcal{J}_i} = C_i^{\mathcal{J}_i}$, where C_i represents a proper variant of the concept C in time i.

3 Representation

We shall now give a systematic account of the representation, introducing consecutive stages in its composition and translating them into DL constructs, which

[2] The only restriction regards the Unique Name Assumption. We address this issue at the end of the paper.

a DL knowledge base has to account for in order to support definitional changes along the proposed lines. The representation has a three-layered structure, consisting of a simple temporal framework, a collection of timestamp restrictions on individuals and a set of timescope restrictions on concepts. The valid time selection is handled on the TBox level, by allowing alternative definitions of a designated concept.

3.1 Temporal Framework and Valid Time Selection

The temporal framework, underlying the representation, is based on interval algebra and defined as a tuple $\langle T, \preceq \rangle$ [Zar98]. A finite, non-empty set T contains elements denoted as chronons, i.e. undecomposable time intervals [JD98], whereas $\preceq$ is a total order on T, i.e. an ordering relation that satisfies the following conditions. For every $t_1, t_2, t_3 \in T$:

1. If $t_1 \preceq t_2$ and $t_2 \preceq t_1$ then $t_1 = t_2$ (antisymmetry).
2. If $t_1 \preceq t_2$ and $t_2 \preceq t_3$ then $t_1 \preceq t_3$ (transitivity).
3. $t_1 \preceq t_2$ or $t_2 \preceq t_1$ (completeness).

In other words, a tuple $\langle T, \preceq \rangle$ constitutes a discrete, finite time axis, whose atoms are chronons.

The framework can be directly embedded in $\mathcal{K}$ by means of a primitive concept $\texttt{TimeInterval} \in N_C$, such that $\texttt{TimeInterval}^{\mathcal{I}} \neq \emptyset$, and a role $\texttt{beforeEq} \in N_R$. The ABox has to be equipped with a set of assertions guaranteeing that $\texttt{beforeEq}$ is a total ordering on $\texttt{TimeInterval}^{\mathcal{I}}$, i.e. it is antisymmetric, transitive and complete[3]. For more flexibility we shall require also an inverse of $\texttt{beforeEq}$:

$$\texttt{afterEq} = \texttt{beforeEq}^{-}$$

Obviously, $\texttt{afterEq}$ is also a total order on $\texttt{TimeInterval}^{\mathcal{I}}$.

The three specified constructs are sufficient to provide a frame of reference for representing temporal aspects of definitional changes, including the dynamics of a domain on the level of individuals. For simplicity, we will assume that there are exactly n individuals in $\texttt{TimeInterval}^{\mathcal{I}}$, all of which are represented in N_I by unique names, whose indexes correspond to positions in the ordering, i.e.:

1. $\{\texttt{interval}_1^{\mathcal{I}}, \ldots, \texttt{interval}_n^{\mathcal{I}}\} = \texttt{TimeInterval}^{\mathcal{I}}$
2. for any $\texttt{interval}_i$ and $\texttt{interval}_j$, if $i \leq j$ then $\mathcal{A}$ contains the following two assertions:
   (a) $\texttt{beforeEq}(\texttt{interval}_i, \texttt{interval}_j)$
   (b) $\texttt{afterEq}(\texttt{interval}_j, \texttt{interval}_i)$

[3] In practice, it should be more convenient to first define an auxiliary role $\texttt{directlyBeforeEq}$ that accounts for identity relation on $\texttt{TimeInterval}^{\mathcal{I}}$ and provides links between all immediate successors and predecessors in the ordering, and then augment it with transitive closure, so that $\texttt{directlyBeforeEq} \sqcup \texttt{directlyBeforeEq}^{+} \equiv \texttt{beforeEq}$.

In order to control the focus of a knowledge base we require a valid time selection mechanism, which will allow pointing to a particular interval, with respect to which relevant concepts and individuals will be activated. There are many ways to achieve such a function. Here we propose a simple method based on a single concept definition using a nominal. For some $i \in \{1, \ldots, n\}$ let $\texttt{CurrentInterval} \in N_C$ be defined as:

$$\texttt{CurrentInterval} \equiv \{\texttt{interval}_i\}$$

As will be shown, the rest of the formalism essentially relies on the interpretation of CurrentInterval, thus the choice of the interval is propagated to the whole knowledge base.

The approach has two apparent advantages. First, it enables a very easy way for switching between versions. Second, it does not affect the base interpretation of the knowledge base, what may potentially simplify certain reasoning tasks. Observe, that if two knowledge bases $\mathcal{K}(\mathcal{T}, \mathcal{A})$ and $\mathcal{K}'(\mathcal{T}', \mathcal{A})$ differ from each other only with respect to the choice of the current interval, none of the primitive concepts obtains a different interpretation, hence, switches entail changes only for extensions of the base interpretation.

3.2 Timestamps and Timescopes

At the next stage we relate individuals and concepts to the time axis. For that purpose we apply two relations: $\texttt{from}, \texttt{to} \in N_r$.

The idea of a timestamp, which links an individual to intervals of time, is very intuitive and boils down to marking exactly two limits of the individual's existence in a domain of application [Zar98]. We adapt this explication in the following general definition:

$$\begin{aligned}\texttt{Timestamped} \equiv\ & (\exists \texttt{from}.\texttt{TimeInterval} \sqcap \geq 1\ \texttt{from} \sqcap \leq 1\ \texttt{from}) \\ & \sqcap\ (\exists \texttt{to}.\texttt{TimeInterval} \sqcap \geq 1\ \texttt{to} \sqcap \leq 1\ \texttt{to})\end{aligned}$$

For any individual a, such that $\texttt{Timestamped}(a)$, the ABox must contain two other assertions: $\texttt{from}(a, \texttt{interval}_i)$ and $\texttt{to}(a, \texttt{interval}_j)$, meaning that a came into existence in $\texttt{interval}_i$ and ceased to exist in $\texttt{interval}_j$. Naturally, it should be assured that $1 \leq i \leq j \leq n$.

Complementarily to timestamps, we specify timescopes for concepts, which denote a period of time during which a given concept remains valid. A general timescope will be represented by the following axiom scheme:

$$\begin{aligned}&\texttt{Timescope}_{(i,j)} \equiv \\ &\texttt{TimeInterval} \sqcap \exists \texttt{afterEq}.\{\texttt{interval}_i\} \sqcap \exists \texttt{beforeEq}.\{\texttt{interval}_j\}\end{aligned}$$

Clearly, given a total ordering of time intervals, an interpretation function maps $\texttt{Timescope}_{(i,j)}$ onto the subset of exactly those intervals that are placed between $\texttt{interval}_i$ and $\texttt{interval}_j$ (including the two) on the time axis. Again, a proper ordering $1 \leq i \leq j \leq n$ should be satisfied.

Recall that the goal of the representation is to restrict classification only to individuals and concepts valid at a given point of time. Hence it has to be assured, that given the current interval the definition of a valid concept can be satisfied only by individuals that exist in that interval, and conversely, that none of the definitions of invalid concepts are satisfiable. We obtain this feature in two steps.

First we allow a specific interplay between concepts $\texttt{CurrentInterval}$ and $\texttt{Timescope}_{(i,j)}$. Let $\texttt{CTimescope}_{(i,j)}$ be a concept defined as follows:

$$\texttt{CTimescope}_{(i,j)} \equiv \texttt{CurrentInterval} \sqcap \texttt{TimeScope}_{(i,j)}$$

Observe, that for any interpretation function $\cdot^{\mathcal{J}}$ it holds that:

$$\texttt{CTimeScope}^{\mathcal{J}}_{(i,j)} = \begin{cases} \texttt{CurrentInterval}^{\mathcal{J}} \textit{ iff } \texttt{CurrentInterval}^{\mathcal{J}} \subseteq \texttt{TimeScope}^{\mathcal{J}}_{(i,j)} \\ \emptyset \textit{ iff } \texttt{CurrentInterval}^{\mathcal{J}} \not\subseteq \texttt{TimeScope}^{\mathcal{J}}_{(i,j)} \end{cases}$$

In other words, an interpretation of $\texttt{CTimeScope}_{(i,j)}$ is a singleton with the current interval as its only member, provided that the respective timescope contains the current interval. Otherwise the interpretation of the concept is an empty set. It is straightforward to employ this property for distinguishing between valid and invalid concepts. Intuitively, valid concepts are those, whose time scopes contain the current interval. At the same time, we want to address only existing individuals, i.e. those that came into existence before or during the current interval and ceased to exist during or after it. We combine the two operations in the second step, by defining $\texttt{CTSRestriction}_{(i,j)}$ as:

$$\texttt{CTSRestriction}_{(i,j)} \equiv$$
$$\exists \texttt{from}.(\exists \texttt{beforeEq}.\texttt{CTimeScope}_{(i,j)}) \sqcap \exists \texttt{to}.(\exists \texttt{afterEq}.\texttt{CTimeScope}_{(i,j)})$$

Augmenting definition of any concept C with $\texttt{CTSRestriction}_{(i,j)}$ results in two possible outcomes. If $\texttt{interval}_k$, being the current interval, is such that $i \leq k \leq j$, then $C^{\mathcal{J}}$ will be restricted only to those timestamped individuals that exist in $\texttt{interval}_k$. If, however, $k < i$ or $j < k$ then C will be unsatisfiable with respect to the TBox, while there is no individual that can satisfy $\texttt{CTSRestriction}_{(i,j)}$. For a more formal explanation see the Appendix.

3.3 Implementation

Finally we shall briefly outline how the representation can be usefully applied in a knowledge base in order to yield desired effects for classification tasks.

Let $\texttt{DynamicConcept} \in N_C$ be a concept whose definition changes over time and $\{\texttt{Variant}_1, \ldots, \texttt{Variant}_m\} \subseteq N_C$ its m consecutive variants, where index $k \in \{1, \ldots, m\}$ uniquely identifies each of them. We want the variants to exclusively and exhaustively cover the time axis. Let us then assume that there is a function τ that assigns to every variant k an ordered pair (i, j), with $i \leq j$, representing the time scope in which the variant is valid, in such a way, that jointly the time scopes generate a partition of the time axis, i.e.:

1. for $k = 1$, $\tau(k) = (1, i)$,
2. for $k = m$, $\tau(k) = (j, n)$, where n is the greatest interval on the axis,
3. for any $1 \leq k < m$, $\tau(k) = (i, j)$ and $\tau(k + 1) = (j + 1, l)$.

Formally, we will define every variant as an intersection of its proper meaning, expressed in terms of some terminological restrictions, and the respective time scope restriction:

$$\texttt{Variant}_k \equiv \texttt{Meaning}_k \sqcap \texttt{CTSRestriction}_{\tau(k)}$$

Finally, the definition of `DynamicConcept` will be simply stated as the union of all its variants:

$$\texttt{DynamicConcept} \equiv \texttt{Variant}_1 \sqcup \ldots \sqcup \texttt{Variant}_m$$

This accomplishes the implementation. Observe that in each point of time there is exactly one valid variant. Therefore, if $\texttt{interval}_c$ represents the current interval, then interpretation of `DynamicConcept` is determined by the following equality:

$$\texttt{DynamicConcept}^{\mathcal{J}_c} = \texttt{Variant}_k^{\mathcal{J}_c}$$

where $\tau(k) = (i, j)$ and $i \leq c \leq j$.

Concluding, independently of the choice of the valid time `DynamicConcept` always denotes these and only these individuals that currently exist and fall under the currently valid meaning of `DynamicConcept`. As an illustration consider the following example. Let $\texttt{DynamicConcept} \equiv \texttt{C}_1 \sqcup \texttt{C}_2 \sqcup \texttt{C}_3$, where the consecutive variants are defined as follows:

$$\begin{aligned}
\texttt{C}_1 &\equiv \texttt{R}_1 \sqcap \texttt{CTSRestriction}_{(1,3)} \\
\texttt{C}_2 &\equiv \texttt{R}_2 \sqcap \texttt{CTSRestriction}_{(4,7)} \\
\texttt{C}_3 &\equiv \texttt{R}_3 \sqcap \texttt{CTSRestriction}_{(8,10)}
\end{aligned}$$

We will focus on three timestamped individuals of the following asserted properties:

$$\begin{aligned}
&\texttt{i}_1 : \texttt{from}(\texttt{i}_1, \texttt{interval}_1), \texttt{to}(\texttt{i}_1, \texttt{interval}_6), \texttt{R}_1(\texttt{i}_1) \\
&\texttt{i}_2 : \texttt{from}(\texttt{i}_2, \texttt{interval}_2), \texttt{to}(\texttt{i}_2, \texttt{interval}_{10}), \texttt{R}_1(\texttt{i}_2), \texttt{R}_2(\texttt{i}_2) \\
&\texttt{i}_3 : \texttt{from}(\texttt{i}_3, \texttt{interval}_6), \texttt{to}(\texttt{i}_3, \texttt{interval}_9), \texttt{R}_3(\texttt{i}_3)
\end{aligned}$$

Observe, that depending on the choice of the current interval the following extensions of `DynamicConcept` will be inferred:

$\texttt{CurrentInterval}^{\mathcal{J}}$	Valid Variant	$\texttt{DynamicConcept}^{\mathcal{J}}$
$\texttt{interval}_1$	$\texttt{C}_1$	$\texttt{i}_1$
$\texttt{interval}_3$	$\texttt{C}_1$	$\texttt{i}_1, \texttt{i}_2$
$\texttt{interval}_5$	$\texttt{C}_2$	$\texttt{i}_2$
$\texttt{interval}_9$	$\texttt{C}_3$	$\texttt{i}_3$
$\texttt{interval}_{10}$	$\texttt{C}_3$	none

Finally, a short comment with respect to implementation in OWL is in order. Notice, that whereas DL naturally complies to the Unique Name Assumption (UNA), OWL does not. From our experience in implementing the representation in OWL ontologies, it follows that the difference can bring about some unexpected classification results, caused by misinterpretation of the time axis by an OWL reasoner. Basically, whenever it is possible a reasoner attempts to equate some of the time intervals in order to obtain a satisfiable interpretation of all concepts in the ontology. To avoid such unintended interpretations, it is necessary to explicitly distinguish between time intervals using `owl:differentFrom` construct.

On the other hand, lack of UNA in OWL allows for an alternative treatment of the valid time selection mechanism. Instead of defining a concept called `CurrentInterval` we can introduce an individual of the same name, which can be easily identified with any interval by means of the `owl:sameAs` construct. As a consequence the choice of the current interval is held outside of the TBox. It may be also useful to employ another auxiliary individual `LastInterval`, equated to the last interval on the axis, to serve as the right limit of timestamps and timescopes of all those individuals and concepts which have not in fact become outdated at the time of implementation. Extending the time axis in an ontology, does not require then updating information about those entities, but merely changing the referent of `LastInterval`.

4 Conclusions and Discussion

In this paper we have proposed a representation for supporting definitional changes in ontologies, motivated predominantly by requirements from the legal domain. The solution can be expressed as a set of TBox and ABox axioms in DL $\mathcal{SHOIN}$ and thus is easily implementable in OWL-DL knowledge bases. The result allows for obtaining correct classifications of individuals in an ontology using different variants of concept definitions, applicable within specified scopes of time intervals. Furthermore, the inherent representation of time provides a framework for capturing the basic dynamics of the domain on the level of individuals, and for including temporal aspects directly into the definitions of concepts.

The solution satisfies the basic requirements with respect to ontology versioning formalisms and exhibits several interesting features. First of all, it provides a convenient mechanism for switching between versions of represented knowledge

and preserves consistency of each representation. Second, the way revisions are handled in the proposed approach prevents an update blowup and maintains backward compatibility between versions. Notice, that a definitional change is implemented monotonically, only by introducing new concepts to the current ontology. All remaining parts, even if referring to the name of a generic concept, are not altered. This property seems especially valuable for reuse and maintenance of legal ontologies, taking into account a typical entrenchment of concepts in a model of legislation. Moreover, the versioning is very space efficient. It is obtained not on the level of whole ontologies but on the level of representation, and so can be enclosed in a single knowledge base with no redundancy involved. Finally, the solution relies exclusively on DL language and standard reasoning tools providing classification services, e.g. Pellet or Racer. Thus no additional formalisms nor external ontology management systems have to be employed.

It has to be noted, however, that demonstrated space efficiency is achieved at the expense of time of classification required whenever a switch between versions is requested. Whether the trade-off is well balanced depends in principle on the intended application scenarios and the actual evaluation should be based on a representative sample of use cases. Presumably, three factors should play a role here: the size of an ontology, the scope and frequency of introduced changes and the frequency of version switching requests. The setting in which the solution might benefit the most is one involving a large model of legislation, to which frequently small amounts of changes are introduced. Clearly, the less requests there are for switching between versions the more suitable the representation proposal seems for the task. In any case one can always obtain separate versions of an ontology, by asserting inferred classifications for particular points of time, while still benefiting from the temporal interpretation of the individuals in the domain.

Elaboration on issues related to efficiency and possible application scenarios, as well as potential extensions for supporting other types of changes or temporal reasoning, will be addressed in future research.

References

[BN03] F. Baader, W. Nutt, "Basic Description Logic", in: F. Baader et al., *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003, 47-100.

[HSV04] P. Haase, Y. Sure, D. Vrandecic. *Ontology Management and Evolution Survey, Methods and Prototype. SEKT Formal Deliverable D3.1.1*, Institute AIFB, University of Karlsruhe, 2004.

[HH00] J. Heflin, J. Hendler, "Dynamic Ontologies on the Web", in: *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000, 443-449.

[JD98] C.S. Jensen, C.E. Dyreson (eds), "A Consensus Glossary of Temporal Database Concepts", in: O. Etzion et al., *Temporal Databases: Research and Practise.* Springer-Verlag, 1998, 367-405.

[KN04] M. Klein, N.F. Noy, "Ontology Evolution: Not the Same as Schema Evolution", in: *Knowledge and Information Systems*, 6(4) (2004), 428-440.

[EK04] J. Eder, C. Koncilia, "Modelling Changes in Ontologies", in: *Proceedings of the On The Move - Federated Conferences*, 2004, 662-673.

[Sto04] L. Stojanovic, *Methods and tools for ontology evolution*, Ph.D. Thesis, University of Karlsruhe, 2004.

[Zar98] G.P. Zarri, "Representation of Temporal Knowledge in Events: The Formalism, and Its Potential for Legal Narratives", in: *Information & Communications Technology Law* 7 (1998), 213-241.

[BIB04] "Changes in the Definition of Dependent. Legislative and Regulatory Fixes Forthcoming (But Not for All Benefits)", in: *Benefits Information Bulletin* 03/2004.

Appendix

Constructors. List of OWL/DL constructors used in the representation, and their semantics:

OWL Constructor	DL syntax	DL semantics
`inverseOf`	r^-	$\{(y,x) \mid (x,y) \in r^{\mathcal{I}}\}$
`Transitive`	$r \sqcup r^+$	$r^{\mathcal{I}} \cup \{(x,z) \mid (x,y),(y,z) \in r^{\mathcal{I}}\}$
`intersectionOf`	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
`unionOf`	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
`oneOf`	$\{a\}$	$\{a^{\mathcal{I}}\}$
`someValuesFrom`	$\exists r.C$	$\{x \mid \exists_y (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
`hasValue`	$\exists r.\{a\}$	$\{x \mid (x,a^{\mathcal{I}}) \in r^{\mathcal{I}}\}$
`minCardinality`	$\geq n\ r$	$\{x \mid card(\{y \mid (x,y) \in r^{\mathcal{I}}\}) \geq n\}$
`maxCardinality`	$\leq n\ r$	$\{x \mid card(\{y \mid (x,y) \in r^{\mathcal{I}}\}) \leq n\}$

Table 1.

Validation of Concepts. Let $C \equiv \texttt{CTSRestriction}_{(i,j)}$ and $\texttt{CurrentInterval} \equiv \{\texttt{interval}_k\}$. Consider two cases:

1. C **is valid in** $\texttt{interval}_k$, i.e. $i \leq k \leq j$. Then the following holds:

$$\texttt{CTimeScope}_{(i,j)} \equiv \texttt{CurrentInterval}$$

$$C \equiv \texttt{CTSRestriction}_{(i,j)} \equiv$$
$$\exists \texttt{from}.(\exists \texttt{beforeEq}.\texttt{CurrentInterval}) \ \sqcap \ \exists \texttt{to}.(\exists \texttt{afterEq}.\texttt{CurrentInterval})$$

Therefore C addresses only currently existing individuals.

2. C **is invalid in** $\texttt{interval}_k$, i.e. $k < i$ or $j < k$. Then the following holds:

$$\texttt{CTimeScope}_{(i,j)} \equiv \bot$$

$$C \equiv \texttt{CTSRestriction}_{(i,j)} \equiv$$
$$\exists \texttt{from}.(\exists \texttt{beforeEq}.\bot) \ \sqcap \ \exists \texttt{to}.(\exists \texttt{afterEq}.\bot)$$

Clearly there are no such individuals that could satisfy this condition, therefore $C^{\mathcal{J}} = \emptyset$ under any interpretation function $\cdot^{\mathcal{J}}$.