MORE2: An Extended Reasoning and Management System for Multi-version Ontologies

Zhisheng Huang, Annette ten Teije, and Frank van Harmelen
(Vrije Universiteit Amsterdam)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.5.3(WP3.5)
MORE is a multi-version ontology reasoning and management system, which is developed based on a temporal logic approach. In this document we introduce MORE2, an extended reasoning and management system for multi-version ontologies. MORE2 is a new prototype of MORE. We discuss the implementation issues of MORE2. Furthermore, we investigate and evaluate the applicability of the system MORE2 with several realistic data sets: SEKT legal ontologies and the Dutch general election data 2006.
Keyword list: ontology versioning, ontology reasoning, temporal reasoning

Document Id.	SEKT/2006/D3.5.3/v1.0.0
Project	SEKT EU-IST-2003-506826
Date	January 30, 2007
Distribution	public

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006 Madrid
Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Empolis GmbH
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Sirma AI EAD, Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Executive Summary

MORE is a multi-version ontology reasoning and management system, which is based on the framework of temporal logic approach. Namely, we consider multi-versions of an ontology as a sequence of ontologies which are connected with each other via change operations. A temporal logic is developed to be an extended query language for reasoning about temporal aspects of ontologies.

In this document, we introduce MORE2, an extended reasoning and management system for multi-version ontologies. MORE2 is a new prototype of MORE with the new functionalities, which include full support of ontology versioning with RDF/RDFS data. MORE2 is integrated with KAON2, a powerful ontology reasoning tool which is developed in SEKT WP3.

In this document, we discuss the implementation issues of MORE2. Furthermore, we report the experiments of MORE2 with several realistic data sets: SEKT legal ontologies and the Dutch general election data 2006.

Contents

Chapter 1

Introduction

It is natural for knowledge engineers to change ontologies in order to update them. When an ontology is changed, knowledge engineers may want to keep only the latest version of the ontology. After many changes have been done to the ontology, knowledge engineers might think that they have committed mistakes regarding some of those modifications, because there exist some unintended properties on the ontology. In this scenario, knowledge engineers usually want to withdraw or adjust the changes to avoid unintended properties. A natural solution is to resume the previous versions of an ontology where an unintended change occurs. That is one of the motivation for ontology versioning. In [HS05a], we provide a detailed discussion of the motivation for ontology versioning.

In [HS05a], we develop a framework of multi-version ontology reasoning and management systems, based on a temporal logic approach. In this approach, we consider multi-versions of an ontology as a sequence of ontologies which are connected with each other via change operations. Based this temporal logic approach, we have developed and implemented the prototype of MORE, which stands for Multiversion Ontology REasoner. MORE allows users to make temporal queries and version retrieval queries on multi-version ontologies. MORE can provide the details of the semantic differences which result from ontology changes, by showing new/obsolete/invariant concept/instance relations among multiple versions of an ontology. MORE supports multiple ontology data formats, which include OWL and DIG.

In this document, we introduce MORE2, an extended reasoning and management system for multi-version ontologies. MORE2 is a new prototype of MORE with the new functionalities. The main functionalities of MORE2 are : a) the full support of ontology versioning with RDF/RDFS data, b) integration of MORE2 with KAON2, a powerful ontology reasoning tool which is developed in SEKT WP3, and c) multiple platform support: Windows/Unix/Linux/Mac. This document provides a detailed reference/manual of MORE2. In this document, we also discuss the implementation issues of MORE2.

One of the three use cases of the SEKT project [BCB+04, CCLCL05, CPC+05b, CPC+05a] is a legal case study, in which an ontology is used to map questions of junior judges to a set of predefined frequently asked questions and their answers by experienced legal experts. In the SEKT deliverable D3.5.2 [HSvH+06], we provide a detailed quantitative evaluation of MORE with ontology data of SEKT legal case studies. In this document, we will continue to report preliminary experiments of SEKT legal ontologies, however, with the qualitative evaluation. More exactly we will provide the guidelines of using the temporal reasoning system by showing several typical query scenarios on SEKT legal ontologies. In this document, we will also report the preliminary experiment of MORE2 with the Dutch general election data 2006, as the second case study of MORE2. By which, we want to show how MORE2 can be used in different application scenarios of ontology versioning in the social science context.

This document is organized as follows: Chapter 2 is an overview on the system MORE, Chapter 3 shows the new functionalities and extensions of MORE2. It also provides the detailed manual of MORE2. Chapter 4 reports the experiment of MORE2 with the SEKT legal ontologies, and provide the qualitative evaluation of SEKT legal ontologies. Chapter 5 reports the preliminary experiment of the 2006 Dutch general election data with MORE2. Chapter 6 discusses further work, and concludes the document.

Chapter 2

MORE: a Multi-version Ontology Reasoning System

In this chapter, we provide a brief overview of the framework of multi-version ontology reasoning and management system which is based on a temporal logic approach. A detailed discussion of the temporal logic approach for ontology versioning can be found in [HS05a, HS05b].

2.1 Version Space and Temporal Logics

MORE is a multi-version ontology reasoning system, which is based on a temporal logic approach [HS05a, HS05b]. In this approach, multi-versions of an ontology are considered as a sequence of ontologies which are connected each other via change operations. Each of these ontologies has a unique name. Thus, a version space S over an ontology set Os is a set of ontology pairs, namely, $S \subseteq Os \times Os$. We use version spaces as a semantic model for our temporal logic, restricting our investigation to version spaces that present a linear sequence of ontologies. A linear version space S on an ontology set Os is denoted as a finite sequence S of ontologies as $S = (o_1, o_2, \cdots, o_n)$. We use $S(i)$ to refer the i_th ontology o_i in the space. For a version space $S = (o_1, o_2, \cdots, o_n)$, We call the first ontology $S(1)$ in the space the *initial version of the version space*, and the last ontology $S(n)$ the *latest version of the version space* respectively. An ordering $<$ with respect to a version space S is introduced as $o < o'$ iff o occurs prior to o' in the sequence S. We use $ontology(S)$ to denote the ontology set $Os = \{o_1, \cdots, o_n\}$ of the version space S.

A temporal logic has been developed in MORE for Multi-version Reasoning [HS05a, HS05b]. The Language $\mathcal{L}+$ of the temporal logic $\mathbf{LTLm}$ is defined as an extension to the ontology language $\mathcal{L}$ with Boolean operators and the backward temporal operators, which include the previous version operator $\mathbf{Prev}\phi$ which denotes that the property ϕ holds in the previous version (with respect to the current version in the version space), the always-in-past operator $\mathbf{H}\phi$ which denotes that the property ϕ always holds in any version before the current version, and the since operator $\phi\mathbf{S}\psi$ which denoted that the property ϕ always holds (till the current version) since the property ψ holds in a version before the current version. The sometimes-in-the-past operator $\mathbf{P}\phi$ is defined in terms of the always-in-past operator as $\neg\mathbf{H}\neg\phi$. In the temporal logic, the evaluation of a temporal formula ϕ on an ontology o (i.e., a version) in a version space S is defined as an entailment relation [HS05a, HS05b]:

$$S, o \models \phi$$

The semantics of the temporal operators is illustrated in Figure 2.1, where arrows denote the sequence relation of ontologies in the version space, and a formula under an ontology denotes that the formula holds on the ontology. For example, the first line in the figure shows that if $\mathbf{Prev}\phi$ holds on an ontology iff the formula ϕ hold on its previous ontology.

The temporal logic can be extended to include the future-oriented temporal operators like those in standard temporal logics [vB95, Bul70, HS91], action operators like those in dynamic logics [Har84, PT91, HTK00], and hybrid operators like those in hybrid logics [BT99, Bla00, FdRS03, AB01]. The hybrid logic extension of MORE will be discussed in the next section.

2.2 Hybrid Logic Extension

Hybrid logics are considered as the extensions to temporal logics. Unlike temporal logic, hybrid logic makes it possible to refer to states (possible worlds) in formulas. This is achieved by a class of formulas called nominals, which are true in exactly one state. In hybrid logics, a nominal is used to refer to an identification of a state. In the temporal models of ontology versioning, a nominal is an identification of a version, i.e., a possible world in the models.

A Hybrid Logic Language $\mathcal{HL}$ is one which introduces nominals directly in its language [BT99, Bla00, FdRS03, BS95, AB01].

There exist variants of hybrid logics, which may contain one or some of the following operators:

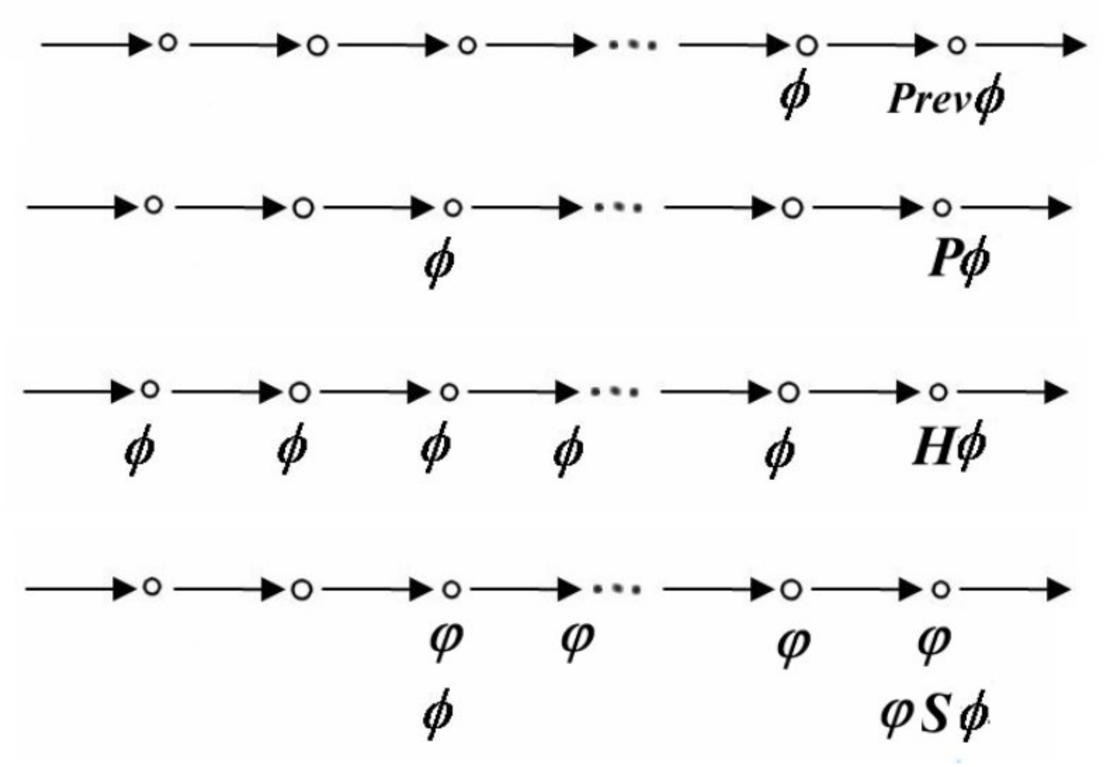

ϕ
$Prev\phi$
ϕ
$P\phi$
ϕ ϕ ϕ ϕ ϕ
$H\phi$
φ φ φ φ
ϕ
$\varphi S \phi$

Figure 2.1: Semantics of Temporal Operators

- **the at-operator** @: it is used to evaluate a formula at a state i, for example, $@_i\phi$ which means that ϕ holds at the state i.
- **the down arrow operator** $\downarrow$: it is used to make a binding of a variable with the current state.
- **the existence quantifier** $\exists$: it is used to express the existence quantifier on states.

The langauge $\mathcal{HL}$ with all of the hybrid logic operators above is defined as an extension to the temporal logic language $\mathbf{LTLm}$ on a primitive proposition set $\mathcal{P}$, a nominal set $\mathcal{N}$, and a variable set $\mathcal{V}$ with Boolean operators, the temporal operators, and the hybrid modal operators as follows:

- $q \in \mathcal{L} \Rightarrow q \in \mathcal{HL}$
- $\phi \in \mathcal{HL} \Rightarrow \neg\phi \in \mathcal{HL}$

- $\phi, \psi \in \mathcal{HL} \Rightarrow \phi \wedge \psi \in \mathcal{HL}$
- $\phi \in \mathcal{HL} \Rightarrow \mathbf{Prev}\,\phi \in \mathcal{HL}$
- $\phi \in \mathcal{HL} \Rightarrow \mathbf{H}\,\phi \in \mathcal{HL}$
- $\phi, \psi \in \mathcal{HL} \Rightarrow \phi\,\mathbf{S}\,\psi \in \mathcal{HL}$
- $i \in \mathcal{N} \cup \mathcal{V} \Rightarrow i \in \mathcal{HL}$
- $i \in \mathcal{N} \cup \mathcal{V}, \phi \in \mathcal{HL} \Rightarrow @_i\phi \in \mathcal{HL}$
- $x \in \mathcal{V}, \phi \in \mathcal{HL} \Rightarrow \downarrow x\phi \in \mathcal{HL}$

A variable assignment g is a mapping $g : \mathcal{V} \to O(S)$ which assigns each variable x an ontology o in the version space S. We use the notation $g \sim_x g'$ to denote that g' is an assignment of values to variables that agrees with g on all arguments save possibly x.

$S, g, o \models q$	iff	$t \models q, for\ q \in \mathcal{L}.$
$S, g, o \models \neg\phi$	iff	$S, g, o \not\models \phi.$
$S, g, o \models \phi \wedge \psi$	iff	$S, g, o \models \phi, \psi.$
$S, g, o \models \mathbf{Prev}\,\phi$	iff	$\exists\langle o', o\rangle$ and $S, g, o' \models \phi.$
$S, g, o \models \mathbf{H}\,\phi$	iff	for any o' such that $o' < o, S, g, o' \models \phi.$
$S, g, o \models \phi\mathbf{S}\psi$	iff	$\exists(o_1 \ldots o_i)(\langle o_1, o_2\rangle, \ldots, \langle o_{i-1}, o_i\rangle \in S$ and $o_i = o)$ such that$S, g, o_j \models \phi$ for $1 \leq j \leq i$ and $S, g, o_1 \models \psi.$
$S, g, o \models o$	iff	$for\ o \in \mathcal{N}.$
$S, g, o \models @_i\phi$	iff	$S, g, i \models \phi$
$S, g, o \models \downarrow x.\phi$	iff	$S, g', o \models \phi,$ where $g \sim_x g' \& g'(x) = o$

Furthermore, we define $S, o \models \phi$ as $S, g, o \models \phi$ for any g.

The following is an example of a query which uses hybrid logic operator. The query asks if the version i is the last version from which a fact ϕ is derivable as the same as the current version. Under this scenario, the fact ϕ holds in the current version x. If the fact ϕ holds in the previous version, then i must be the previous version, as shown in Figure 2.2(a). If the fact ϕ does not hold in the previous version, then i must be the first one in the prior versions from which the fact ϕ holds, as shown in Figure 2.2(b).

This query can be expressed as:

$$((\mathbf{Prev}\phi \to \mathbf{Prev}i) \wedge (\neg\mathbf{Prev}\phi \to \mathbf{Prev}(\neg\phi\mathbf{S}\mathbf{Prev}(\phi \wedge i)))).$$

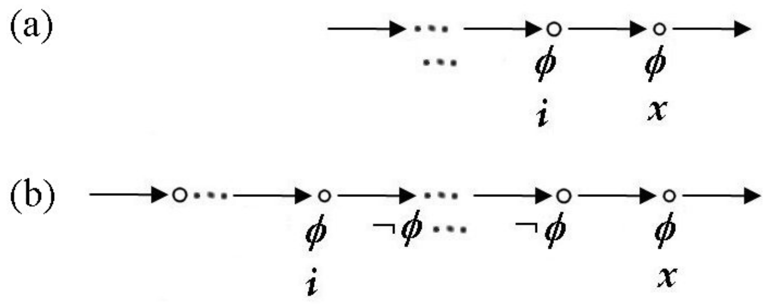


Figure 2.2: Query on Last Version

The main advantage of hybrid logic operators over ordinary temporal logics is that it supports nominal in temporal formulas. However, note that the temporal logic language $\mathbf{LTLm}$ and its extension, Hybrid Logic Language $\mathcal{HL}$, are designed as the query languages for multi-version ontology reasoning in MORE. Those temporal queries consist of two parts: the part corresponds with a temporal formula, and another part states which version space and ontology are targeted by the temporal formula. Usually, a query in MORE has the following form: $S, o \models \phi$. Using hybrid logic operators, the targeted ontology can appear in temporal formula, like this $S \models \phi(o)$. It seems that it does not make any big difference from standard temporal queries. However, the hybrid logic operators provide a variant of temporal reasoning for multiple version ontology reasoning. It will be a different approach from the standard temporal logic approach when the hybrid logics serve as a specification language rather than a query language. That will be one of the future researches of MORE.

Chapter 3

Prototype of MORE2

We have implemented the prototype of MORE2, as an extension of MORE. MORE is implemented by using Prolog. MORE is powered by the XDIG interface [HV04], an extended DIG description logic interface for Prolog[1]. MORE is designed to be a simple API for a general reasoner with multi-version ontologies. It supports extended DIG requests from other ontology applications or other ontology and metadata management systems and supports multiple ontology languages, including OWL and DIG [BMC03][2]. This means that MORE can be used as an interface to any description logic reasoner as it supports the functionality of the underlying reasoner by just passing requests on and provides reasoning functionalities across versions if needed.

3.1 Functionalities of MORE

In SEKT Deliverable D3.5.1 [HS05a] and SEKT Deliverable D3.5.2 [HSvH$^+$06], we have reported that the prototype of MORE (version 1.0 and version 1.5) have the following functionalities:

- **Temporal Reasoning Queries**: supports the temporal logic **LTLm**.
- **Ontology Comparison Queries**: supports new/obsolete/invariant concept/role/instance queries.
- **Versioning Queries** : supports version retrieval, relative version numbering, and absolute version numbering.

[1] http://wasp.cs.vu.nl/sekt/dig

[2] http://dl.kr.org/dig/

- **Ontology Languages**: supports the DIG/OWL/RDF/RDFS data format.

In this document, we would like to report the new functionalities of the prototype of MORE2. It has the following new features:

- **Integration with KAON2**: MORE2 has been integrated with KAON2, a reasoning tool which is developed in SEKT WP3.
- **Hybrid Logic Extension**: support hybrid logic operators.
- **Multiple Platform**: supports the Windows/Unix/Linux/Mac platforms.
- **Composite Concept Queries**: supports instance queries and ontology comparison queries on composite concepts for RDF/RDFS data.

3.2 Installation and Test Guide

1. **Download**: The MORE2 package is available from the MORE website:

   `http://wasp.cs.vu.nl/sekt/more/more2.zip`

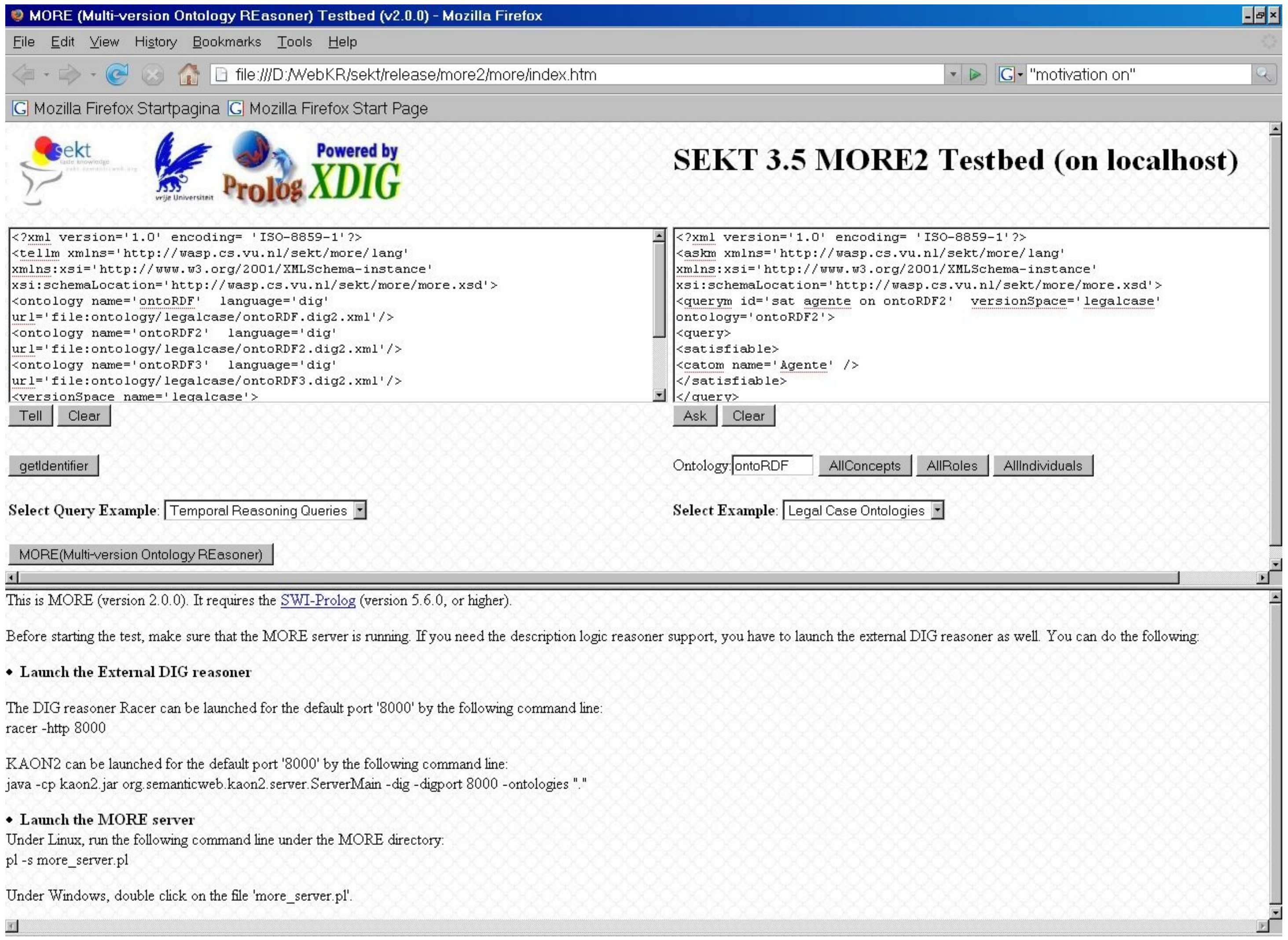
   Unzip the MORE package into a directory.

2. **Installation of SWI-Prolog**: MORE2 requires that SWI-Prolog (version 5.6.0 or higher) has been installed on your computers. It can be downloaded from the SWI-Prolog website:

   `http://www.swi-prolog.org`

3. **Installation of External DL Reasoner**: MORE requires an external DL reasoner which supports the DIG interface, like RACER and KAON2, for reasoning with OWL/DIG data. MORE2 needs no external DL reasoners for reasoning with RDF/RDFS data. KAON2 can be downloaded from the KAON website:

   `http://kaon2.semanticweb.org/`


MORE (Multi-version Ontology REasoner) Testbed (v2.0.0) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///D:/WebKR/sekt/release/more2/more/index.htm
"motivation on"
Mozilla Firefox Startpagina
Mozilla Firefox Start Page
Powered by XDIG
Tell
Clear
Ask
Clear
getIdentifier
AllConcepts
AllRoles
AllIndividuals
MORE(Multi-version Ontology REasoner)

SEKT 3.5 MORE2 Testbed (on localhost)

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<tellm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
<ontology name='ontoRDF'  language='dig'
url='file:ontology/legalcase/ontoRDF.dig2.xml'/>
<ontology name='ontoRDF2'  language='dig'
url='file:ontology/legalcase/ontoRDF2.dig2.xml'/>
<ontology name='ontoRDF3'  language='dig'
url='file:ontology/legalcase/ontoRDF3.dig2.xml'/>
<versionSpace name='legalcase'>
```

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
<querym id='sat agente on ontoRDF2'  versionSpace='legalcase'
ontology='ontoRDF2'>
<query>
<satisfiable>
<catom name='Agente' />
</satisfiable>
</query>
```

Ontology: ontoRDF

Select Query Example: Temporal Reasoning Queries

Select Example: Legal Case Ontologies

This is MORE (version 2.0.0). It requires the SWI-Prolog (version 5.6.0, or higher).

Before starting the test, make sure that the MORE server is running. If you need the description logic reasoner support, you have to launch the external DIG reasoner as well. You can do the following:

- **Launch the External DIG reasoner**

The DIG reasoner Racer can be launched for the default port '8000' by the following command line:
racer -http 8000

KAON2 can be launched for the default port '8000' by the following command line:
java -cp kaon2.jar org.semanticweb.kaon2.server.ServerMain -dig -digport 8000 -ontologies "."

- **Launch the MORE server**

Under Linux, run the following command line under the MORE directory:
pl -s more_server.pl

Under Windows, double click on the file 'more_server.pl'.

Figure 3.1: MORE testbed.

The MORE2 testbed 'more_test.htm' is a MORE2 client with a graphical interface, which is designed as a webpage. Therefore it can be launched from a web browser which supports Javascript. A screenshot of the MORE2 testbed, is shown in Figure 3.1.

The current version of the MORE testbed supports tests for which both the MORE server (with default port: 8003) and the external DL reasoner (with the default port: 8000) are running on the localhost. The default hostname of the MORE server and the external DL reasoner is 'localhost'. For a MORE server which runs on a remote host, change the host and port data in the 'moremain.htm' file. Namely, replace the URL 'http://127.0.0.1:8003' with another valid URL.

Before starting the MORE2 test, make sure that the MORE server and the external DL reasoner are running at the host with the correct ports.

1. **Launch DL Reasoner**: Racer can be launched by the following command:

   ```
   racer -http 8000
   ```

   Alternatively, click on the file 'racer8000.bat' for Windows users if the MORE package includes the Racer reasoner.

   KAON2 can be launched by the following command:

   ```
   java -cp kaon2.jar  org.semanticweb.kaon2.server.ServerMain
          -dig -digport 8000 -ontologies "."
   ```

2. **Launch MORE server**: click on the file '*more_server.pl*' in the MORE directory. If you encounter the global stack limit problem because of a big amount of test data, you should increase the size of the global stack. The windows users can edit the path setting of 'plwin.exe' in the file 'moreserver_bigGlobalStack.bat', then launch it.

The TELLm and ASKm request data can be copied into the TELL text area and the ASK text area respectively. After that, you can click on the buttons 'Tell' and 'Ask' to make the corresponding requests. The request data can also be posted from any application without using the MORE testbed. That is useful if the test data exceeds the text area limit.

3.3 Query Interface

A detailed query interface of MORE can be found in SEKT deliverable D3.5.1 [HS05a]. The appendix of this document is the summary of the ontology comparison query interface.

In the following subsections, we provide the detailed query interface of the new functionalities of MORE2.

3.3.1 Composite Concept Queries

In the previous release of MORE, it supports only instance queries with named concepts, such as:

```
<?xml version="1.0" encoding= "ISO-8859-1" ?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="new instances of the concept Annotation of the ontology
           election1012 compared with the ontology election1011"
            versionSpace="election"  ontology="election1012">
  <newOnIndividual  concept="Annotation"
    comparedOntology="election1011"/>
</querym>
</askm>
```

Which asks for the new instances of the concept 'Annotation' of the ontology '1012', compared with those in the ontology '1011'. A named concept 'Annotation' is specified in this kind of queries. To extend to it, MORE2 supports not only instance queries on named concepts, but also on composite concepts for RDF/RDFS data[3]. Here is a query example of instance queries of composite concepts:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="new Annotation with subject pvdacampagne"
     versionSpace="election" ontology="election1007">
<newOnIndividual>
   <instancesRDFS>
      <catom name="http://www.content-analysis.org/vocabulary/net#Annotation"/>
```

[3]Support for instance queries of composite concepts with OWL/DIG data will be done in the future release of MORE

Type	Syntax	Semantics
Named Concept	<catom name=C />	$\{I \vert S, o \models C(I)\}$
Role With Individual	<hasRoleWithIndividual role=R individual=I'/>	$\{I \vert S, o \models rdf(I, R, I')\}$
Role With Literal	<hasRoleWithLiteral role=R literal=I'/>	$\{I \vert S, o \models rdf(I, R, literal(type(\_, I')))\}$[4]
Role With Anything	<hasRoleWithAnything role=R />	$\{I \vert S, o \models rdf(I, R, \_I')\}$
Named Role	<ratom name=R> C <ratom/>	$\{I \vert S, o \models rdf(I, R, I') \wedge C(I')\}$

Figure 3.2: Composite Concepts on instanceRDFS

```
        <hasRoleWithIndividual
           role="http://www.content-analysis.org/vocabulary/net#Subject"
            individual="http://www.content-analysis.org/vocabulary/
                 ontologies/k06#kenobj-61821-pvdacampagne"/>
   </instancesRDFS>
</newOnIndividual>
</querym>
</askm>
```

Which asks for the new instances of the concept 'Annotation' with the subject 'pvdacampagne' of the ontology 'election1007'. In this query, the tag 'instancesRDFS' is used to contain a composite concept which is defined in the body of the tag element. The semantics of composite concepts are defined as the conjunction of the interpreation of its elements in the body. We use $rdf(X, Y, Z)$ to denote $\langle X, Y, Z\rangle$ is a RDF triple in the ontology and use $C(I)$ to denote that the individual I is an instance of the concept C. The evaluation of the tag 'instancesRDFS' is to check whether or not an individual I is an instance of the composite concept which is defined in the body of the tag 'instancesRDFS' in an ontology o of the version space S. Elements in the body of the tag 'instancesRDFS' can be a named concept statement, a named role statement, a 'hasRoleWithIndividual' statement, a 'hasRoleWithIndividual' statement, or a 'hasRoleWithAnything' statement. The semantics of these elements is defined in Figure 3.2.

As an example, the following statement defines a composite concept, more exactly, a set of individuals

$$\{I|S, o \models C1(I) \wedge rdf(I, R1, I1) \wedge rdf(I, R2, I2) \wedge rdf(I2, R3, I') \wedge C2(I')\}$$

```
   <instancesRDFS>
```

```
      <catom name=C1/>
      <hasRoleWithIndividual role=R1 individual=I1/>
      <ratom name=R2>
        <ratom name=R3>
          <catom name=C2/>
        </ratom>
      </ratom>
   </instancesRDFS>
```

The following query is an example of an instance query on composite concepts:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="Annotation with subject pvdacampagne"
       versionSpace="election" ontology="election1002">
<query>
     <instancesRDFS>
      <catom name="http://www.content-analysis.org/vocabulary/net#Annotation"/>
      <hasRoleWithIndividual
          role="http://www.content-analysis.org/vocabulary/net#Subject"
          individual="http://www.content-analysis.org/vocabulary/
                  ontologies/k06#kenobj-61821-pvdacampagne"/>
     </instancesRDFS>
</query>
</querym>
</askm>
```

This query asks for the instance set of the composite concept on the ontology, i.e., $\{I|election, election1002 \models Annotation(I) \wedge rdf(I, net : Subject, k06 : kenobj\text{-}61821\text{-}pvdacampagne)\}$[5].

The following is an example which queries on the annotations which have the role 'net:Quality' with the literal '-50':

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http:// www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http:// wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="Annotation with the quality literal -50"
       versionSpace="election" ontology="election1002">
<query>
```

[5]Where we use the namespaces net and k06.

```
   <instancesRDFS>
     <catom name="http://www.content-analysis.org/
         vocabulary/net#Annotation"/>
     <hasRoleWithLiteral
        role="http://www.content-analysis.org/vocabulary/net#Quality"
        literal="-50"/>
   </instancesRDFS>
</query>
</querym>
</askm>
```

If we do not care for any particular value or a particular individual as in the example above, we can use an instance query of 'hasRoleWithAnything' like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http:// www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http:// wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="Annotation with any quality"
       versionSpace="election" ontology="election1002">
<query>
   <instancesRDFS>
     <catom name="http://www.content-analysis.org/
         vocabulary/net#Annotation"/>
     <hasRoleWithAnything
        role="http://www.content-analysis.org/
           vocabulary/net#Quality"/>
   </instancesRDFS>
</query>
</querym>
</askm>
```

3.3.2 Queries with Hybrid Logic Operators

The interface for hybrid logic operators is shown in Figure 3.3.

We can use the at operator in hybrid logics to design a formula which states that it will be evaluated at a particular nominal, i.e, a version, like this:

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
<querym id='what are new on child concept of Agente
```

Type	Syntax	Semantics
Nominal	<nominal value=o />	$S, o \models o$
Variable	<variable value=x />	$S, o \models x$
At Operator	<at nominal=i > ϕ </at>	$S, o \models @_i\phi$
Downarrow Operator	<downarrow variable=X ϕ </downarrow>	$S, o \models \downarrow x.\phi$

Figure 3.3: Hybrid Logic Interface

```
at ontoRDF3 compared with ontoRDF1?' versionSpace='legalcase'>
  <at nominal='ontoRDF3'>
    <newOnConcept  concept='Agente' type='children'
       comparedOntology='ontoRDF'/>
   </at>
</querym>
</askm>
```

The query above asks for the new children concept relations of 'Agente' at the nominal 'ontoRDF3', compared with those in the ontology 'ontoRDF'.

In the MORE interface, we know that if a query does not state any ontology then the corresponding formula will be evaluated with respect to the last ontology in the version space by default. Thus, we can use the nominal operator to make a query to check if an ontology is the last version in the version space, like this:

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
<querym id='is ontoRDF3 the last version of this version space?'
  versionSpace='legalcase'>
   <nominal value='ontoRDF3'/>
</querym>
</askm>
```

Chapter 4

Experiment I: SEKT Legal Ontologies

The aim of SEKT is to develop and exploit semantically-based knowledge technologies in order to support document management, content management, and knowledge management in knowledge intensive workplaces. Specifically, SEKT aims at designing appropriate utilities to users in three main areas: digital libraries, the engineering industry, and the legal domain, providing them with quick access to the right pieces of information at the right time. In this chapter, we report on the ontologies of the legal case study of the SEKT project and their versioning. We will first briefly introduce the legal case study, before describing the relevant version space for the OPJK ontology in more detail.

4.1 The SEKT legal case study

More details regarding the legal case study can be found in SEKT deliverable D10.2.1 [CPC$^+$05b, BPOC$^+$05, CCV$^+$07].

In the legal case study, the accomplished tasks so far provide both the quantitative and qualitative data necessary to assess both the context of users, (newly recruited Spanish judges), and their specific needs with regard to the technology under development. In particular, these data give an insight on institutional, organizational, and individual constraints that could either facilitate or block the introduction of SEKT technologies in judicial units. From this set of data, the Ontology of Professional Judicial Knowledge is built.

Until now, the legal ontologies have been built up with several purposes: information retrieval, statute retrieval, normative linking, knowledge management or legal reasoning. Although the legal domain remains very sensitive to the features of regional or national statutes and regulations, some of the Legal-Core Ontologies (LCO) are intended to share a common kernel of legal notions. However, LCO remain in the domain of a general knowledge shared by legal theorists, national or international jurists and comparative lawyers. This use case study fed on the idea of the existence of a Professional Legal Knowledge (PLK) which is: (i) shared among members of the judicial professional group (e.g. judges, attorneys, prosecutors...); (ii) learned and conveyed formally or most often informally in specific settings (e.g. the Judicial School, professional associations -the Bar, the Judiciary...-); (iii) expressible through a mixture of natural and technical language (legalese, legal slang); (iv) non-equally distributed among the professional group; (v) non-homogeneous (elaborated on individual bases); (vi) universally comprehensible by the members of the profession (there is a sort of implicit identification principle).

The data (nearly 800 competency questions) gathered from judicial interviews by the legal case study, allows the construction of an ontology based on professional judicial knowledge. Knowledge derived from the daily practice at courts. The Ontology of Professional Judicial Knowledge (OPJK) developed by the legal case study team has been learnt from scratch out of nearly 400 competency questions and has, currently, nearly 104 concepts, 100 relations and 561 instances. The following top domain concepts have been identified: $acto\_procesal$, $organo\_judicial$, $calificacion\_jurdica$, $documento\_procesal$, $fase\_procesal$, $jurisdiccion$, $proceso\_judicial$, $profesion\_jurdica$, $rol\_procesal$, $rol\_familiar$ and $sancion$. In this deliverable, we will use the ontolgy OPJK as the test data to evaluate the system of ontology versioning and management.

The Legal Case Study Prototype [CCV$^+$06] has been designed taking into account two main considerations: (i) an accurate searching system, with advanced technology, that goes beyond traditional searching algorithms, capable of reliable search over a vast FAQ repository; (ii) a design that supports a fast, usable, modular, extensible, scalable, improving implementation. The first point might be achieved by using some techniques like ontologies to model legal case domains and NLP techniques. The second point is achieved by leveraging on some software technology patterns, like a multistage searching cycle for successive approach to FAQ pair target or pluggable searching stage engines. The final design is flexible, modular, scalable, customizable and suitable for the prototype.

4.2 The OPJK versioning space

The OPJK ontology lies at the core of the legal case study within SEKT, as it links natural language questions (formulated by newly recruited Spanish judges) with a repository of frequently asked questions (FAQs) with their corresponding answers provided by more experiences judges. To this aim, OPJK contains definitions for the most relevant judicial terms in the professional knowledge of law professionals with respect to this set of FAQs.

As this is highly specialized knowledge producing a new version of the OPJK is a difficult task. In practice, a team of legal experts meets in regular intervals to decide on the relevant changes to the OPJK where the decision is taken with respect to the questions in the FAQs.

The OPJK versioning space has been built to reflect on this structured way of dealing with the frequently asked questions, and in principle, it is a linear space with usually one new version per FAQ. Not only is this a direct mirror of the creation process, it also constitutes an opportunity to analyze the epistemic process in a systematic way. Technically, we collected 27 versions of the OPJK up to now.

4.3 Analysis of the OPJK versioning space

We can now analyze the OPJK versioning space with two different motivations: first, as a well-constructed (and documented) versioning space of a complex ontology, and secondly, as a formalization of an epistemological process, in which knowledge elicitation is made explicit in ontological changes.

In SEKT deliverable D3.5.2 [HSvH$^+$06], we addressed only the first issue, namely, the quantitative evaluation of OPJK versioning. In this document, we address the second issue, namely, the qualitative evaluation of OPJK versioning. More concretely, we will analyze the OPJK versioning space from two perspectives. First, we will study measures on the temporal dimension of the version space, i.e. search whether we can detect peculiarities in the process at a whole. Secondly, we will study measures on the concept level of the version space, i.e. investigate the ramification of changing a single concept.

In the following subsections, we will show several scenarios how these kinds of questions can be answered. These scenarios can be considered as use cases of MORE for better understanding the dynamic aspects of SEKT legal ontologies. The description of these scenarios can be considered as concrete examples and guidelines how the system MORE can be used. In the following, queries and their answers are mainly shown by using the interface language instead of logical formulas, so that it is much easier for users to copy similar queries to test their own data.

4.3.1 Scenario One: Data Examination

It is very useful for knowledge engineers and users to examine data to see how those data are changed dynamically and temporally in a version space. For example, we may want to know how many facts (Hecho) are claimed in a particular version by posting an instances query, like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns=http://wasp.cs.vu.nl/sekt/more/lang
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="instances of Hecho"
   versionSpace="legalcase2" ontology="opjk27">
<query>
     <instances>
      <catom name='http://protege.stanford.edu/kb#Hecho'/>
     </instances>
     </query>
</querym>
</askm>
```

The answer to this query is a list of instances of the concept 'Hecho', like this:

```
<response>
<answer id="instances of Hecho">
  <individualSet>
     <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30011"/>
     <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30121"/>
     <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30128"/>
     <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30129"/>
       ......
     <individual name="http://protege.stanford.edu/kb#oplk_Instance_32"/>
     <individual name="http://protege.stanford.edu/kb#oplk_Instance_33"/>
  </individualSet>
</answer>
</response>
```

We use an instance query of hasRoleWithAnything to obtain the instances of the concept $Acto\_Procesal$ on opjk27. Those instances are required to have the relation with $realizado\_por$. The corresponding query is like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http:// www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="Instances of Acto Procesal with realizado por"
versionSpace="legalcase2" ontology='opjk27'>
  <query>
    <instancesRDFS>
      <catom name='http://protege.stanford.edu/kb#Acto_Procesal'/>
      <hasRoleWithAnything role='http://protege.stanford.edu/
         kb#realizado_por'/>
    </instancesRDFS>
```

```
   </query>
</querym>
</askm>
```

The answer to this query shows a list of the instances:

```
<response>
<answer id="Instances of Acto Procesal with realizado por">
  <individualSet>
  <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_10001"/>
  <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30032"/>
  <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30045"/>
  <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30046"/>
   ......
  <individual name="http://protege.stanford.edu/kb#ontoRDF2_Instance_30136"/>
  <individual name="http://protege.stanford.edu/kb#onto_Instance_10036"/>
  <individual name="http://protege.stanford.edu/kb#onto_Instance_10037"/>
  <individual name="http://protege.stanford.edu/kb#onto_Instance_10043"/>
  <individual name="http://protege.stanford.edu/kb#onto_Instance_10047"/>
  <individual name="http://protege.stanford.edu/kb#onto_Instance_10057"/>
  <individual name="http://protege.stanford.edu/kb#opjk_Instance_5"/>
  <individual name="http://protege.stanford.edu/kb#opjk_new_v0.2_Instance_10007"/>
 <individual name="http://protege.stanford.edu/kb#opjk_new_v0.2_Instance_10027"/>
  <individual name="http://protege.stanford.edu/kb#opjk_new_v0.2_Instance_10061"/>
  <individual name="http://protege.stanford.edu/kb#oplk_Instance_12"/>
  <individual name="http://protege.stanford.edu/kb#oplk_Instance_6"/>
  </individualSet>
 </answer>
</response>
```

From the OPJK instance naming convention, we know that most instances in this list were created in the very beginning of the OPJK ontology (ontoRDF2), and there are only few instances which have been created on later OPJK versions

We can use the query of instanceRDFS to check a concrete instance relation, like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http:// www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="is ontoRDF2_Instance_10001
   an instance  of Acto Procesal with realizado por"
   versionSpace="legalcase2" ontology='opjk27'>
  <query>
    <instanceRDFS>
     <individual name="http://protege.stanford.edu/
```

```
              kb#ontoRDF2_Instance_10001"/>
      <catom name='http://protege.stanford.edu/
              kb#Acto_Procesal'/>
      <hasRoleWithAnything role='http://protege.stanford.edu/
            kb#realizado_por'/>
   </instanceRDFS>
  </query>
 </querym>
</askm>
```

The answer to this query is positive, like this:

```
<response>
   <true id="is ontoRDF2_Instance_10001
        an instance  of Acto Procesal with realizado por"/>
</response>
```

4.3.2 Scenario Two: Concept Addition and its Ramification

In version 'opjk27', the concepts $NegocioJuridico$ and $Infraccion\_Penal$ are added as subconcepts of $AbstraccionJuridica$, and the concepts $Delito$ and $Falta$ are created to be two subconcepts of the concept $Infraccion\_Penal$. We want to know what the ramification of those changes is. Namely, we want to know what new/obsolete concept relations are in the effect space by the following queries:

```
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
 <querym  id='new concept relations at opjk27'
     versionSpace='legalcase2' ontology='opjk27'>
  <newOnConcept/>
 </querym>
<querym  id=' obsolete concept relations at opjk27'
   versionSpace='legalcase2' ontology='opjk27'>
  <obsoleteOnConcept/>
</querym>
</askm>
```

The answers to those queries show that there are no obsolete concept relations from those changes, however, there are many new concept relations are introduced by them, as shown in the following:

```
<response>
  <answer id="new concept relations at opjk27">
    <conceptRelated concept="kb:AbstraccinJurdica">
     <children>
       <catom name="kb:Infraccin_Penal"/>
       <catom name="kb:Negocio_Jurdico"/>
     </children>
</conceptRelated>
<conceptRelated concept="kb:Delito">
  <parents>
    <catom name="kb:Infraccin_Penal"/>
  </parents>
  <ancestors>
    <catom name="kb:Infraccin_Penal"/>
    <catom name="kb:AbstraccinJurdica"/>
    <catom name="kb:Abstraccin"/>
    <catom name="kb:Entidad"/>
  </ancestors>
</conceptRelated>
<conceptRelated concept="kb:Falta">
 <parents>
   <catom name="kb:Infraccin_Penal"/>
</parents>
  <ancestors>
    <catom name="kb:Infraccin_Penal"/>
    <catom name="kb:AbstraccinJurdica"/>
    <catom name="kb:Abstraccin"/>
    <catom name="kb:Entidad"/>
   </ancestors>
</conceptRelated>
<conceptRelated concept="kb:Infraccin_Penal">
  <children>
    <catom name="kb:Delito"/>
    <catom name="kb:Falta"/>
   </children>
  <parents>
   <catom name="kb:AbstraccinJurdica"/>
  </parents>
  <ancestors>
    <catom name="kb:AbstraccinJurdica"/>
    <catom name="kb:Abstraccin"/>
    <catom name="kb:Entidad"/>
  </ancestors>
  <descendants>
    <catom name="kb:Delito"/>
  </descendants>
</conceptRelated>
<conceptRelated concept="kb:Negocio_Jurdico">
```

```
  <parents>
    <catom name="kb:AbstraccinJurdica"/>
 </parents>
 <ancestors>
   <catom name="kb:AbstraccinJurdica"/>
   <catom name="kb:Abstraccin"/>
   <catom name="kb:Entidad"/>
</ancestors>
</conceptRelated>
  <conceptRelated concept="kb:Rol">
    <children>
     <catom name="kb:Rol_Negocio_Jurdico"/>
    </children>
</conceptRelated>
 <conceptRelated concept="kb:Rol_Negocio_Jurdico">
 <parents>
<catom name="kb:Rol"/>
</parents>
 <ancestors>
  <catom name="kb:Rol"/>
  <catom name="kb:Situacin"/>
  <catom name="kb:Suceso"/>
  <catom name="kb:Entidad"/>
  </ancestors>
</conceptRelated>
</answer>
<answer id="obsolete concept relations at opjk27">
  </answer>
</response>
```

For the answers above, we observe the fact that all of the new concept relations are always involved a newly created concepts ($NegocioJuridico$, $Infraccin\_Penal$, $Delito$, $Falta$, or $Rol\_Negocio\_Jurdico$). That means that there exist no unintended impact by those changes.

4.3.3 Scenario Three: Temporal Change Checking

It is useful for users to know what changes have been done by knowledge engineers on a particular version. For example, the following query asks which obsolete instance relations are within opjk25:

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
```

```
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
<querym id='obsolete instance relations on opjk25'
  versionSpace='legalcase2'  ontology='opjk25'>
  <obsoleteOnIndividual/>
</querym>
</askm>
```

The answer to this query shows that there exist several obsolete instance relations on opjk25, like this:

```
<response>
<answer id="obsolete instance relations on opjk25">
  <individualRelated concept="http://protege.stanford.edu/
      kb#Entidad">
     <instances>
       <individual   name="http://protege.stanford.edu/
          kb#ontoRDF2_Instance_30094"/>
     </instances>
  </individualRelated>
  <individualRelated concept="http://protege.stanford.edu/
       kb#Rol">
    <instances>
      <individual name="http://protege.stanford.edu/
          kb#ontoRDF2_Instance_30094"/>
     </instances>
</individualRelated>
  <individualRelated concept="http://protege.stanford.edu/
       kb#Rol_Familiar">
    <instances>
      <individual name="http://protege.stanford.edu/
          kb#ontoRDF2_Instance_30094"/>
    </instances>
</individualRelated>
<individualRelated concept="http://protege.stanford.edu/
         kb#Situacin">
   <instances>
     <individual name="http://protege.stanford.edu/
        kb#ontoRDF2_Instance_30094"/>
  </instances>
</individualRelated>
<individualRelated concept="http://protege.stanford.edu/
      kb#Suceso">
  <instances>
     <individual name="http://protege.stanford.edu/
        kb#ontoRDF2_Instance_30094"/>
  </instances>
</individualRelated>
</answer>
```

We observe that fact that the individual $ontoRDF2\_Instance\_30094$ is not an instance of the concept $Entidad$ any more. Thus, we want to know whether or not this instance relation always holds in all prior versions, by posting the following temporal query:

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
 <querym id='in all prior version of opjk25,
    is instance30094 an instance of Entidad?'
   versionSpace='legalcase2'  ontology='opjk25'>
  <allPriorVersions>
    <query>
      <instance>
        <individual
      name='http://protege.stanford.edu/
          kb#ontoRDF2_Instance_30094'/>
       <catom name='http://protege.stanford.edu/
          kb#Entidad'/>
     </instance>
    </query>
  </allPriorVersions>
 </querym>
</askm>
```

Note that the instance query uses its DIG query format. Thus, it should be contained by the tag 'query' inside the tag 'allPriorVersion' in the temporal query, as shown above.

The answer to this query tells us that the instance relation always holds in all prior versions of opjk25, like this:

```
<response>
<true id="in all prior version of opjk25, is
   instance30094 an instance of Entidad?"/>
</response>
```

Moreover, we can make a version retrieval query to know in which versions this instance relation holds. The query is:

```
<?xml version='1.0' encoding= 'ISO-8859-1'?>
<askm xmlns='http://wasp.cs.vu.nl/sekt/more/lang'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://wasp.cs.vu.nl/sekt/more/more.xsd'>
<querym id='version retrieval of instance30094
```

```
      as an instance of Entidad?' versionSpace='legalcase2'>
  <ontologies>
    <query>
      <instance>
        <individual   name='http://protege.stanford.edu/
            kb#ontoRDF2_Instance_30094'/>
       <catom name='http://protege.stanford.edu/kb#Entidad'/>
     </instance>
   </query>
  </ontologies>
</querym>
</askm>
```

The answer to this version retrieval query tells us that this instance relation holds only from version opjk1 till version opjk24, as shown in the following:

```
<response>
 <ontologySet id="version retrieval of
   instance30094 as an instance of Entidad?">
     <ontology name="opjk24"/>
     <ontology name="opjk23"/>
     <ontology name="opjk22"/>
     ......
    <ontology name="opjk3"/>
    <ontology name="opjk2"/>
    <ontology name="opjk1"/>
    </ontologySet>
</response>
```

4.3.4 Scenario Four: Temporal Aspects of Ontology Change

In the new children log of the ontology $ontoRDF2$[1], we find that the concept $Hecho$ has the bottom concept as its new child. We examine further on the ontology $ontoRDF$ and $ontoRDF2$, and find that the child concept $Llamada$ of the concept $Hecho$ has been changed into an individual $Llamada$ on the ontology $ontoRDF2$. We want to know whether or not this change is stable, namely, this child concept relationship is never resumed in all sequel versions, as illustrated in Figure 4.1. Thus, we make the following query on the version space:

$$\neg(LLamada \sqsubseteq Hecho)\mathbf{SH}(LLamada \sqsubseteq Hecho).$$

[1]ontoRDF, ontoRDF2, ontoRDF3 are earlier versions of OPJK, which were developed in 2004.

The answer to this query is negative. Then, by the version retrieval query on the subsumption relation $(LLamada \sqsubseteq Hecho)$, we can find that the child concept relation is actually resumed on the ontology $ontoRDF3$.

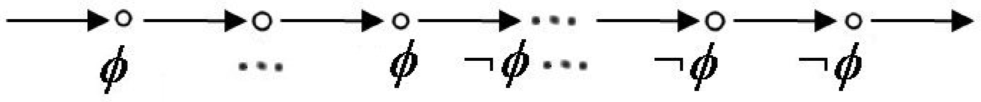


Figure 4.1: Stable change

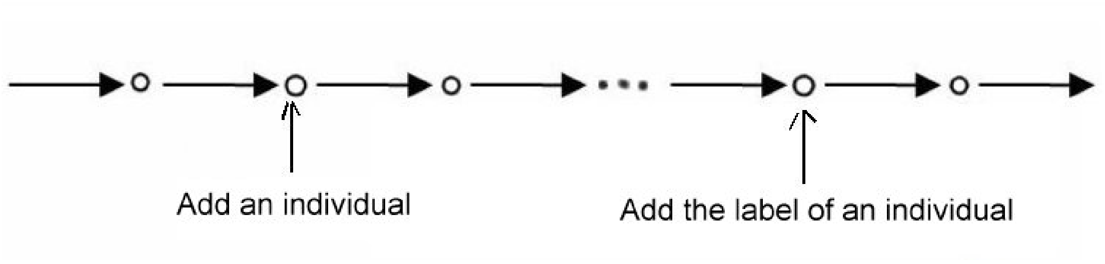


Figure 4.2: Changes on different versions

Sometimes we want to know whether or not the changes on a concept/role/individual are done at different versions. For example, an individual may be added in a version, but their other information is added later on another version, as illustrated in Figure 4.2.

Let ψ be a formal property which is identified as the existence of an individual in an ontology, and ϕ is a formal property which is identified as the other information about the individual is available in an ontology. This scenario can be illustrated semantically, as shown in Figure 4.3.

The query on this can be formalized as a temporal formula:

$$(\phi \wedge \psi)\mathbf{S}(\mathbf{Prev}((\psi \wedge \neg\phi)\mathbf{S}\neg\psi)).$$

As concrete examples of this kinds of queries on the OPJK ontologies, The formula ϕ can be like this:

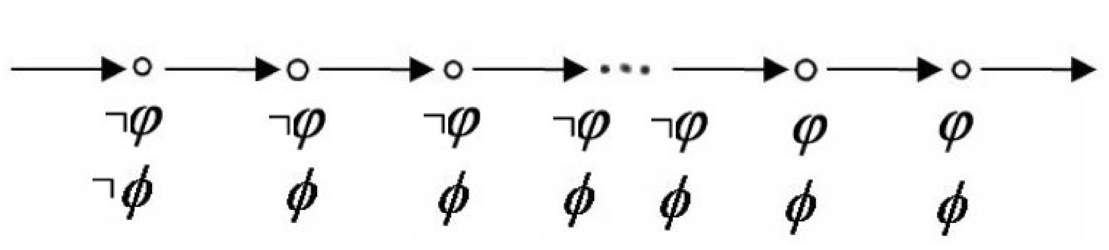


Figure 4.3: Semantics of changes on different versions

```
  <instanceRDFS>
    <individual name="http://protege.stanford.edu/kb#onto_Instance_10057"/>
    <catom name='http://protege.stanford.edu/kb#Acto_Procesal'/>
    <hasRoleWithAnything role='http://www.w3.org/2000/01/rdf-schema#label'/>
 </instanceRDFS>
```

which checks the label information of the instance $onto\_Instance\_10057$. The formula ψ can be like this:

```
<instance>
  <individual name="http://protege.stanford.edu/kb#onto_Instance_10057"/>
   <catom name='http://protege.stanford.edu/kb#Acto_Procesal'/>
 </instance>
```

which checks the instance relation. The answer to this query is negative.

4.4 Summary of the Analysis

We have shown several scenarios how MORE can be used for reasoning about multi-version ontologies in the SEKT legal ontologies. Those scenarios range from simply data examination in the version space, concept addition and its ramification checking, to more complex temporal reasoning on dynamic aspects of the SEKT legal ontologies. Those preliminary experiment on the SEKT legal ontologies shows that the system MORE is flexible and convenient for querying temporal and dynamic aspects of multiple version ontologies.

However, note that our experiment is still quite preliminary, because of the limited number (27) of ontologies in the version space. Changes in the version space are not very complicated. Most of changes are simply adding new concepts or new instances in the OPJK ontology. Few deletions occur. No repositioning of a concept, i.e., moving a concept from a branch into another branch in the concept hierarchy, happens inside these limited version space.

Chapter 5

Experiment II: 2006 Dutch General Election Data

In this chapter we evaluate the applicability of the MORE2 system based on a case study with realistic data sets from the Dutch general election of November 2006 [KSvA$^+$07]. This data set contains annotations of the content of news articles based on an ontology, as well as this ontology itself. In section 5.1 details of the data set are given. In section 5.2 and section 5.3 we analyze this data by using the MORE2 tool. In 5.2 our focus is on the ontology that is used for annotating the news articles, and in 5.3 we look at the annotations of the articles themselves. Based on these analyses and the process of doing these analyses we give in 5.4 our lessons learned. Finally, section 5.4 concludes this work.

5.1 Case study: Dutch general election data set

We use a realistic data set from the Dutch general election of November 2006 [KSvA$^+$07]. The data is collected in the project kies2006 (`www.kies2006.nl`). This data set contains annotations of paragraphs of news article based on an ontology. For instance a paragraph from a given newspaper article can be annotated by stating that an actor (e.g. a member of a particular political party) is positive about a particular issue, which can itself be classified as, for instance, a left-issue). The data has been collected on daily basis, over a period two months, and is separated in an ontology part and the part containing the actual annotations.

Concerning the ontology we have used all the different versions of the ontology in the period of Oct 2nd 2006 until November 24th 2006, which are 18 versions. The ontology is expressed in RDF Schema (RDF-S). The following gives an impression of the structure and size of this ontology:

- the average number of classes is 3640, with a minimal number of 2480 and a maximum number of 5027
- the average number of subclasses is 1745, with a minimal number of 1526 and a maximum number of 2006
- The RDF-S files have an average number of 11095 triples(ranging between 9770 and 12659)
- the ontology part of the data set both increases and decreases in size during the period that we have taken into account.

The annotations are modeled as individuals in RDF, referring to concepts from the ontology. The size of the annotation-part is rather large[1]. Again to give an impression of the size of the 16 versions that have been used:

- the number of annotations is on average roughly 26000 (with max 43300 and min 7400)
- the number of triples in each RDF file is on average roughly 1.180.000 (with max 2.100.000 and min 300.000)
- the annotation part of the data set increases in size during the period that we have taken into account. Deletion is possible, but rare.

5.2 Analysis: election ontology

In this section we describe the experiments with the election ontology. First we describe the experiment, then some hypotheses are formulated. We comment on these hypotheses by discussing the figures which summarize the results of the experiment.

5.2.1 Experiment 1: Concept changes

Description

[1]At the moment MORE2 can not handle more data

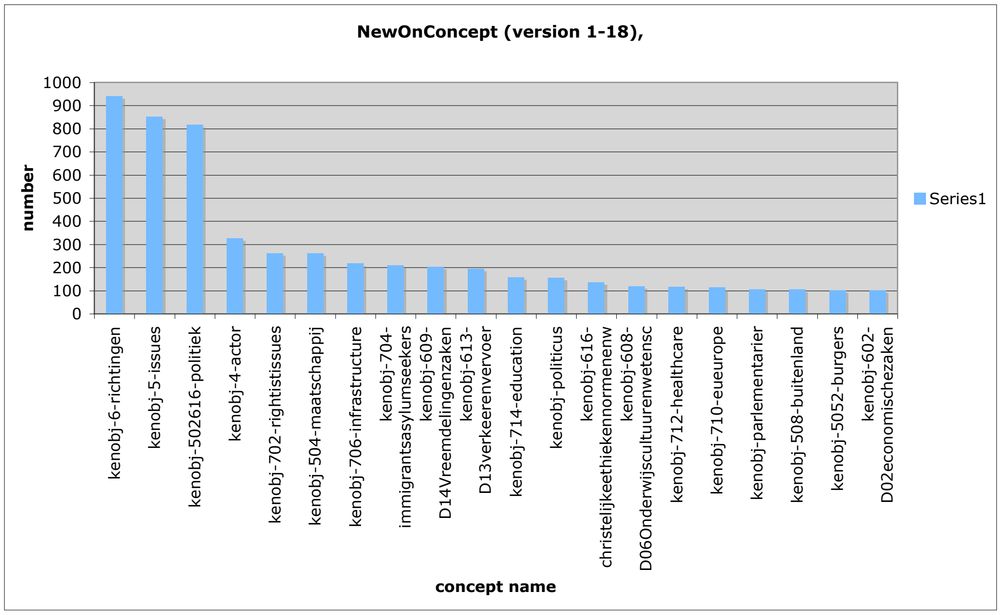


Figure 5.1: New on concept

In this experiment we want to know which concept definitions are changed most frequently during the development of an ontology. The MORE2 tool offers a query for ”new on concept” and ”obsolete on concept” (see Appendix). For 18 versions of the ontology, we queried for the new on concept and obsolete on concept.

Hyp1: Most changes of an ontology are related to a few concepts of the ontology.

Observation: Fig 5.1 shows us that a few concepts are relatively often involved in additions of definitions, which means a new parent or new child. The top three concepts are ”kenobj-6-richtingen”, ”kenobj-5-issues”, ”kenobj-502616-politiek”. The figure shows only the concepts with a value above 100. The total number of concepts is 1014 (!). This means that only 0,3 % of the concepts are responsible for the majority of all changes.

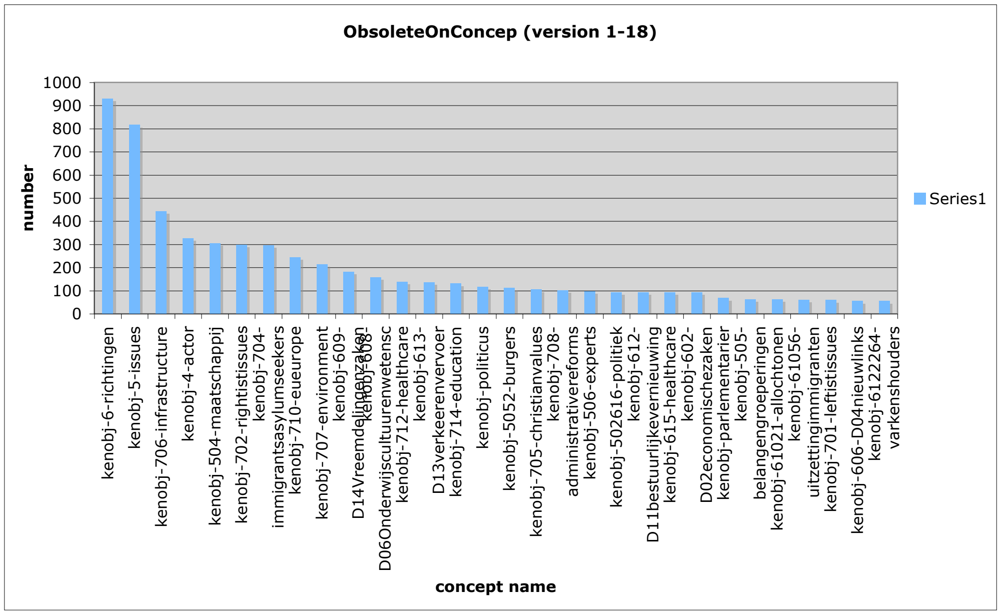


Figure 5.2: Obsolete on concept

In fig 5.2 we see the concepts that are affected by a *deletion* of a parent or child. Again we see that a few concepts are responsible for most of the changes. The total number of concepts that are affected by a deletion is 694.

Hyp2: The concepts which change frequently on new concepts (either by parent, child, ancestors or descendants) are the same concepts as those which are frequently changed in obsolete on concepts.

Observation Figure 5.1 and 5.2 show an overlap between the concepts that are affected by new on concepts or obsolete on concepts. The top 10 concepts from the new-on-concept belongs to the top 20 concepts from the obsolete-on-concepts. The highly affected concepts from obsolete on concepts are relatively high in the ranking of new on concepts. For instance the concept "710-eueurope", "707-environment", "D06Onderwijscultuurenwetenschappen", "712-healthcare" are respectively ranked 8,9, 11,12 in the obsolete on concept list and are 16, 54, 14 and 15 in the new on concept ranking.

5.2.2 Experiment 2 : Versioning

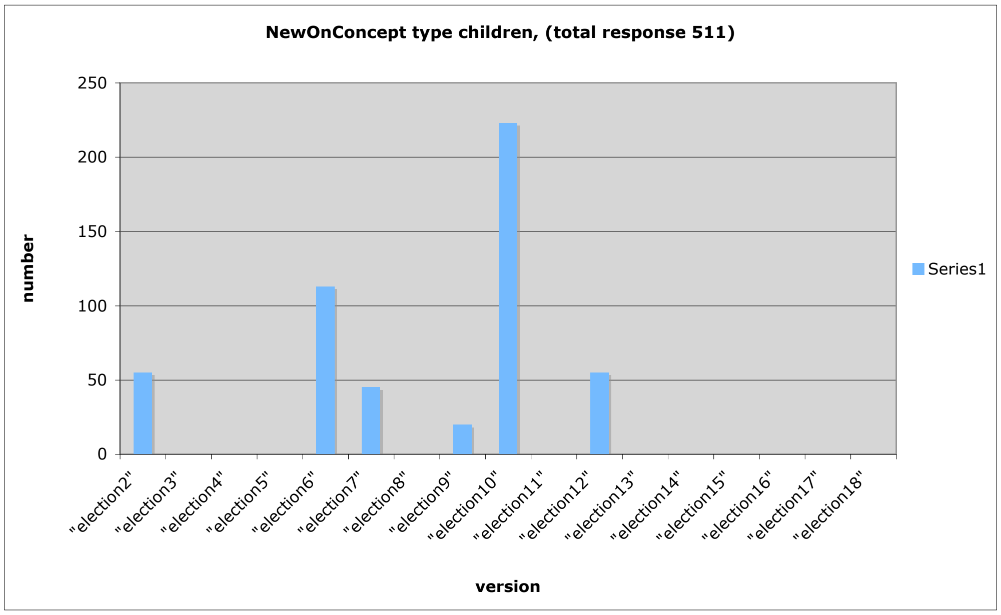


Figure 5.3: New on concept (type=children)

Description: In this experiment we are interested in how often changes happen during the use of the ontology, independent of which concept is changing. We use queries for ”new on concept”, and respectively the ”obsolete on concept” both with type children. For 18 versions of the ontology we queried for the new on concept and obsolete on concept.

Hyp1 : A few versions are responsible for large updates.

Observation The figure 5.3 shows the timeline of the ”new on concept” with type children. This shows us that one large update is in version ”election10”, with

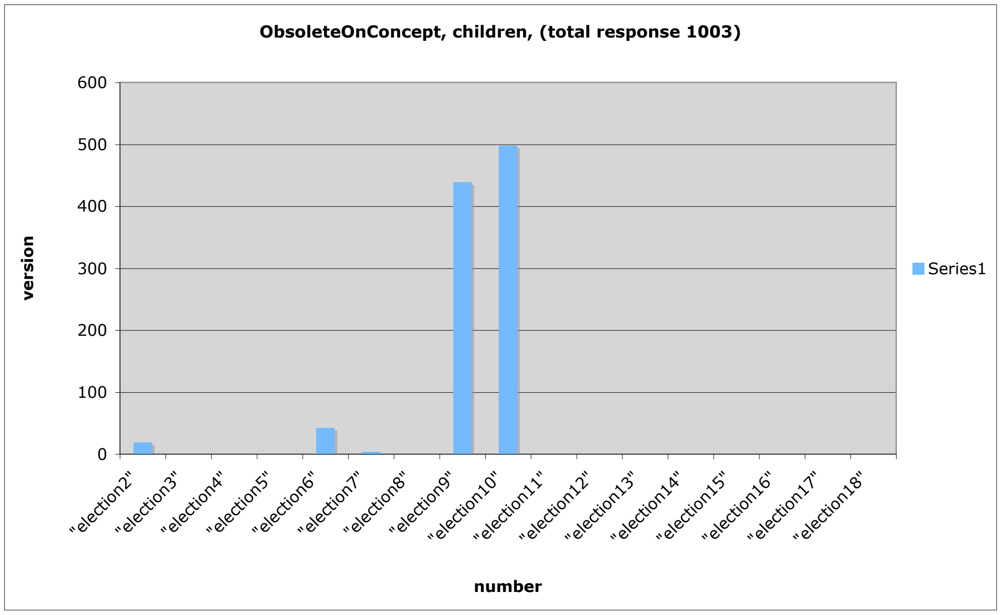


Figure 5.4: Obsolete on concept (type=children)

5 further small version changes, and no changes in 12 versions. Figure 5.4 shows the timeline for the ”obsolete concepts”. Here we see two large updates (”election9” and ”election10”) and a few small updates.

Of course these new on concept and obsolete on concept have their corresponding effect in the invariant on concepts. In figure 5.5 the election versions with many new or obsolete concepts (”election9” and ”election10”) has a smaller part that is invariant (unchanged). Fig 5.5 shows us that the invariant part of the ontology up to version 8 is larger then the invariant part from version 12 to 18. The reason for this is the number of deletions of concepts in the version ”election9” and ”election10”. The ontology became smaller and therefore the invariant-part became smaller. Taking into account the size of the ontology figure 5.6 gives a better measure of the stability of the ontology.

Hyp2 : If there are deleted concepts then there will be added concepts too. The idea is that a concept is changed (added somewhere else) but not removed completely from

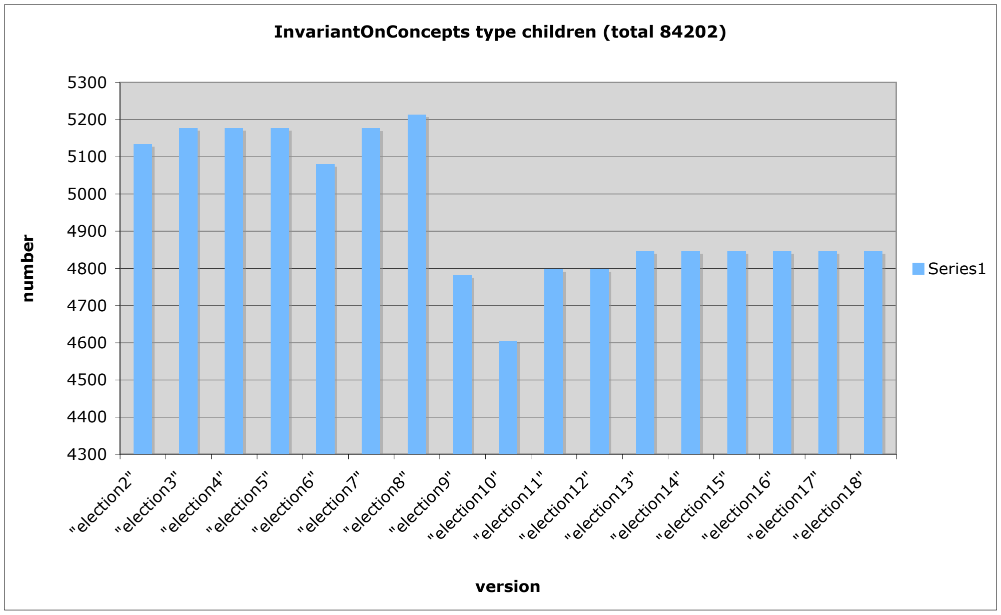


Figure 5.5: Invariant on concept (type=children)

the ontology. Furthermore only additions of concepts (without obsolete concepts) are realistic updates.

Observation Considering the figures 5.3 and 5.4, indeed for each version where concepts are deleted, concepts are also added. The contrary is not the case, for instance ”election12” has no deletions but does have additions.

Hyp3 : Deleted concepts are low in the hierarchy, whereas the compensated new concept is higher in the hierarchy.

Observation The versions ”election10” and ”election11” has a higher number for obsolete concepts, then for new concepts.

Notice that the versions that do not have affected concepts by new on concept (type children) or obsolete on concept (type children) will probably have changes for type parent. However, it turns out that this is not the case, see figure 5.7. The explanation is that

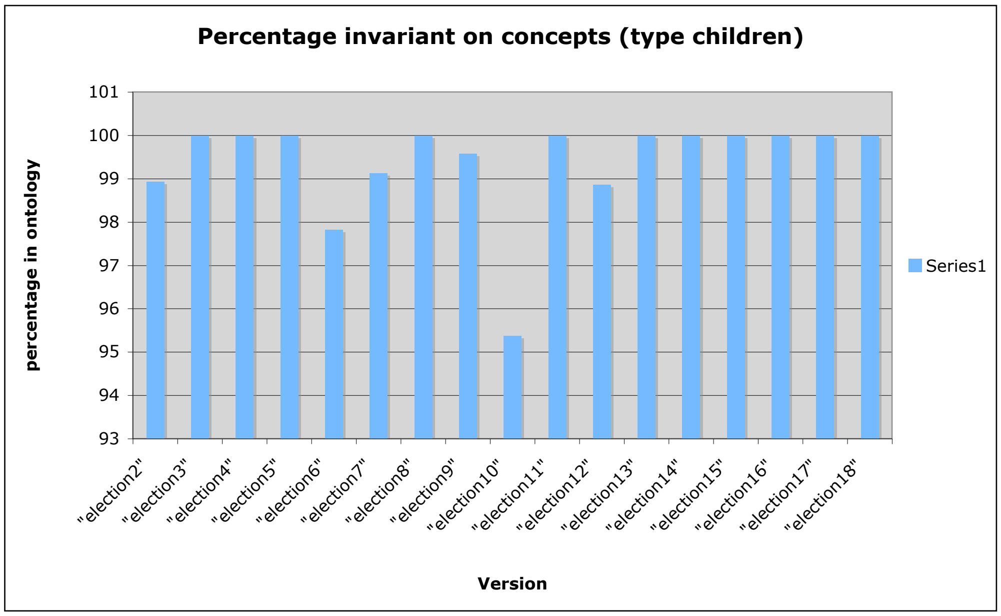


Figure 5.6: Invariant on concept (type=children)

these versions do have deletions or additions but delete a relation that was redundant or add a relation that is redundant. In other words, the affected versions from figure 5.3, 5.4, and 5.7 are the serious changes in the ontology.

5.2.3 Experiment 3: Stability of concepts

Description: Combining the information that we have obtained from the experiment 1 and 2, we now consider the changes of the new on concept with type children for the versions 2, 6, 7, 9, 10 and 12.

Hyp1 : most concepts are only changed in at most two versions

Observation Figure 5.8 shows the most affected concepts based on the query new on concept (type children). For instance the concept ”christelijke ethiek en

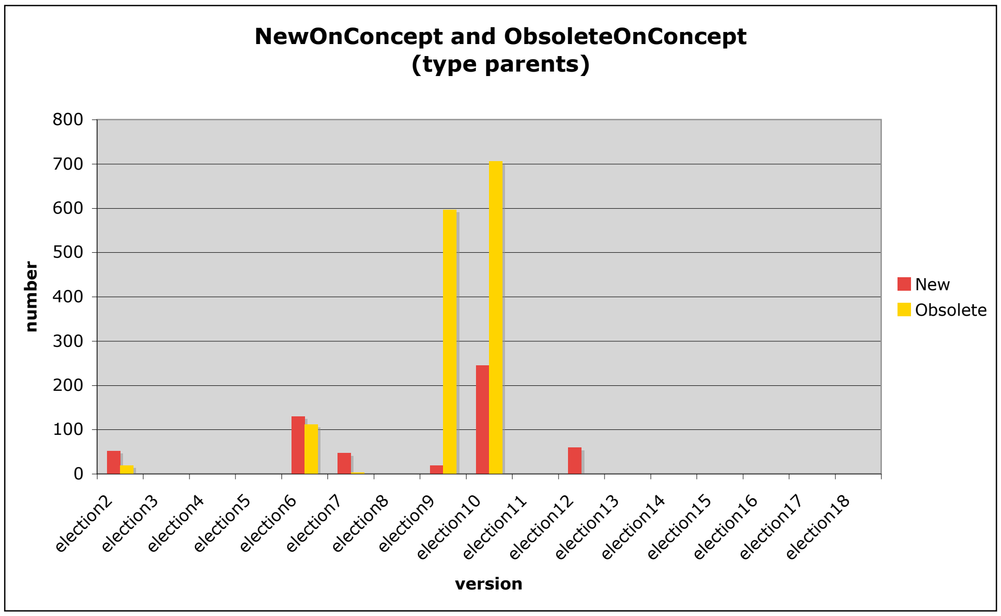


Figure 5.7: New on concept and Obsolete on concept (type-parents)

normen en waarden" (this concept has new children or is itself a new child) is for 45% affected by verion6, and 55% affected by version 10. The figure shows that all these concepts are involved in changes in one or two versions. To be more precise 14 concepts are only affected by one version, and 16 version are involved in two versions.

Hyp2 : changes in a version have an effect on a large number of concepts.

Observation In figure 5.8 we see that changes effects a large number of concepts. For instance the stat-10 effects most of the concepts in the figure. Notice that this is just a snapshot from the concepts that are affected (in total 116 concepts).

This section has illustrated that it is indeed possible to analyse several versions of the Dutch election ontology over time with the MORE2 tool. In section 5.4 we will discuss our lessons from these experiments.

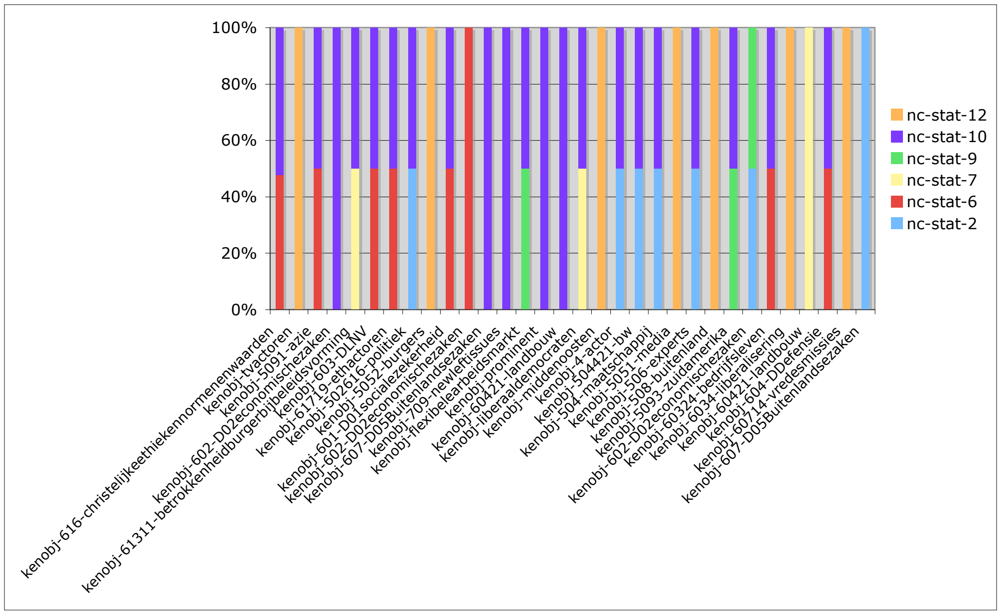


Figure 5.8: New on concept (type=children)

5.3 Analysis: annotations of news articles

In this section we describe some experiments with the annotations of the news articles. We also give some suggestions for further testing. The role of this section is just to *illustrate* the kind of analysis that can be done with the MORE2 system. The chosen versions of the annotation-part of the data is basically the same as the one for the ontology. However due to limitations of the size of data that MORE2 can handle at the moment, we fall back on 16 version instead of 18 versions. Furthermore, these versions were new versions with respect to the ontology part, however for the annotation-part of the data there are some versions in between the one we have token.

5.3.1 Experiments

As explained above, the annotation data are only ”individuals” in the sense of RDF Schema. MORE2 offers new, obsolete and invariant queries for individuals. We performed queries of ”new individual on annotations” with a particular property, for instance the Annotations with a subject of ”left-issue”. This is realized by the MORE extension of ”hasRoleWithIndividual” in the query in Section 3.3.1.

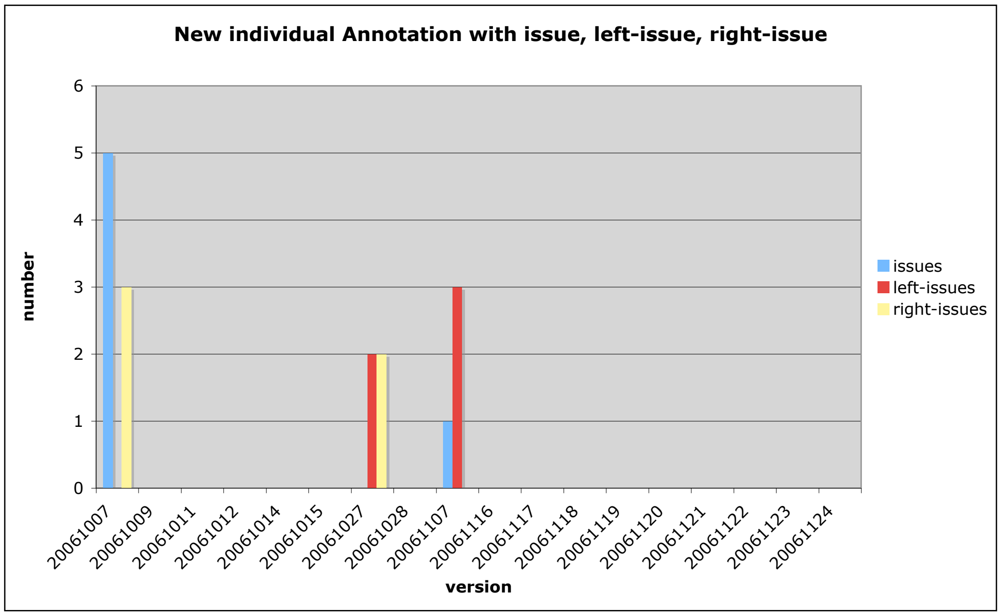


Figure 5.9: New on Individual Annotation with issues, right-issues, left-issues

Figure 5.9 shows us that there are not that many annotations with the subject of respectively ”issues”, ”left-issues” and ”right-issues”. An explanation could be that these concepts are too general and that the more specific issues are used in the annotations. A possible follow-up experiment would be to use the more specific left-issues like ”pensioen” or ”inkomensnivellering”.

Another experiment based on the ”new individual on Annotations” is described in figure 5.10. For the value of subject and object we have chosen the two biggest parties in

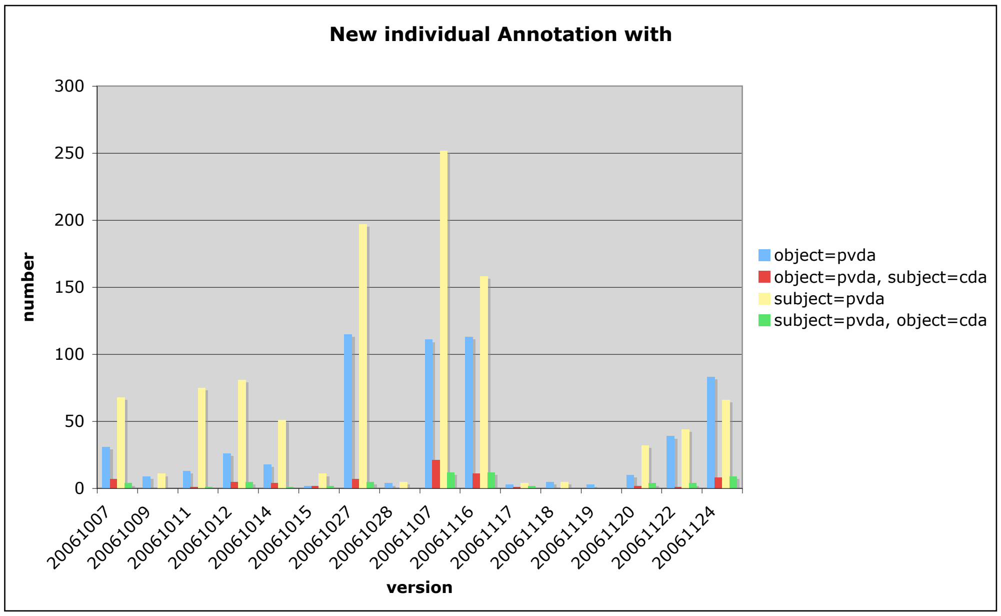


Figure 5.10: New Individuals on Annotations with particular values for object and subject.

The Netherlands: ”PvdA” and ”CDA” . The figure shows us the following:

- in all versions there are more annotations with ”PvdA” as subject then as object.
- only a relatively small number of annotations is in combination with ”CDA” (as either subject or object).
- A larger part of the annotations with object ”PvdA” are annotations with object ”PvdA” and subject ”CDA” than the part of the annotations with subject ”PvdA” that has subject ”PvdA” and object ”CDA”. This is illustrated in figure 5.11: the percentages of respectively annotations with object ”PvdA” and annotations with subject ”PvdA”.

Notice that the choice of the versions was not very convenient, because of the gaps in the sequence of versions. For instance the high peeks for versions ”20061027”,

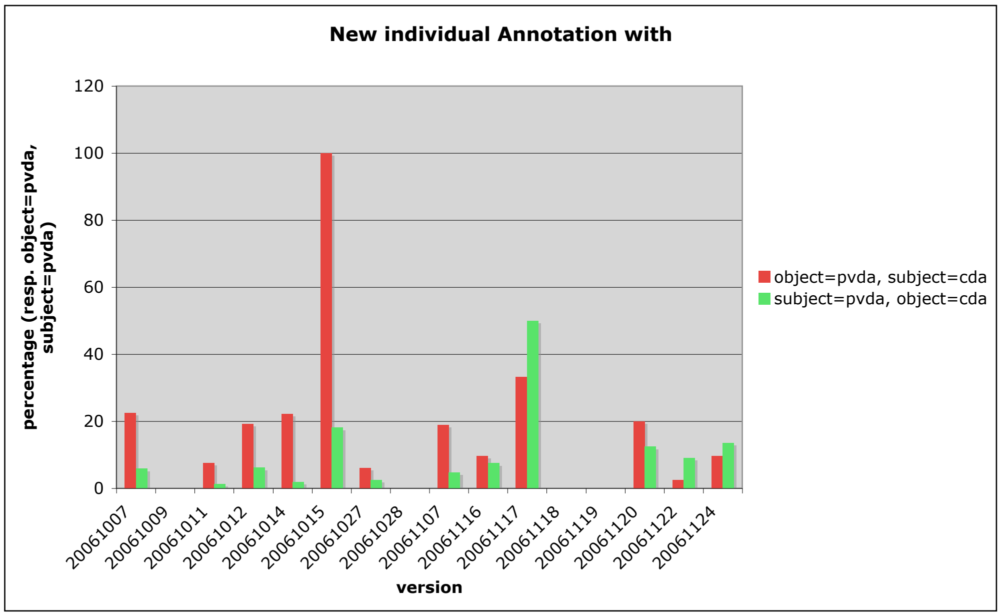


Figure 5.11: New Individuals on Annotations with subject "PvdA" and object "CDA" in percentages of subject "PvdA", and New Individuals on Annotations with object "PvdA" and subject "CDA" in percentages of object "PvdA"

"20061107" and "20061117" in figure 5.10 are due to the missing versions in the experiment. In future experiments a selection of versions will have to be made independent of the version selection of the ontology part, but making a meaningful selection of a "next" version in the experiment, such as an increase of the data-set over two days.

Further experiments in this line are for instance experiments that take into account the quality of the sentence (very negative, negative, neutral, positive, very positive) or other important parties in The Netherlands like SP and VVD.

5.3.2 Proposals for experiments

As already described above, the experiments concerning the annotation data are preliminary, and are just illustrations for how to use the MORE2 tool for analyzing this data set. Below are a number of hypotheses that we want to verify which use the same kind of queries as in the previous section. These hypotheses came up during several discussions with the domain expert.

Hyp1: Do actors change their opinion? This can be validated by analyzing queries on "new on individual" on annotations using the hasRoleWithIndividual and hasRoleWithLiteral features (see Section 3.3.1) for the desired properties. A change in an opinion can be observed by analyzing the number of "new on individual on annotation" with:

- hasRolewithIndividual for subject with a particular value for an actor (for instance $subject = "Bos"$(leader of the PvdA party))
- hasRolewithIndividual for object with a particular value for an issue (for instance $object = "Afganistan"$)
- hasRoleWithLiteral for quality with value 50 (=positive) or 100 (=very positive).

and the numbers of "new on individual on annotation" with the same actor and issue but with different values for quality, namely -50 (=negative) or -100 (=very negative). These numbers should be analysed over the sequence of all versions.

Hyp2: Do two actors have the same opinion on a particular issue? This can be validated with queries again on "new on individual on annotations" with respectively:

- hasRolewithIndividual for subject with actor $Actor1$
- hasRolewithIndividual for object with issue $Issue$
- hasRoleWithLiteral for quality with value 50 (=postive) or 100 (=very positive).

and

- hasRolewithIndividual for subject with actor $Actor2$
- hasRolewithIndividual for object with issue $Issue$
- hasRoleWithLiteral for quality with value -50 (=negative) or -100 (=very negative).

Notice that using variables like $Actor1$ and $Issue$ is not possible in a MORE2 query. This means that we have to decide which instance of the query we want to analyze.

Hyp3: If two actors do not have the same opinion on a particular issue, do they get a conflict with each other? Based on the previous analysis, we can continue with the analysis of ”new on individual on annotations” with

- hasRolewithIndividual for subject with actor $Actor1$
- hasRolewithIndividual for object with actor $Actor2$
- hasRoleWithLiteral for quality with value -50 (=negative) or -100 (=very negative)

or vice versa $Actor1$ and $Actor2$

- hasRolewithIndividual for subject with actor $Actor2$
- hasRolewithIndividual for object with actor $Actor1$
- hasRoleWithLiteral for quality with value -50 (=negative) or -100 (=very negative)

Hyp4: Criticism leads to criticism? Again an analysis with ”new on individual on annotations” can validate this question, namely with the following properties:

- hasRolewithIndividual for subject with actor $Actor1$
- hasRolewithIndividual for object with actor $Actor2$
- hasRoleWithLiteral for quality with value -50 (=negative) or -100 (=very negative)

and vice versa for $Actor1$ and $Actor2$

- hasRolewithIndividual for subject with actor $Actor2$
- hasRolewithIndividual for object with actor $Actor1$
- hasRoleWithLiteral for quality with value -50 (=negative) or -100 (=very negative)

Notice that for the above hypotheses the timeline is very important, and therefore MORE2 can be applied very well.

An orthogonal application of MORE2 on this data set is to concentrate on the process of making annotations for news articles, i.e. the manual coding process. For instance: do annotators who are coding fast also change their coding more often, is there agreement between coders who are coding the same articles, etc. Focusing on the coding process gives also possibilities for the use of obsolete on individuals. We will investigate content focused experiments, like the ones discussed above.

5.4 Conclusions and Lessons learned

This chapter has shown that indeed, the MORE2 tool can be used for analysing change patterns in substantial and realistic ontologies and datasets.

Implementation: The MORE2 tools was used on a dataset that was not known at the time of development, and although some adjustments and additions were made to MORE2 for the purposes of the experiment, it is fair to say that MORE2 was shown to be useable on an independent dataset. (In particular the need for analyzing changes on instances required the development of new MORE2 functionality during the experiment).

Also, the experiments were performed by a user of MORE2 had not been involved in the development of the tool, hence showing the technical usability of MORE2 by third parties.

Finally, the experiments were mostly run on a Apple hardware, running OS/X, while MORE2 was developed on a mixture of Windows and Linux hardware. This shows that MORE2 is useable across the major hardware and OS platforms.

Query language: The experiments from this chapter have exploited only a small fragment of the query-language offered by MORE2, in particular the new on concept, obsolete on concept, invariant on concept, and new individuals on annotations. Future experience will have to show if there is also good use for the other elements of MORE2's richt query language.

Also, there were some additional elements that would have been useful in MORE2's query language, such as allowing an analysis of the depth in the ontology where changes take place.

Efficiency: The experiments reported in this chapter did push MORE2 to its limits. The ontology experiments were all well within the computational bounds of MORE2, but the instance experiments had to be scaled down in order to make them feasible under the current MORE2 implementation, and even then many of the queries required run-times of several hours each.

Usability: Although useable by a third party, MORE2 has clear signs of being alpha-status software, and is not ready for independent use without access to advise by its developer. This is as expected of an academic prototype.

Besides access to technical MORE2 expertise during the experiments, it was also necessary to have access to a domain expert to explain the meta-model of the ontology involved in the experiment, since the MORE2 queries rely on knowing this meta-model in great detail. For example, for some of the queries above, it was crucial to know that natural persons (such as politicians) were (for technical reasons) modeled as classes, instead of the more obvious choice to model them as instances.

Finally, the XML-based representation of MORE2's output can only be interpreted meaningfully when represented in the form of tables, charts of spreadsheets. For the experiments in this chapter, ad hoc scripts were written for this conversion, but more tool support on the output side of MORE2 would greatly enhance its attractiveness to third parties.

Chapter 6

Discussion and Conclusions

In this document, we discussed MORE2, an extended reasoning and management system for multi-version ontologies. The approach to multi-version management that is based on the idea of using temporal logic for reasoning about commonalities and differences between different versions. For this purpose, we defined the logic **LTLm** that consists of operators for reasoning about derivable statements in different versions in [HS05a]. In MORE2, we introduced the hybrid logic operators as an extension to the temporal logic **LTLm** for multiple version ontology reasoning and management. We show that the temporal logic can be used to formulate typical reasoning and retrieval queries that occur in the context of managing multiple versions. One of the future work for MORE2 is to use the temporal logic as a specification language for ontology versioning, by which temporal and dynamic aspects of ontology versioning can be specified directly as part of the version space specification, rather than as part of temporal queries. The hybrid logic approach will play a distinguished role if the temporal logic serves as a specification language for ontology versioning, because that hybrid logic approach allows users to talk about version names directly in the specification. Therefore, the work of the hybrid logic extension reported in this document can be considered as an initial and significant step for the goal.

In this document, we reported that MORE2 provides the full support of ontology versioning with RDF/RDFS data. In particular, MORE2 supports queries on reasoning about commonalities and differences between different versions with respect to composite concepts, rather than only named concepts, as that is supported in the previous system of MORE. This composite concept support allows users to explore more complex aspects with respect to commonalities and differences between different versions. In this document, we provided various query examples to show how that can be achieved.

We have implemented a prototypical implementation of the system MORE2 that is based on the extended temporal logic framework. We have discussed the implementation issues of MORE2. The implementation of MORE2 has been integrated with KAON2, a powerful DL reasoner that is implemented by our SEKT research partner in Karlsruhe. In addition, this document provides a detailed manual of MORE2.

MORE has been successfully tested on realistic ontologies [HS05a, HSvH$^+$06]. In this document we reported two additional experiments of MORE2 with realistic ontology data: the first one is the SEKT legal ontologies, and the second one is the 2006 Dutch general election ontology data. For the experiment of SEKT legal ontologies, we provided various scenario examples how MORE2 can be used to explore temporal and dynamic aspects of SEKT legal ontology versioning. Those scenarios include how to examine data change by instance checking queries, how to examine the ramification after a concept is added by new/invariant/obsolete concept queries, how to check ontology change by complex temporal queries, etc. Those experiments show that we can obtain better understanding on temporal and dynamic aspects of ontology versioning by using the system MORE2.

Chapter 7

Appendix

7.1 Interface for Temporal and Hybrid Logic Formulas

Type	Interface	Formula
Query on Single Ontology	<query> Atomic Query </query>	q
Negation	<not> Q </not>	$\neg\phi$
Conjunction	<and> Q1 $\cdots$ Qn </and>	$\phi_1 \wedge \cdots \wedge \phi_n$
Disjunction	<or> Q1 $\cdots$ Qn </or>	$\phi_1 \vee \cdots \vee \phi_n$
Implication	<implies> Q1 Q2 </implies>	$\phi_1 \rightarrow \phi_2$
If and only if	<iff> Q1 Q2 </iff>	$\phi_1 \leftrightarrow \phi_2$
Previous	<previousVersion> Q </previousVersion>	$\mathbf{Prev}\phi$
Always in the Past	<allPriorVersions> Q </allPriorVersions>	$\mathbf{H}\phi$
Sometimes in the Past	<somePriorVersion> Q </somePriorVersion>	$\mathbf{P}\phi$
Since	<since> $Q_1 Q_2$ </since>	$\phi_1 \mathbf{S} \phi_2$
Nominal	<nominal value=o />	o
At Operator	<at nominal=i > ϕ </at>	$@_i\phi$
Downarrow Operator	<downarrow variable=X ϕ </downarrow>	$\downarrow x.\phi$

7.2 Interface for Concept Comparison

New concept	<querym id=ID versionSpace=S ontology=o> <newOnConcept concept=c type=Type/> </querym>	$new_{Type}(S, o, c)$
Obsolete concept	<querym id=ID versionSpace=S ontology=o> <obsoleteOnConcept concept=c type=Type/> </querym>	$obsolete_{Type}(S, o, c)$
Invariant concept	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept concept=c type=Type/> </querym>	$obsolete_{Type}(S, o, c)$
New concept on all concepts	<querym id=ID versionSpace=S ontology=o> <newOnConcept type=Type/> </querym>	$new_{Type}(S, o, c)$ for all c
Obsolete concept on all concepts	<querym id=ID versionSpace=S ontology=o> <obsoleteOnConcept type=Type/> </querym>	$obsolete_{Type}(S, o, c)$ for all c
Invariant concept on all concepts	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept type=Type/> </querym>	$invariant_{Type}(S, o, c)$ for all c
New concept without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept/> </querym>	$new_{type}(S, o, c)$ for all c and for all $type$
Obsolete concept without a type	<querym id=ID versionSpace=S ontology=o> <obsoleteOnConcept/> </querym>	$obsolete_{type}(S, o, c)$ for all c and for all $type$
Invariant concept without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept/> </querym>	$invariant_{type}(S, o, c)$ for all c and for all $type$
New concept compared with arbitrary ontology	<querym id=ID versionSpace=S ontology=o> <newOnConcept concept=c type=Type comparedOntology=o'/> </querym>	New $type$ concept of c in o, compared with o'
New concept relation for a concept list	<querym id=ID versionSpace=S ontology=o> <newOnConcept type=Type > <catom name = c_1> ... <catom name = c_n/> < /newOnConcept> </querym>	$new_{Type}(S, o, c)$ for all $c \in \{c_1, \cdots, c_n\}$

where $type$ =children/parents/ancestors/descendants

7.3 Interface for Role Comparison

New role	<querym id=ID versionSpace=S ontology=o> <newOnRole role=r type=Type/> </querym>	$new_{Type}(S, o, r)$
Obsolete role	<querym id=ID versionSpace=S ontology=o> <obsoleteOnRole role=r type=Type/> </querym>	$obsolete_{Type}(S, o, r)$
Invariant role	<querym id=ID versionSpace=S ontology=o> <invariantnRole role=r type=Type/> </querym>	$obsolete_{Type}(S, o, r)$
New role on all roles	<querym id=ID versionSpace=S ontology=o> <newOnRole type=Type/> </querym>	$new_{Type}(S, o)$
Obsolete role on all roles	<querym id=ID versionSpace=S ontology=o> <obsoleteOnRole type=Type/> </querym>	$obsolete_{Type}(S, o)$
Invariant role on all roles	<querym id=ID versionSpace=S ontology=o> <invariantOnRole type=Type/> </querym>	$invariant_{Type}(S, o)$
New role without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnRole/> </querym>	$new_{type}(S, o)$ for all $type$
Obsolete role without a type	<querym id=ID versionSpace=S ontology=o> <obsoleteOnRole/> </querym>	$obsolete_{type}(S, o)$ for all $type$
Invariant role without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnRole/> </querym>	$invariant_{type}(S, o)$ for all $type$
New role compared with arbitrary ontology	<querym id=ID versionSpace=S ontology=o> <newOnRole role=c type=Type comparedOntology=o'/> </querym>	New $type$ role of r in o, compared with o'
New role relation for a role list	<querym id=ID versionSpace=S ontology=o> <newOnRole type=Type > <ratom name = r_1> $\cdots$ <catom name = r_n/> < /newOnRole> </querym>	$new_{Type}(S, o, r)$ for all $r \in \{r_1, \cdots, r_n\}$

where $type$ =rchildren/rparents/rancestors/rdescendants

7.4 Interface for Individual Comparison

New individual	<querym id=ID versionSpace=S ontology=o> <newOnIndividual concept=c/> </querym>	$new_{instances}(S, o, c)$
Obsolete individual	<querym id=ID versionSpace=S ontology=o> <obsoleteOnIndividual concept=c/> </querym>	$obsolete_{instances}(S, o, c)$
Invariant individual	<querym id=ID versionSpace=S ontology=o> <invariantOnIndividual concept=c/> </querym>	$obsolete_{instances}(S, o, c)$
New individual on all concepts	<querym id=ID versionSpace=S ontology=o> <newOnIndividual/> </querym>	$new_{instances}(S, o)$
Obsolete individual on all concepts	<querym id=ID versionSpace=S ontology=o> <obsoleteOnIndividual/> </querym>	$obsolete_{instances}(S, o)$
Invariant individual on all concepts	<querym id=ID versionSpace=S ontology=o> <invariantOnIndividual/> </querym>	$invariant_{instances}(S, o)$
New individual compared with arbitrary ontology	<querym id=ID versionSpace=S ontology=o> <newOnIndividual concept=c comparedOntology=o'/> </querym>	new instances of c in o, compared with o'
New individual relation for a concept list	<querym id=ID versionSpace=S ontology=o> <newOnIndividual> <catom name = c_1> $\cdots$ <catom name = c_n/> < /newOnIndividual> </querym>	$new_{instances}(S, o, c)$ for all $c \in \{c_1, \cdots, c_n\}$

Bibliography

[AB01] C. Areces and P. Blackburn. Bringing them all together. *Journal of Logic and Computation*, 11(5):657–669, 2001. Special Issue on Hybrid Logics. Areces, C. and Blackburn, P. (eds.).

[BCB$^+$04] V.R. Benjamins, J. Contreras, M. Blázquez, L. Rodrigo, P. Casanovas, and M. Poblet. The sekt use legal case components: ontology and architecture. In T.B. Gordon, editor, *Legal Knowledge and Information Systems*, pages 69–77. IOS Press, Amsterdam, 2004.

[Bla00] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

[BMC03] S. Bechhofer, R. Möller, and P. Crowther. The DIG description logic interface. In *International Workshop on Description Logics (DL2003)*. Rome, September 2003.

[BPOC$^+$05] M. Blazquez, R. Pena-Ortiz, J. Contreras, R. Benjamins, P. Casanovas, J.J. Vallbe, and N. Casellas. Legal case study: Prototype. Project Report Report D10.3.1, SEKT, 2005.

[BS95] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.

[BT99] P. Blackburn and M. Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.

[Bul70] R. Bull. An approach to tense logic. *Theoria*, 36:282–300, 1970.

[CCLCL05] V.R. Benjamins P. Casanovas, J. Contreras, J. M. López-Cobo, and L. Lemus. Iuriservice: An intelligent frequently asked questions system to assist newly appointed judges. In V.R. Benjamins et al, editor, *Law and the Semantic Web*, pages 205–522. Springer-Verlag, London, Berlin, 2005.

[CCV$^+$06] P. Casanovas, N. Casellas, J.-J. Vallbe, M. Poblet, V.R. Benjamins, M. Blazquez, R. Pena-Ortiz, and J. Contreras. Semantic web: A legal case study. In J. Davies, R. Studer, and P. Warren, editors, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, pages 259–280. John Wiley & Sons, Chichester, UK, 2006.

[CCV$^+$07] P. Casanovas, N. Casellas, J. J. Vallbe, M. Poblet, M. Blazquez, J. Contreras, V.R. Benjamins, and J.-M. Lopez-Cobo. After analysis: First steps of the iuriservice implementation at the spanish judicial school. Project Report Report D10.4.1, SEKT, 2007.

[CPC$^+$05a] P. Casanovas, M. Poblet, N. Casellas, J. Contreras, R. Benjamins, and M. Blázquez. Supporting newly-appointed judges: A legal knowledge management case study. *Journal of Knowledge Management*, 2005.

[CPC$^+$05b] P. Casanovas, M. Poblet, N. Casellas, J-J. Vallbé, F. Ramos, V.R. Benjamins, M. Blázquez, J. Contreras, and J. Gorronogoitia. Legal scenario. case study intelligent integrated decision support for legal professionals. Project Report Report D10.2.1, SEKT, 2005.

[FdRS03] M. Franceschet, M. de Rijke, and B. H. Schlingloff. Hybrid logics on linear structures: expressivity and complexity. In *Proceedings TIME 2003*, pages 166–173. IEEE Computer Society Press, 2003.

[Har84] D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands, 1984.

[HS91] J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, October 1991.

[HS05a] Z. Huang and H. Stuckenschmidt. Reasoning with multiversion ontologies. Deliverable D3.5.1, SEKT, 2005.

[HS05b] Z. Huang and H. Stuckenschmidt. Reasoning with multiversion ontologies: a temporal logic approach. In *Proceedings of the 2005 International Semantic Web Conference*, 2005.

[HSvH$^+$06] Zhisheng Huang, Stefan Schlobach, Frank van Harmelen, Michel Klein, Nuria Casellas, and Pompeu Casanovas. Reasoning with multiversion ontologies: Evaluation. Deliverable D3.5.2, SEKT, 2006.

[HTK00] D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, 2000.

[HV04] Z. Huang and C. Visser. Extended DIG description logic interface support for PROLOG. Deliverable D3.4.1.2, SEKT, 2004.

[KSvA^{+}07] J. Kleinnijenhuis, O. Scholten, W. van Atteveldt, A. van Hoof, A. Krouwel, D. Oegema, J.A. de Ridder, N.Ruigrok, and J. Takens. *Nederland Vijfstromenland: De rol van media en stemwijzers bij de verkiezingen van 2006*. Amsterdam: Bert Bakker, 2007.

[PT91] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93(2):263–332, 1991.

[vB95] J. van Benthem. Temporal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 241–350. Oxford, Clarendon Press, 1995.