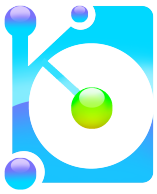

WP5-D5.1

A Dynamic Knowledge Base

Project full title:	Knowledge Driven Data Exploitation
Project acronym:	K-Drive
Grant agreement no.:	286348
Project instrument:	EU FP7/Maria-Curie IAPP/PEOPLE WP 2011
Document type:	D (deliverable)
Nature of document:	S (Specification)
Dissemination level:	PU (public)
Document number:	IBM/WP5-D5.1/D/PU/b1
Responsible editors:	Alessandro Faraotti, Marco Monti
Reviewers:	Yuan Ren and Jeff Z. Pan
Contributing participants:	IBM, UNIABDN
Contributing workpackages:	WP5
Contractual date of deliverable:	31 December 2013
Actual submission date:	15 January 2014

Abstract
In this deliverable, we conduct case studies in both public and private domains to identify the dynamics in K-Drive use cases. Following these studies, we categorise the dynamics in knowledge bases and investigate existing approaches for dealing dynamic knowledge in the Semantic Web community. Following a conclusion of the state of the art, we identify the techniques demanded by K-Drive use cases and propose our solutions to deal with the dynamic knowledge not only in the data level but also in the conceptual level. This dynamic knowledge base will be an infrastructure of WP5 and also WP6.

Keyword List
dynamic knowledge Base, healthcare, scalable semantic data, stream semantic data, query generation

Project funded by the European Commission within the 7th Framework Programme/Maria Curie Industry-Academia Partnerships and Pathways schema/PEOPLE Work Programme 2011.

A Dynamic Knowledge Base

Alessandro Faraotti[1], **Marco Monti**[1], **Guido Vetere**[1], **Honghan Wu**[2], **and Yuting Zhao**[2]

[1] IBM Italy, Roma, Italy
[2] Department of Computing Science, Aberdeen University, UK

15 January 2014

Abstract
In this deliverable, we conduct case studies in both public and private domains to identify the dynamics in K-Drive use cases. Following these studies, we categorise the dynamics in knowledge bases and investigate existing approaches for dealing dynamic knowledge in the Semantic Web community. Following a conclusion of the state of the art, we identify the techniques demanded by K-Drive use cases and propose our solutions to deal with the dynamic knowledge not only in the data level but also in the conceptual level. This dynamic knowledge base will be an infrastructure of WP5 and also WP6.

Keyword List
dynamic knowledge Base, healthcare, scalable semantic data, stream semantic data, query generation

Contents

1 Introduction

The goal of this deliverable it to produce a specification for a knowledge-base (KB) which is able to store, access and analyse domain independent dynamic information. Such specification will include the definition of the dynamic KB which will be a generalization of the one described in WP3, with additional information including time attributes of the maintained knowledge. In particular tasks related to this deliverable drive the implementation of a dynamic medical KB to be used as underling infrastructure to store data extracted from the Heuristic-Based Clinic System (HBCS), described in WP2, and to answer generated semantic queries and stream queries (WP5, WP6).

In this deliverable, we first conduct use case studies in both public and private domains to identify the dynamics in various situations. Following this analysis, we look into the existing approaches which can be utilised to deal with dynamic knowledge. Finally, we propose our solution which is dedicated to the dynamic characteristics in K-Drive use cases.

This document has been structured as follows: section 2 explores characteristics of data-sets, related to the two case studies, in order to outline actual an prospective dynamics; in section 3.3 the state of the art about dynamic knowledge bases is briefly investigated; in section 4.1 identified dynamics are considered and a description of the expected dynamic KB is presented; finally section 5 concludes this deliverable.

2 Case studies for dynamics analysis

The comprehension of domain and data–sets dynamics is crucial to provide K-Drive with a suitable test–bed. Different domains may reveal different data dynamics which are rooted in their underlying data variation generating mechanisms. The statistical features of the identified data dynamic process may provide suggestions on the best strategies on how to deal with them from an analytical point of view and on how to model their assumed generating mechanisms.

Stability but not-stationariness, for example, may represent statistical features that are usually paired in biology and in medicine. Fluctuations may in fact occur over time, between evolution phases, but usually, they are constrained within specific ranges. The temporary, step–by–step changing information (i.e. non stationarity condition) can be paired with the more permanent/long term asymptotic information coming from the averages and distributions (i.e. stability condition). By the interaction between the short term and long term information evolutions, we may have to design different strategies for representing the dynamic knowledge.

For this purpose, we reviewed case studies descriptions reported in WP1 and WP2 deliverables to analyse dynamics that the knowledge–base should be able to support.

2.1 Sport and History Domains

Regarding the first case study the two datasets, Sport and History from DBPedia, have been considered. The dynamics of these datasets are quite different to each other, with the sports dataset being more volatile than the history one. In particular, in the sports dataset, an important change happens in the rosters of the teams, namely the players that currently play in them. Given that transfers take place at least twice a year, this is a relatively frequent change that results into the outdating of several statements in the dataset and their replacement with new ones. The same happens for the managers of the teams. On the other hand, changes in the history dataset are primarily of an additive nature (existing facts are only deleted if proven wrong) with new events and their related information being periodically added. In both datasets, dynamics have mostly to do with the data as the ontology schemas are rather stable.

2.2 The Oncological Domain

The oncological domain is a critical field in which knowledge is constantly evolving, being strictly connected to the most advance research. As any other medical domain, cancer care is very broad and complex, to model, it includes the description of the human organism and its dynamics whose insights come from different disciplines (e.g. anatomy, cytology, physiology). In particular, the oncology focuses on the relation between the presentations (e.g. oncological diseases and other comorbidities) and the identified treatments considering symptoms, causes and prognoses.

The scientific progress affects the evolution of both the comprehension and the representation of such a great complexity, changes in the terminology, in the boundary of concepts, in the relations/hierarchies make the dynamic domain description a complex and challenging task to achieve.

Another important aspect of affecting the complexity of the domain comprehension and representation is related to the different speeds at which knowledge "layers" evolve.

From a semantic perspective, different "objects" may change their concept-boundaries/definitions at different speeds. In our experience, this aspect is reflected by the evolution of the concepts used to specify the clinical presentations (yearly), the oncological history and the extra-oncological history (monthly/daily). Also the analyzed recommended treatment programs as well as the performed treatments and the patient's clinical status have revealed such a conceptual dynamics.

In particular, in the presented HBCS case study, in which treatment programs (TPs) performed by physicians are analyzed in order to find deviations from TPs recommended by the hospital clinical guidelines, deviations analysis criteria and guidelines themselves are subjects to change, both in instances (data) and in concepts. When considering performed TPs it is easy to notice that each patient has his own history made by sequences of clinical pictures that succeed one another over the time according to the disease evolution and to the healthcare programs. Physicians iteratively take decisions about treatments to deliver basing on the outcomes of performed clinical tests or on the assumptions on new ones. Data under examination can be even incomplete or partially outdated, requiring a reconsideration of the information set and therefore, introducing dynamics in the data acquisition, classification and processing.

Clinicians have to plan for new data acquisition on a daily basis and this aspect of the process characterizes their decision-making process. The use case original datasets contains clinical data periodically updated which reveal the dynamics trough which the information acquisition, comprehension and representation evolve over the time. In particular in the HBCS use case we do not process sensors data from Point-of-Services (blood pressure, heart-rate) are not available because we are not dealing with emergency clinical treatments. Nonetheless stream queries are possible by relying on medical examinations and tests reports that which availability is time-dependent; for example a performed TP may turns to be a deviation with respect to a newer version of a related guideline.

If the scientific research for sure contributes to the evolution of concepts other forces like the technology and the marketing may alter the perception and identification of domain-specific concepts even in a relatively short time window frame.

Nonetheless, if all the discovery and knowledge representation processes are aimed to extend the boundaries of actual medical knowledge, they are also aimed to preserve the scientific value of the previously accumulated knowledge and therefore its consistency; this aspect has a significant impact on the knowledge maintenance.

2.3 Considerations

Case studies analysis make us aware that dynamics have to be considered from different point of view when designing the knowledge-base. In particular domain and data evolution affect three different "angles" of knowledge dynamics:

1. data flow (e.g. new clinical facts, new football matches)

2. intra–domain (e.g. time bounded data, sequences of clinical facts and patterns, guidelines versions)

3. domain evolution (e.g. the way through which the domain and its dynamics are considered and classified)

In our research we investigated different strategies that can be adopted to represent and take advantage of dynamics behind each "angle". In particular we considered ontology design methodologies to model the domain, stream reasoning to deal with data flows, and belief revision approaches to handle data and domain evolution.

3 Stream Reasoning approaches

Here we introduce a generic formalization of dynamic knowledge base and then, by analyzing the state of the art, we show techniques that can be used to handle incoming information streams.

3.1 A generic Dynamic Knowledge Base

Firstly we formalize a *dynamic Knowledge Base* as a steam of knowledge bases, and then we use *change* to capture the difference between KBs.

Definition 1 ***(Dynamic Knowledge Base (DKB))***
A Dynamic Knowledge Base (DKB) *is a steam of knowledge base* K_1, K_2, ..., K_n, *a* change $C(i)$ *is the difference between* K_{i-1} *and* K_i, i.e.,

$$C(i) = (K_i - K_{i-1}) \cup (K_{i-1} - K_i). \tag{1}$$

in where $(K_i - K_{i-1})$ *is the set of data ADDED in* K_i, *donated by* $C_{add}(i)$; *and* $(K_{i-1} - K_i)$ *is the set of data REMOVED from in* K_{i-1}, *donated by* $C_{del}(i)$. *So we have:*

$$C_{add}(i) = \{k | k \in K_i \text{ and } k \notin K_{i-1}\}; \tag{2}$$

$$C_{del}(i) = \{k | k \in K_{i-1} \text{ and } k \notin K_i\}. \tag{3}$$

Obviously above definition of "change" reflects the difference between the current KB (*e.g.* K_i) and the former KB (*e.g.* K_{i-1}). In general it contains 2 parts of information: the "new data" added in the current KB, and the "outdated data" removed from the previous KB.

3.2 Category of changes in DKB

Based on the Dynamic Knowledge Base described as above (Def 3.1), there could be various kinds of changes on the data in a DKB. In general "changes" (Def 1) in a DKB would be categorized in terms of meta-properties such as rigidity, cyclicity, monotonicity, etc.

Without losing generality, the behavioural properties of "changes" in Stream data could be categorized as:

1. **come and remain (CM)**: A fact is detected and will never change, e.g. a person is born: Birthday(Mary)

2. **removed forever (RF)**: A fact is removed from the KB forever, e.g. if a person is dead, then all in treatment data will be removed from the current KB, and move to the other KB.

3. **come and leave (CL)**: A fact appeared for a while and then disappeared, e.g. a patient is pregnant: Pregnant(Mary)

4. **periodical changes (PC)**: A fact has a value which will change periodically with repeated values, e.g. women's Menstrual or changes of 4 seasons.

5. **monotonic changing the value (MC)**: A fact has a value which will change monotonically, e.g. ages of a patient: Age(Mary)

6. **non-monotonic changes (NC)**: A fact has a value which will change NON-monotonically, e.g. body temperature of a patient: Temperature (Mary)

In general, if we model a stream KB as you did, i.e. an ordered set of KBs sharing the same TBox where KBt means the stream KB at time t, we could come up with a complete analysis in terms of meta-properties such as rigidity, cyclicity, monotonicity, etc, as you suggested. Then, one of the tasks could be that of inferring such meta properties for the KB TBox, based on observations of the stream KB at different times. Once you have a compelete dynamic meta-level characterization of the TBox, you can decide how to optimize the access of the dynamic KB in many use cases.

3.3 The State of the art of handling dynamic knowledge

Following the categorisation of dynamics in knowledge bases, in this subsection, we investigate existing approaches for dealing with dynamic knowledge in the Semantic Web community. In terms of surveys in this regard, we found a very good tutorial [1] about stream reasoning in ISWC 2013, which covers most of the interesting researches. Hence, in this subsection, we follow some of online slides to summarise the related researches. We also reused many of the figures in those slides. Hereby we thank all those authors.

Specifically, first, we look into the dynamics in the data level i.e., the extension of RDF data model to support dynamic data. Second, we investigate the dynamic queries i.e., the extension in SPARQL queries to support dynamic data acquisition. Finally, we do a brief survey in system level to get a brief idea about how these techniques are used in different applications.

[1] http://streamreasoning.org/sr4ld2013

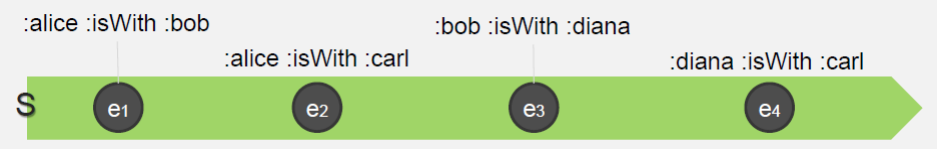

Figure 1: A RDF stream without timestamp

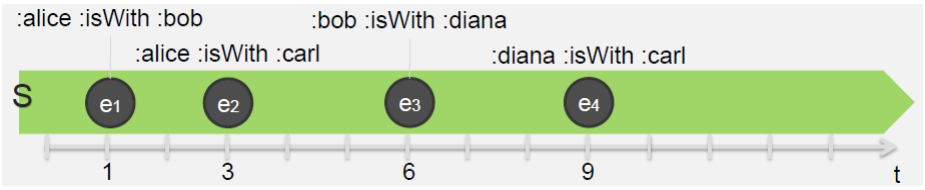

Figure 2: A RDF stream with one timestamp attribute

3.3.1 Continuous RDF model extensions

A data item is a self-consumable informative unit. A data item can be a triple or an RDF graph. A data stream is an (infinite) ordered sequence of data items. So the main task is to extend RDF to model data streams. In this taks, one of the key questions is how to model the time in the data. There are several considerations.

- Time should not (but could) be part of the schema
- Time should not be accessible through the query language
- Time as object would require a lot of reification

A timestamp is a temporal identifier associated to a data item. The application time is a set of one or more timestamps associated to the data item. Two data items can have the same application time. The one that generates the data stream. There are several categories of stream modelling according to the timestamps.

The first type of stream data is that there is no timestamp associated with data items. As depicted in Figure 1, the stream data is an ordered sequence of data items. Such kind of stream data supports stream queries which are only concerned about the sequence of events e.g. Does Alice meet Bob before Carl?

The second type of stream data is that each data item is associated with one timestamp, which is about when the data item occurs. In Figure 2, each data item has a timestamp. For example, the data item $e1$ (¡:alise, :isWith, :bob¿) occurred at time 1. Obviously such data model can be ustilised to answer queries in a time window. For example, How many people has Alice met in the last 5m?

One step further from the second type is that one data item can be associated with two timestamps. Let us look at the first data item $e1$ in this case. This time we not only know that $e1$ happened at time slot 1 but also we know that it was removed at time slot 5. So in this model, each data item has life cycle which is a period of time in which it holds true. Obviously this model supports both types of queries we discussed in the previous two models. In addition, it also supports queries asking the innervations among different data items.

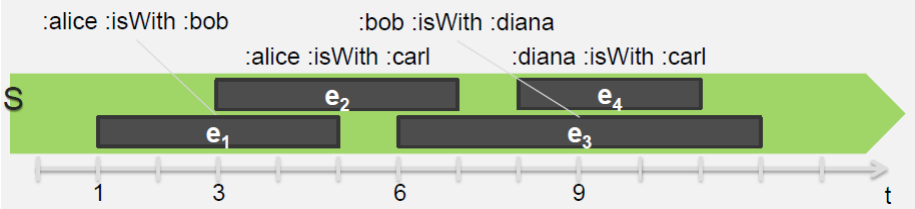

Figure 3: A RDF stream with two timestamp attributes

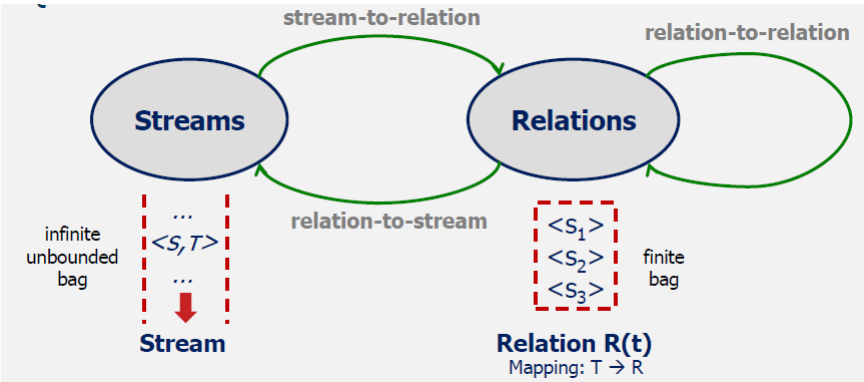

Figure 4: CQL Model

3.3.2 Continuous extensions of SPARQL

In database community, continuous query language model (cf. Figure 4) is introduced for querying data streams. At some specific time point or time period, the snapshots of stream data is captured as static relational data in a conventional database system. In this sense, the stream data, infinite unbounded bag of data, is converted into discrete snapshots each of which is a finite bag of relations. Such snapshot capturing is illustrated as stream-to-relation operation in Figure 4. Obviously, for each snapshot, the traditional query answering mechanisms could be used to answer queries. Hence the main task of supporting stream querying is turned to be selecting correct snapshots. To output streams, a reverse operation called relation-to-stream can be used to put the finite bag of relations into the infinite data stream by attaching timestamp on each relation.

Inspired by CQL model in database community, Semantic Web community also introduced similar ideas to processing RDF data streams. Depicted in Figure 5, the main difference is the two converting operations. For RDF data streams, the operations are called S2R window operators and R2S operators.

3.3.3 Overview of existing systems

There are several RDF Stream Processing systems.

- **C-SPARQL:** It applies a combined architecture with two components of RDF Store and Stream processor (cf. Figure 6).

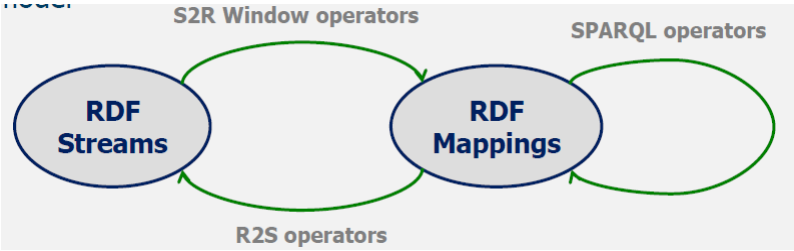

Figure 5: CQL Model for RDF Data Stream

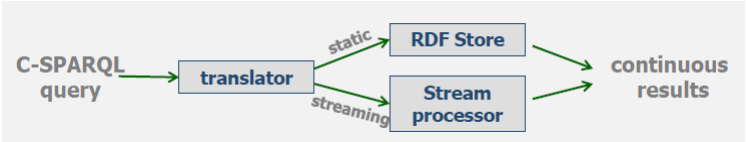

Figure 6: C-SPARQL

- **CQELS:** It was implemented from scratch. The main focus was on performance. The ideas is to use native and adaptive joins for static data and streaming data (cf. Figure 7).

- **EP-SPARQL:** This approach applies Event Processing in RDF Stream Processing. The authors implemented a complex-event detection method which is based on SEQ and EQUALS operators (cf. Figure 8).

- **SPARQLStream:** It is implemented based on the idea of ontology-based stream query answering. The idea is to generate virtual RDF views using R2RML mappings. Then, SPARQL stream queries can be implemented over the original data streams(cf. Figure 9).

The classification of existing systems can be found in Figure 10.

In particular the contrasting needs of adopting a logic expressive enough to represent dynamic domain and to support efficient reasoning and scalability arose; for this purpose DLs tailored for efficient reasoning have been considered, including OWL2 QL and EL+ profiles. Moreover in order to have a more precise domain description, rather than using restricted logics, OWL full has been used to write the ontologies and ontology approximation techniques have then been used.

Figure 7: CQELS

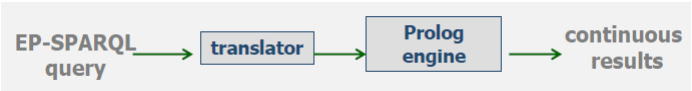

Figure 8: EP-SPARQL

Figure 9: $SPARQLStream$

	Model	Continuous execution	Union, Join, Optional, Filter	Aggregates	Time window	Triple window	R2S operator	Sequence, Co-ocurrence	Time function
TA-SPARQL	TA-RDF	✗	✔	Limited	✗	✗	✗	✗	✗
tSPARQL	tRDF	✗	✔	✗	✗	✗	✗	✗	✗
Streaming SPARQL	RDF Stream	✔	✔	✗	✔	✔	✗	✗	✗
C-SPARQL	RDF Stream	✔	✔	✔	✔	✔	✗	✗	✔
CQELS	RDF Stream	✔	✔	✔	✔	✔	✗	✗	✗
SPARQLStream	(Virtual) RDF Stream	✔	✔	✔	✔	✗	✔	✗	✗
EP-SPARQL	RDF Stream	✔	✔	✔	✗	✗	✗	✔	✗
Instans	RDF	✔	✔	✔	✗	✗	✗	✗	✗

Figure 10: Classification of existing systems

3.4 The summary of existing approaches

To conclude the investigation of the state-of-the-art, most existing approaches focus on handling continuous updating data. The extensions are mainly focused on either the data model level to support time attributes of data units (triples) or the query language level to enable querying specific data portions in the interested time spans. Little work has been focused on modelling the dynamics in domain knowledges, or in other words the T-Box level. In K-Drive's use cases, e.g. the HBCS use case, dynamics in T-Box level are very crucial. For example, the cancer treatment guidelines might change over time due to the advances in the medical techniques. Without the ability to capture such changes in the domain knowledge might lead to severe consequences. Hence, in K-Drive, in addition to handling dynamic data, we also have to propose techniques to handle the dynamics in the domain knowledge level.

4 Handling Dynamic Knowledge in K-Drive

The dynamics in K-Drive use cases are two-fold. On the one hand, data keeps coming and going in the healthcare systems. On the other hand, the cancer treatment guidelines, and the advances in oncological and medical sciences will result in constant updates in the domain knowledge. Most existing approaches surveyed in previous sections target the former dynamics. We can utilise these techniques in handling the data dynamics. For the latter type of dynamics, there is little techniques we can reuse. Hence, in this section, we propose our own solutions mainly focusing on dealing with the conceptual level dynamics.

Specifically, our solution is composed of two main techniques. In the knowledge modelling aspect, we propose a meta-level approach to enable modelling dynamic concepts in our domain. To support consistency-preserving knowledge updating, we propose a belief revision technique based on Datalog+/-
.

4.1 Meta-level approach

Case studies analysis outlined that dynamics may arise both at data level and at conceptual level therefore the KB should be able to:

- represent and store dynamic information (including time and space related concepts)
- handle schema level evolution
- support efficient reasoning and query answering
- support stream queries
- be scalable in order to store ever growing amount of information

These challenging aspects lead to investigate the state-of-the-art of relevant technologies related to description logics knowledge bases.

In this section we present a medical ontology aimed at considering dynamics from the domain point of view. In such ontology we modeled concepts form the HBCS case study such as symptoms, TPs and deviation form recommended TPs in a general framework able to represent fundamental concepts together with dynamic aspects related to succession of events and chains of planned actions. T his general framework has been obtained by building an ontology made of tree modules with decreasing level of abstraction as follows:

- Top Level: a foundation ontology
- Planning Level: an ontology which Allows descriptions of planning activities
- K-Drive Clinical Level: an ontology modeling the use case

In addition a meta-level has been defined to characterize ontology entities themselves by means of meta-properties. In the following we will describe the methodology that lead the design of this ontology and then will give a brief description of each level.

4.1.1 The Methodology and Meta-level Ontology

The methodology, we adopted to design the ontology which is being used in K-Drive, derives from OntoClean [13] and is aimed at considering domain dynamics. OntoClean defines formal and domain independent properties of concepts (i.e. meta-properties) that aid the knowledge engineer to make correct decisions when building ontologies and therefore to avoid common mistakes. OntoClean originally defines four meta-properties as follow:

- **Identity:** the criteria for defining 'sortal' classes; classes all of whose instances are identified in the same way
- **Unity:** a property that only holds of individuals that are 'wholes'
- **Rigidity:** a property that only holds of individuals that have it and cannot change, is 'essential'
- **Dependence:** a property that only holds for a class if each instance of it implies the existence of another entity

According the methodology those meta-properties are used to analyze concepts in order to check hierarchies correctness, for example a 'rigid' concept like Person cannot be subsumed by an 'not-rigid' concept like Physician.

Currently this methodology lacks in characterizing ontology elements from the dynamic point of view and we believe that refining it by adding meta-properties about dynamics can be useful, when evolving domains (and datasets) are involved, both to design a well formed ontology and also to support reasoning. As an illustrative example we may have an individual which acquires a 'status' that cannot be changed any more, such kind of property is not rigid but become rigid somehow once acquired; we can imagine the case of person which once vaccinated against poliomyelitis remains immunized throughout his life. We called such meta-property 'Reversibility'. In a stream reasoning context, knowing that a property is not reversible makes it possible to leverage previous results so to avoid re-computation.

Other extensions we are going to add are related to the cyclicality, trend (e.g. monotonic or non monotonic) and variability (i.e. frequency of changes) of properties which allow to distinguish entities according their dynamics. In K-Drive we started from the Standford implementation of OntoClean for Protégé [2] to design our Meta-Level ontology represented in figure 4.1.1

[2] http://protege.stanford.edu/ontologies/ontoClean/ontoCleanOntology.html

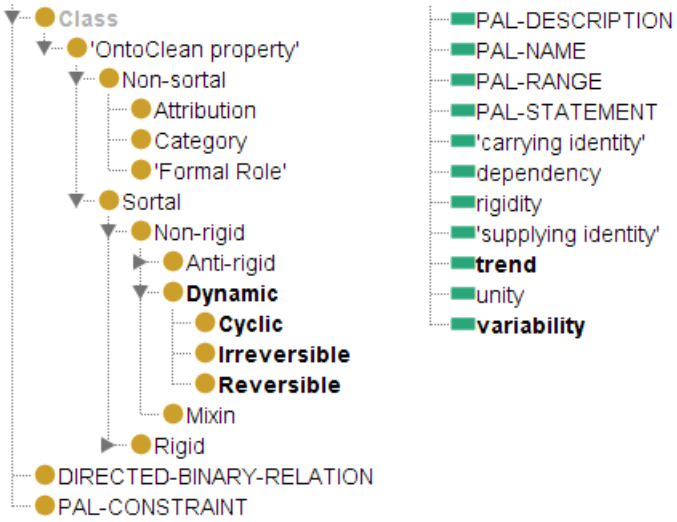

Figure 11: Meta Level

4.1.2 Ontology Description

Top Level Ontology

Foundational ontologies are formal conceptualizations aimed at describing general concepts, relations and constraints that can be considered common in any domain. Such kind of ontologies derives from philosophical studies about the reality which tries to answer to the question "What exists?", and are at the basis of the Semantic Web. Foundational (or upper) ontologies allow to give a formal and non-ambiguous description of a shared vocabulary to be used across different organizations and to support a coherent integration of heterogeneous models. Moreover such kind of ontologies allow to handle dynamics both related to domain evolution by isolating a solid and stable reference to core concepts and axioms which are not affected by updates, and also dynamics which belongs to the domain itself, such as temporal and spacial dynamics, by modeling related basic notions (e.g. TemporalRegion and Process). For this reasons in K-Drive we adopted a foundational ontology as first layer, in particular the TopLevel ontology we present has been derived from the "Lite" version of DOLCE [3] which is the first module of the WonderWeb Foundational Ontologies Library developed by the Laboratoy of Applied Ontology (CNR). In the figure 4.1.2 are shown the OWL Classes and the OWL DataProperties we included and a brief description of most important entities is given in the table 1.

Table 1 – continued from previous page

OWL Classes	
Entity	Anything is identifiable by humans as an object of experience or thought
	Continued on next page

[3] Descriptive Ontology for Linguistic and Cognitive Engineering `http://www.loa.istc.cnr.it/DOLCE.html`

Table 1 – continued from previous page

Collection	Arbitrary sum of entities
NonPhysicalEntity	Entities out of the physical space, possibly localizable in time, like abstractions, ideas, concepts.
Quality	Characterization of individual qualities
Region	A non-empty, open, mutually related set of symbols
Relation	Abstract collection
SocialEntity	A.k.a. 'social objects': entities established within human societies as a result of any kind of 'linguistic act' and recognized on the basis of any kind of 'collective intentionality' (Searle)
GeopoliticalEntity	A geographical area associated with some sort of political structure
InformationEntity	A social (abstract) entity which conveys information, through one or more (concrete) realizations
Language	Set of rules and conventions to rule communication or other information flow
Measure	Description of a set of coordinated measurement values
Organization	A group of people consciously cooperating
SemioticEntity	Social entities that signify something
Description	A complex semiotic entity that encodes a conceptualization, and is conveyed by some information object
Classification	A systemic set of classifiers
Regulation	Legislative regulations such as taxation
Sign	Atomic semiotic entity, conveyed by some information object
SocialStatus	The honor or prestige attached to one's position in society
PtisicalEntity	Entities localizable in space and time, like objects or events
Continuant	Concrete entities without temporal parts
Group	Concrete collection
Occurrent	Concrete entities with temporal parts

OWL Object Properties	
relatedTo	Generic relatedness
dependentOn	Existential dependency
conveyedBy	Dependency of semiotic entities from information entities where they are conveyed
yieldedBy	Abstract relationship that refers to the execution by which an artifact was yielded
instanciatedBy	A concrete instance which is represented by 'something'
involves	The relationship of an Event with the Entities it involves
agent	Dependence of a process upon an object that 'intentionally' initiates it. Defining 'intentionality' is out of the scope of this ontology
location	Relationship of a Physical entity of being in a Spacial Location
part	Relationship that binds an entity (domain) to other, possibly non-dependent, entities to which it supplies some unity criterion
abstractPart	Non spatio-temporal parts

Continued on next page

Table 1 – continued from previous page

member	Collection membership
concrete art	Spatio-temporal parts
constituent	For objects, the relation with other objects whose lack would cause the whole to cease, or change identity, or change relevant properties
phase	For events, the relation with their temporal slices
signifies	The relationship of a social entity with other entities it stands for, under some respect and within some communication context
denotes	Signification that refers to explicit criteria
describes	The relationship between a description and the state of affairs that may correspond to it from the standpoint of some social & agent
refers	Signification that associates a sign to a specific object, under some naming convention

Table 1: Main Top Level Enties

Planning Level Ontology

Once we defined the basic ontology framework we added a layer to represent clinical guidelines that play an essential role in the HBCS use case, we described in WP2 and briefly introduced in section 2.2. Medical guidelines are recommendation indicating the most appropriate sequence of treatment to be performed by physicians, in the form of preferred choices or normative rules, and can have different grain from simple text to formal workflow description. In [14] is shown how guidelines can be theoretically modeled as workflow patterns that have to be applied according to a given healthcare situation. As a consequence of describing guidelines in terms of plans and procedures become possible to discover deviations by comparing them against effective tasks performed by physicians. The Planning Ontology Level include the abstract representation of a Plan composed by Procedures organized into Sequences. In the following figure 4.1.2 we show OWL Classes and the OWL DataProperties that can be used to define abstract plans while in the table 2 a brief description of most important entities is given

Table 2 – continued from previous page

OWL Classes	
Plan	The description of a coordinated sequence of actions (tasks)
Procedure	The description of a kind of action as part of a plan
ProcedureRelation	The description of a relationship between tasks
Alternative	The relationship between procedures that may be alternatives within some plan
Dependency	The relationship between a given procedure and the ones that must be accomplished in order for it to get started
Sequence	The relationship between procedures that should be subsequent within some plan
FollowUp	A sequence of two adjacent and related procedures

Continued on next page

Table 2 – continued from previous page

EvaluationResult	The result of an evaluation process
Action	Processes initiated by some agent
DescriptionEvaluation	The evaluation of a process with respect to a plan or procedure
PlanEvalutaion	The description of the execution of a plan with respect to a process
ProcedureEvaluation	The description of the execution of a procedure with respect to an action

OWL Object Properties	
followingTask	A relation that holds between a task and the nearest follower in a plan. If t1 follows t2, then exist an instance of task atomic sequence.
evaluatedDescription	relates plans or procedures with their evaluation
evaluatedProcess	relates processes or actions with their evaluation
evaluationResult	relationship to report the result of an evaluation
hasTask	membership relation of a task belonging to a plan
finalTask	final task of a plan
initialTask	initial task of a plan
participantInPlan	Humans of a Social Role participation in a plan or a procedure
leaderInPlan	Humans of a Social Role leading a plan or a procedure
predecessor	abstract predecessor relationship for sequences
successor	abstract successor relationship for sequences

Table 2: Main Planning Level Entities

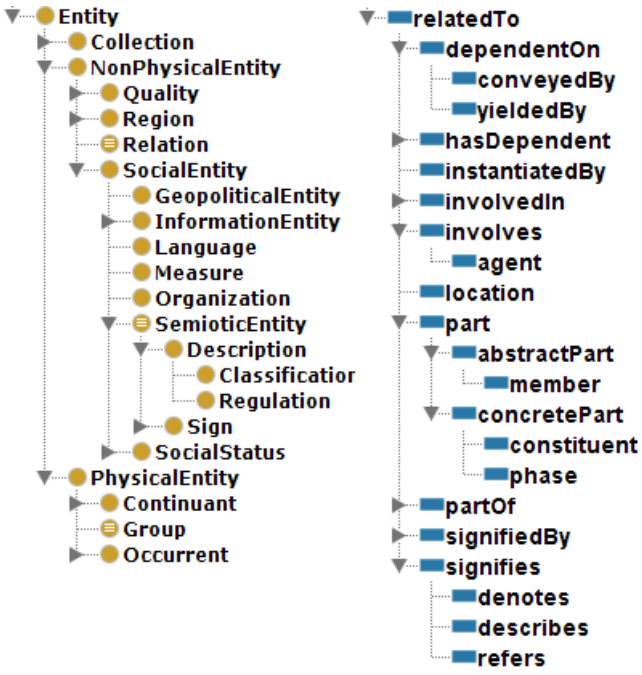

Figure 12: Top Level

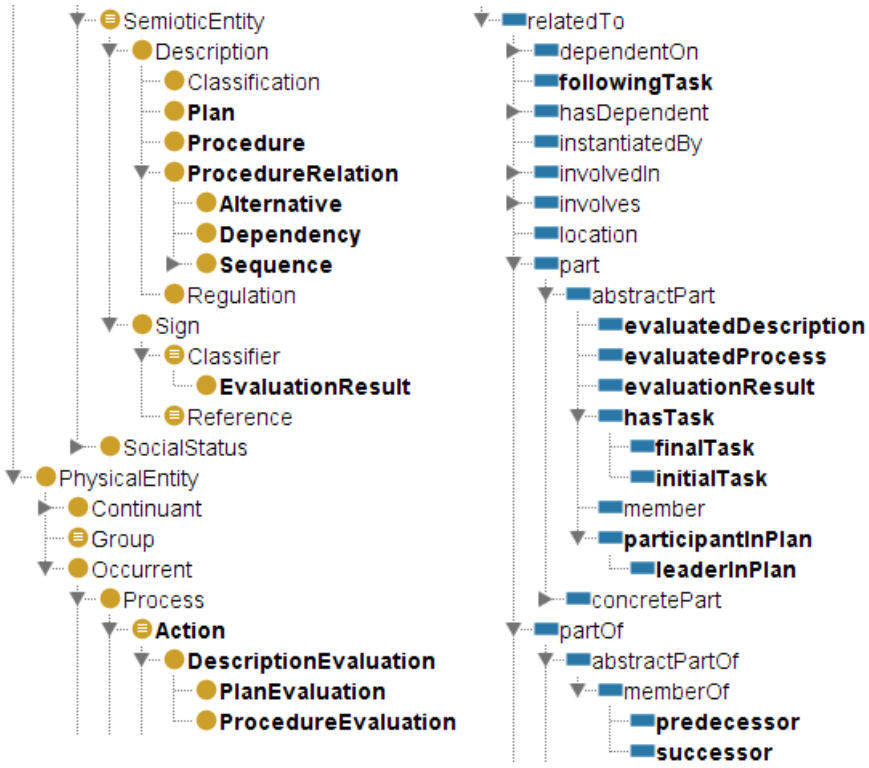

Figure 13: Planning Level Entities

K-Drive Clinical Level Ontology

Finally the K-Drive Clinical Ontology contains the most specific entities which accommodates HBCS use case terminology. Here we present a first version of the ontology, mainly focused on treatment programs and medical events, which serves as template for the knowledge-base development. Treatment programs are represented as ('is-a') plans and comprises a sequence of at least one treatment which 'is-a' procedure; treatments are then specialized according the use case in Chemo, Radio and Surgery. Similarly performed treatments are modeled as medical events which are actions performed by agents; and again medical events are specialized in chemo and radio therapies and surgery operation. Through this model and leveraging on the upper ontology, we consider planned treatments as expected 'descriptions' of happened medical events and we are able to evaluate this events sequences against deviations from related plans (e.g can be discovered that an extra radio therapy has been supplied). In the figure **??** main OWL Classes are shown. Finally figure 4.1.2 displays how individuals are instanced, note that each individual also references linked datasets throw an identifier. In the example a Treatment-Program has been added together to an applied treatment (PlanEvaluation) and can be seen the program composed by a Surgery and a Chemo while only a ChemoTherapy has been performed.

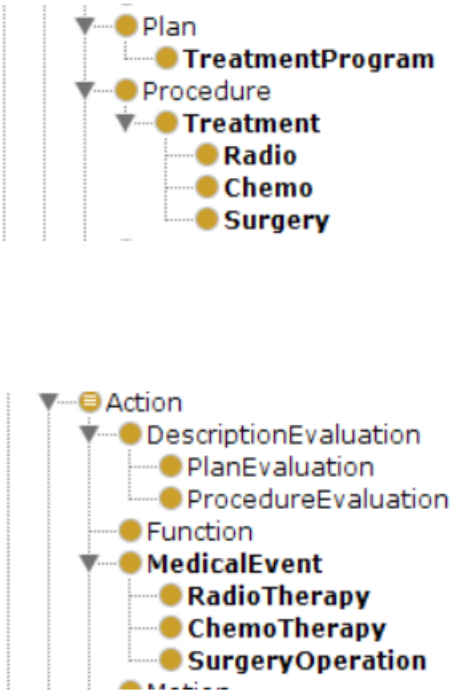

Figure 14: HBCS specific Classes

An example on how Individuals are instanced is shown in figure 4.1.2 in which

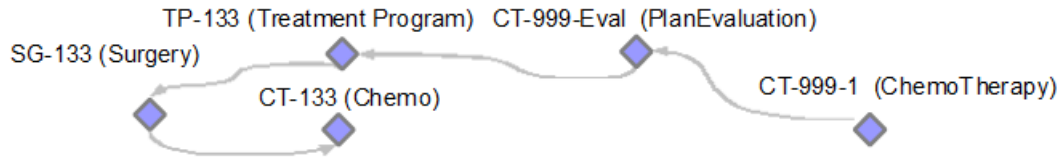

Figure 15: An example: Adding Idividuals

4.2 Belief Revision approaches

In this section we investigate the dynamic knowledge base, in which data/knowledge may evolve over time, from the point of view of Belief Revision. Belief revision deals with the problem of adding new information to a knowledge base in a consistent way. Particularly, our study is based on Datalog+/- ontologies.

4.2.1 Belief Revision

In general a *belief* is a piece of knowledge or a logic result in the Knowledge Base. *Belief revision* is the process of changing beliefs in order to take into account a new piece of knowledge.

In general, there are two kinds of changes in a dynamic knowledge base:

update Old beliefs are always overlapped by new beliefs. In the manner of "update", the new information is about the situation at present, while the old one refers to the past. "Update" is the operation of changing the old beliefs and add the new belief;

revision Old beliefs are regarded being less reliable than the new one, if there is an inconsistency between them. "Revision" is the process of inserting the new information into the set of old beliefs and avoiding inconsistency.

Given a knowledge Base K, and a new observation o,

4.2.2 Datalog+/- ontologies

Datalog+/- [3] has its origin from database technologies, encompasses and generalizes the tractable description logics EL and DL-Lite, which can be used to represent lightweight ontologies in Semantic Web [6].

Datalog+/- enables a modular rule-based style of knowledge representation. Its properties of decidability of query answering and good query answering complexity in the data complexity allows to realistically assume that the database **D** is the only really large object in the input. These properties together with its expressive power make Datalog+/- a useful tool in modelling real applications such as ontology querying, web data extraction, data exchange, ontology-based data access and data integration.

For both classic logic and description logic, belief revision has been studied intensively and many literature exist, such as [1, 7]. There are some extensions of Datalog+/- to deal with incomplete or inconsistency information, including inconsistent handling method [4], probability extension [5, 8], and well-founded semantics extension [9], however, to the best of our knowledge, there is no belief revision method for Datalog+/- before.

In this section, we address the problem of belief revision for Datalog+/- ontologies and propose a *kernel-incision* based belief revision operator. Kernel consolidation was originally introduced by Hansson [10] based on the notion of kernels and incision function. The idea is that, given a knowledge base KB that needs to be consolidated (*i.e.*, KB is inconsistent), the set of kernels is defined as the set of all minimal inconsistent subsets of KB. For each kernel, a set of sentences is removed (i. e. , an incision is made) such that the remaining formulas in the kernel are consistent. Note that it is enough to remove any single formula from the kernel because they are minimal inconsistent sets. The result of consolidating KB is then the set of all formulas in KB that are not removed by the incision function.

We adopt the kernel-based consolidation idea into Datalog+/- ontologies, and give an approach to deal with belief revision in the face of new information which contains two parts of message: (i) the *new facts* A, and (ii) the *unwanted set* Ω. With our belief revision operator, a Datalog+/- ontology KB is able to be undated by adding new facts A in its database, and removing some database element to prevent any result in the unwanted set Ω, such as inconsistency $\bot$. We study the properties of proposed approach using extended postulates, and then give algorithms to revise the Datalog+/- ontologies. We finally give the complexity results by showing that query answering for a revised Datalog+/- ontology is tractable if a linear KB is considered.

The paper is organized as follows. In Section 2 we introduce some preliminary knowledge about Datalog+/-. In Section 3 we give our revision operator on Datalog+/- ontologies. The properties of the operator are investigated in Section 4. In Section 5 we give the algorithm and provide complexities results. Related works and the conclusion are given in Section 6 and 7 respectively.

4.2.3 Preliminaries on Datalog+/-

In this section, we briefly recall some necessary background knowledge on Datalog+/-.

Databases and Queries We assume (i) an infinite universe of *(data)constants* constants Δ (which constitute the normal domain of a database), (ii) an infinite set of *(labelled) nulls* Δ_N (used as fresh Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (unique name assumption), while different nulls may represent the same value. We also assume that there is a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ sequences of variables $X_1, ..., X_n$ with $k \geq 0$.

We assume a relational schema $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula*(or *atom*) **a** has the form $P(t_1, ..., t_n)$, where P is an n-arry predicate, and $t_1, ..., t_n$ are terms.

We assume a *Database (instance) D* for $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. A *conjunctive query* (CQ) over R has the form $Q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but without nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu : \Delta \cup \Delta_N \cup \mathcal{V} \rightarrow \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$ and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms.

The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples t over Δ for which there exists a homomorphism $\mu : \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu : \Phi(X, Y) \subseteq D$ and $\mu(X) = t$. The *answers* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, *a tuple-generating dependency* (TGD) σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and *head* the head of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D , there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. All sets of TGDs are finite here. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ.A TGD σ is *linear* iff it contains only a single atom in its body. *Query answering* under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of models of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. It is proved that query answering under general TGDs is undecidable, even when the schema and TGDs are fixed.

For a BCQ Q we say that $(D \cup \Sigma)$ *entail* Q if answer for a BCQ Q to D and Σ is *Yes*, $D \cup \Sigma$ *entail* a set of BCQs if it entail some element of it.

Negative constraints (NC): A negative constraints γ is a first-order formula $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the body of γ) is a conjunction of atoms (without nulls and not necessarily guarded).

Equality-generating dependencies (EGD): A equality-generating dependency σ is a first-order formula of the form $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of Σ, denoted $body(\sigma)$, is a (without nulls and not necessarily guarded) conjunction of atoms, and X_i and X_j are variables from X . Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y}) \subseteq D$, it holds that $h(X_i) = h(X_j)$.

The Chase The *chase* was first introduced to enable checking implication of dependencies, and later also for checking query containment. It is a procedure for repairing a database relative to a set of dependencies, so that the result of the chase satisfies the dependencies. By *chase*, we refer both to the chase procedure and to its result.

The chase works on a database through TGD and EGD chase rules. Let D be a database, and Σ a TGD of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. Then, Σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(X, Y)$ to atoms of D. Let Σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$ where z_j is a fresh null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* Σ on D adds to D the atom $h_1(\Psi(X, Z))$ if not already in D.

The chase algorithm for a database D and a set of TGDs consists of essentially an exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) chase for D and Σ.

The chase relative to TGDs is a *universal model*, that means, there exists a homomorphism from $chase(D, \Sigma)$ onto every $B \in mods(D, \Sigma)$. This implies that BCQs Q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models Q$ is equivalent to $chase(D, \Sigma) \models Q$. For guarded TGDs Σ, such BCQs Q can be evaluated on an initial fragment of $chase(D, \Sigma)$ of constant depth k$|Q|$, which is possible in polynomial time in the data complexity.

Datalog+/- ontology A *Datalog+/- ontology* KB=(D, Σ), where D is database, $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a database D, a set of TGDs Σ_T, a set of non-conflicting EGDs Σ_E, and a set of negative constraints Σ_{NC}. We say KB is *linear* iff Σ_T is linear. A Datalog+/- ontology KB=(D, Σ) is *consistent*, iff $model(D, \Sigma) \neq \emptyset$, otherwise it is inconsistent.

4.2.4 Belief base revision in Datalog+/- ontology

We give an approach for performing belief base revision for Datalog+/- ontology in this section. We present a framework based on kernels and incision functions to deal with the problem of belief revision for Datalog+/- ontologies.

4.2.5 Revision

Firstly we give the definition of kernel in Datalog+/- ontologies.

Definition 2 ***Kernel*** *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω. A kernel is a set $c \subseteq D \cup A$ such that (c, Σ) entail Ω, and there is no $c' \subset c$ such that (c', Σ) entail Ω. We denote $(D \cup A) \perp\!\!\!\perp \Omega$ the set of all kernels.*

Example 1 *A (guarded) Datalog+/- ontology KB=(D, Σ) is given below. Here, the formulas in Σ_T are tuple-generating dependencies (TGDs), which say that each person working for a department is an employee (σ_1), each person that directs a department is an employee (σ_2), and that each person that directs a department and works in that department is a manager (σ_3). The formulas in Σ_{NC} are negative constraints, which say that if X supervises Y, then Y cannot be a manager (ν_1), and that if Y is supervised by someone in a department, then Y cannot direct that department (ν_2). The formula Σ_E is an equality-generating dependency (EGD), saying that the same person cannot direct two different departments.*

D=$\{directs(tom, d_1), directs(tom, d_2), worksin(john, d_1), worksin(tom, d_1)\}$;
Σ_T=$\{\sigma_1 : worksin(X, D) \rightarrow emp(X), \sigma_2 : directs(X, D) \rightarrow emp(X),$
$\sigma_3 : directs(X, D) \wedge worksin(X, D) \rightarrow Manager(X)\}$;
Σ_{NC}=$\{\nu_1 : supervises(X, Y) \wedge manager(Y) \rightarrow \bot,$
$\nu_2 : supervises(X, Y) \wedge worksin(X, D) \wedge directs(Y, D) \rightarrow \bot\}$;
Σ_E=$\{\nu_3 : directs(X, D_1) \wedge directs(X, D_2) \rightarrow D_1 = D_2\}$.

If new observation is supervises(tom, john), unwanted sentence is emp(tom), then kernel is $c_1 = \{worksin(tom, d_1)\}$.
If new observation is supervises(tom, john), unwanted atom is $\bot$, then kernels are
$c_1 = \{supervises(tom, john), direct(tom, d_1), worksin(john, d_1)\}$
$c_2 = \{supervises(tom, john), direct(tom, d_1), worksin(tom, d_1)\}$
$c_3 = \{direct(tom, d_1), direct(tom, d_2)\}$.

We then give the definition of incision function in Datalog+/- ontologies.

Definition 3 (Incision Function) *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω, an incision function is a function σ that satisfies the following two properties*

1. $\sigma((D \cup A) \perp\!\!\!\perp \Omega) \subseteq \bigcup((D \cup A) \perp\!\!\!\perp \Omega)$.
2. *if* $X \in ((D \cup A) \perp\!\!\!\perp \Omega)$ *then* $X \cap \sigma((D \cup A) \perp\!\!\!\perp \Omega) \neq \emptyset$.

We now give the definition of revision operator. The idea is to add the new information to ontology and then cut unwanted information from it in a consistent way. Intuitively, the revision intends to add A and avoid Ω in a rational way. Note that we cut unwanted database Ω, which is a generation of just cutting $\bot$ from the ontology to avoid the contravention. In this point we are like Ribeiro *et al.*[11].

Definition 4 (Revision Operator) *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω, let $D' = (D \cup A) \backslash \sigma((D \cup A) \perp\!\!\!\perp \Omega)$, where σ is an incision function, then revised ontology $KB * A = (D', \Sigma)$.*

Example 2 *We continue with example 1. If new observation is supervises(tom, john), unwanted atom is emp(tom), let incision function be $\{worksin(tom, d1)\}$, then revised database is*
$D' = \{directs(tom, d_1), directs(tom, d_2), worksin(john, d_1)\}$.
If new observation is supervises(tom, john), unwanted atom is $\perp$,
let incision function be $\{supervises(tom, john), direct(tom, d_2)\}$, then revised database is
$D' = \{directs(tom, d_1), worksin(john, d_1), worksin(tom, d_1)\}$.

4.2.6 General Properties

It is important to ensure the revision operator behave rationally, so we analyze some general properties of it in this section. We first propose the revision postulates for Datalog+/- ontologies, which is an adaptation of known postulates for semi-revision, then prove that these new postulates are indeed satisfied by the operator.

Definition 5 (Postulates) *Given a Datalog+/- ontology $KB = (D, \Sigma)$, a new observation database instance A, an unwanted database instance Ω, let $KB \diamond A = (D', \Sigma)$ be the revised ontology, then the postulates are:*

1. *(Consistency) Any element of Ω is not entailed by $KB \diamond A$.*
2. *(Inclusion)) $D' \subseteq D \cup A$.*
3. *(Core-retainment) if $\beta \in D$ and $\beta \notin D'$, then there is D'' such that $D'' \subseteq D \cup A$, $(D'' \cup \{\beta\}, \Sigma)$ entail Ω, but (D'', Σ) does not entail Ω.*
4. *(Internal change) If $A \subseteq D, B \subseteq D$ then $KB \diamond A = KB \diamond B$.*
5. *(Pre-expansion) $(D \cup A, \Sigma) \diamond A = KB \diamond A$.*

The intuitive meaning of the postulates are as follows: *consistency* means no atom in the unwanted database should be entailed; *inclusion* means that no new atoms were added; *core-retainment* means if an atom is deleted, then there must be some reason; *internal change* means that every time an ontology is revised by any of its own elements of database, then the result ontology should be same; *pre-expansion* means that if an ontology is expanded by a database and then revised by it, the result should be the same as revising the original ontology by the database.

We now prove that the revision operator given in the last section satisfy these postulates.

Theorem 1 *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance* A*, unwanted database instance Ω, let $KB * A = (D', \Sigma)$ be the revised ontology defined as above, then the revision satisfies the postulates in Def. 4.2.6.*

Proof 1 *Inclusion, internal change, pre-expansion follows directly from the construction. To prove consistency, assume by contradiction that it is not. Then there is an element of Ω that is entailed by $KB \diamond A$, as Datalog+/- is a fragments of first-order logic, from compacity of first-order logic it follows that there is a $Z \in D \cup A$, such that there is an element of Ω that is entailed by (Z, Σ). We can then infer by monotonicity that there is a $Z' \subseteq Z$ such that $Z' \in (D \cup A) \perp\!\!\!\perp \Omega$. Then we must have $Z' \neq \emptyset$, and by*

construction there must be $\epsilon \in \sigma((D \cup A)) \cap Z'$*, but if this is true then* $\epsilon \notin (D \cup A)$ *and* $\epsilon \in Z' \subseteq (D \cup A)$*, which is a contradiction. To prove core-retainment, we have that if* $\beta \in D$ *and* $\beta \notin D'$*, then there is* $\beta \in \sigma((D \cup A) \perp\!\!\!\perp \Omega)$*, That is, there is a* $X \in (D \cup A) \perp\!\!\!\perp \Omega$ *such that* $\beta \in X$*. Considering* $D'' = X/\beta$*, then* $D'' \subseteq D \cup A$*,* $(D'' \cup \{\beta\}, \Sigma)$ *entail* Ω*, but* (D'', Σ) *does not entail* Ω*.*

4.2.7 Algorithms

In the section, we first give an algorithm to compute all kernels when an atom is unwanted, and then give the algorithm of revision. We deal with *linear* ontology in this section.

Algorithm 1 :Computing kernels for an atom

Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted atom ω, we give in this subsection a method to calculate all kernels $(D \cup A) \perp\!\!\!\perp \omega$.

We first give the method when unwanted atom λ is a node in *Chase* of $(D \cup A, \Sigma)$. The idea is to travel chase graph bottom-up and then cut non-minimal ones from result sets of atoms. Note that this idea is similar to the one used in Lukasiewicz etc [4], which use a bottom-up travel of chase graph starting from the matching a body every rule of Σ_{NC} to compute the culprits of an inconsistent ontology.

We now give algorithm Justs(λ). Min(Justs(λ)) are exactly kernels $(D \cup A) \perp\!\!\!\perp \lambda$, where Min is used in the usual sense of subset inclusion. Note that *just* below is not a set of nodes, but a set of sets of nodes. Note also that step 2 can be done because we can collect all information needed from *Chase*.

We now have the following theorem.

Theorem 2 *Given a linear Datalog+/- ontology* $KB = (D, \Sigma)$*, new observation database instance A, Let* λ *be a node in chase graph of Datalog+/- ontology* $(D \cup A, \Sigma)$*, then Min(Justs(*λ*))=*$(D \cup A) \perp\!\!\!\perp \lambda$*. The algorithm run in polynomial time in the data complexity.*

Proof 2 *We first prove that* $(Min(Justs(\lambda)), \Sigma)$ *entail* λ*, that is, for every model M of* $(Min(Justs(\lambda)), \Sigma)$*, there is a homomorphism* μ *such that* $\mu(\lambda) \subseteq M$*.*

In fact, we will show that for all nodes that were produced in the bottom-up travel process, it holds that the node is entailed by $(Min(Justs(\lambda)), \Sigma)$*, Then, as a result, the* λ *is also entailed by* $(Min(Justs(\lambda)), \Sigma)$ *automatically.*

Suppose the derivation level of λ *is N. We prove level by level from 1 to N. If the node level is 1. let M be a model of* $(Min(Justs(\lambda)), \Sigma)$*, then* $M \supseteq Min(Justs(\lambda))$*, but from the definition of satisfaction of a rule, whenever there is a homomorphism that map the body to the database, there is a extended homomorphism that map the head to the database,so, this homomorphism map the node to M, the entailment holds.*

Suppose for all node whose derivation level is n, it is right. That is, there is a a homomorphism that map the node to M, now we consider the node whose derivation level is n+1. Consider the rule that applicable and can get this node, since all parent nodes of the node has a level smaller or equal to n, so there is a homomorphism that map these nodes to M, as the rule itself is satisfied by M, we can then construct a new homomorphism by extend the above homomorphism to map the node to M. So we have $(Min(Justs(\lambda)), \Sigma)$ *entail the node.*

We now show that there are no other subsets of $D \cup A$ *that along with* Σ *entail* λ *and is smaller than Min(Justs(*λ*)). Otherwise, suppose it is not, that is, there is a subset of* $D \cup A$ *that is smaller than Min(Justs(*λ*)) and along with* Σ *entail* λ*. Then we have a ChaseGraph that end with* λ*, however, according to the construction of the Min(Justs(*λ*)), this set should be equal to some element of Min(Justs(*λ*)), this is a contraction.*

Algorithm 1 Justs(λ)

Require: a linear Datalog+/- ontology $KB = (D, \Sigma)$, a new observation database instance A, and a node λ in *Chase* of $(D \cup A, \Sigma)$.

Ensure: Justs(λ)

```
just = ∅
for all Φ_i ⊆ (nodes in Chase) such that there is rule r : Φ(X, Y) → ∃Z Ψ(X, Z) which is applicable to Φ_i and produces λ do
    just = just ∪ {Φ_i}
end for
for all Φ_i ∈ just do
    for all Φ_i^j ∈ Φ_i do
        if Justs(Φ_i^j) ≠ ∅ then
            just = Expand(just, Φ_i^j)
        end if
    end for
end for
return just
```

```
Expand(just, a)
for all φ ∈ just do
    if a ∈ φ then
        just = just/φ
        for all j_a ∈ Justs(a) do
            just = just ∪ {φ/a ∪ j_a}
        end for
    end if
end for
return just
```

We finally shows that computing Min(Justs(λ)) in the linear case can be done in polynomial time in the data complexity. Note that the ChaseGraph is constant-depth and polynomial-size for a linear ontology due to the result in [3]. Note also that Justs(λ) is a recursive procedure, it will be called $N \times M$ times at most, where N is the depth of the graph and M is the numbers of rules in the Σ, and that at each time the algorithm is running, the time complexity exclusive the recursive procedure is polynomial, so the algorithm Min(Justs(λ)) run in polynomial time.

We now give the algorithm Kernels(ω) to compute kernels for an atom ω. Note that by match(ω, *Chase*) we mean the procedure of finding the same node as ω in *Chase*, if it is successful, return this node, otherwise return $\emptyset$.

Theorem 3 (Correctness and Complexity of Kernels(ω)) *Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation A, an unwanted atom ω, algorithm Kernels(ω) compute $(D \cup A) \perp\!\!\!\perp \omega$ correctly in polynomial time in the data complexity.*

Proof 3 *If L=$\emptyset$, then $KB' = (D \cup A, \Sigma)$ do not entail the atom as ChaseGraph is sound and complete with respect to query answering. If L$\neq \emptyset$ then the atom is entailed by KB', in this case, MinJust(L) are*

Algorithm 2 Kernels(ω)

Require: a linear Datalog+/- ontology $KB = (D, \Sigma)$, a new observation A, and an atom ω
Ensure: Kernels(ω)
1: compute the *Chase* of $KB = (D \cup A, \Sigma)$
2: L=match(ω, *Chase*)
3: **if** L=$\emptyset$ **then**
4: return $\emptyset$
5: **else**
6: return Min(Justs(L))
7: **end if**

all minimal sets of atoms that belong to database $D \cup A$ and along Σ entail ω according to theorem 1. So Kernels(ω) compute $(D \cup A) \Subset \omega$ correctly in both cases.

The complexity of the algorithm depends on the match procedure, as the ChaseGraph is polynomial-size and constant-depth for a linear ontology, the travel of ChaseGraph can be done in polynomial time, so the Kernels(ω) can be done in polynomial time.

Algorithm 2 :Revision

We now give the algorithm to revise a linear Datalog+/- ontology named as *RevisionKB*.

Algorithm 3 RevisionKB

Require: a linear Datalog+/- ontology $KB = (D, \Sigma)$, a new observation A, unwanted instance Ω
Ensure: Revised ontology $KB * A$
1: **for** every atom ω, $\omega \in \Omega$ **do**
2: compute Kernels(ω)
3: **end for**
4: get all combinations of Kernels(ω, every combination corresponds to a way of choosing each element from Kernels(ω), where $\omega \in \Omega$.
5: $(D \cup A) \Subset \Omega$=minimal subsets of all combinations
6: $KB * A$=$((D \cup A) \backslash \sigma((D \cup A) \Subset \Omega), \Sigma)$

Note that $((D \cup A), \Sigma)$ may be an inconsistent ontology, however, inconsistency can be removed in the revised ontology by making $\bot \subseteq \Omega$.

We now show that the algorithm can compute revision of ontology and give the complexity.

Theorem 4 (Correctness and Complexity of RevisionKB) *Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω, algorithm RevisionKB can compute revision correctly in polynomial time in the data complexity.*

Proof 4 *Note that $(D \cup A) \Subset \Omega$ can be obtained by combining $(D \cup A) \Subset \omega$ for every elements $\omega \in \Omega$ and cut from it the non-minimal ones, the algorithm's correctness then follows directly from the definition of revision operator.*

The complexity of the algorithm depends basically on the task of finding Kernels(ω), as it run in polynomial time due to Theorem 3, so we have the conclusion.

4.2.8 Related Works in Belief Revision

There is strong relationship between Datalog+/- ontology and description logic as they can be translated to each other in many cases. In the area of belief revision in description logic, Ribeiro *et al.* [11] bear much similarities to our work since they also used a kernels-based semi-revision method, they give a belief revision method to a monotonic logic, take description logic as a spacial case. However, there are difference between their work and this paper. They deal with monotonic logic, but Datalog+/- has a different syntax and semantic and cannot been cover by it. Furthermore, Ribeiro *et al.* [11] compute kernels by invoke classical reasoning, but this paper give a direct method to calculate kernels and prove that the complexity of computing revision is tractable.

In the area of Datalog+/-, Lukasiewicz *et al.* [4] give an inconsistency reasoning method for Datalog+/- ontologies, theirs work is close related to ours work since they use culprit to resolve the inconsistency of ontologies, and the culprit is equivalent with the kernel of this paper in the case of atom unwanted is $\bot$. However, there are some difference. Although the area of belief change is closely related to the management of inconsistent information, they are still quite different in both goals and constructions. Inconsistency can be handled by using kernels and clusters in [4], ours work can also deal with inconsistency, however revision operator given in ours work can do more except this, for example, it can choose a set of atoms as unwanted information, not only $\bot$, thus give more flexibility to resolve the inconsistency, and in this sense this work is more general than theirs work. Note also that in [4] the properties of the reasoning result are not clear even in the case of inconsistency handling because they did not study the properties of the operation from the viewpoint of belief revision.

There are still some works that extend Datalog+/- with the capability of dealing with uncertainty. In Lukasiewicz *et al.* [5], they developing a probabilistic extension of Datalog+/-. This extension uses Markov logic networks as the underlying probabilistic semantics and focus especially on scalable algorithms for answering threshold queries. Riguzzi *et al.* [8] apply the distribution semantics for probabilistic ontologies (named DISPONTE) to the Datalog+/- language. Lukasiewicz *et al.* [9] tackle the problem of defining a well-founded semantics for Datalog rules with existentially quantified variables in their heads and negations in their bodies, thus provide a kind of nonmonotonic reasoning capability to Datalog+/-. Our work also deal with the commonsense reasoning in the background of Datalog+/- language, however, we focus on the problem of belief revision, instead of adding quantitative or qualitative uncertainties to ontologies.

4.2.9 Summary and Outlook

In this section, we address the problem of belief revision for Datalog+/- ontologies. In our approach, we introduce a kernel based belief revision operator, and study the properties using extended postulates, we then provide algorithms to revise Datalog+/- ontologies, and give the complexity results by showing that query answering for a revised linear Datalog+/- ontology is tractable.

In the future, we plan to study how to implement belief revision when some heuristic information, *i.e.*, different trust [12] or reputation [2] levels of both database and rule set due to the different source of information, can be added to Datalog+/- ontologies.

4.3 The Dataset

De-identified data from the HBCS has been enriched by extracting time related information (e.g. guidelines creation and expiration dates, medical events...), mapped to the ontology and converted into RDF format. Due to the intelligent property restrictions, we cannot present the details of the dataset in this public deliverable.

5 Conclusion

This deliverable reports research activities, performed in WP5, aimed at providing K-Drive with a underling extendible knowledge base which is not only able to be a test bed, in which to apply latest query generation techniques, but also enables giving evidence of challenges which can arise when dealing with a concrete and crucial domain as the oncological one can be. The outcome of this deliverable is a solution of two main components. The first one is a specification, for the definition of a dynamic KB; such specification has been presented by means of technical approaches and as a methodology for ontology design. The first version of the medical ontology has been included. The second component is a consistency-preserving knowledge updating technique based on Datalog +/-.

Acknowledgement

This research has been funded by the European Commission within the 7th Framework Programme/Marie Curie Industry-Academia Partnerships and Pathways schema/PEOPLE Work Programme 2011 project K-Drive number 286348 (cf. `http://www.kdrive-project.eu`).

References

[1] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis and Holger Wache. *Inconsistencies, Negations and Changes in Ontologies*. In Proc. of the 21st National Conference on Artificial Intelligence (AAAI-06), 1295-1300. 2006.

[2] Andrew Koster and Jeff Z. Pan. *Ontology, Semantics and Reputation*. In Agreement Technologies, ISBN 978-94-007-5582-6, Springer. 2013.

[3] Thomas Lukasiewicz, Andrea Cali and Georg Gottlob, *A General Datalog-Based Framework for Tractable Query Answering over Ontologies*. Journal of Web Semantics, 2012, Vol 14, Pages 57-83

[4] Thomas Lukasiewicz, Maria Vanina Martinez and Gerardo I. Simari, *Inconsistency Handling in Datalog+/- Ontologies*, in the Proceedings of the 20th European Conference on Artificial Intelligence (ECAI) 2012. Pages 558-563.

[5] Thomas Lukasiewicz, Maria Vanina Martinez and Gerardo I. Simari, *Query Answering under Probabilistic Uncertainty inDatalog+/- Ontologies*, Annals of Mathematics and Artificial Intelligence, 2013, Pages 195-197

[6] Jeff Z. Pan, Edward Thomas, Yuan Ren and Stuart Taylor. *Exploiting Tractable Fuzzy and Crisp Reasoning in Ontology Applications*. In IEEE Computational Intelligence Magazine, 7(2):45-53. 2012.

[7] Guilin Qi, Peter Haase, Zhisheng Huang, Qiu Ji, Jeff Z. Pan, Johanna VÃ¶lker. *A Kernel Revision Operator for Terminologies* . In Proc. of the 7th International Semantic Web Conference (ISWC2008). 2008.

[8] Fabrizio Riguzzi, Elena Bellodi, and Evelina Lamma, *Probabilistic Datalog+/- under the distribution semantics*. In the Proceedings of the 25th International Workshop on Description Logics (DL), Aachen, Germany, 2012, pages 519-529.

[9] Thomas Lukasiewicz, Maria Vanina Martinez and Gerardo I. Simari, *Well-Founded Semantics for Extended Datalog and Ontological Reasoning*. In the Proceedings of the 32nd ACM Symposium on Principles of Database System. ACM Press. 2013

[10] Sven Hansson, *Semi-revision*, Journal of Applied Non-Classical Logics, Vol 7, Issue 1-2, 1997.

[11] Marcio M. Ribeiro and Renata Wassermann. *Base Revision for Ontology Debugging*. Journal of Logic and Computation. Vol 19, Issue 5, 2009, pages 721-743.

[12] Murat Sensoy, Achille Fokoue, Jeff Z. Pan, Timothy Norman, Yuqing Tang, Nir Oren and Katia Sycara. *Reasoning about Uncertain Information and Conflict Resolution through Trust Revision*. In Proc. of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2013). 2013.

[13] Guarino, N., and Welty, C. *Evaluating ontological decisions with ontoclean*. Commun. ACM 45, 2 (Feb. 2002), 61-65.

[14] Smith, B., Pisanelli, D. M., Gangemi, A., and Stefanelli, M. *Clinical guidelines as plans - an ontological theory*. In Methods of Information in Medicine (2006).