Development of a WSML Reasoning Infrastructure

Uwe Keller

September 1, 2005

Abstract

This document provides an outline of the development of a WSML reasoning infrastructure. We give a brief outline of the current state of affairs and then summarize the plan for future development along with some estimated timelines.

1 Overview

We provide an outline of the development of a WSML reasoning infrastructure. The document is structured as follows: Section 2 presents the big picture, i.e. what kind of algorithms and systems have to be designed and implemented to realize a reasoning infrastructure for WSML. Then, Section 3 summarizes the current state of the development of the reasoning infrastructure. Finally, Section 4 describes our future plans and gives some timelines for achieving the goals.

2 Reasoning with WSML

In Computer Science, *Reasoning* is commonly understood as the process of inferring ,,new" (i.e. not explicitly stated) information about some domain of discourse from a given (formal) model of that domain that contains a set of explicit descriptions of properties of the domain, i.e. a given knowledge base. Taking a broader perspective on reasoning, one can understand this term in a much more generic way as the computation based on explicit and formalized knowledge or computation based on logics. Reasoning then allows to exploit information represented in the formal model of the domain of discourse.

In Artificial Intelligence and its applications various forms of reasoning have been investigated for a long time, for instance checking for logical entailment, consequence finding, abductive reasoning, logic programming etc. Respective algorithms have been investigated for various logics. Taking this general perspective, a traditional database task, namely finding answers to given queries with respect to some database (query answering), can be seen as a reasoning

task as well. This task has been extensively studied in the deductive database community. Therefore it is not surprising that recent approaches to information integration are based on logics as data models and query answering.

Therefore one has to be more precise when talking about *Reasoning* and distinguish between the various forms of reasoning: they serve significantly different purposes, are implemented by means of a specific algorithm, that takes a well-defined set of input information and is expected to deliver results of a certain kind. The single forms of reasoning (i.e. computation with explicit formal models based on logics) differ significantly in all these aspects. In the following, we will call these different forms of reasoning *reasoning tasks*.

To stress the necessity of such a distinction we stress that usually an algorithm to perform some sort of reasoning (in the broader sense) is designed to fulfill a single, well-defined reasoning task. Algorithms for different reasoning task are based on very different techniques and thus differ significantly. In general, they will even not share the same infrastructure in terms of the datastructures on which the algorithm operate.

The reasoning tasks that we consider as being relevant in the context of WSML are the following:

- **Logic-based reasoning tasks.** Reasoning tasks that are not specific to ontologies but are usually considered with different kinds of knowledge bases. As the most relevant examples for such reasoning tasks, we consider at present:
  - Query Answering (QA).
  - Logical Entailment (LE).
  - Consequence Finding (CF).
- **Ontology-specific Reasoning tasks (ORT).** Reasoning tasks that specifically refer to ontologies and their modelling primitives. Typical examples from the Description Logic Community are:
  - Concept Satisfiablity
  - Concept Subsumption
  - Relation Subsumption
  - Classification tasks: Building taxonomies according to concept definitions, finding some most-specific concepts in an ontology that covers a given concept description etc.
- **Reasoning tasks specific to other top-level elements in WSMO.** Reasoning tasks that specifically refer to web services, goals and mediators and the respective additional modelling primitives in WSML. Examples would be:
  - Goal-Capability matching

– Choreography matching

We believe that many reasoning tasks of the third-mentioned class are in fact reducible to tasks of the first and second class. At least they will be based on reasoning tasks of the first and second kind to a large extend. For the moment, we will not consider the third class at all, since the respective tasks are not really clear right now and WSML is defined too vaguely for some of the involved modelling elements.

Please note that there are a lot more reasoning tasks, such as abductive reasoning for the first class of reasoning tasks mentioned above, that might be investigated in the long run, in case that senseful application could be identified.

When it comes to the design and implementation of algorithms for a specific reasoning tasks, another important aspect has to be considered, that potentially results in very different techniques, datastructures and systems: the concrete logic (or formal language) in which input to the reasoning task (such as the knowledge base representing the domain of discourse) is represented.

In our specific context, we have to consider here the single variants of WSML, namely: WSML-Core, WSML-Flight, WSML-Rule, WSML-DL, WSML-Full and perhaps the monotonic subset of WSML-Full (WSML-FOL).

From an implementation perspective, the two above mentioned dimensions are (largely) orthogonal to each other and span the space of the various different reasoners (or more precisely reasoning algorithms) that need to be integrated into a common WSML Reasoner Framework. Figure 1 illustrates the space to be considered. It deliberately leaves out some parts of the space that are not relevant right now or can not be achieved in a forseeable period of time.

3 Current State of Affairs

Recently, we started with the implementation of a prototype of the WSML Reasoner Framework described in Section 2. During this work, we focused on one specific reasoning task, i.e. *query answering*. We developed and a flexible architecture that allows for easy integration of external reasoning components for query answering by means of wrapper classes.

Approach to realize Query Answering. Query answering is a well-understood reasoning task for the Rule-based WSML Variants, namely, WSML-Core, WSML-Flight and WSML-Rule. More precisely, WSML-Core and WSML-Flight can be mapped to Datalog, that means the function-free Horn-subset of First-order logic and to Datalog$^{\neg}$, that means Datalog extended by a default negation. WSML-Rule can be mapped to Prolog, which means Datalog$^{\neg}$ extended by function symbols[1].

For Datalog, Datalog$^{\neg}$ and Prolog, there are already various available implementations that can be used to perform query answering on rule bases. Thus,

[1] We consider function symbols to have a positive arity.

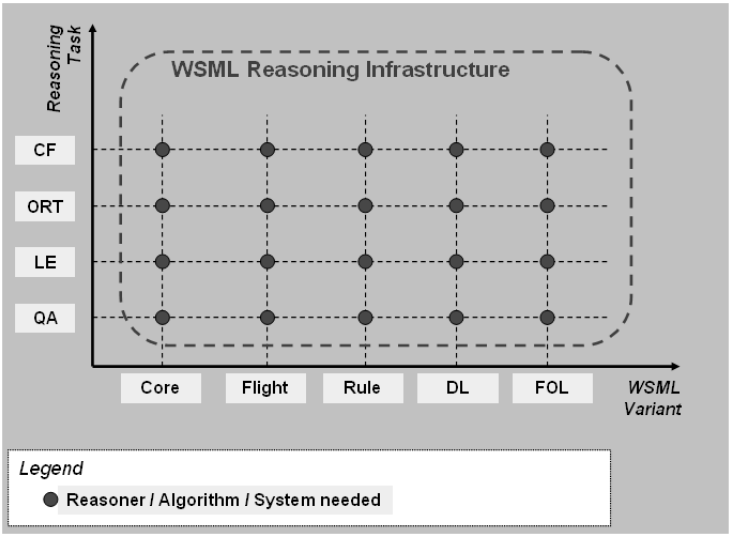

Figure 1: The Space of Reasoners to be considered in the WSML Reasoning Framework.

we decided to start with these languages and come up with an architecture that allows us to integrate different tools easily and exploit several person-years of development efforts that has been invested in other groups to come up with powerful systems for this specific task.

We realized the architecture for the query answering reasoner for WSML-Core and WSML-Flight as shown in Figure 2: We defined a general API for creation and interaction with a WSML reasoner of any kind. WSML Ontologies are transformed to a syntactically simpler form and finally to a pure logical representation in terms of Datalog rules. The specific semantics of object-oriented modelling primitives in WSML is captured by auxilliary rules, at is has been done in the Ontobroker system. An API for wrapping external datalog engines has been defined and instantiated for a few different systems. A specific auxilliary datastructure that can be reused for different wrappers is a *symbol map* which takes care of translating the WSML specific naming convention, in particular IRIs, into symbol names that are actually valid for a specific tool.

This architecture is flexible and enables the reuse of various ontology transformation (or normalization steps) that are needed in the context of other reasoning tasks and other languages as well. Extending the architecture to WSML-Rule is straightforward and essentially means to extend the Datalog layer to a Prolog layer by adding support for function symbols.

Integrated external reasoning components. So far we integrated the following systems:

- DLV: A disjunctive Datalog engine.

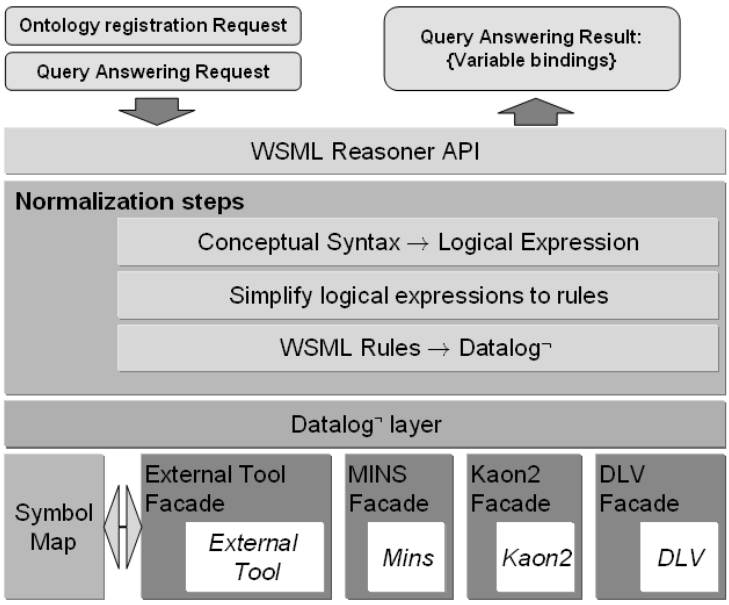

Figure 2: An architecture for Query Answering in WSML-Core and WSML-Flight

- Mandarax: A naive Java-implementation of a resolution-based Prolog engine.
- Kaon2: An ontology reasoner (for almost all of OWL-DL) based on a disjunctive Datalog.
- MINS: The DERI rule reasoner (which currently is nothing more than a lean version of the SILRI system).

Candidates for additional integration are:

- XSB: A powerful implementation of a Prolog engine.
- Flora2: A deductive database system based on F-Logic.

As our work shows, integration of new systems does not take longer than around 2 - 4 hours, which means that within a day one can come up with a reasoner for query answering in WSML[2] which supports all features of language[3] and is based on a specific external tool.

Current limitations. At present, we can not deal with all WSML variants and not with all features of the supported variants: We can support WSML-Core and WSML-Flight. Extension to WSML-Rule will take a few days.

[2] At least the WSML Rule fragment
[3] Depending on what part of WSML the external tool actually can deal with itself.

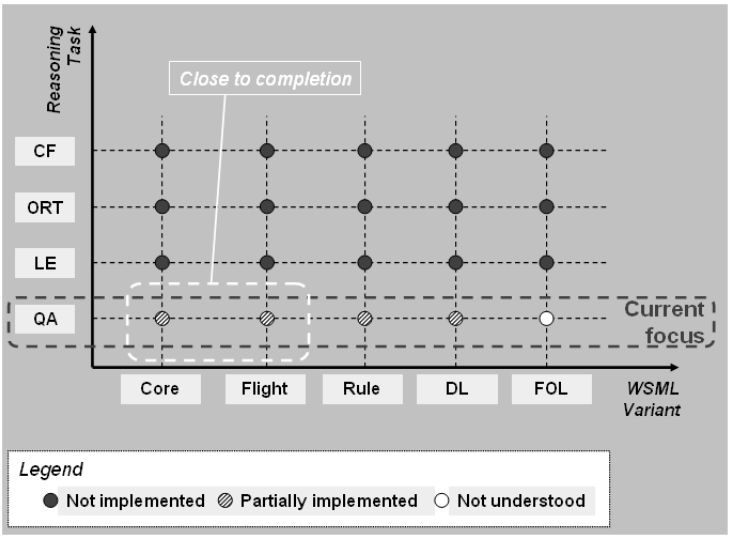

Figure 3: Current status of the implementation of the WSML Reasoning Framework.

However, datatypes are not implemented correctly in WSMO4j at present, such that the reasoner framework can not handle datatypes and built-in predicate yet. As soon as the implementation is fixed, the extension of to datatypes can be done within a few days for the various integrated tools.

Moreover, the reasoner simply ignores constraints in Ontologies, since handling of constraints is an issue that needs to be dealt with by a Knowledge Base Management or Ontology Management Environment and not a reasoner itself. Provision of some very basic support for constraints within the reasoner framework can be done within a few hours.

Finally, there is no normalization step implemented at present which resolves Anonymous Identifies in WSML. Hence, anonyoumous Ids can not be dealt with in the present prototype. An implementation of the respective normalization step is easy and will take a few hours.

Figure 3 overviews the current state with respect to the reasoner space shown in Figure 1.

4 Future Plans and Timelines

In the near future we will continue to focus on query answering mainly. Additionally, we will implement some of the ontology related reasoning tasks.

Query Answering. We plan to complete the support for query answering in WSML-Core and WSML-Flight such that all features of WSML are covered.

That means datatypes and anonymous ids have to be dealt with. This can be done until end of September.

Ontology import will be dealt with as well in a simple form, since a serious approach will require an implementation of some sort of sophisticated (distributed) Ontology registry. At present, we believe that a simple prototype for such a distributed registry could be realized with moderate effort on top of the P-Grid system, developed at the ETH Laussane, Switzerland.

Integration of XSB for Datalog has not primary priority and thus will be skipped for the next few weeks.

Basic constraint handling will be included until mid of September.

Until mid of October the extension to WSML-Rule should be completed and thus a reasoner for WSML-Rule supporting all WSML features should be available.

In parallel, we will start with the development of the DERI rule engine, called MINS[4] which essentially is a reimplementation of the SILRI rule system. This reimplementation will take some time. It is unrealistic that it will be ready until the end of the year in a stable version. However, some first prototype system will be possible, depending on how much time the developers can spend on this project.

Ontology related Reasoning tasks. We will implement some of the ontology related reasoning tasks for WSML-Core, WSML-Flight, WSML-Rule and WSML-DL until the end of the year. For those tasks that can be simply implemented as query answering problems, this will only need a few days. Hence, they will be available already until

Logical entailment checking. Logical entailment checking for WSML-Core is not difficult to implement and will be realized until the end of November. For WSML-DL some protoype should be possible until the end of the year.

For the other WSML Variants contain non-monotonic features (i.e. default negation) this will be more difficult and not be ready until the end of the year. However and implementation for the non-monotonic part of these languages should be possible until the end of the year as well.

Everything else. It is unrealistic that we will consider other reasoning tasks and languages until the end of the year. Hence, they are out of scope for now. For some of them, more research and clarification is needed.

[4]MINS is an acronym for ,,MINS is not SILRI".