# Overview on Proposals for Master-Thesis Projects and Bachelor-Thesis Projects

Uwe Keller
Semantic Technology Institute (STI)
University of Innsbruck, Austria.
uwe.keller@sti2.at

December 25, 2007

**Abstract**

This document collects proposals for Master-thesis projects and Bachelor thesis projects that are open at present or already running. It gives a short motivations and introduction to the single projects and overviews the aspects to be addressed in the project and the respective thesis.

## Contents



## 1 Open Master-Thesis Projects

This section collects a couple of open Master-thesis projects.

## 1.1 Development and Integration of a Tool supporting Personal Task Management into an agent-based Enterprise Information System

Everybody works, most often facing various projects or activities at the same time, usually working together in a team of people, taking over well-defined tasks to accomplish an overall goal. At the same time. people working in enterprises often feel stressed and overloaded. A constant feeling of loosing control over the things that need to be done in a complex setting with various concurrent activities and projects, both, professionally and privately, causes troubles.

For this reason, various methodologies for personal time and task management have been developed and proposed over time. A particularly prominent one in recent time is the so-called "Getting Things Done "(GTD) Methodology [1] by David Allen.

The aim of the project is to (1) get familiar with and understand the methodology, (2) design and develop a tool for personal task management based on the "Getting Things Done "(GTD) Methodology [1,2,3] that considers typical information infrastructure that one can find in enterprises (e.g. eMail Servers) and (3) to integrate it into an agent-based enterprise information system (developed within the SEnSE project).

The project requires technical skills (e.g. implementation in Java, working with eMail infrastructure, ontologies ...), curiosity and creativity. A basic need is to get familiar with the GTD methodology and how it can be supported in a simple way by technology (software systems and semantic annotations).

A specifically interesting conceptual question would be, how semantics and domain ontologies could be used to further support users in there needs within an enterprise setting (e.g. in the context of product engineering).

### References


- [1] Getting Things Done Methodology at WIKIPEDIA: `http://en.wikipedia.org/wiki/Getting_Things_Done`
- [2] Website of David Allen (inventor of the methodology): `http://www.davidco.com/`
- [3] The GTD book at Amazon: `http://www.amazon.com/Getting-Things-Done-Stress-Free-Productivity/dp/0142000280`


## 1.2 Ordered Binary Decision Diagrams for Efficient Evaluation of Datalog Programs

Ordered Binary Decision Diagrams (OBDDs) are very successful datastructures that are used in various areas of computer science to deal with huge sets and implement efficient operations on such sets. A nice characteristic of OBDDs is that in practice they often allow for very compact

representations of subsets of potentially extremely large domains (e.g. domains with 10 to the power of 12 elements). Examples for such sets are sets of states of an execution of a program that have a certain property (for instance raise a null-pointer exception), application areas of OBDDs are e.g. solvers for propositional satisfiability problems and model-checkers.

Datalog is a well-studied rule-based knowledge representation framework that is extensively used in classical databases. In fact, there is a close link between datalog programs, relational algebra and SQL.

Evaluation of datalog programs is needed to computer answers to queries of the knowledge base. It can be done in various ways (e.g. bottom-up or top-down). However, in many application scenarios, one has to face large amounts of data (e.g. relations in terms of relational databases). In such scenarios, when answering queries (independent of the specific evaluation strategy) one is interested in set-based evaluation techniques that allows to deal with sets (instead of individual tuples) at a time and to apply efficient techniques for dealing with such sets.

The purpose of this project is the investigation of OBDDs in such set-based evaluation strategies. We are especially interested in the extension of existing work on this topic towards more general settings, namely:

- How to deal with domains that are not constant or known upfront
- How to efficiently deal with elements in sets that themselves have not a suitable binary encoding
- How to deal with built-in predicates

The proposed solution should be implemented (preferably in Java) and evaluated. The implementation will be done in the context of a datalog engine that is currently under development at DERI.

In this context, certain interesting questions should be addressed:

- How to determine suitable orders for the BDDs (e.g. optimization by machine learning techniques, genetic algorithms etc. ...)

We are seeking for motivated people that want to learn about some basic elements of databases, knowledge representation, artificial intelligence and OBDDs, to want to develop creative solutions to open problems and to apply their knowledge to an interesting problem in a concrete setting.

# 2 Open Bachelor-Thesis Projects

This section collects a couple of open Bachelor-thesis projects.

## 2.1 Extending a Tableau-based Description Logic Reasoner to handle Terminological Background Knowledge

Description Logics (DLs) are a family of class-based knowledge representation formalisms characterised by the use of various constructors to build complex classes from simpler ones, and by an emphasis on the provision of sound, complete and (empirically) tractable reasoning services. They

have a range of applications, but are most widely known as the basis for ontology languages such as OWL.

A specifically simple Description Logic is the logic ALC. ALC can be seen as a syntactic variant of the (multi-)modal logic K.

Recently, we developed a novel, experimental DL reasoner. At present the reasoner is restricted to deal with ALC and to perform concept satisfiability tests only. In particular, the reasoner can not take into account terminological background knowledge (so-called TBoxes) when performing the satisfiability test.

The goal of the project is to extend the current system by all needed components to integrate termonological background knowledge (TBoxes). We start with a simple well-known method and add various optimizations that have been proposed during the last years to make the system more practical. Further, the system should be extended by another API that allows clients to invoke and interact with our reasoner over the Web in a stateful manner. We therefore aim at implementing a DIG-interface for our reasoner. DIG (DL Interest Group) can be understood as a application-domain specific Web-protocol that layers on top of HTTP.

Implementation language is Java. Implementation is done in a team that develops the system concurrently. We use standard Software Engineering tools such as Version Control and Bug Tracking systems.

The project has a well-defined focus and can be completed in a reasonable amount of time, given continuous involvement of the respective student.

The project will add an important feature to our system and is therefore valuable work that will be used further on.

## 2.2 Extending a Datalog engine with Dynamic Filtering

Datalog is a simple yet universal knowledge representation formalism. Its main characteristic is that it is a rule-based formalism, where one can express simple if-then rules to describe derivations of knowledge. Further, datalog underlies many industrial data management systems, i.e. it is the core (and extends) the well-known relational datamodel and SQL as the respective industrial query language.

Models in datalog are expressed as so-called programs, i.e. a set of simple rules. These rules capture schematic knowledge about the models. An example is the following rules describing reachability in a graph.

(1) reachable(?x,?y) ¡= edge(?x, ?y) (2) reachable(?x,?y) ¡= reachable (?x, ?z) and edge(?z, ?y)

Further, a datalog program contains so-called facts, which is assertional knowledge or a database respectively.

In our graph example, this could be the edges of a specific graph:

edge(bregenz, innsbruck) edge(innsbruck, salzburg) edge(innsbruck, brenner) edge(innsbruck, muenchen) edge(salzburg, linz) edge(linz, steyr) edge(linz, wien) edge(muenchen, wien) edge(wien, muenchen) edge(muenchen, augsburg) edge(augsburg, ulm)

located-in(muenchen, germany) located-in(augsburg, germany) located-in(ulm, germany) located-in(innsbruck, austria) ...

The fundamental task to be solved by a datalog engine is query answering, which is database-style queries. An example is to ask what cities in Germany can be reached from Innsbruck in our specific model (i.e. the graph described above):

?- reachable(innsbruck, ?destination) and located-in(?destination, germany)

which would result in a range of answers (bindings for the variable ?destiantion in the query): muenchen, augsburg, ulm ...

There are various algorithms how to compute the result for a given query. All these algorithms have their own strengths and weaknesses. One of these algorithms is called *Dynamic Filtering* which is elegant and provides an a basis for a highly-concurrent or even distributed implementation.

We are currently developing a datalog engine called IRIS [3] which implements a couple of standard evaluation strategies on top of common datastructures. The aimof this project is to implement dynamic filtering within the IRIS system and to evaluate the method wrt. to other algorithms.

Implementation will be done in Java.

The project provides some insight in some interesting topic in the intersection of knowledge representation and databases, allows you to learn about a specific elegant evaluation strategy for queries over datalog programs and to learn how to integrate it in an existing system. The focus of the project is design and implementation of a component providing a specific functionality, it is not theory work. Finally, one might want to think about optimizations of the implementation.

## 2.3 Implementation and Evaluation of (simple) SAT solvers for Propositional Logic

Propositional Logics are the most fundamental class of logics. Despite their simplicity, they enjoy real industrial interest. Consequently, a good number of SAT solvers has been developed over the last decades, some of which are industrial-strength systems that can cope with problems arising in industrial applications. The huge variety of systems however is implemented on the basis of a smaller number of well-understood algorithm families.

Recently, we developed a novel experimental platform for reasoning in Description Logics (or multi-modal logics). DLs (MMLs) can be seen as expressive extensions of propositional logics. At present we are developing a prototype of our reasoner. A core component is a propositional SAT solver. For this reason, we implemented a range of propositional solvers.

The goal of this project is two-fold: (1) it should extend our arsenal of SAT solvers by implementations of a few distinct approaches and (2) it should evaluate the performance of various solvers in our framework systematically, derive recommendations and potentially optimizations for the integration of SAT solvers in our framework for DL reasoning.

Proposals for interesting algorithms are available.

The implementation will be done in Java. We develop the system in a small team concurrently. We therefore use state-of-the art software engineering tools such as Version control systems, Unit Testing and Bug Tracking systems.

We seek for motivated students that want to learn about specific techniques to solve the SAT problem, implement the techniques in (simple) components and are curious to experiment with their components to find out how to use them best.

## 2.4 Parts of a Master-thesis project from above

In general, it should be possible to split any of the given Master-thesis project into two or three well-defined Bachelor-thesis proposals. In case of questions, please feel free to ask.