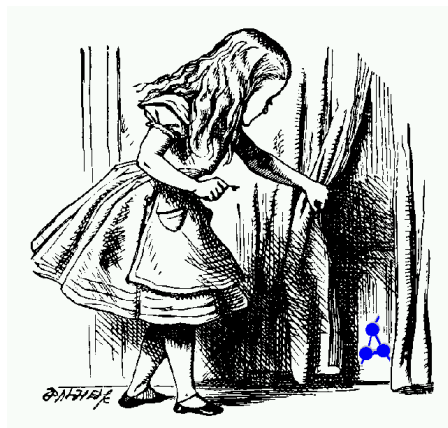


Reasoner Demonstrator

Implementing new reasoner with datatypes support

Dmitry Tsarkov & Ian Horrocks
University of Manchester
Kilburn Building
Oxford Road
Manchester M13 9PL
email: tsarkov@cs.man.ac.uk



Identifier	Del 14
Class	Deliverable
Version	1.0
Date	30-06-2004
Status	Final
Distribution	Public
Lead Partner	VUM

WonderWeb Project

This document forms part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2001-33052.

For further information about WonderWeb, please contact the project co-ordinator:

Ian Horrocks
The Victoria University of Manchester
Department of Computer Science
Kilburn Building
Oxford Road
Manchester M13 9PL
Tel: +44 161 275 6154
Fax: +44 161 275 6236
Email: wonderweb-info@lists.man.ac.uk

Contents

Executive Summary	1
1 Introduction	2
2 Overview of FaCT++	2
3 Software Details	3
3.1 Performance evaluation	3
4 Obtaining FaCT++	4
5 Discussion	4

Executive Summary

This document provides an overview of the FaCT++ reasoner. FaCT++ is a Description Logic reasoner, which is a re-implementation of the FaCT system, extended with new features and optimisations. The current version supports the OWL Lite language; the extended one will support the OWL DL language. The latest version of FaCT++ can be downloaded from the project website:

<http://wonderweb.semanticweb.org/deliverables/D14.shtml>

1 Introduction

The OWL language,¹ which is designed for representing Semantic Web ontologies, allows the user to introduce datatypes into ontologies (in addition to the usual mechanisms for describing “abstract” classes, such as boolean conjunctive, quantifiers and cardinality constraints). In order to use these ontologies one should have an efficient, robust reasoner, which supports all the features of the language. We have set out to satisfy these requirements by creating the FaCT++ DL reasoner.

FaCT++ was started as re-implementation of the well-known highly optimised DL reasoner FaCT [1]. It used the established FaCT algorithms, but with a different internal architecture. Additionally, the implementation language C++ was chosen in order to create a more efficient software tool, and to maximise portability. During the implementation process, new optimisations were also introduced, and some new features were added.

2 Overview of FaCT++

FaCT++ can be used for the following tasks:

- Checking the consistency of an ontology;
- Checking the satisfiability of a single concept / group of concepts;
- Checking the subsumption relationship between two concepts;
- Classifying the whole ontology (creating a taxonomy);

FaCT++ provides full reasoning support for the $\mathcal{SHIF}(\mathcal{D})$ Description Logic. In addition, it has some extra features:

First-order logic interface. FaCT++ can create FOL problems for subsumption and satisfiability checking in the \mathcal{SHOIQ} logic using a standard syntax (TPTP) that can be read by most first order theorem provers.² See [3] for a comparison of the performance of the FaCT++ reasoner with a state of the art first order theorem prover running on TPTP problems created using FaCT++.

Limited individual support. FaCT++ currently has limited support for individuals. All individuals are treated as concepts, and reasoning is performed on the modified ontology. This approach results in no loss of inferences if the ontology contains no relations between individuals, a restriction that holds for many of useful ontologies.

Even in case where relations between individuals are present, it is sometimes possible to get a correct answer to satisfiability and/or subsumption query. In this case, FaCT++ gives answer together with correctness information.

¹<http://www.w3.org/TR/owl-features/>

²see <http://www.cs.miami.edu/~tptp/>

Ontology	FaCT v2.33 time, sec	FaCT++ v0.8 time, sec	FaCT++ v0.98 time, sec
GALEN	49	790	104
GO	227	12	15
wine	—	11	2

Table 1: Comparison between FaCT and FaCT++

Role domain and range absorbtion. If the reasoning tool has no support of range and domain constructions (as in FaCT), it must treat them as general axioms of a kind which are not absorbable. This slows the reasoning process significantly. In FaCT++, native support of these constructions was implemented, and this results in speedups of as much as two orders of magnitude for some ontologies. Besides that, new absorbtion methodic based on domain and range constructions was implemented, which leads to more efficient reasoning on several tasks [4].

3 Software Details

FaCT++ is currently available as a following software products:

- Standalone software component. May works only in batch mode. In order to either perform classification/consistency checking of an ontology, or to test satisfiability/subsumption of given concept(s), users must create a configuration file. This file contains all necessary information about the input ontology, reasoning task(s) and configuration options for the reasoner.
- DIG reasoner, implemented as C++ dynamic library. There is a Java implementation file, which implements DIGReasoner interface.³
- Servlet, implementing HTTP DIG reasoner functionality. Can be used with arbitrary servlet containing environment like tomcat.⁴

3.1 Performance evaluation

Some results from a comparison between FaCT++ and FaCT can be found in Table 1. Current FaCT++ version is 0.98, version 0.8 was used in previous deliverable and used for comparison purposes.

The GALEN ontology is a quite large (2800 classes) and complex ontology used in medicine. FaCT was optimised and tuned specifically to perform well on this particular ontology, and is still able to outperform FaCT++ in this test. This is mainly due to various caching optimisations, not all of which are currently implemented in FaCT++. These optimisations are currently being added to FaCT++, and when completed should allow FaCT++ to perform at least as well as FaCT on this ontology.

³freely available at <http://dig.sourceforge.net>

⁴<http://jakarta.apache.org/tomcat>

The Gene Ontology (GO) is large (>13500 classes and individuals), but has a very simple structure. There are no relations between individuals, so it is possible to use our individual-to-class translation without loss of correctness. This approach, together with the range of optimisations employed, makes FaCT++ at least 10 times faster than FaCT on this ontology (and on other large but simply structured ontologies).

The wine ontology is part of OWL test suite. It contains 9 roles with range and domain restrictions together with individuals and oneOf constructions (nominals). This makes it unsolvable by FaCT (it exceeds resource limits before completing the task), while FaCT++ can solve it in only a few seconds. This is mainly due to FaCT++'s native support for role range and domain restrictions, and improved absorption optimisations [2].

4 Obtaining FaCT++

FaCT++ is freely available under the GNU General Public License either in sources or as a precompiled binaries for several platforms. The latest version of FaCT++ can be downloaded from the project website:

<http://wonderweb.semanticweb.org/deliverables/D14.shtml>

Binary package contains standalone version (together with examples of usage), dynamically linked library (together with Java interface file), suitable for usage from Java programs, and servlet which behaves as an HTTP reasoner with DIG interface.

5 Discussion

FaCT++ provides highly optimised reasoning support for the OWL Lite language, and will eventually be extended to support OWL DL. It already outperforms FaCT by 1–2 orders of magnitude on several ontologies, and new features and optimisations are still being added.

References

- [1] I. Horrocks. The FaCT system. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence, pages 307–312. Springer-Verlag, Berlin, May 1998.
- [2] I. Horrocks. Implementation and optimisation techniques. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 306–346. 2003.
- [3] Dmitry Tsarkov and Ian Horrocks. DL reasoner vs. first-order prover. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, volume 81, pages 152–159. CEUR (<http://ceur-ws.org/>), 2003.

- [4] Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, volume 104, pages 41–50. CEUR (<http://ceur-ws.org/>), 2004.