Term-Indexing for the Beagle Theorem Prover

Timothy Clarence Richard Cosgrove

A subthesis submitted in partial fulfillment of the degree of
Bachelor of Science (Honours) at
The Department of Computer Science
Australian National University

September 2013

Typeset in Palatino by TeX and LaTeX 2_ε.

Except where otherwise indicated, this thesis is my own original work.

Timothy Clarence Richard Cosgrove
10 September 2013

For Dana.

Acknowledgements

Thank you to my Supervisor and all…

Abstract

This should be the abstract to your thesis. . .

Contents

Chapter 1

Introduction

1.1 Motivation

- Describe beagle
- Advantages of beagle
- drawbacks
- some intstrumentation

1.1.1 A Theoretical Framework

Chapter 2

Background

2.1 First-Order Logic Terms and Notation

This thesis focuses around the extension of *beagle*, a *first-order logic* (FOL) theorem prover. In order to understand beagle's purpose and functions a basic understanding of the FOL logical system is required. This section provides a rudimentary overview of FOL syntax and uses; but also includes an explanation of any specialised terms and notation used throughout the paper.

2.1.1 FOL basics

2.1.2 Calculi and FOL problems

2.1.3 The Superposition Calculus

Should contain

- Variables
- Symbols
- Predicates
- Quantifiers
- Notion of soundness and completeness
- Description of a 'calculus'

2.1.4 Specialised Syntax in this Paper

- Positions

2.2 Automated Reasoning and Theorem Proving

Automated Reasoning is a rapidly growing field of research where computer programs are used to solve problems stated in first order logic statments or other formal logics.

Some existing theorem provers include:

SPASS

[Weidenbach et al. 1999]

Vampire

[Riazanov and Voronkov 1999]

E

[Schulz 2002]

Should contain

- Theorem prover examples

2.3 Term Indexing

Term indexing is a technique used to better locate logical terms which match rules in a prover's calculus.

Top Symbol Hashing

Discriminant Trees

2.4 Fingerprint Indexing

Fingerprint Indexing is a recent technique developed by Schulz [2012], the creator of the E prover.

Table 2.1: Fingerprint matches for Unification [Schulz 2012, p6]

	f_1	f_2	**A**	**B**	**N**
f_1	**Y**	**N**	**Y**	**Y**	**N**
f_2	**N**	**Y**	**Y**	**Y**	**N**
A	**Y**	**Y**	**Y**	**Y**	**N**
B	**Y**	**Y**	**Y**	**Y**	**Y**
N	**N**	**N**	**N**	**Y**	**Y**

Table 2.2: Fingerprint matches for Matching [Schulz 2012, p6]

	f_1	f_2	**A**	**B**	**N**
f_1	**Y**	**N**	**N**	**N**	**N**
f_2	**N**	**Y**	**N**	**N**	**N**
A	**Y**	**Y**	**Y**	**N**	**N**
B	**Y**	**Y**	**Y**	**Y**	**Y**
N	**N**	**N**	**N**	**N**	**Y**

2.5 The Beagle Theorem Prover

The core implementation of Beagle was developed by Peter Baumgartner et al. of NICTA. Its purpose was to demonstrate the capabilities of the *Weak Abstraction with Heirachic Superposition Calculus*; which allows the incorporation of prior knowledge via a 'background reasoning' modules.

2.5.1 The Heirachic Superposition with Weak Abstraction Calculus

2.5.2 Beagle's Shortcomings

2.6 Tools Used

2.6.1 Scala

As mentioned above *beagle*is written in *Scala*, the Scalable Language. Scala is a functional language and may be confusing to those who are not familiar with the functional programming paradigm. This thesis will contain occasional snippets of Scala code; but note that any snippets used will be accompanied by an explanation and in general an understanding of Scala/funcitonal programming is not required.

2.6.2 VisualVM

2.6.3 Eclipse

Chapter 3

Implementing Fingerprint Indexing

3.1 Structure of *beagle*

To be able to make any significant contribution to the *beagle*project, I first had to gain a solid understanding of the existing Scala codebase.

Figure of Clause/Expression syntax tree

Refer to results for instrumentation.

3.2 Implementing Fingerprint Indexing

To compare two fingerprints with each other we look at them side-by-side and check that each position shows a Y in the Fingerprint unifcation table.

```
/** Check two Fingerprint features for compatibility based
  * on the unification table (See page 6 of [Shulz 2012]).
  * This table is reduced to 4 cases:
  *  - True if Features are equal,
  *  - True if at least one Feature is B,
  *  - True if at least one Feature is A; but no Ns,
  *  - False otherwise  */
 def compareFeaturesForUnification
         (a:FPFeature, b:FPFeature) : Boolean =
  (a == b) ||
  (Set(a,b) match {
    case x if (x contains FPB) => true
    case x if (x contains FPA) => !(x contains FPN)
    case _ => false})
```

Listing 1: Scala implementation of the Fingerprint unification table. [Schulz 2012, p6]

The following block of Scala code extracts a single Fingerprint Feature from a Term at the given position.

```
/** Extract the operator at position pos. Note that matching Var
  * and Funterm is an exhaustive pattern for Term. */
 def extractFeature(term: Term, pos: Position) : FPFeature = pos match {
   case Nil     => term match {
     case t:FunTerm => FPF(t.op) // Found function symbol, return it
     case t:Var     => FPA       // Found variable, return A
   }
   case p :: ps => term match {
     case t:FunTerm => try   {extractFeature(t.args(p), ps) }
                       //Attempted to index non-existent position, return N
                       catch {case e:IndexOutOfBoundsException => FPN}
     // Found variable BEFORE end of position, return B
     case t:Var     => FPB
   }
 }
```

Listing 2: Scala code to extract fingerprint features for matching.

3.3 Adding Indexing to *Beagle*

3.3.1 Refactoring Current Implementation

Refer to class diagram from 3.2

3.3.2 Initial Problems

3.4 Tailoring to *Beagle*'s Heirachic Superposition with Weak Abstraction Calculus

3.4.1 Foreground and Background Terms

In the Heirachic Superposition with Weak Abstraction Calculus all terms have a concept of being 'Foreground' or 'Background'.

Chapter 4

Results

4.1 Beagle Before Indexing

4.2 Indexing Subsumption

4.2.1 False Positives

4.2.2 Speed

4.2.3 Comparison

4.3 Tailored Improvements

4.3.1 Backgroundedness

4.4 Further Indexing

Chapter 5

Conclusion

5.1 Why this is a Very Clever Thesis

Appendix A

Some Other Stuff

A.1 Why I Did It

Appendix B

More Stuff

Bibliography

RIAZANOV, A. AND VORONKOV, A. 1999. Vampire. In *Automated Deduction – CADE-16*, Volume 1632 of *Lecture Notes in Computer Science*, pp. 292–296. Springer Berlin Heidelberg. (p. 4)

SCHULZ, S. 2002. E – A Brainiac Theorem Prover. *Journal of AI Communications 15*, 2/3, 111–126. (p. 4)

SCHULZ, S. 2012. Fingerprint indexing for paramodulation and rewriting. In B. GRAMLICH, D. MILLER, AND U. SATTLER Eds., *Automated Reasoning*, Volume 7364 of *Lecture Notes in Computer Science*, pp. 477–483. Springer Berlin Heidelberg. (pp. 4, 5, 7)

WEIDENBACH, C., AFSHORDEL, B., BRAHM, U., COHRS, C., ENGEL, T., KEEN, E., THEOBALT, C., AND TOPIĆ, D. 1999. System description: Spass version 1.0.0. In *Automated Deduction – CADE-16*, Volume 1632 of *Lecture Notes in Computer Science*, pp. 378–382. Springer Berlin Heidelberg. (p. 4)