Term Indexing for the Beagle Theorem Prover

Tim Cosgrove

COMP4006 Honours Research Project

Research School of Computer Science,
Australian National University

u4843619@anu.edu.au

Supervisor: Peter Baumgartner

October 14, 2013

The Beagle Theorem Prover

- Beagle is a First-Order-Logic resolution theorem prover with equality, built to show off the capabilities of the Hierarchic Superposition with Weak Abstraction Calculus.

The Beagle Theorem Prover

- Beagle is a First-Order-Logic resolution theorem prover with equality, built to show off the capabilities of the Hierarchic Superposition with Weak Abstraction Calculus.
- This calculus is capable of *hierarchic reasoning* by incorporating a *background prover*.
- Background provers act as a black box which can instantly prove known facts. For example integer arithmetic.

The Beagle Theorem Prover

- Beagle is a First-Order-Logic resolution theorem prover with equality, built to show off the capabilities of the Hierarchic Superposition with Weak Abstraction Calculus.
- This calculus is capable of *hierarchic reasoning* by incorporating a *background prover*.
- Background provers act as a black box which can instantly prove known facts. For example integer arithmetic.
- The calculus is carefully constructed with a process known as *weak abstraction* in order to ensure consistency and completeness.

Extensions to Beagle

- Beagle has some major shortcomings which prevent it being more than a proof of concept.
- In particular, it lacks an efficient manner of locating terms for inference

Extensions to Beagle

- Beagle has some major shortcomings which prevent it being more than a proof of concept.
- In particular, it lacks an efficient manner of locating terms for inference
- Enter 'Term Indexing', a method for efficiently managing and collecting these terms.

Terminology Used in this Presentation

- First Order Logic

Terminology Used in this Presentation

- First Order Logic
- Positions

Terminology Used in this Presentation

- First Order Logic
- Positions
- Substitutions:
  - s is '*unifiable*' with t : $\sigma s = \sigma t$
  - s '*subsumes*' t : $\sigma s = t$

The Superposition Calculus

- Normal Superposition rule

Positive Superposition

$$\frac{l \approx r \vee C \qquad s[u] \approx t \vee D}{(s[r] \approx t \vee C \vee D)\sigma}$$

Where (i) $\sigma = \text{simple mgu } (l, u)$,
and (ii) u is not a variable.

The Hierarchic Superposition with Weak Abstraction Calculus

- Extension of the Superposition Calculus to accommodate hierarchic reasoning.

Positive Superposition $\dfrac{l \approx r \vee C \qquad s[u] \approx t \vee D}{\mathrm{abstr}((s[r] \approx t \vee C \vee D)\sigma)}$

Where (i) $\sigma =$ simple mgu (l, u),

(ii) u is not a variable,

(iii) $r\sigma \not\succeq l\sigma$,

(iv) $t\sigma \not\succeq s\sigma$,

(v) l and u are not pure background terms,

(vi) $(l \approx r)\sigma$ is strictly maximal in $(l \approx r \vee C)\sigma$,

and (vii) $(s \approx t)\sigma$ is strictly maximal in $(s \approx t \vee D)\sigma$.

Term Indexing Techniques

- Term indexers aim to collect all FOL terms which potentially match a ‘query’ term.

Term Indexing Techniques

- Term indexers aim to collect all FOL terms which potentially match a ‘query’ term.
- Top-Symbol Hashing.
- Discrimination Trees.
- Path Indexing.

Fingerprint Indexing

- Maintain a collection of *fingerprints* for terms.

Fingerprint Indexing

- Maintain a collection of *fingerprints* for terms.
- A term fingerprint is an array over $F \cup \{\mathbf{A}, \mathbf{B}, \mathbf{N}\}$, the *Fingerprint Features*.

Fingerprint Indexing

- Maintain a collection of *fingerprints* for terms.
- A term fingerprint is an array over $F \cup \{\mathbf{A}, \mathbf{B}, \mathbf{N}\}$, the *Fingerprint Features.*

Table: Fingerprint Feature comparison tables for *unification* (left) and *subsumption* (right)

	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

	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

Fingerprint Indexing

- Maintain a collection of *fingerprints* for terms.
- A term fingerprint is an array over $F \cup \{\mathbf{A}, \mathbf{B}, \mathbf{N}\}$, the *Fingerprint Features.*

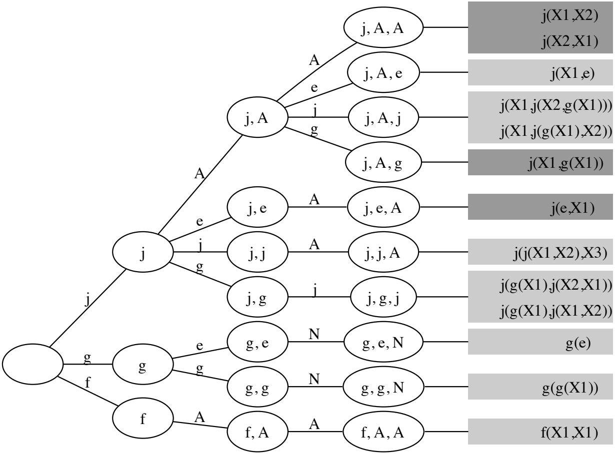
Table: Fingerprint Feature comparison tables for *unification* (left) and *subsumption* (right)

	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

	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

- Schulz, Stephan: Fingerprint Indexing for Paramodulation and Rewriting. In: Lecture Notes in Computer Science volume 7364 pp. 447–483 (2012).

Fingerprint Indexing – Example Fingerprint Index



Why Fingerprint Indexing?

- New and not thoroughly tested technique.

Why Fingerprint Indexing?

- New and not thoroughly tested technique.
- Currently showing very promising results.

Why Fingerprint Indexing?

- New and not thoroughly tested technique.
- Currently showing very promising results.
- Highly customisable and configurable.

Creating the Fingerprint Index

- Addition of terms.

Creating the Fingerprint Index

- Addition of terms.
- Requires an Fingerprint generation along with implementation and traversal of the Index tree structure.

Creating the Fingerprint Index

- Addition of terms.
- Requires an Fingerprint generation along with implementation and traversal of the Index tree structure.
- Retrieval of terms.

Creating the Fingerprint Index

- Addition of terms.
- Requires an Fingerprint generation along with implementation and traversal of the Index tree structure.
- Retrieval of terms.
- Requires implementation of the comparison tables and a more complex Index traversal algorithm.

Beagle's Main Procedure

- Maintain two Clause Sets, *new* and *old*.
  Remove Clauses from *new* one at a time, simplify them and then attempt inference rules.

Beagle's Main Procedure

- Maintain two Clause Sets, *new* and *old*.
  Remove Clauses from *new* one at a time, simplify them and then attempt inference rules.
- Two key areas of improvement ($O(n)$ operations):
- Inferences via the Superposition rules.
- Simplifying Clauses.

Indexing Superposition

- Refer to rule. Requires...

Indexing Simplification

- Simplification rules exist to implement special cases of the rules in the Hierarchic Superposition with Weak Abstraction Calculus.
- These special cases allow redundant Clauses to be removed; preventing clutter in the inference process.

Indexing Simplification

- Simplification rules exist to implement special cases of the rules in the Hierarchic Superposition with Weak Abstraction Calculus.
- These special cases allow redundant Clauses to be removed; preventing clutter in the inference process.
- The two main simplification rules used by Beagle are *Negative Unit Simplification* and *Demodulation*

Indexing Simplification

- Simplification rules exist to implement special cases of the rules in the Hierarchic Superposition with Weak Abstraction Calculus.
- These special cases allow redundant Clauses to be removed; preventing clutter in the inference process.
- The two main simplification rules used by Beagle are *Negative Unit Simplification* and *Demodulation*
- These rules require checking for *subsumption*, which requires implementing a new comparison table for Fingerprint Indexing.

Indexing Negative Unit Simplification

Negative Unit Simplification $\dfrac{l \not\approx r \qquad s \approx t \vee C}{C}$

Where $(i)\ \ \exists\sigma\ \ s.t.\ \ (l \approx r)\sigma \equiv s \approx t.$

The clause $s \approx t \vee C$ may be removed.

Indexing Negative Unit Simplification

Negative Unit Simplification $\dfrac{l \not\approx r \qquad s \approx t \vee C}{C}$

Where $(i)\ \ \exists\sigma\ \ s.t.\ \ (l \approx r)\sigma \equiv s \approx t.$

The clause $s \approx t \vee C$ may be removed.

- Searching for valid subsuming Literals is extremely time consuming.
- Requires an index capable of matching Equations rather than Terms.

Indexing Demodulation

Demodulation $$\frac{l \to r \qquad s[u] \approx t \vee D}{s[r\sigma] \approx t \vee D}$$

Where $l\sigma = u$

The clause $s[u] \approx t \vee D$ may be removed.

Indexing Demodulation

Demodulation $$\frac{l \to r \qquad s[u] \approx t \lor D}{s[r\sigma] \approx t \lor D}$$

Where $l\sigma = u$

The clause $s[u] \approx t \lor D$ may be removed.

- Like in Negative Unit Simplification, the most costly operation is searching for subsuming l Terms.
- We must perform this search for every possible subterm u of s.

Fingerprint Indexing for the Hierarchic Superposition with Weak Abstraction Calculus

- The Hierarchic Superposition with Weak Abstraction Calculus imposes many restrictions on what can be used for inference.

Fingerprint Indexing for the Hierarchic Superposition with Weak Abstraction Calculus

- The Hierarchic Superposition with Weak Abstraction Calculus imposes many restrictions on what can be used for inference.
- We may take advantage of some of these conditions to increase the effectiveness of Fingerprint Indexing.

Table: Fingerprint comparison table for unification; extended by considering the term hierarchy.

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

Metrics for Analysing Indexing Performance

- Speed - Not necessarily relevant

Metrics for Analysing Indexing Performance

- Speed - Not necessarily relevant
- False Positives - Relevant, but can be misleading depending on number of positions being sampled.

Metrics for Analysing Indexing Performance

- Speed - Not necessarily relevant
- False Positives - Relevant, but can be misleading depending on number of positions being sampled.
- Time Spent *per Inference* - Booyah

Comparing Varieties of Beagle

Table: Totalled inference counts and indexing statistics for various versions of beagle.

	Inference Counts			Indexing Results		
Version	Sup	Demod	NegUnit	TotalFound	SupFP	SimpFP
Unmodified [1]	414216	29097	1826	0	0	0
Standard	162881	41414	2452	61884768	15525	39778148
Enhanced	162997	41435	2454	58535681	15401	39779224

Table: Totalled timing results for various versions of beagle.

	Time Spent (seconds)					
Version	Indexing	Retrieving	Sup	Demod	NegUnit	Total
Unmodified [1]	0	0	730.44	9.44	31.99	5623.21
Standard	28.4	38.73	254.17	41.66	3.18	381.36
Enhanced	22.91	20.29	180.54	32.6	2.51	281.38

[1] This version failed to solve two out of the fifty problems within 8 hours.

Results Analysis

- Times per inference

Results Analysis

- Times per inference
- Extreme examples

Fingerprint Sampling Varieties

- Reasoning. Cite shulz and FP/Speed balance

Fingerprint Sampling Varieties

- Reasoning. Cite shulz and FP/Speed balance
- Different position samples

Fingerprint Sampling Varieties

Table: Totalled inference counts and indexing statistics for various Fingerprint sampling sets.

	Inference Counts			Indexing Results		
Sample Set	Sup	Demod	NegUnit	TotalFound	SupFP	SimpFP
FP3W	164574	42402	2473	59433145	72051	40037948
FP4M	150154	35709	1964	58989318	29469	40073471
FP6M	146861	35326	1960	58119897	17641	39916687
FP7	161411	41005	2441	58530669	23903	39818531
FP8X2	161741	40876	2439	58336597	11754	39823989

Table: Totalled timing results for various Fingerprint sampling sets.

	Time Spent (seconds)					
Sample Set	Indexing	Retrieving	Sup	Demod	NegUnit	Total
FP3W	12.48	14.86	177.82	28.98	1.79	265.65
FP4M	14.37	15.02	173.32	31.53	1.83	261.84
FP6M	18.74	17.58	168.79	30.56	2.12	259.02
FP7	22.26	19.82	180.13	35.23	2.38	282.22
FP8X2	51.73	34.23	195.18	42.75	4.07	331.01