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 15, 2013

Outline
Project Overview
Motivation
Background
First Order Logic Terminology
The Beagle Theorem Prover
Term Indexing
Fingerprint Indexing
Implementation
Implementing Fingerprint Indexing
Indexing Applications
Tailoring to Beagle
Results
Evaluation Metrics
Beagle Comparisons
Sample Position Comparisons
Conclusion

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

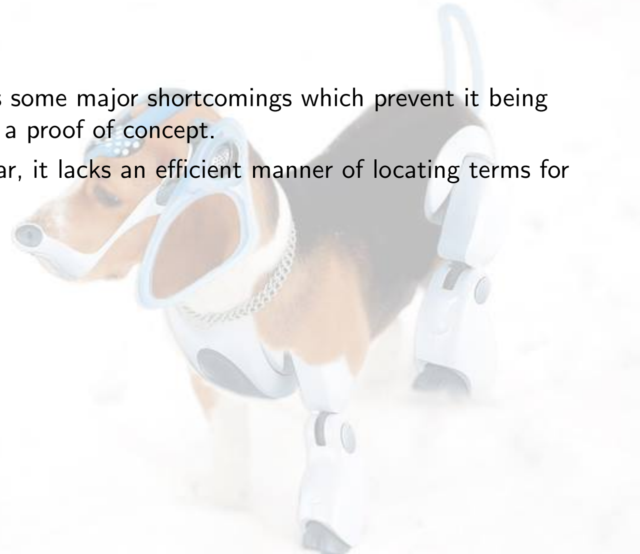
- Beagle is a First-Order-Logic resolution theorem prover with equality, built to show off the capabilities of the Hierarchic Superposition with Weak Abstraction Calculus.
- This calculus is capable of *hierarchic reasoning* by incorporating a *background prover*.
- Background provers act as a black 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.

Extending 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.



Extending 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.

Extending 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.
- Research Questions:
  - How can Term Indexing (in particular Fingerprint Indexing) be implemented and applied to Beagle?
  - How can Fingerprint Indexing be improved; generally and with respect to Beagle's specific calculus?
  - What sort of improvement will this implementation yield in the prover?

Terminology Used in this Presentation

- Position: List of integers indicating a precise subterm. $s[u]$ refers to a term s with a subterm u.

Terminology Used in this Presentation

- Position: List of integers indicating a precise subterm. $s[u]$ refers to a term s with a subterm u.
- 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) $\sigma = \mathrm{mgu}\ (l, u)$,

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

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.*

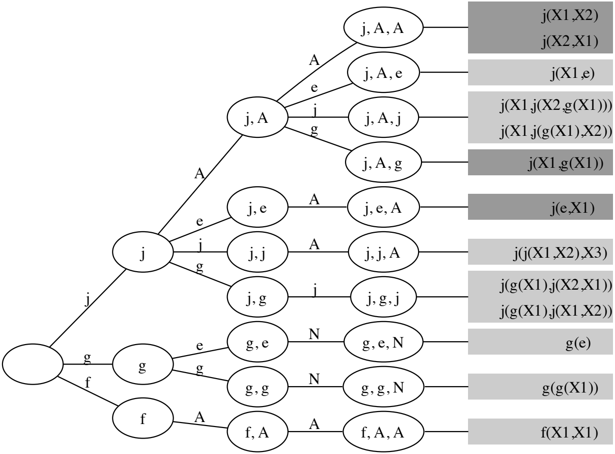
Table: Fingerprint Feature comparison tables for *unification* (left) and *subsumption* (right)

	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

- Two main tasks: Add terms to the index and retrieve them.

Creating the Fingerprint Index

- Two main tasks: Add terms to the index and retrieve them.
- Addition requires Fingerprint generation along with implementation and traversal of the Index tree structure.

Creating the Fingerprint Index

- Two main tasks: Add terms to the index and retrieve them.
- Addition requires Fingerprint generation along with implementation and traversal of the Index tree structure.
- Retrieval requires implementation of the comparison tables and a more complex Index traversal algorithm. We must collect all compatible leaves and union them together.

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. Results are added to *new*, simplified Clause is added to *old*.

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. Results are added to *new*, simplified Clause is added to *old*.
- Two key areas of improvement:
  - Inferences via the Superposition rules. $O(|old|)$
  - Simplifying Clauses. $O(|old| + |new|)$

Indexing Superposition

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 $(s \approx t \vee D)\sigma$.

- Requires that we index all subterms. Furthermore we must implement two separate cases for *from* and *into*.

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 operate only on *unit Clauses*, so using our current index clogged with subterms is inefficient.
- Thus we will create new indexes which behave differently depending on a configuration object.

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 operate only on *unit Clauses*, so using our current index clogged with subterms is inefficient.
- Thus we will create new indexes which behave differently depending on a configuration object.
- The rules also 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 $(i) \quad \exists \sigma \quad s.t. \quad (l \approx r)\sigma \equiv s \approx t.$

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

Where $l\sigma = u$

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

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.

- For a simple example, with a Literal $X \to f(a)$ we may replace all occurrences of X with $f(a)$.

Indexing Demodulation

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

Where $l\sigma = u$

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

- For a simple example, with a Literal $X \rightarrow f(a)$ we may replace all occurrences of X with $f(a)$.
- 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.

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

- We may measure the performance of Fingerprint Indexing by comparing run statistics of a subset of problems from the TPTP (Thousands of Problems for Theorem Provers) library.
- A subset of 50 problems was created; spanning a range of problem categories and difficulties.

Metrics for Analysing Indexing Performance

- We may measure the performance of Fingerprint Indexing by comparing run statistics of a subset of problems from the TPTP (Thousands of Problems for Theorem Provers) library.
- A subset of 50 problems was created; spanning a range of problem categories and difficulties.
- Total run time - Need to be careful to consider all factors.

Metrics for Analysing Indexing Performance

- We may measure the performance of Fingerprint Indexing by comparing run statistics of a subset of problems from the TPTP (Thousands of Problems for Theorem Provers) library.
- A subset of 50 problems was created; spanning a range of problem categories and difficulties.
- Total run time - Need to be careful to consider all factors.
- False Positives - Relevant, but can be misleading depending on number of positions being sampled.

Metrics for Analysing Indexing Performance

- We may measure the performance of Fingerprint Indexing by comparing run statistics of a subset of problems from the TPTP (Thousands of Problems for Theorem Provers) library.
- A subset of 50 problems was created; spanning a range of problem categories and difficulties.
- Total run time - Need to be careful to consider all factors.
- False Positives - Relevant, but can be misleading depending on number of positions being sampled.
- Run time *per Inference* - Most accurate measure of performance. Must still take care when interpreting.

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	146861	35326	1960	58119897	17641	39916687

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	18.74	17.58	168.79	30.56	2.12	259.02

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

Time Spent Per Inference

Table: Superposition time for the 6 most extreme problem examples.

Version	Superposition	Demodulation	NegUnit Simplification
Unmodified	1.7ms	0.3ms	17.5ms
Standard	1.5ms	1.0ms	1.3ms
Enhanced	1.1ms	0.8ms	1.0ms

Time Spent Per Inference

Table: Superposition time for the 6 most extreme problem examples.

Version	Superposition	Demodulation	NegUnit Simplification
Unmodified	1.7ms	0.3ms	17.5ms
Standard	1.5ms	1.0ms	1.3ms
Enhanced	1.1ms	0.8ms	1.0ms

- Most typical application of Demodulation is a Literal like $X \to f(a)$. X will match anything, making fingerprint indexing a waste of time.

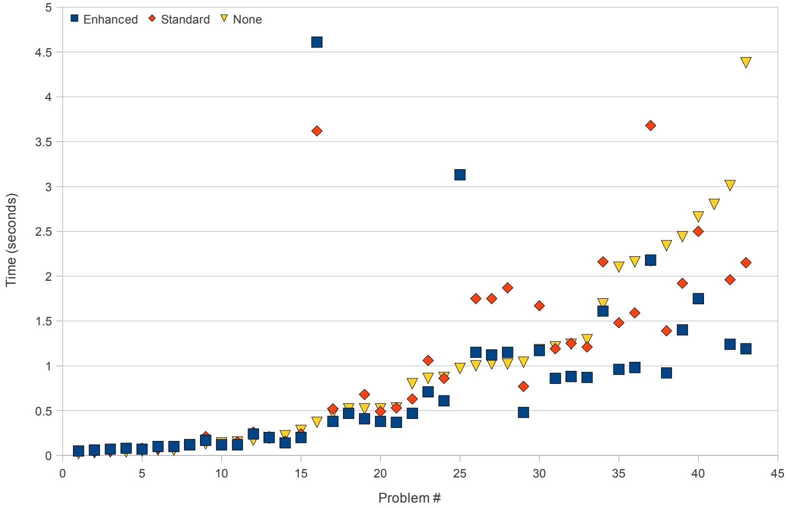
Time Spent Per Inference

Table: Superposition time for the 6 most extreme problem examples.

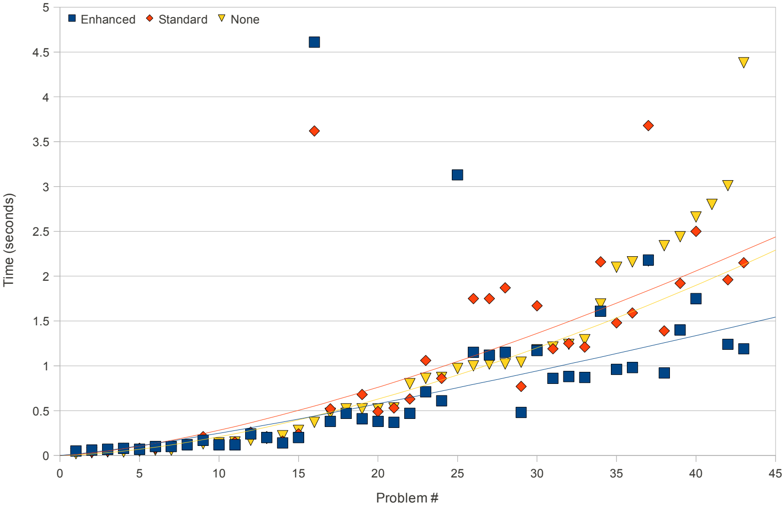
Version	Superposition	Demodulation	NegUnit Simplification
Unmodified	1.7ms	0.3ms	17.5ms
Standard	1.5ms	1.0ms	1.3ms
Enhanced	1.1ms	0.8ms	1.0ms

- Most typical application of Demodulation is a Literal like $X \to f(a)$. X will match anything, making fingerprint indexing a waste of time.
- When excluding PUZ037-1.p we have 0.29, 0.39 and 0.31 milliseconds per Demodulation.

Runtimes under 5 seconds



Runtimes under 5 seconds



Results Analysis

Table: Superposition time for the 6 most extreme problem examples.

Problem	Enhanced	Standard	Unmodified
DAT050=1.p	17.53	31.54	48.62
DAT039=1.p	13.2	22.51	130.77
DAT040=1.p	14.49	21.29	190.71
DAT038=1.p	12.53	24.04	294.86
DAT043=1.p	18.67	26.08	**N/A**
DAT048=1.p	17.65	35.77	**N/A**

Results Analysis

Table: Superposition time for the 6 most extreme problem examples.

Problem	Enhanced	Standard	Unmodified
DAT050=1.p	17.53	31.54	48.62
DAT039=1.p	13.2	22.51	130.77
DAT040=1.p	14.49	21.29	190.71
DAT038=1.p	12.53	24.04	294.86
DAT043=1.p	18.67	26.08	**N/A**
DAT048=1.p	17.65	35.77	**N/A**

- When taking number of inferences into account for DAT038=1.p we observe 1.2 milliseconds per superposition for the full implementation versus 2.2 milliseconds per superposition for unindexed beagle.

Fingerprint Sampling Varieties

- We have yet to consider the impact of varying the configuration for Fingerprint Indexing; in particular the varying the list of positions which we are sampling.

Fingerprint Sampling Varieties

- We have yet to consider the impact of varying the configuration for Fingerprint Indexing; in particular the varying the list of positions which we are sampling.
- It is important to strike a balance between accuracy and a simple index structure.

Fingerprint Sampling Varieties

- We have yet to consider the impact of varying the configuration for Fingerprint Indexing; in particular the varying the list of positions which we are sampling.
- It is important to strike a balance between accuracy and a simple index structure.
- We present some results for some of the most successful position sets from Shulz's paper.
  - FP3W: ϵ, 1, 2
  - FP4M: ϵ, 1, 2, 1.1
  - FP6M: ϵ, 1, 2, 3, 1.1, 1.2
  - FP7: ϵ, 1, 2, 1.1, 1.2, 2.1, 2.2
  - FP8X2:ϵ, 1, 2, 3, 4, 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, 1.1.1, 2.1.1

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	162218	42402	2472	13913606	69429	1815992
FP4M	147798	35709	1963	13469779	26847	1851515
FP6M	144505	35326	1959	12601762	16406	1694731
FP7	159055	41005	2440	13011130	21281	1596575
FP8X2	159385	40876	2438	12819184	11229	1602033

Table: Totalled timing results for various Fingerprint sampling sets.

	Time Spent (seconds)					
Sample Set	Indexing	Retrieving	Sup	Demod	NegUnit	Total
FP3W	11.52	14.02	170.37	9.26	1.78	237.75
FP4M	13.09	14.12	164.95	9.51	1.82	230.68
FP6M	16.82	16.5	159.93	10.78	2.11	229.59
FP7	19.98	18.74	170.83	12.37	2.37	249.22
FP8X2	45.56	32.59	181.43	21.45	4.06	294.8

Note that for more relevant comparisons these results exclude PUZ037-1.p.

The Benefits of Indexing Beagle

- The Hierarchic Superposition with Weak Abstraction Calculus is a great leap forward for hierarchic reasoning; and the benefits that it offers are worth exploring thoroughly.

The Benefits of Indexing Beagle

- The Hierarchic Superposition with Weak Abstraction Calculus is a great leap forward for hierarchic reasoning; and the benefits that it offers are worth exploring thoroughly.
- By adding Fingerprint Indexing to Beagle, this project has made this task far more approachable by greatly increasing the number of logic problems which can be solved within a reasonable timeframe.

The Benefits of Indexing Beagle

- The Hierarchic Superposition with Weak Abstraction Calculus is a great leap forward for hierarchic reasoning; and the benefits that it offers are worth exploring thoroughly.
- By adding Fingerprint Indexing to Beagle, this project has made this task far more approachable by greatly increasing the number of logic problems which can be solved within a reasonable timeframe.
- We have also shown that tailoring indexing to the specific needs of a calculus is worthwhile; improving results by 50% in some cases.

The Benefits of Indexing Beagle

- The Hierarchic Superposition with Weak Abstraction Calculus is a great leap forward for hierarchic reasoning; and the benefits that it offers are worth exploring thoroughly.
- By adding Fingerprint Indexing to Beagle, this project has made this task far more approachable by greatly increasing the number of logic problems which can be solved within a reasonable timeframe.
- We have also shown that tailoring indexing to the specific needs of a calculus is worthwhile; improving results by 50% in some cases.
- In Shulz's paper Fingerprint Indexing appeared to be a very promising technique. This project has verified it's viability in the term indexing field and has made Beagle viable as far more than a proof of concept.