Computation of Alias Sets from Shape Graphs for Comparison of Shape Analysis Precision

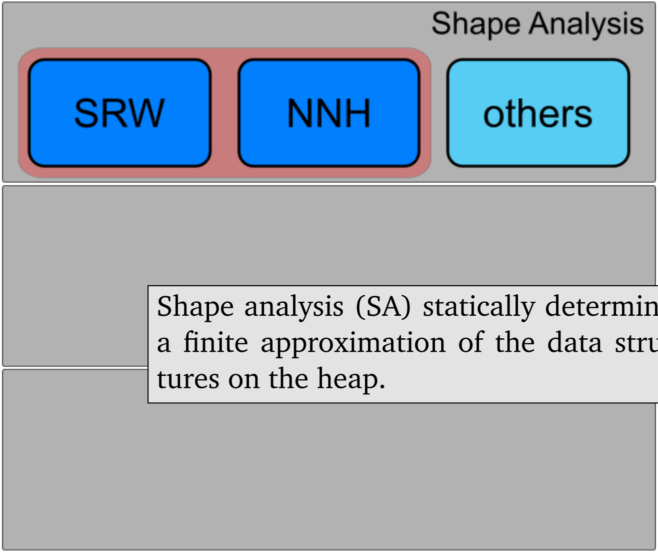
Viktor Pavlu,[1] Markus Schordan,[2] Andreas Krall[1]

[1]Vienna University of Technology
[2]University of Applied Sciences Technikum Wien

SCAM 2011, Williamsburg

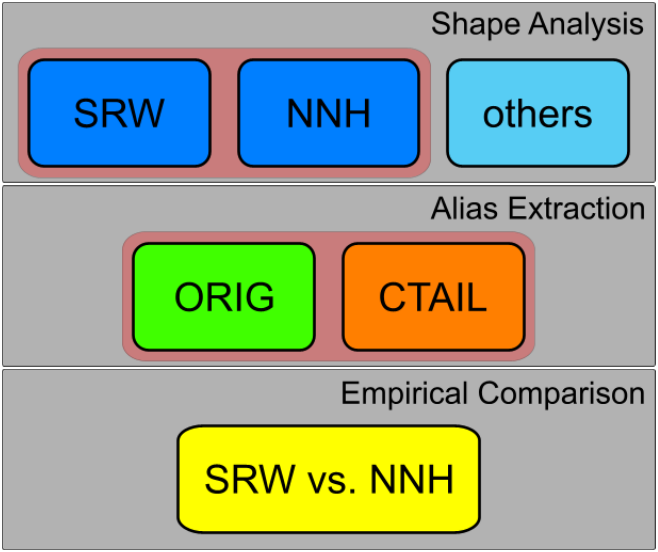
September 2011

Overview



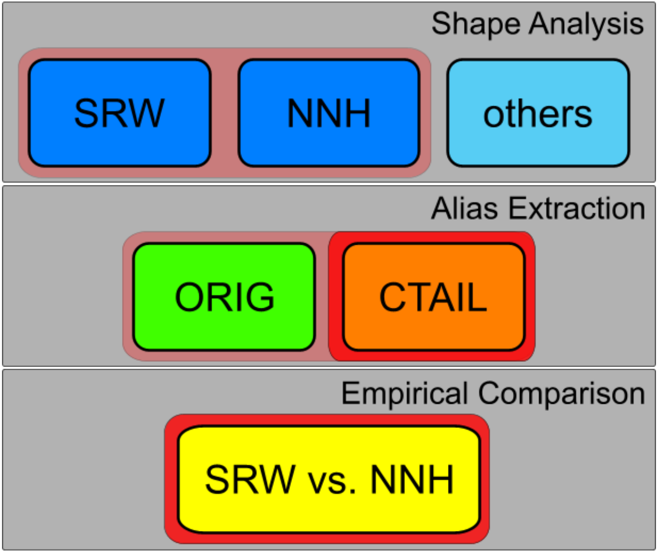
Shape analysis (SA) statically determines a finite approximation of the data structures on the heap.

Overview



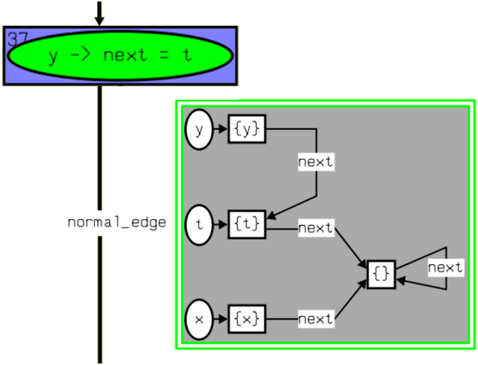
Shape Analysis
SRW
NNH
others
Alias Extraction
ORIG
CTAIL
Empirical Comparison
SRW vs. NNH

Overview


Shape Analysis
SRW
NNH
others
Alias Extraction
ORIG
CTAIL
Empirical Comparison
SRW vs. NNH

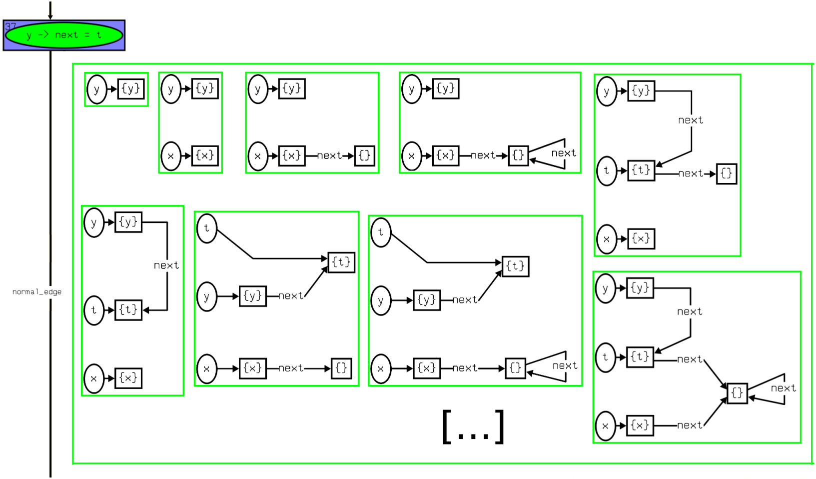
SRW Shape Analysis

[SRW98] Mooly Sagiv, Thomas W. Reps, and Reinhard Wilhelm. Solving shape-analysis problems in languages with destructive updating. ACM Transactions on Programming Languages and Systems (TOPLAS), 20(1):1–50, January 1998.



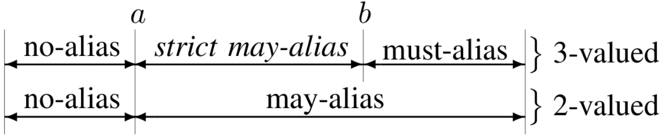
NNH Shape Analysis

[NNH05] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. Principles of Program Analysis, chapter Shape Analysis, pages 102–129. Springer, 2005.



Comparing the Precision of Shape Analyses

- Comparing Shape Analysis results directly does not work.
- Naive idea: convert representation, then compare.
  - ➥ Compare usefulness for a specific application
  - ➥ In our work: Aliasing Information for Compiler Optimizations



Relation between 2-valued and 3-valued alias sets.

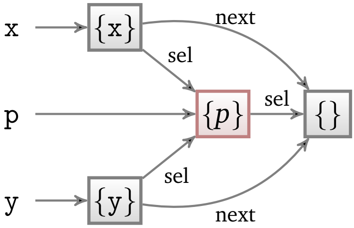
Benchmarks for Graph-based Shape Analyses

- [RS01] Noam Rinetzky, Mooly Sagiv. Interprocedural Shape Analysis for Recursive Programs. LNCS 2027, 133–149, 2001.
- [SRW98] Mooly Sagiv, Thomas W. Reps, and Reinhard Wilhelm. Solving shape-analysis problems in languages with destructive updating. ACM Transactions on Programming Languages and Systems (TOPLAS), 20(1):1–50, January 1998.

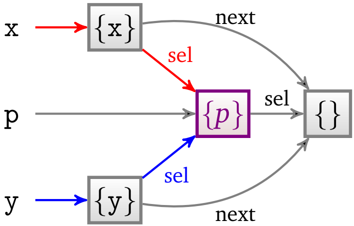
Operations on Linked Lists

- `insert`, `append`, `remove`, `delall`, `search`, `reverse`, `merge`
- Two versions each: with loops (`lp`), and unrolled (`nb`)

Existing Approach (RSW02):
Do path expressions refer to same node?



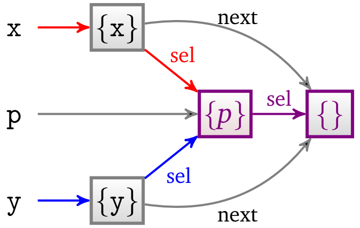
Alias Computation for a single SG_i



Existing Approach (RSW02):
Do path expressions refer to same node?

$$(\langle x, sel \rangle, \langle y, sel \rangle) \qquad \in \textbf{Must}$$

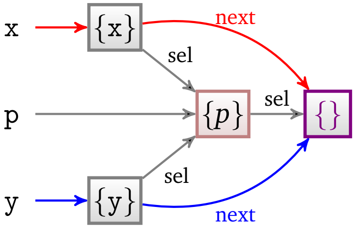
Alias Computation for a single SG_i



Existing Approach (RSW02):
Do path expressions refer to same node?

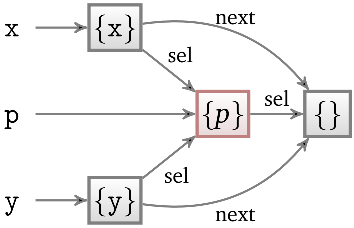
$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{May}$$

Alias Computation for a single SG_i



Existing Approach (RSW02):
Do path expressions refer to same node?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{May}$$
$$(\langle x, next\rangle, \langle y, next\rangle) \in \textbf{May}$$

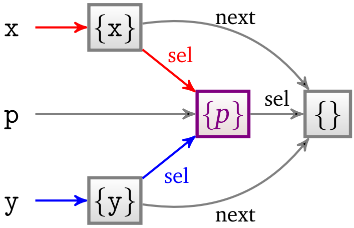


Existing Approach (RSW02):
Do path expressions refer to same node?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{May}$$
$$(\langle x, next\rangle, \langle y, next\rangle) \in \textbf{May}$$

Our "Common Tail" Improvement:
Do path expressions meet at same named node and share a common tail of selectors?

Alias Computation for a single SG_i



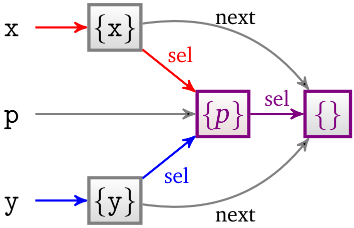
Existing Approach (RSW02):
Do path expressions refer to same node?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{May}$$
$$(\langle x, next\rangle, \langle y, next\rangle) \in \textbf{May}$$

Our "Common Tail" Improvement:
Do path expressions meet at same named node and share a common tail of selectors?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$

Alias Computation for a single SG_i



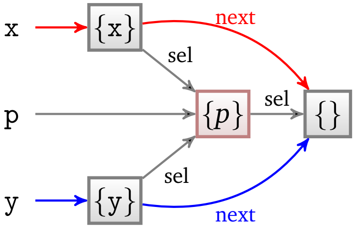
Existing Approach (RSW02):
Do path expressions refer to same node?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{May}$$
$$(\langle x, next\rangle, \langle y, next\rangle) \in \textbf{May}$$

Our "Common Tail" Improvement:
Do path expressions meet at same named node and share a common tail of selectors?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{Must}$$

Alias Computation for a single SG_i



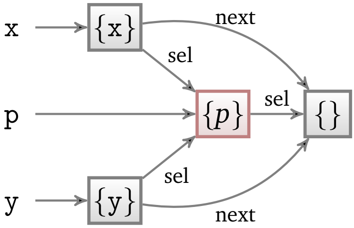
Existing Approach (RSW02):
Do path expressions refer to same node?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{May}$$
$$(\langle x, next\rangle, \langle y, next\rangle) \in \textbf{May}$$

Our "Common Tail" Improvement:
Do path expressions meet at same named node and share a common tail of selectors?

$$(\langle x, sel\rangle, \langle y, sel\rangle) \in \textbf{Must}$$
$$(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) \in \textbf{Must}$$
$$(\langle x, next\rangle, \langle y, next\rangle) \in \textbf{No}$$

Alias Computation for a single SG_i



Existing Approach (RSW02):
Do path expressions refer to same node?

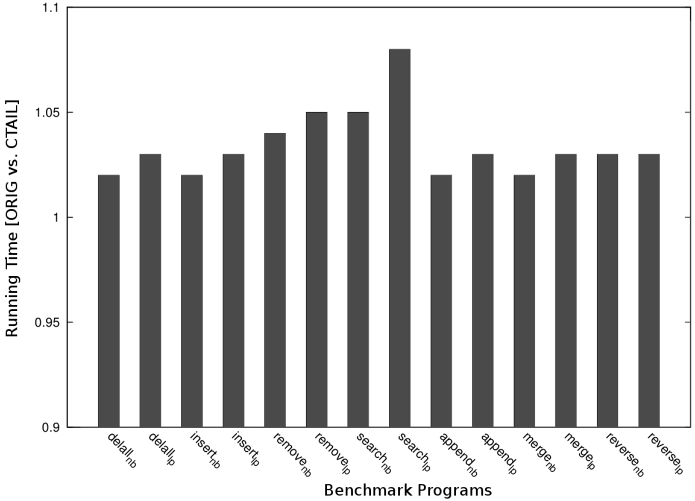
$$
\begin{array}{ll}
(\langle x, sel\rangle, \langle y, sel\rangle) & \in \textbf{Must} \\
(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) & \in \textbf{May} \\
(\langle x, next\rangle, \langle y, next\rangle) & \in \textbf{May}
\end{array}
$$

Our "Common Tail" Improvement:
Do path expressions meet at same named node and share a common tail of selectors?

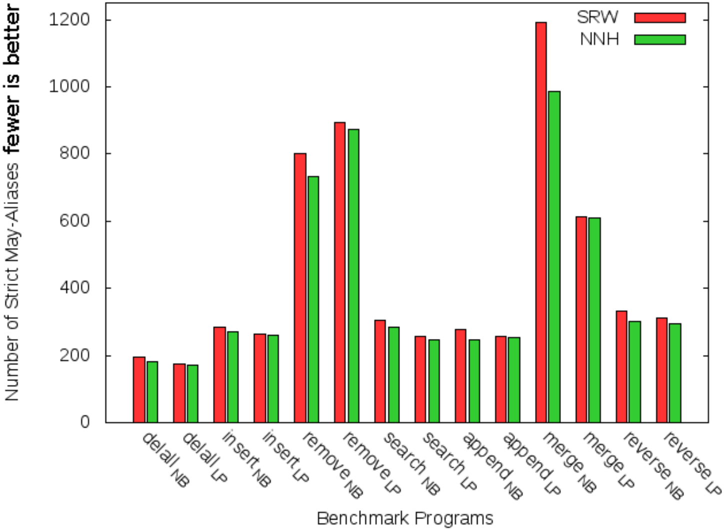
$$
\begin{array}{ll}
(\langle x, sel\rangle, \langle y, sel\rangle) & \in \textbf{Must} \\
(\langle x, sel, sel\rangle, \langle y, sel, sel\rangle) & \in \textbf{Must} \\
(\langle x, next\rangle, \langle y, next\rangle) & \in \textbf{No}
\end{array}
$$

Runtime of CTAIL vs. ORIG

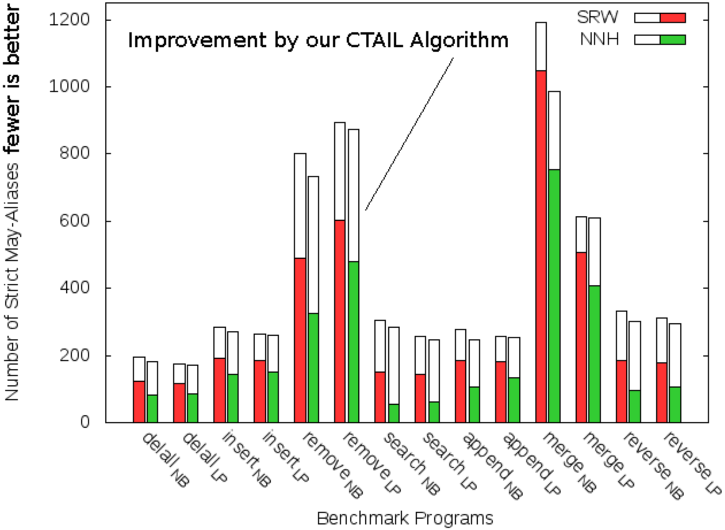
- Path-Checks not too expensive
- Overhead for CTAIL well below 10% of ORIG's runtime



SRW vs. NNH – using ORIG



SRW vs. NNH – using CTAIL



Conclusion

Contributions

- Improved Alias Extraction Algorithm of [RSW02] comparing paths instead of only the final nodes
- Up to $5\times$ more precise with only $< 10\%$ runtime overhead
- First implementation of the NNH Shape Analysis algorithm
- Empirical comparison of SRW and NNH Shape Analyses

Result

- With CTAIL we can extract from NNH shape graphs $1.62\times$ more precise aliasing information than from SRW results on average for our benchmarks.
- Without CTAIL (ORIG) only $1.06\times$ more precise aliasing information can be extracted.

Some More Pointers

http://www.complang.tuwien.ac.at/vpavlu

Bibliography:

- [NNH05] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. Principles of Program Analysis, chapter Shape Analysis, pages 102–129. Springer, 2005.
- [RSW02] Thomas W. Reps, Mooly Sagiv, and Reinhard Wilhelm. Shape analysis and applications. In The Compiler Design Handbook: Optimizations and Machine Code Generation, pages 175–218. CRC Press, 2002.
- [SRW98] Mooly Sagiv, Thomas W. Reps, and Reinhard Wilhelm. Solving shape-analysis problems in languages with destructive updating. ACM Transactions on Programming Languages and Systems (TOPLAS), 20(1):1–50, January 1998.

Improved Precision with CTAIL

	Algorithm: ORIG		Algorithm: CTAIL	
	$n_\emptyset \notin is$	$n_\emptyset \in is$	$n_\emptyset \notin is$	$n_\emptyset \in is$
with	$\frac{1}{2}$	$\frac{1}{2}$	1	1
without	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$

$n_\emptyset \in (\notin)\ is$: the summary location is shared (unshared).
with (without): there is a (no) intermediate node from which both expressions share a common tail of selectors.