A Relational Approach to Context-Free Language Reachability

Nicholas Hollingum Bernhard Scholz

The University of Sydney
SAPLInG 2015

20th Nov, 2015

CFL-R

The CFL-R Context

- Computational engine
- Extends context-free language recognition for graphs [Yannakakis, 1990]
  - Find paths in an edge-labeled graph
  - Spell words in the input language
- Wide range of applications for CFL-R:
  - Program analysis [Reps, 1998]
  - Logic programming [Yannakakis, 1990]
  - Formal verification [Dolev et al., 1982]

Example - Points-To Analysis

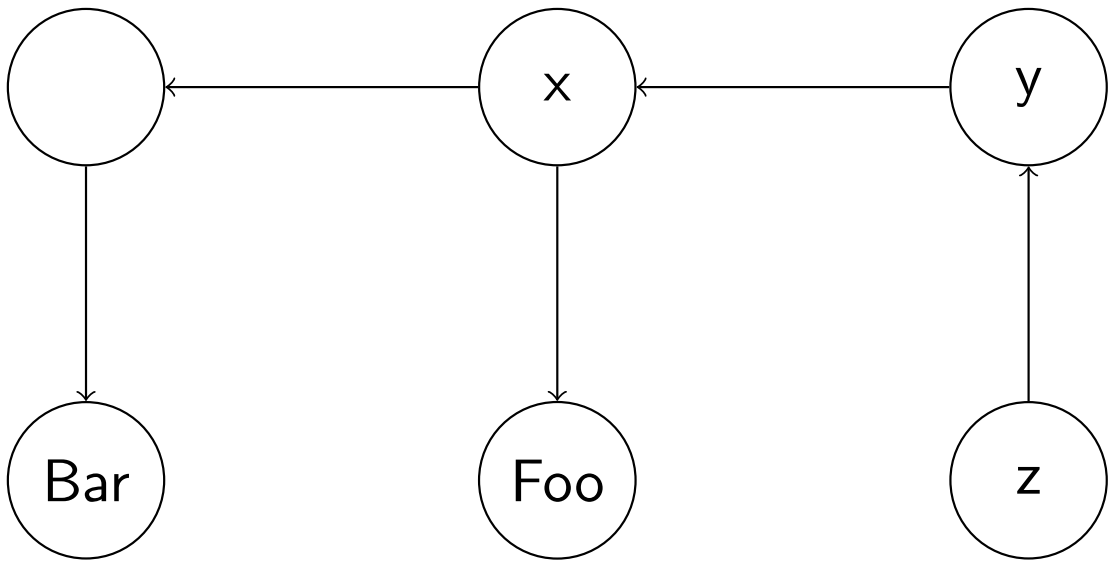
- Points-To analysis
- Necessary for many optimisations and other analyses
- Which variables point-to which heap-locations at runtime

```
x = malloc(Foo)
*x = malloc(Bar)
y = x
z = *y
```

Example - Points-To Analysis

- Points-To analysis
- Necessary for many optimisations and other analyses
- Which variables point-to which heap-locations at runtime

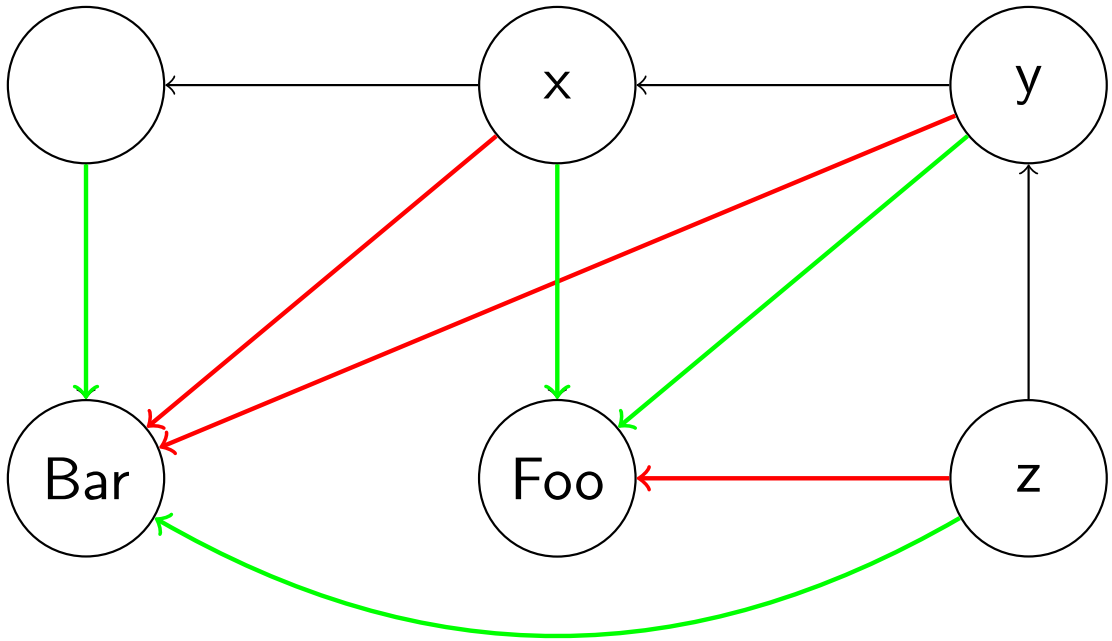
```
x = malloc(Foo)
*x = malloc(Bar)
y = x
z = *y
```



Example - Points-To Analysis

- Transitive closure/Reachability
- Imprecise: over-approximates the correct answer
- Program semantics can be used to improve accuracy

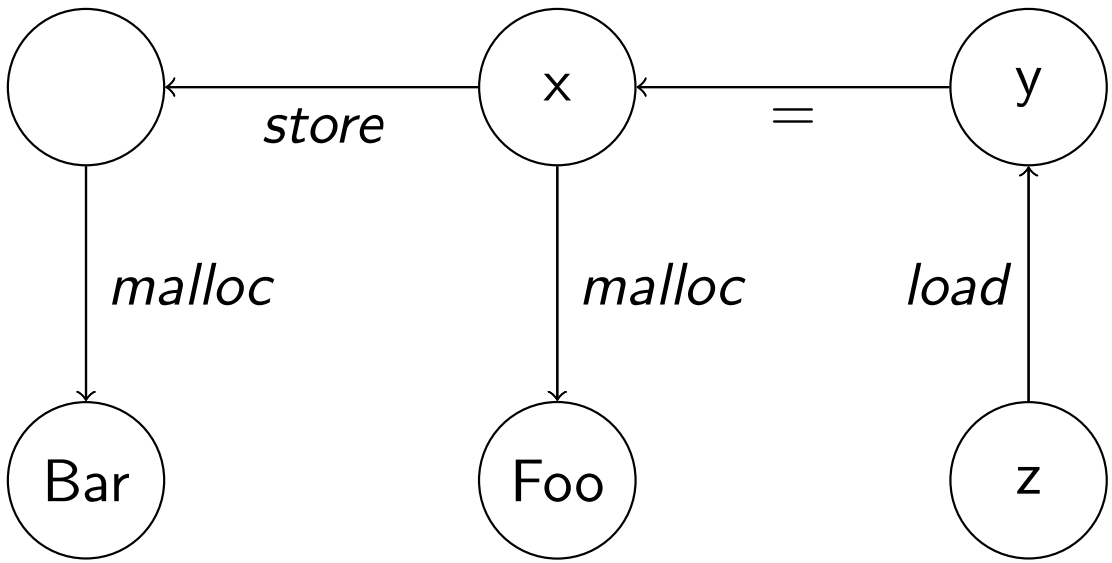
```
x = malloc(Foo)
*x = malloc(Bar)
y = x
z = *y
```



Example - Points-To Analysis

- Label edges
- Paths spell a word
- Words in the language

```
x = malloc(Foo)
*x = malloc(Bar)
y = x
z = *y
```

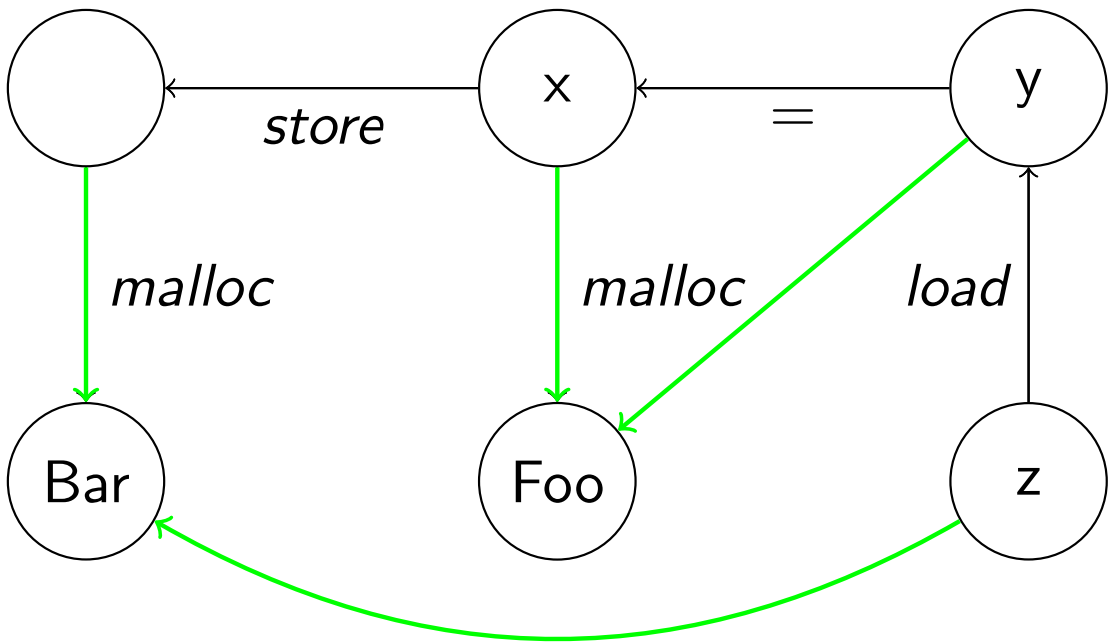


Example - Points-To Analysis

- Label edges
- Paths spell a word
- Words in the language

- $PT \rightarrow malloc \mid = PT \mid \approx PT$
- $alias \rightarrow PT\ \overline{PT}$
- $\approx \rightarrow load\ alias\ store$

```
x = malloc(Foo)
*x = malloc(Bar)
y = x
z = *y
```



State-of-the-Art Algorithms

- Melski and Reps [Melski and Reps, 2000]
  - Worst case runtime complexity: $\mathcal{O}(n^3)$, $(n = |V|)$
  - Dynamic programming with path-summarisation

- Chaudhuri [Chaudhuri, 2008]
  - Worst case runtime complexity: $\mathcal{O}(\frac{n^3}{\log n})$
  - Adapts the Four Russians' trick

State-of-the-Art Algorithms

- Melski and Reps [Melski and Reps, 2000]
  - Worst case runtime complexity: $\mathcal{O}(n^3)$, $(n = |V|)$
  - Dynamic programming with path-summarisation
  - Cubic bottleneck [Heintze and McAllester, 1997] slows down analyses
- Chaudhuri [Chaudhuri, 2008]
  - Worst case runtime complexity: $\mathcal{O}(\frac{n^3}{\log n})$
  - Adapts the Four Russians' trick

State-of-the-Art Algorithms

- Melski and Reps [Melski and Reps, 2000]
  - Worst case runtime complexity: $\mathcal{O}(n^3)$, $(n = |V|)$
  - Dynamic programming with path-summarisation
  - Cubic bottleneck [Heintze and McAllester, 1997] slows down analyses

- Chaudhuri [Chaudhuri, 2008]
  - Worst case runtime complexity: $\mathcal{O}(\frac{n^3}{\log n})$
  - Adapts the Four Russians' trick
  - Logarithmic speedup only
  - Significant memory cost, requires $\Theta(n^2)$ storage

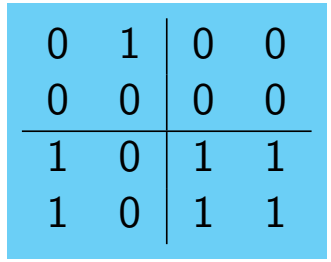
A Relational Approach

Semi-naïve Evaluation

- Adapted from the datalog context
- Use recently discovered (Δ) paths to compute new paths
  - $A \to BC$
  - New Δ_A relations come from at least one Δ_B, Δ_C
- Order the choices of Δ to avoid eager computation
  - $A \to BC$, choose Δ_B, Δ_C before Δ_A
- Conclude when no new paths are discovered
- Depends on **relational operations**
  - union, intersection, difference, identity
  - composition (join)

Relations - Quadtrees

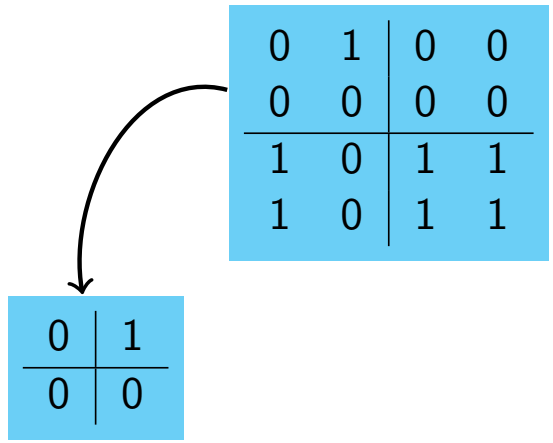
$$\begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{matrix}$$



- Input edges encoded in an adjacency matrix
- Quadtrees to represent adjacency matrices
- Sparse representation, performs well for sparse matrices
- Matrix multiplication can be used to join paths, $\mathcal{O}(m^2 n \log n)$

Relations - Quadtrees

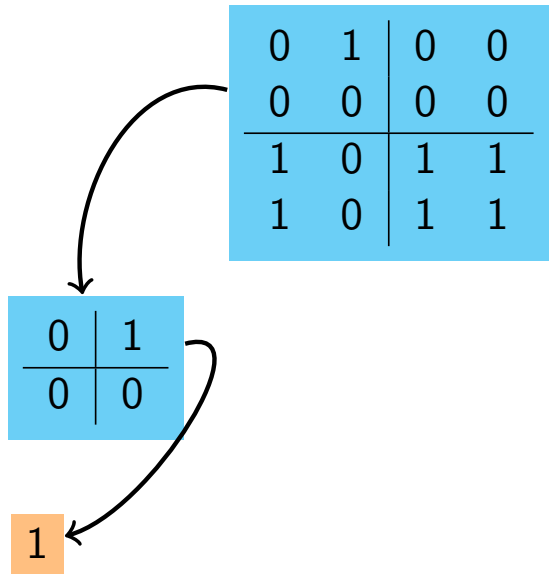
0	1	0	0
0	0	0	0
1	0	1	1
1	0	1	1



- Input edges encoded in an adjacency matrix
- Quadtrees to represent adjacency matrices
- Sparse representation, performs well for sparse matrices
- Matrix multiplication can be used to join paths, $\mathcal{O}(m^2 n \log n)$

Relations - Quadtrees

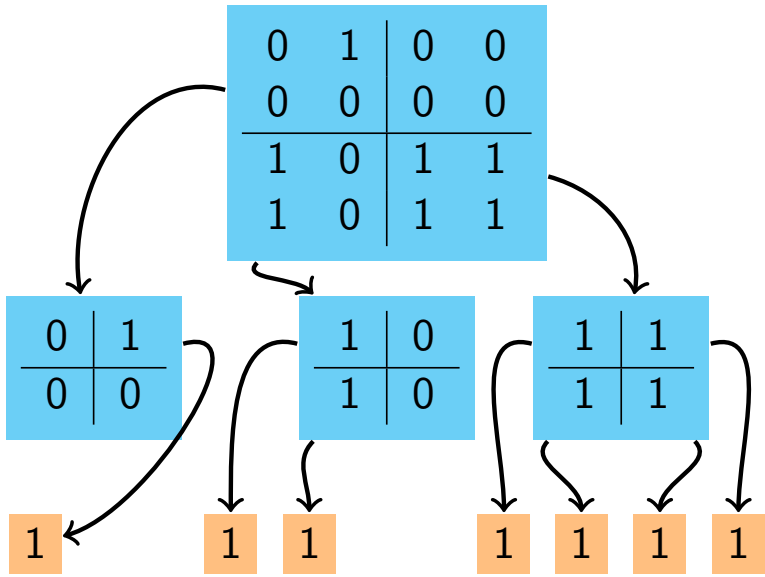
0	1	0	0
0	0	0	0
1	0	1	1
1	0	1	1



- Input edges encoded in an adjacency matrix
- Quadtrees to represent adjacency matrices
- Sparse representation, performs well for sparse matrices
- Matrix multiplication can be used to join paths, $\mathcal{O}(m^2 n \log n)$

Relations - Quadtrees

0	1	0	0
0	0	0	0
1	0	1	1
1	0	1	1



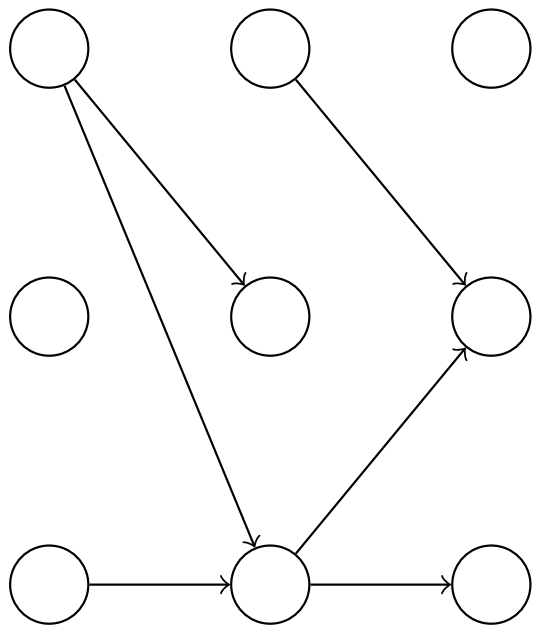
- Input edges encoded in an adjacency matrix
- Quadtrees to represent adjacency matrices
- Sparse representation, performs well for sparse matrices
- Matrix multiplication can be used to join paths, $\mathcal{O}(m^2 n \log n)$

Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors

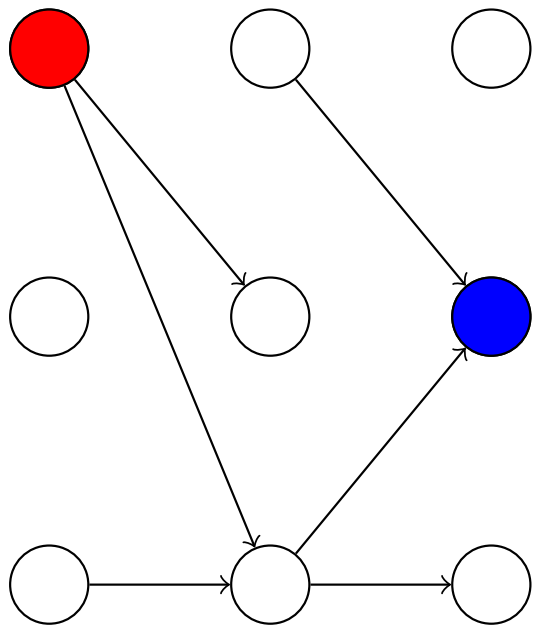
Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors



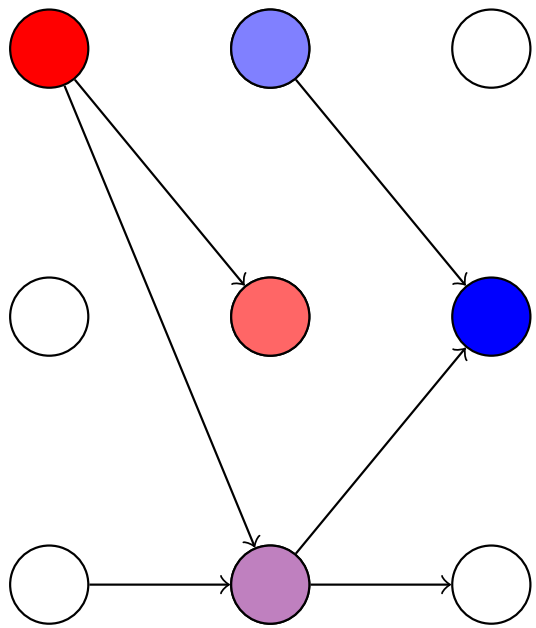
Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors



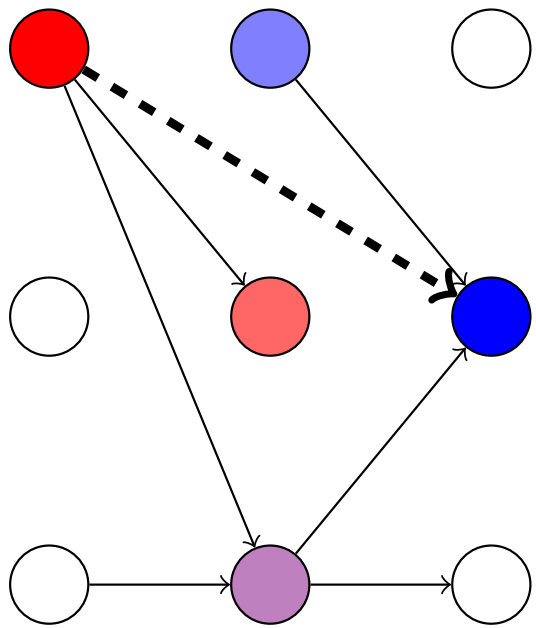
Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors



Relations - Neighbourhood Maps

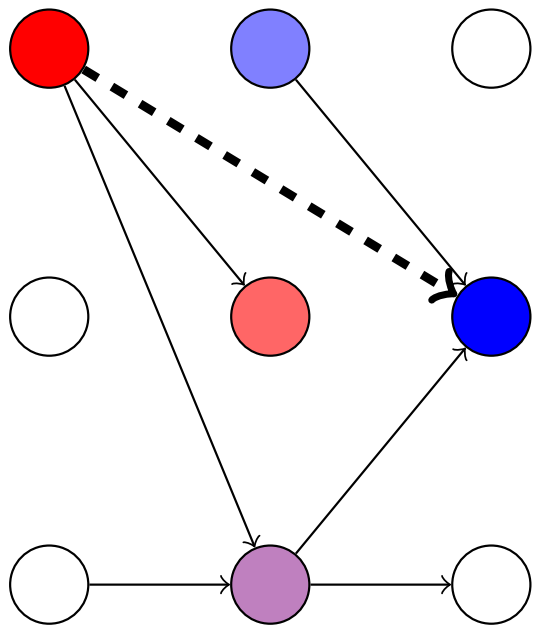
- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors

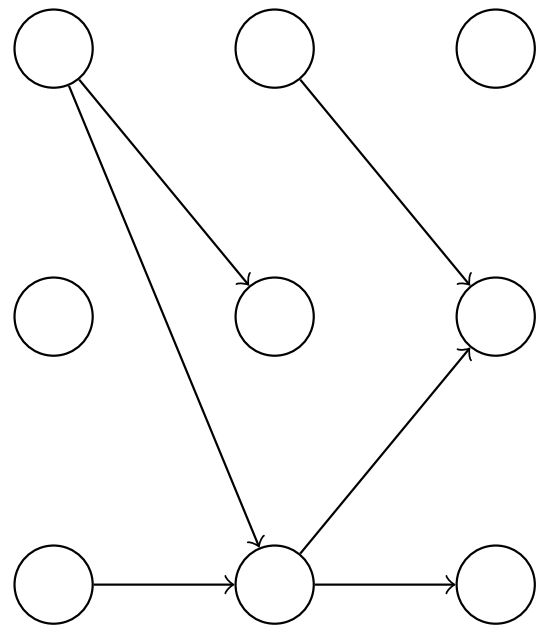


- $\mathcal{O}(n^2 m)$

Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors

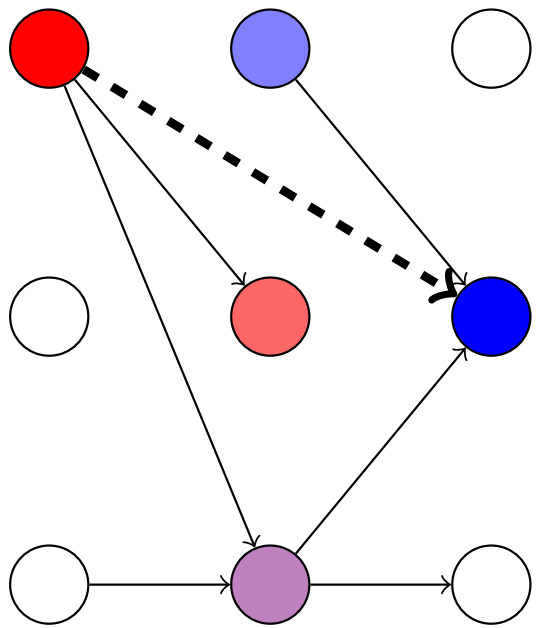


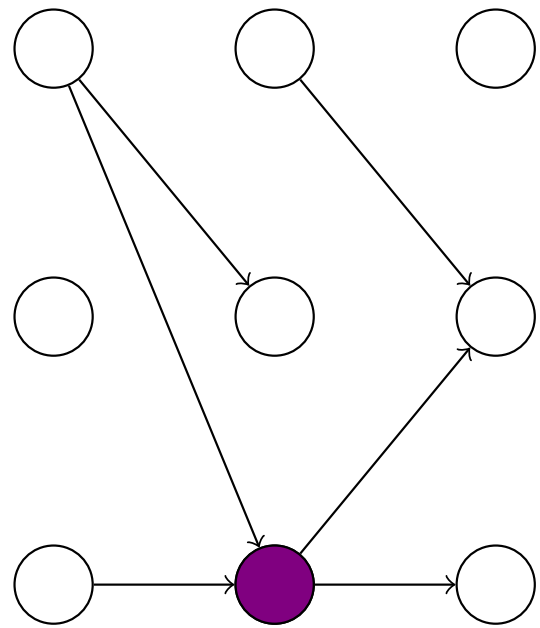


- $\mathcal{O}(n^2 m)$

Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors

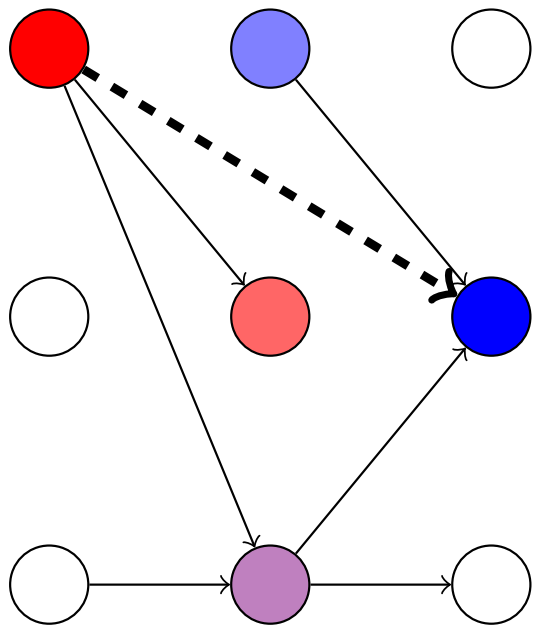


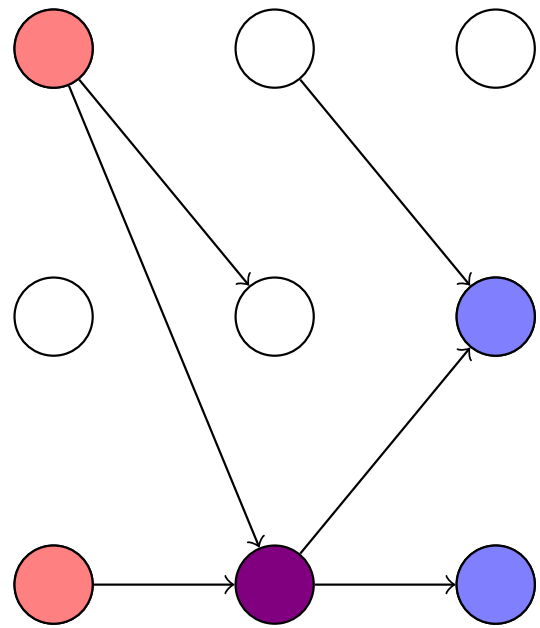


- $\mathcal{O}(n^2 m)$

Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors

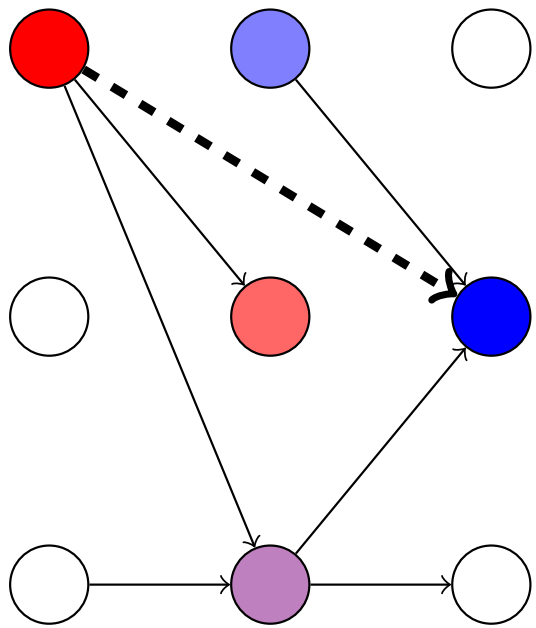




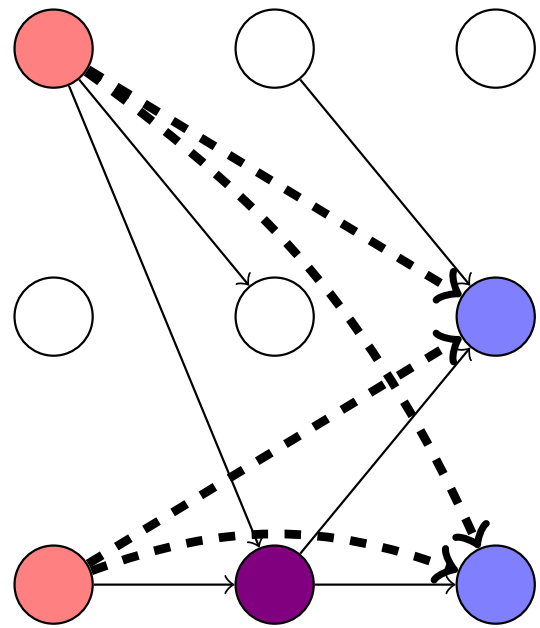
- $\mathcal{O}(n^2 m)$

Relations - Neighbourhood Maps

- Standard technique for relational composition
- Maintain a map from a node to its predecessors/successors



- $\mathcal{O}(n^2 m)$

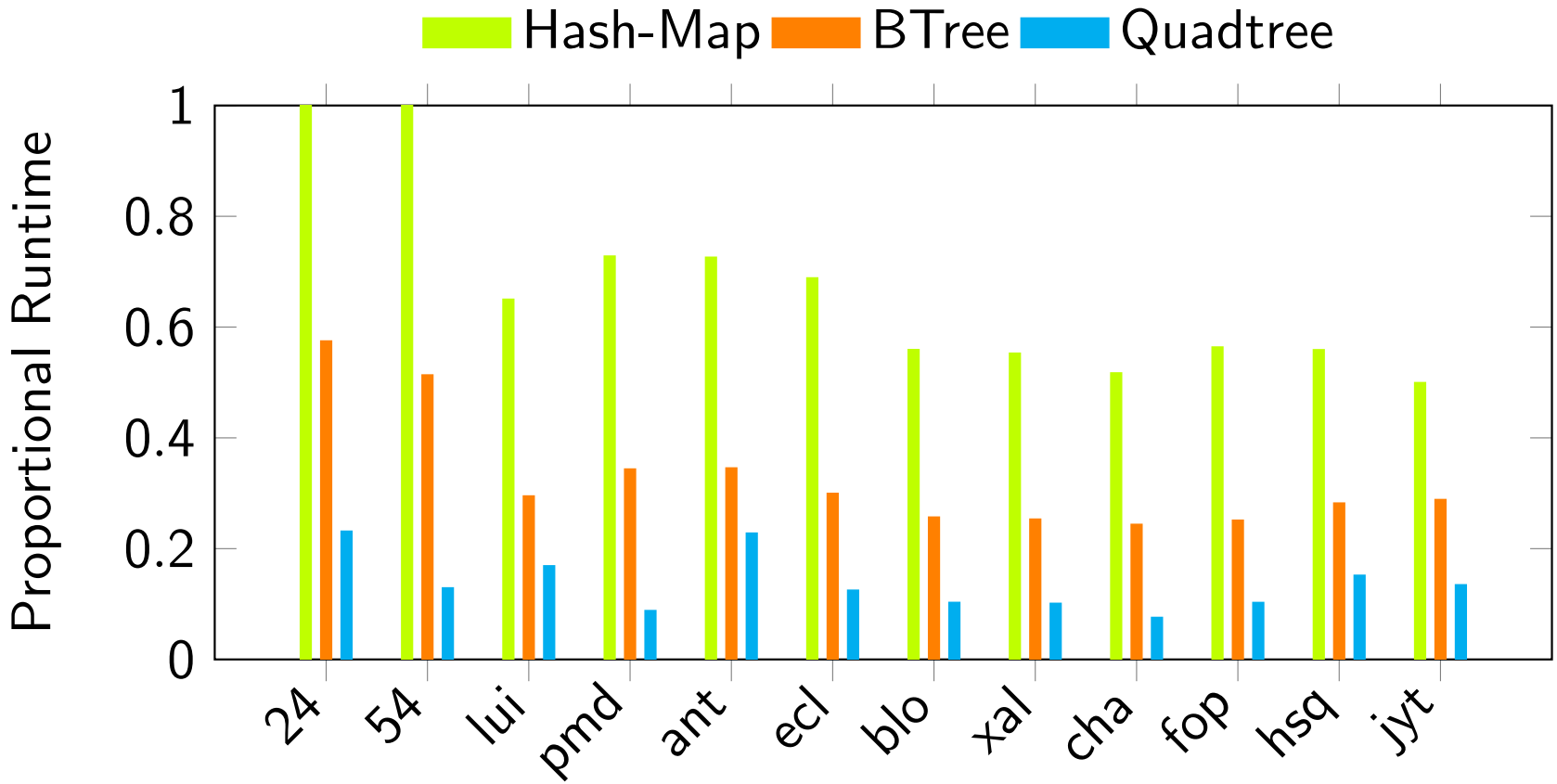


- $\mathcal{O}(m^2 + n)$

Experimental Results

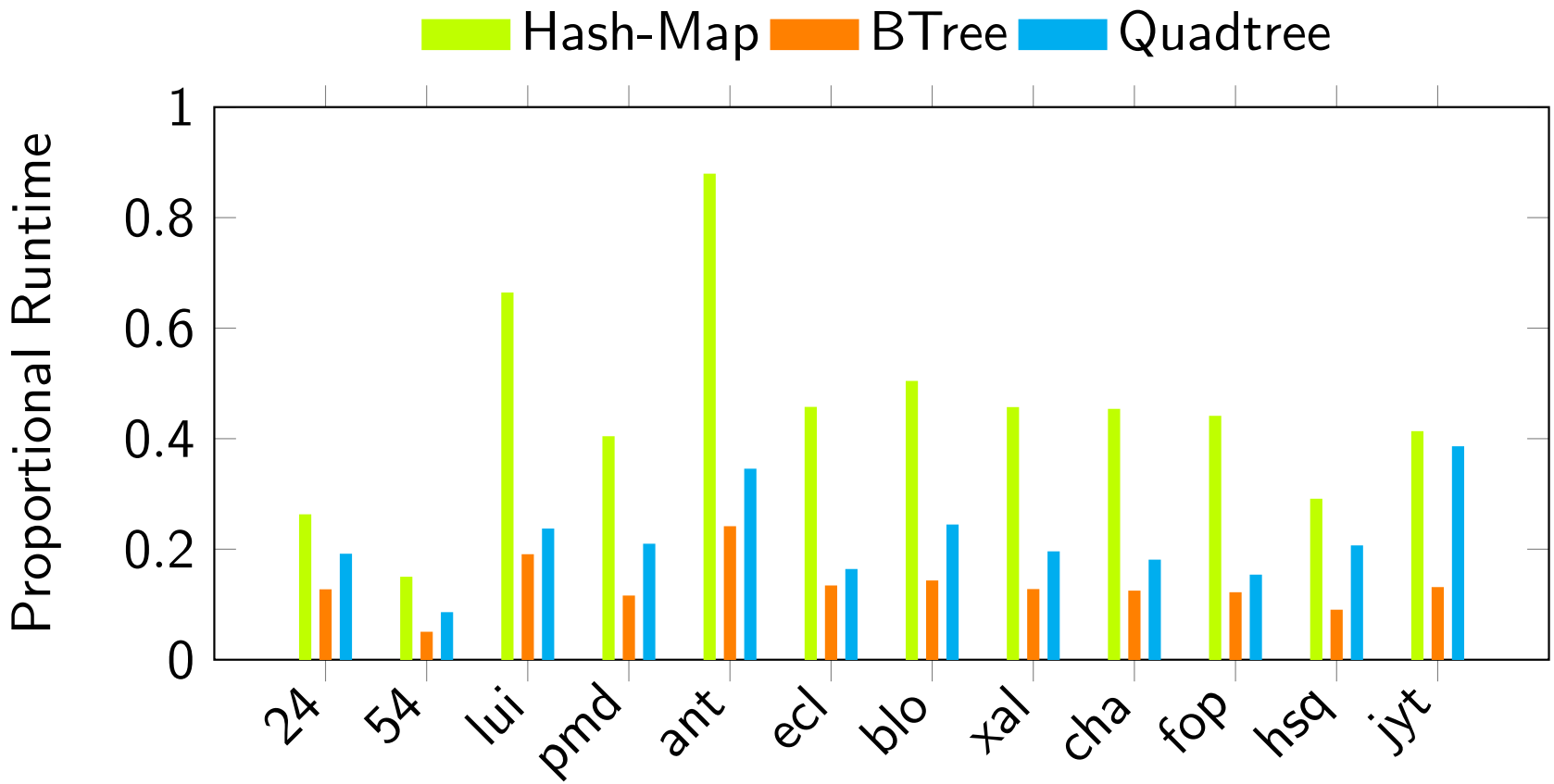
Speedup vs WL

- Significant improvements, especially for larger benchmarks
- Quadtree universally superior



Speedup vs WL

- Improved memory, memory/time tradeoff
- B-Tree implementation is superior



Summary

- CFL-R: Useful computational framework
  - Points-to, security, model checking
- Current algorithms are ineffective
  - Inefficient/redundant computations, $\mathcal{O}(n^3)$
- Relational approach
  - Semi-naïve: cuts down on redundant computations
  - Quadtrees: efficient sparse representation
  - Neighbourhood maps: standard technique
- Experimental evaluation
  - Semi-naïve is good
  - memory/time tradeoff

References I

[0] Chaudhuri, S. (2008).
Subcubic algorithms for recursive state machines.
In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, pages 159–169, New York, NY, USA. ACM.

[0] Dolev, D., Even, S., and Karp, R. (1982).
On the security of ping-pong protocols.
Information and Control, 55(13):57 – 68.

[0] Heintze, N. and McAllester, D. (1997).
On the cubic bottleneck in subtyping and flow analysis.
In *Logic in Computer Science, 1997. LICS'97. Proceedings., 12th Annual IEEE Symposium on*, pages 342–351. IEEE.

References II

[0] Melski, D. and Reps, T. (2000).
Interconvertibility of a class of set constraints and context-free-language reachability.
Theoretical Computer Science, 248(12):29 – 98. PEPM'97.

[0] Reps, T. (1998).
Program analysis via graph reachability.
Information and Software Technology, 40(11-12):701–726.

[0] Yannakakis, M. (1990).
Graph-theoretic methods in database theory.
In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '90, pages 230–242, New York, NY, USA. ACM.