Constraint-Based Static Analyses of Java Bytecode

Đurica Nikolić

1. - Dipartimento di Informatica, University of Verona (Italy)
2. - Microsoft Research - University of Trento Centre for Computational and Systems Biology

MOVEP - December 3rd, 2012

Julia - a static analyzer for Java and Android

www.juliasoft.com:

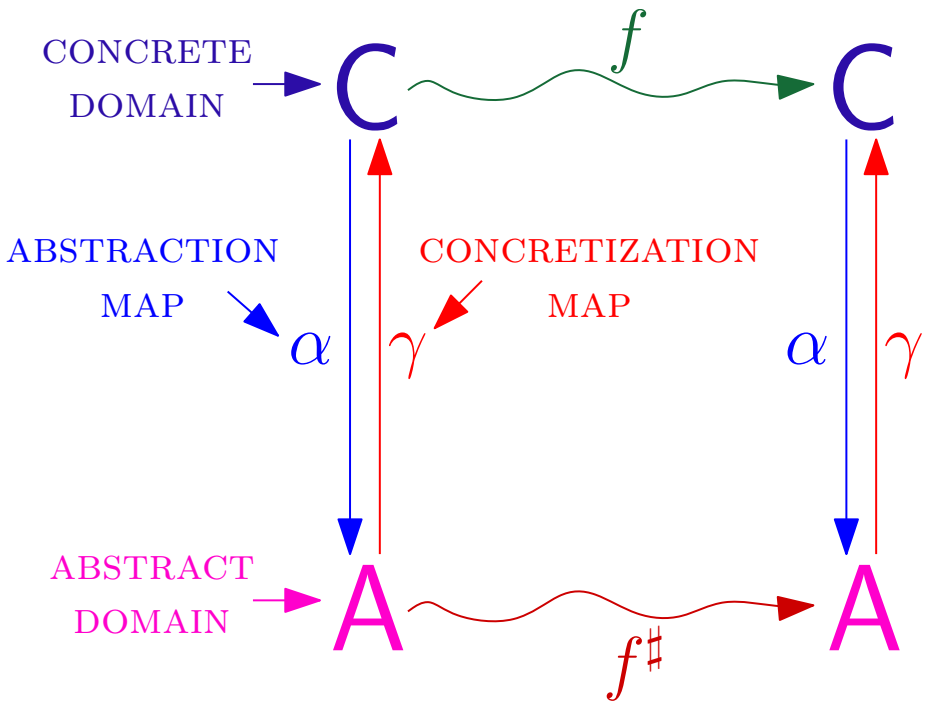
- Julia analyzer finds bugs in Java and Android programs well before they are run.
- It is a semantical tool, based on the general theory of approximations known as abstract interpretation.
- Julia checks all possible executions of a software and find all possible bugs inside the categories considered by the tool (simple checks, nullness, termination).

Julia - a static analyzer for Java and Android

Nullness and Termination analyses use some auxiliary analyses which purpose is to improve the precision of the former analyses.

- Definite Expression Aliasing Analysis
- Definite Aliasing Analysis
- Possible Sharing Analysis
- Possible Side Effects Analysis
- Possible Creation Point Analysis
- Possible Reachability Analysis
- Definite Array Initialization Analysis

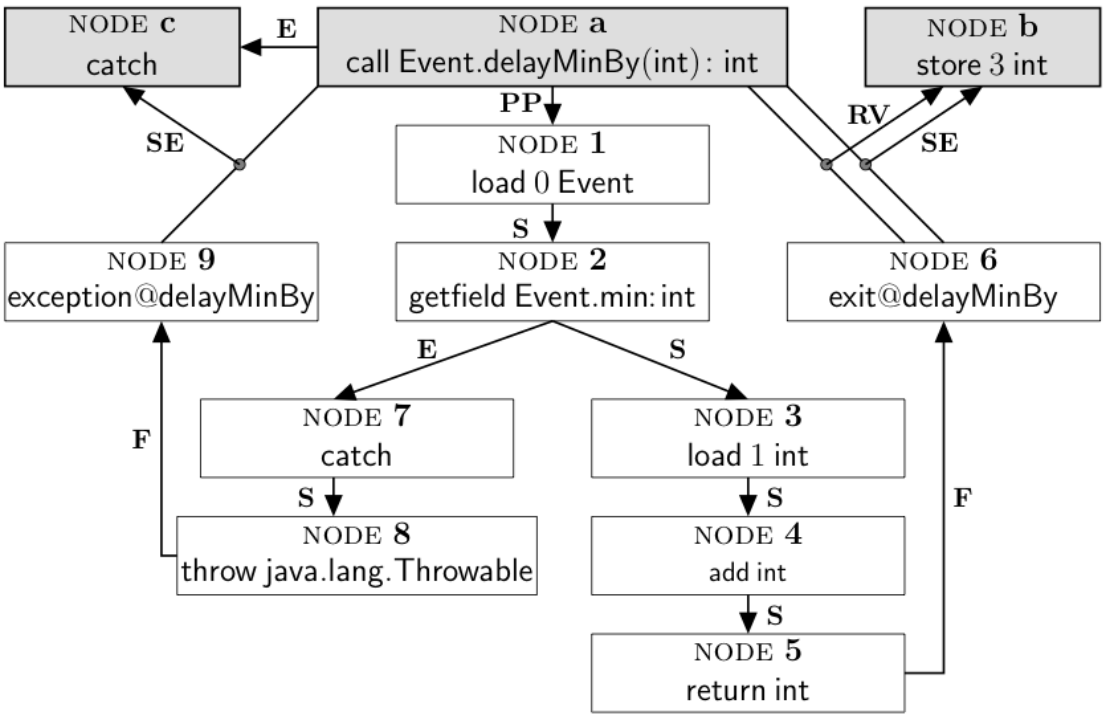
Abstract Interpretation Framework [CousotCousot77]



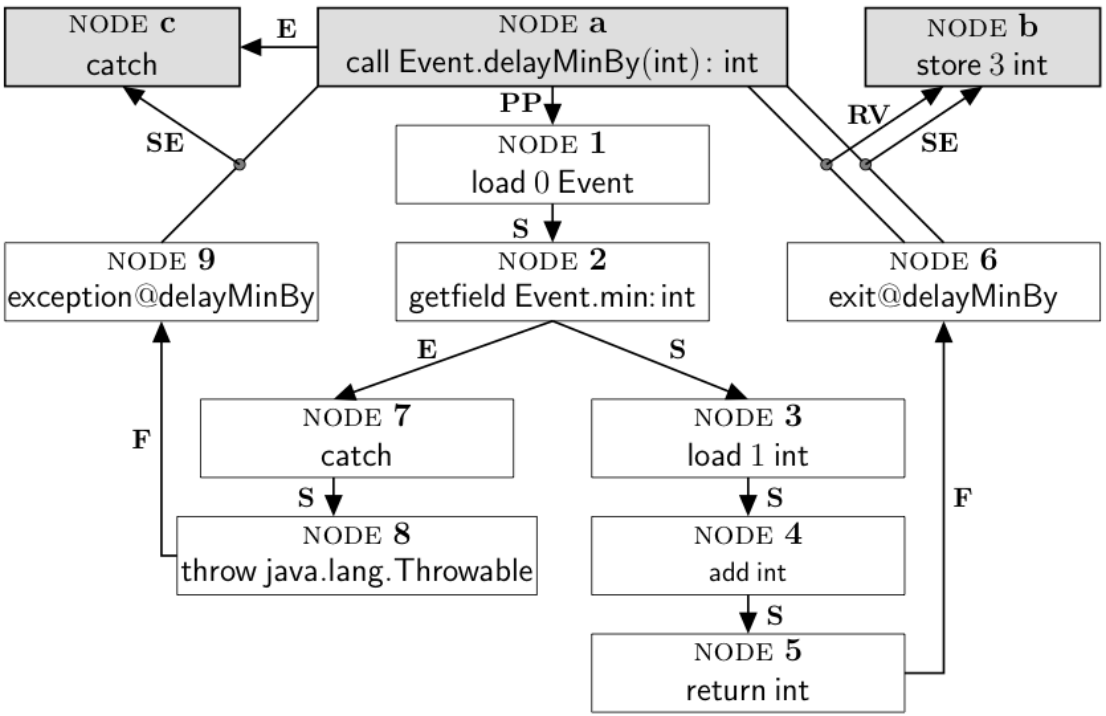
BEST CORRECT APPROXIMATION: $f^{bca} = \alpha \circ f \circ \gamma$

IN PRACTICE: $f^{\sharp}$ IS LESS PRECISE THAN f^{bca} AND INTRODUCES OVER-APPROXIMATION

Abstract Constraint Graph [NikolicSpoto2012]



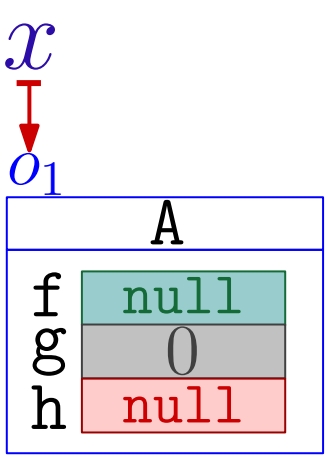
Abstract Constraint Graph [NikolicSpoto2012]

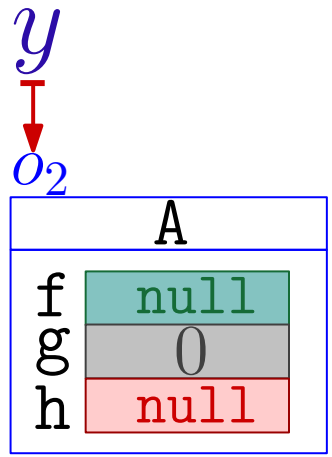


$$
\begin{aligned}
\Pi^{PP}(A_{\mathbf{a}}) &\sqsubseteq A_{\mathbf{1}} \\
\Pi^{S}(A_{\mathbf{1}}) &\sqsubseteq A_{\mathbf{2}} \\
\Pi^{S}(A_{\mathbf{2}}) &\sqsubseteq A_{\mathbf{3}} \\
\Pi^{S}(A_{\mathbf{3}}) &\sqsubseteq A_{\mathbf{4}} \\
\Pi^{S}(A_{\mathbf{4}}) &\sqsubseteq A_{\mathbf{5}} \\
\Pi^{F}(A_{\mathbf{5}}) &\sqsubseteq A_{\mathbf{6}} \\
\Pi^{E}(A_{\mathbf{2}}) &\sqsubseteq A_{\mathbf{7}} \\
\Pi^{S}(A_{\mathbf{7}}) &\sqsubseteq A_{\mathbf{8}} \\
\Pi^{F}(A_{\mathbf{8}}) &\sqsubseteq A_{\mathbf{9}} \\
\Pi^{RV}(A_{\mathbf{a}}, A_{\mathbf{6}}) \sqcup \Pi^{SE}(A_{\mathbf{a}}, A_{\mathbf{6}}) &\sqsubseteq A_{\mathbf{b}} \\
\Pi^{E}(A_{\mathbf{6}}) \sqcup \Pi^{SE}(A_{\mathbf{a}}, A_{\mathbf{6}}) &\sqsubseteq A_{\mathbf{c}}
\end{aligned}
$$

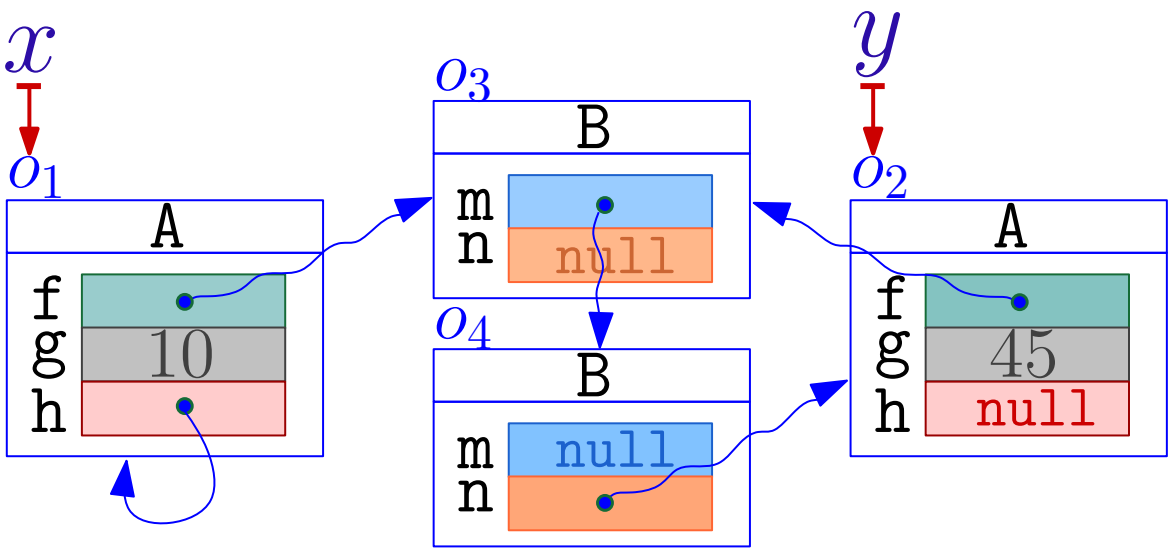
Reachability Analysis of Program Variables

Intuitive definition of Reachability



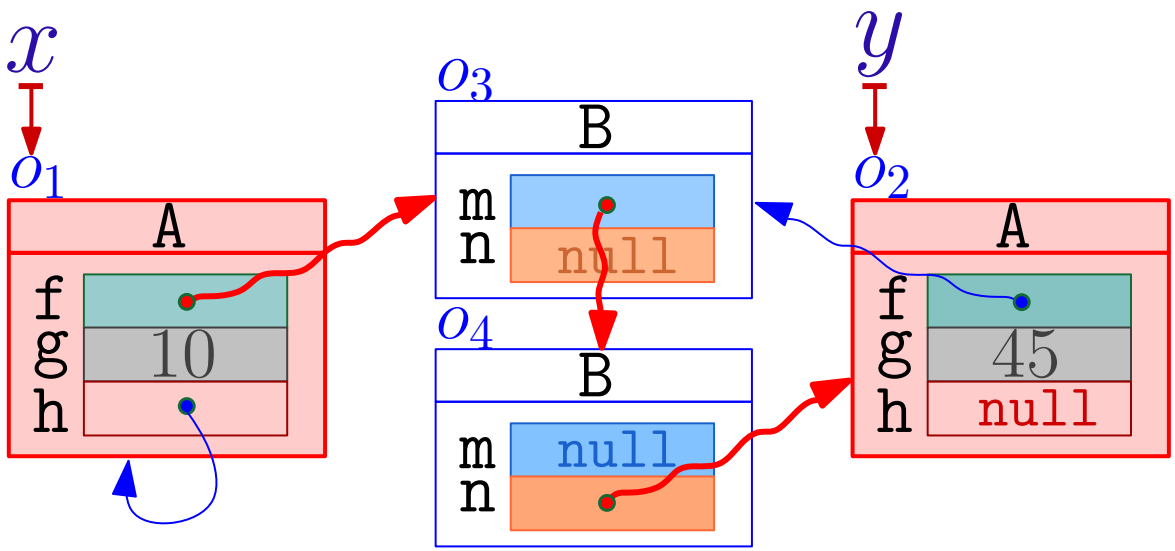


Intuitive definition of Reachability



Is there a sequence of fields $f_1, \ldots, f_k$ such that $x.f_1.\ldots.f_k = y$?

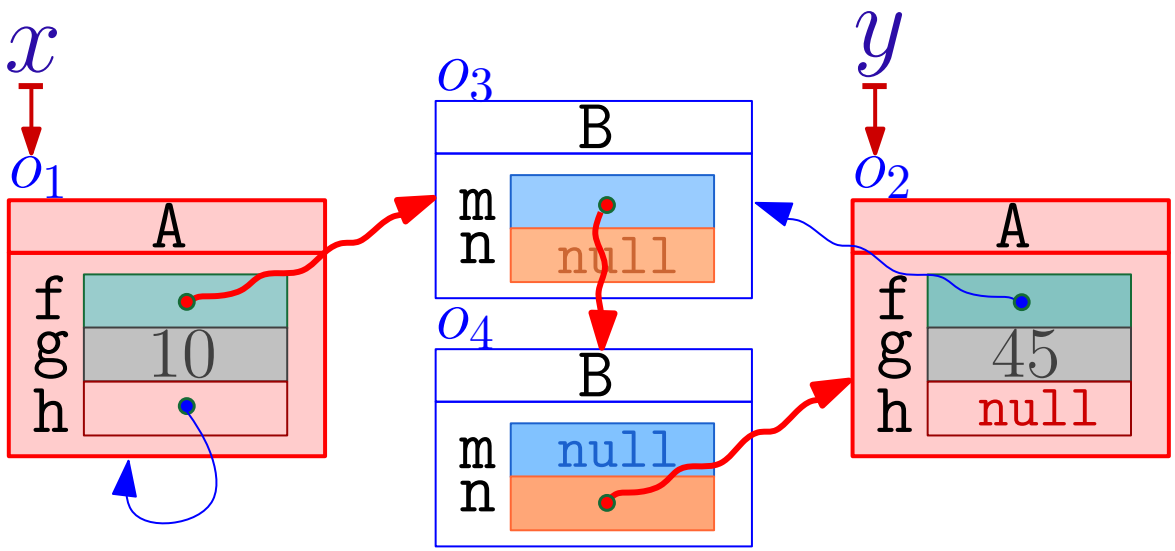
Intuitive definition of Reachability



Is there a sequence of fields $f_1, \ldots, f_k$ such that $x.f_1.\ldots.f_k = y$?

$$x.f.m.n = y$$

Intuitive definition of Reachability



Is there a sequence of fields $f_1, \ldots, f_k$ such that $x.f_1.\ldots.f_k = y$?

$x.f.m.n = y \quad \Rightarrow \quad x$ reaches y

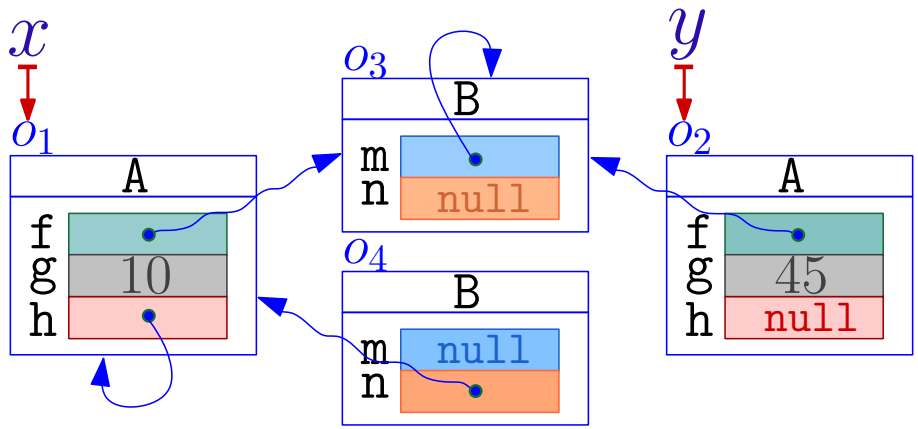
Haven't we solved this problem yet?

There is a lot of pointer analyses: [Hind01] surveys more than 75 papers

Haven't we solved this problem yet?

There is a lot of pointer analyses: [Hind01] surveys more than 75 papers

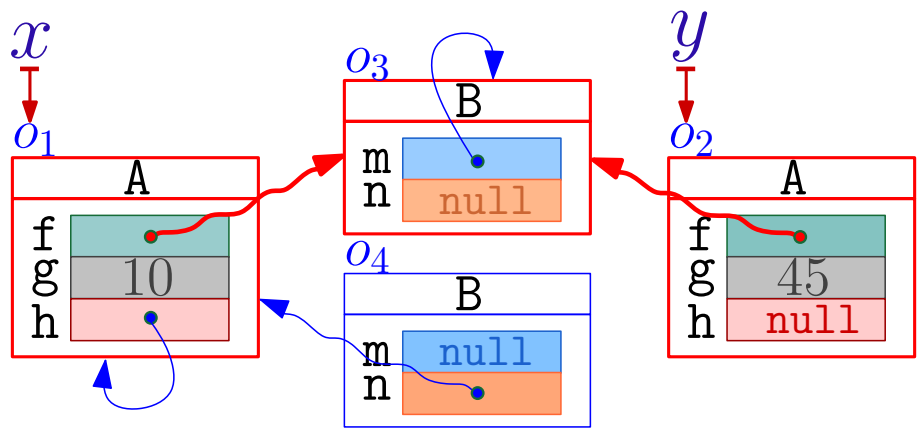
- Sharing Analysis



Haven't we solved this problem yet?

There is a lot of pointer analyses: [Hind01] surveys more than 75 papers

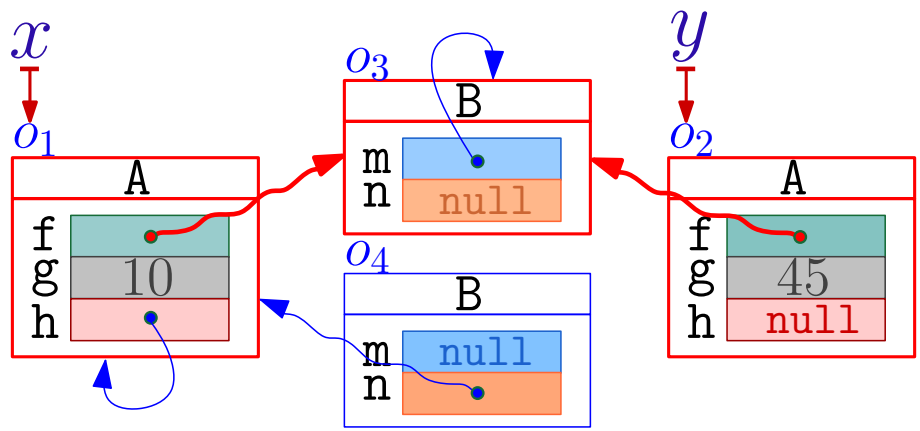
- Sharing Analysis



Haven't we solved this problem yet?

There is a lot of pointer analyses: [Hind01] surveys more than 75 papers

- Sharing Analysis

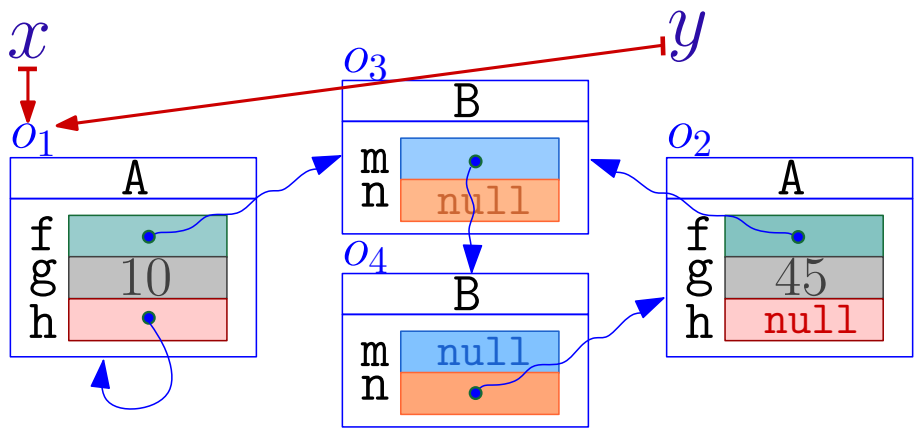


- REACHABILITY entails SHARING
- SHARING ~~entails~~ REACHABILITY

Haven't we solved this problem yet?

There is a lot of pointer analyses: [Hind01] surveys more than 75 papers

- Sharing Analysis
- Aliasing Analysis



- ALIASING ENTAILS REACHABILITY
- REACHABILITY ~~ENTAILS~~ ALIASING

Haven't we solved this problem yet?

There is a lot of pointer analyses: [Hind01] surveys more than 75 papers

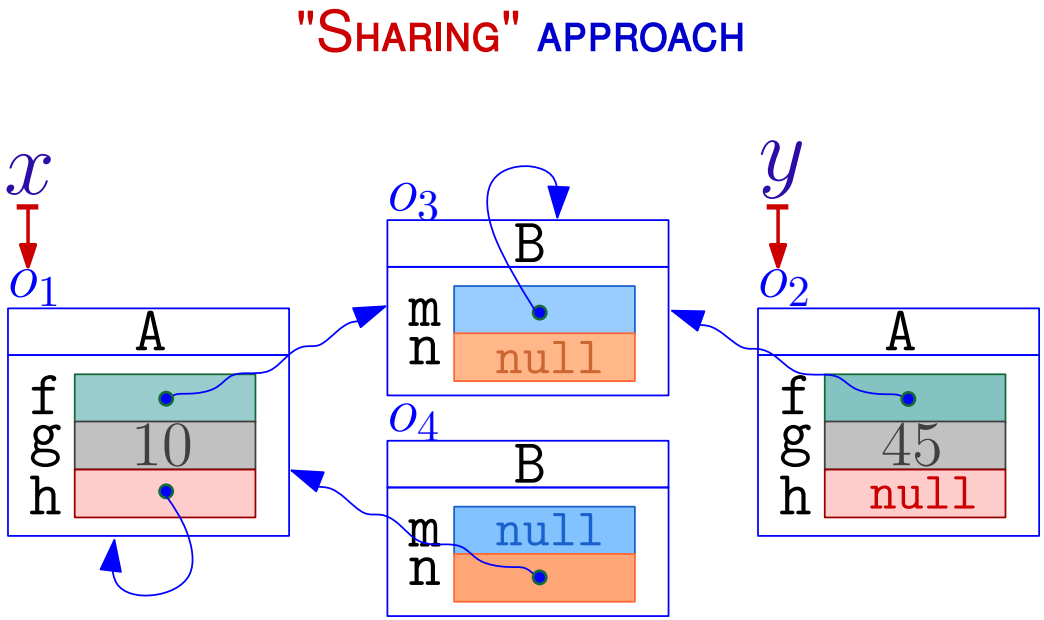
- Sharing Analysis
- Aliasing Analysis
- Shape Analysis

Where can it be useful?

Cyclicity Analysis: An assignment $y.\mathrm{h} = x$ might make y cyclical?

Where can it be useful?

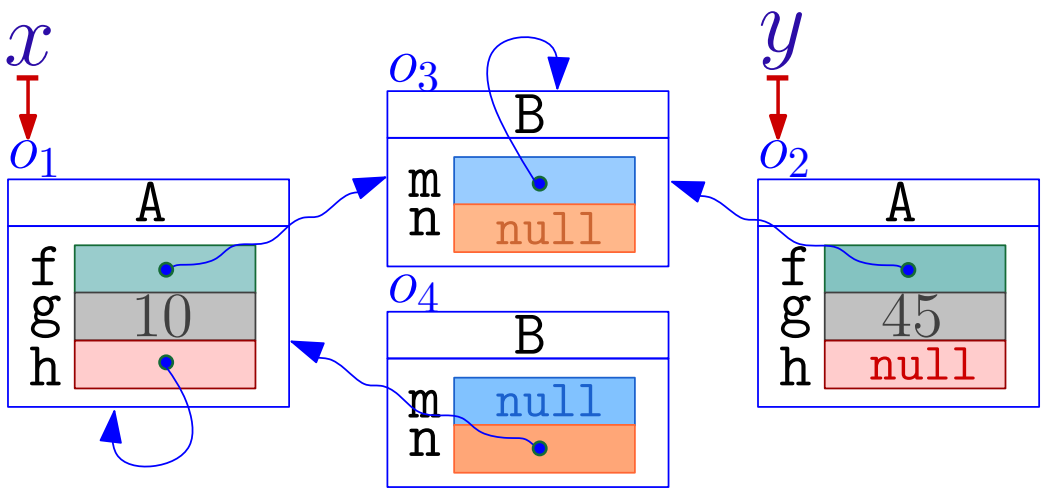
Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?



Where can it be useful?

Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

"Sharing" approach

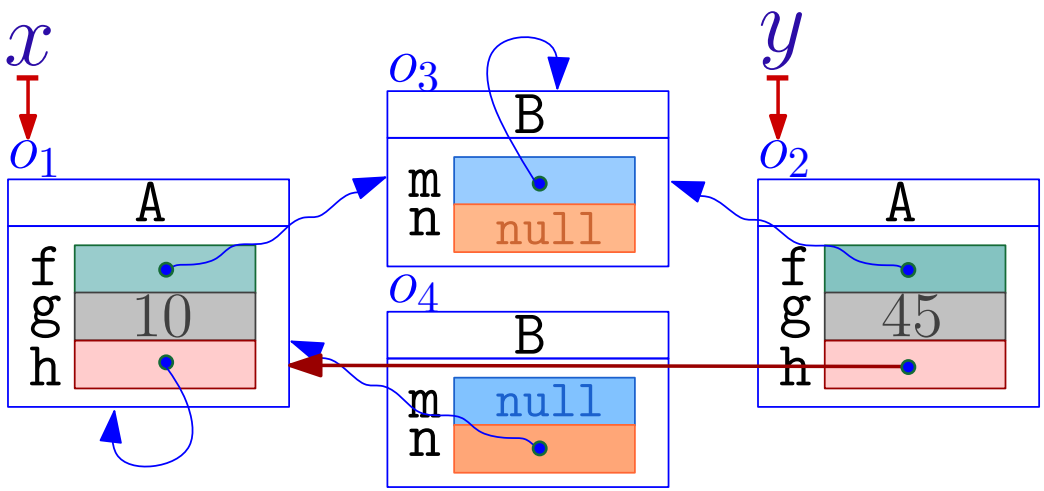


$y.h = x$ makes y cyclical?

Where can it be useful?

Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

"Sharing" approach

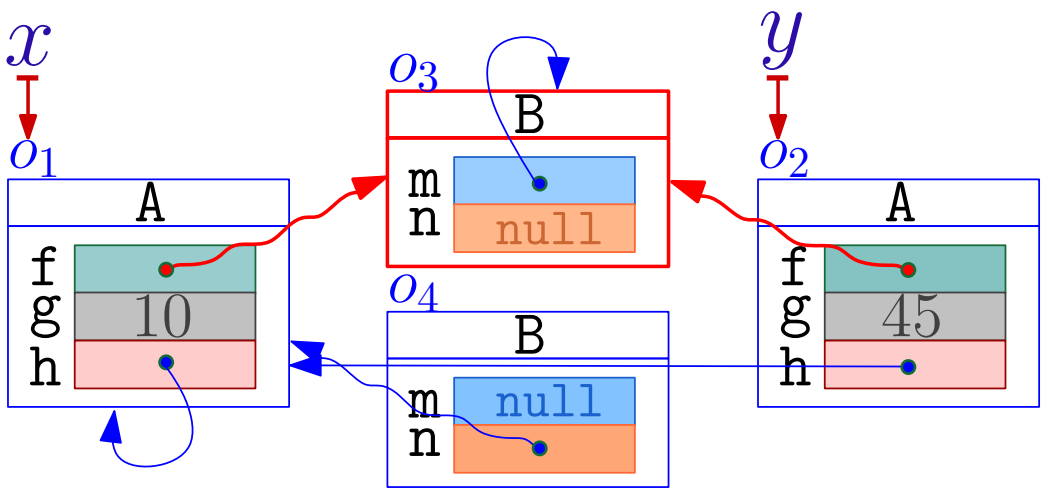


$y.h = x$ makes y cyclical?

Where can it be useful?

Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

"Sharing" approach

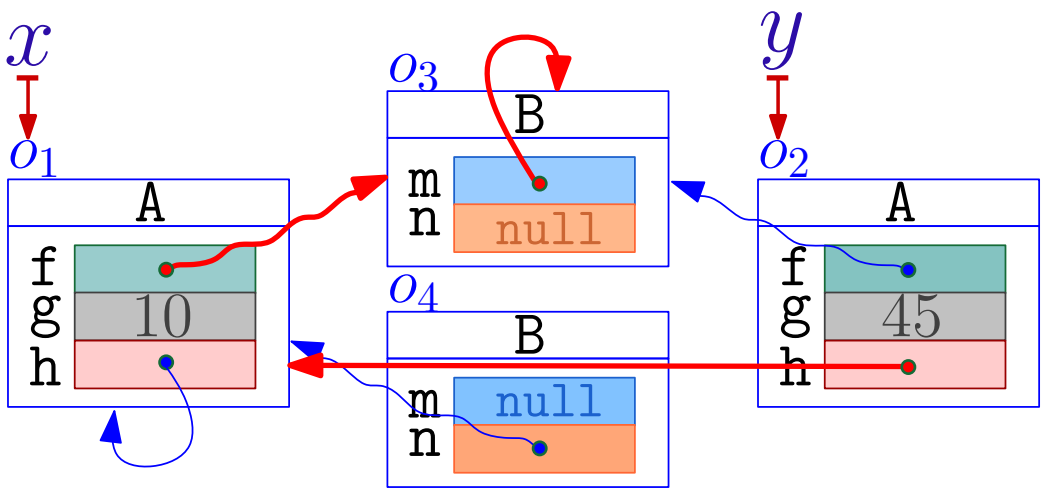


$y.h = x$ makes y cyclical?
if x shares with y?

Where can it be useful?

Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

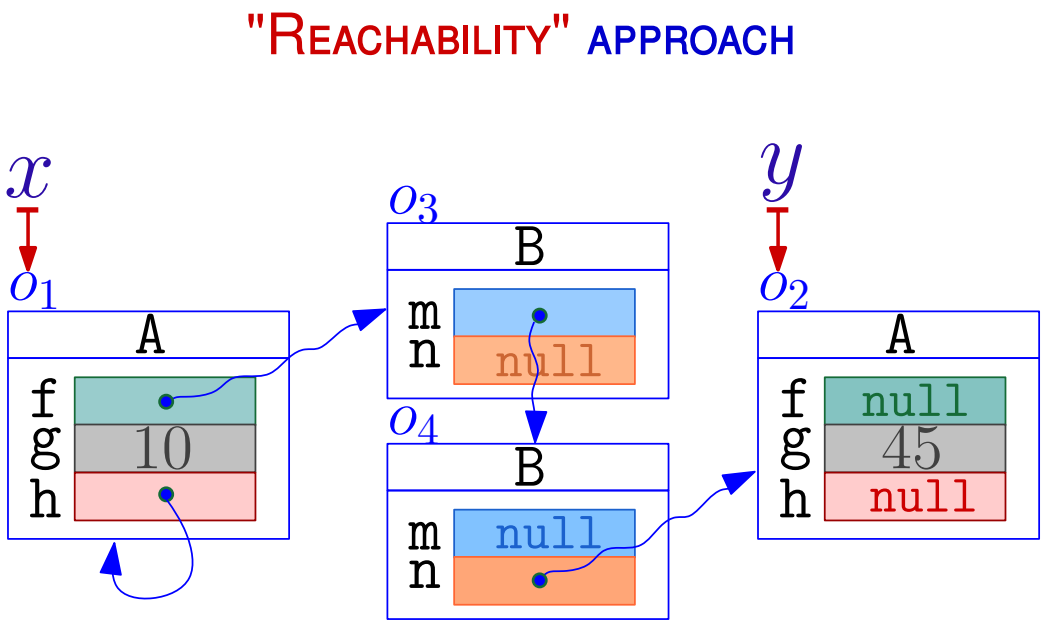
"Sharing" approach



$y.h = x$ makes y cyclical? No!

Where can it be useful?

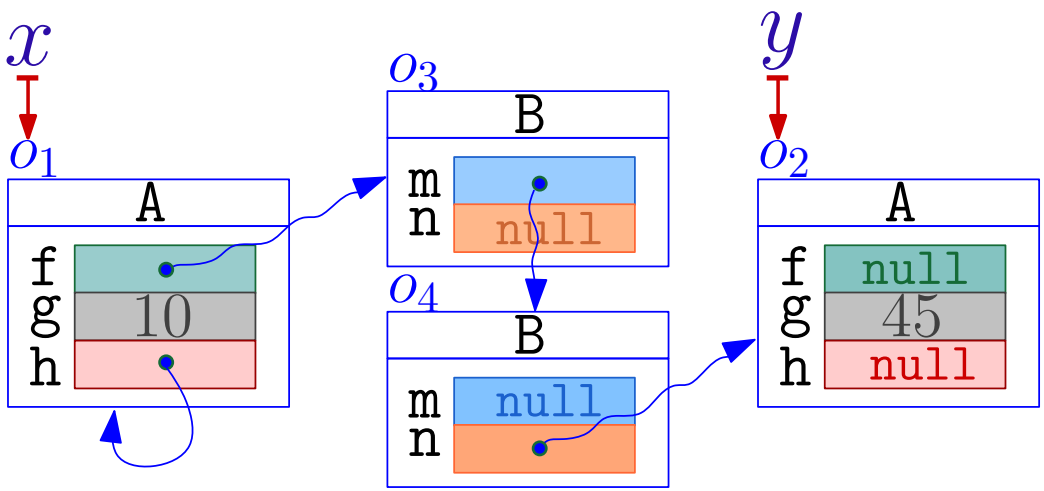
Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?



Where can it be useful?

Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

"Reachability" approach

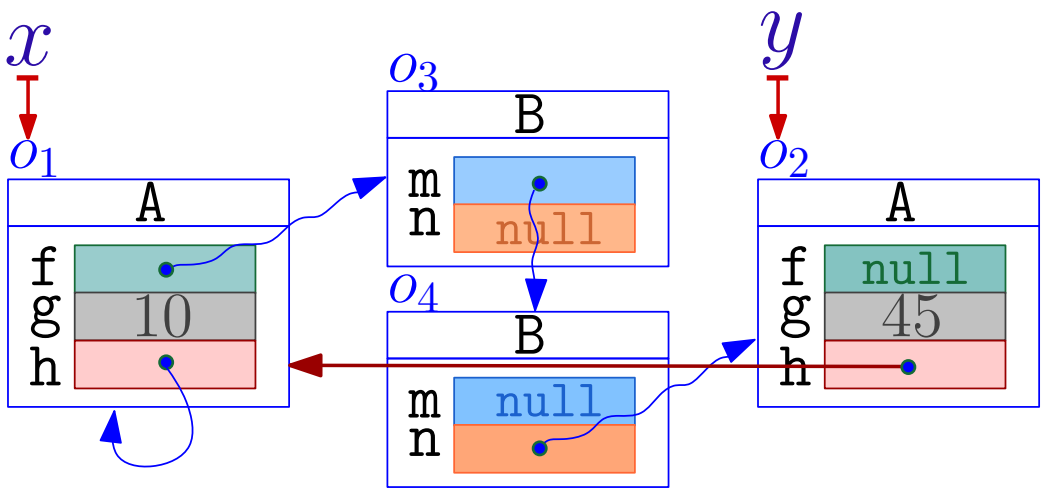


$y.h = x$ makes y cyclical?

Where can it be useful?

Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

"Reachability" approach



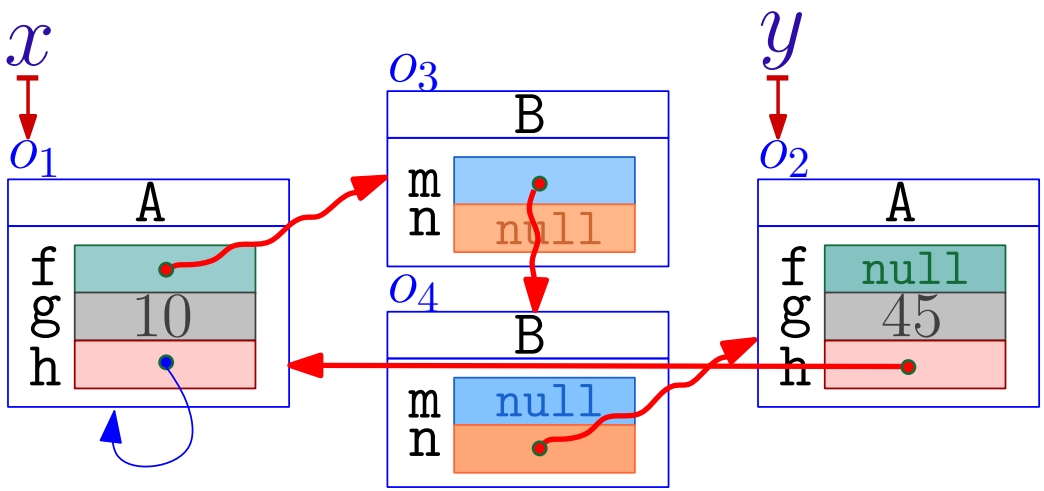
$y.h = x$ makes y cyclical?

if x reaches y

Where can it be useful?

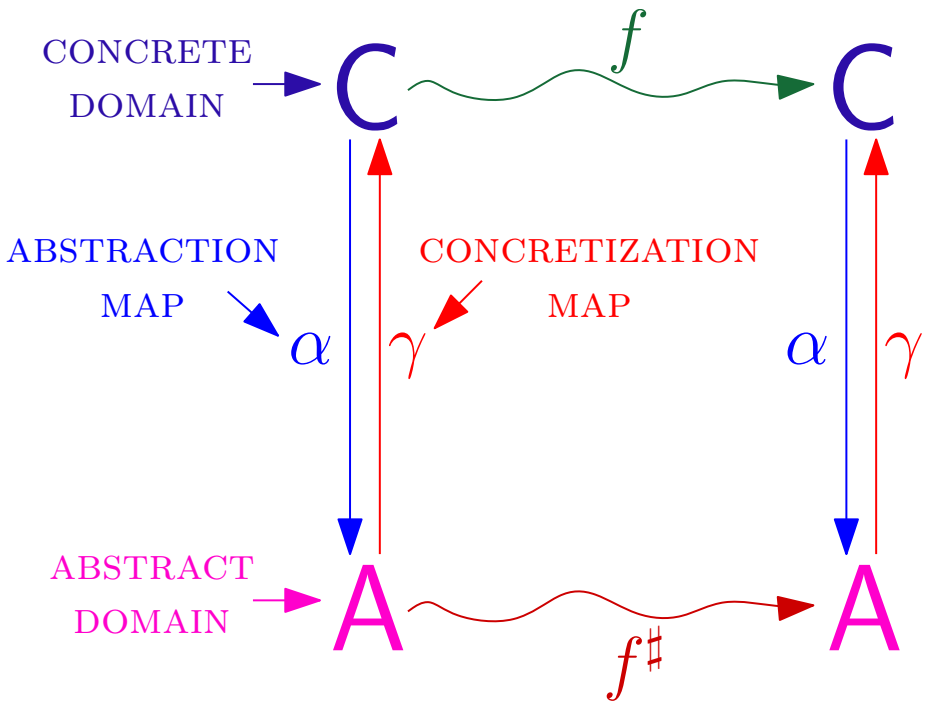
Cyclicity Analysis: An assignment $y.h = x$ might make y cyclical?

"Reachability" approach



$y.h = x$ makes y cyclical? Yes!

Abstract Interpretation Framework [CousotCousot77]



BEST CORRECT APPROXIMATION: $f^{bca} = \alpha \circ f \circ \gamma$

IN PRACTICE: $f^{\sharp}$ IS LESS PRECISE THAN f^{bca} AND INTRODUCES OVER-APPROXIMATION

Concrete and Abstract Domains

- Σ - set of all states
- V - set of all variables
- Concrete Domain: $C = \langle \wp(\Sigma), \subseteq \rangle$
- Abstract Domain: $A = \langle \wp(V \times V), \subseteq \rangle$
  - an abstract element $R \in A$ represents those concrete states whose reachability information is over-approximated by the pairs of variables in R
  - we write $a \rightsquigarrow b$ to denote $\langle a, b \rangle$
- Concretization Map:

$$\gamma(R) = \{\sigma \in \Sigma \mid \forall a, b \in V. a \rightsquigarrow^{\sigma} b \Rightarrow a \rightsquigarrow b \in R\}$$

Experimental Evaluation

Reachability Analysis	Side-effects Analysis	Field Initializat. Analysis
45.07%	−23.47%	+3.46%

	Nullness Analysis	Termination Analysis
runtime	−7.77%	−1.62%
warnings	−3.38%	0%

Conclusion

- We have defined a framework for the static analyses of Java bytecode, based on constraint generation and solving, and for the proof of correctness of these analyses;
- The framework is able to deal with the exceptional flows and the side-effects of non-pure methods;
- We have formalized, proven correct and implemented inside the Julia analyzer different static analyses for Java bytecode concerning heap related properties: reachability analysis of program variables [IJCAR2012], definite aliasing expression analysis [ICTAC2012], etc;
- This technique is not limited to Java bytecode but can be used for other low-level as well as for high-level languages.

Thank You!!!