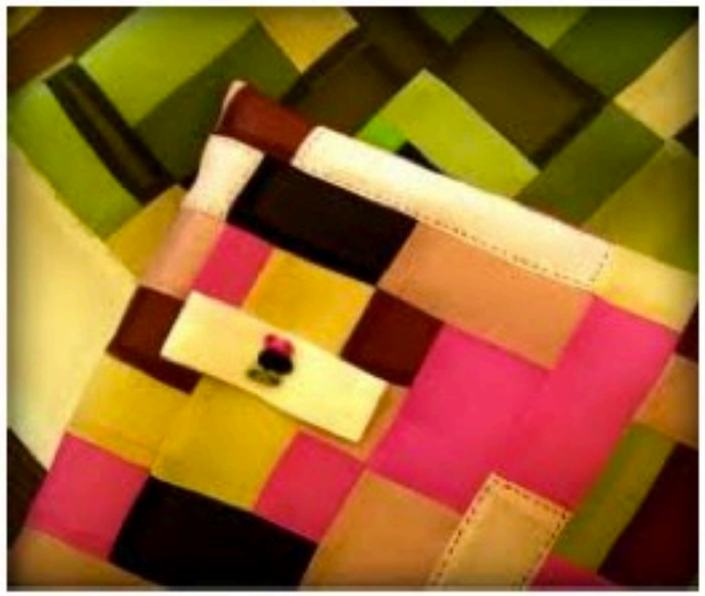
Course Outline

Collage of Static Analysis



- 0.5hr: Static Analysis Overview
- 1.5hr: Static Analysis Design Framework
- 1.0hr: Static Analysis Engineering Framework
- 1.0hr: Static Analysis of Multi-Staged Programs

Need of Static Analysis Design Theory

- How can we derive correct equations from program text?
  - Does the equations capture all the execution dynamics?
  - Non-obvious: pointers, heap structures, exceptions, high-order functions, typeless low-level hacks, etc.

```
x = readInt;
while (x ≤ 99)
        x++;
end
```

$\overset{\text{how?}}{\Longrightarrow}$

$$
\begin{array}{lcl}
x_1 & = & [-\infty, +\infty] \; or \; x_3 \\
x_2 & = & x_1 \; and \; [-\infty, 99] \\
x_3 & = & x_2 \dot{+} 1 \\
x_4 & = & x_1 \; and \; [100, +\infty]
\end{array}
$$

- Is there always a solution to the derived equations?
  - How do we compute the solution in a finite time?

$$
\begin{array}{lcl}
x_1 & = & [-\infty, +\infty] \; or \; x_3 \\
x_2 & = & x_1 \; and \; [-\infty, 99] \\
x_3 & = & x_2 \dot{+} 1 \\
x_4 & = & x_1 \; and \; [100, \infty]
\end{array}
\quad \overset{\text{how?}}{\Longrightarrow} \quad
\begin{array}{lcl}
x_1 & = & [-\infty, +\infty] \\
x_2 & = & [-\infty, 99] \\
x_3 & = & [-\infty, 100] \\
x_4 & = & [100, +\infty]
\end{array}
$$

Need of Static Analysis Design Theory

- How can we derive correct equations from program text?
  - Does the equations capture all the execution dynamics?
  - Non-obvious: pointers, heap structures, exceptions, high-order functions, typeless low-level hacks, etc.

```
x = readInt;
while (x ≤ 99)
    x++;
end
```

how? ⟹

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \ or \ x_3 \\ x_2 & = & x_1 \ and \ [-\infty, 99] \\ x_3 & = & x_2 \dot{+} 1 \\ x_4 & = & x_1 \ and \ [100, \infty] \end{array}$$

Define a Semantics Function

- Is there always a solution to the derived equations?
  - How do we compute the solution in a finite time?

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \ or \ x_3 \\ x_2 & = & x_1 \ and \ [-\infty, 99] \\ x_3 & = & x_2 \dot{+} 1 \\ x_4 & = & x_1 \ and \ [100, \infty] \end{array}$$

how? ⟹

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \\ x_2 & = & [-\infty, 99] \\ x_3 & = & [-\infty, 100] \\ x_4 & = & [100, +\infty] \end{array}$$

Need of Static Analysis Design Theory

- How can we derive correct equations from program text?
  - Does the equations capture all the execution dynamics?
  - Non-obvious: pointers, heap structures, exceptions, high-order functions, typeless low-level hacks, etc.

```
x = readInt;
while (x ≤ 99)
    x++;
end
```

how? $\Longrightarrow$

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \ or \ x_3 \\ x_2 & = & x_1 \ and \ [-\infty, 99] \\ x_3 & = & x_2 \dot{+} 1 \\ x_4 & = & x_1 \ and \ [100, \infty] \end{array}$$

Define a Semantics Function $\hat{F}$

- Is there always a solution to the derived equations?
  - How do we compute the solution in a finite time?

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \ or \ x_3 \\ x_2 & = & x_1 \ and \ [-\infty, 99] \\ x_3 & = & x_2 \dot{+} 1 \\ x_4 & = & x_1 \ and \ [100, \infty] \end{array}$$

how? $\Longrightarrow$

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \\ x_2 & = & [-\infty, 99] \\ x_3 & = & [-\infty, 100] \\ x_4 & = & [100, +\infty] \end{array}$$

Need of Static Analysis Design Theory

- How can we derive correct equations from program text?
  - Does the equations capture all the execution dynamics?
  - Non-obvious: pointers, heap structures, exceptions, high-order functions, typeless low-level hacks, etc.

```
x = readInt;
while (x ≤ 99)
    x++;
end
```

how? $\Longrightarrow$

$$
\begin{array}{lcl}
x_1 & = & [-\infty, +\infty] \ or \ x_3 \\
x_2 & = & x_1 \ and \ [-\infty, 99] \\
x_3 & = & x_2 + 1 \\
x_4 & = & x_1 \ and \ [100, \infty]
\end{array}
$$

Define a Semantics Function $\hat{F}$

- Is there always a solution to the derived equations?
  - How do we compute the solution in a finite time?

$$
\begin{array}{lcl}
x_1 & = & [-\infty, +\infty] \ or \ x_3 \\
x_2 & = & x_1 \ and \ [-\infty, 99] \\
x_3 & = & x_2 + 1 \\
x_4 & = & x_1 \ and \ [100, \infty]
\end{array}
$$

how? $\Longrightarrow$

$$
\begin{array}{lcl}
x_1 & = & [-\infty, +\infty] \\
x_2 & = & [-\infty, 99] \\
x_3 & = & [-\infty, 100] \\
x_4 & = & [100, +\infty]
\end{array}
$$

Compute an UB of

Need of Static Analysis Design Theory

- How can we derive correct equations from program text?
  - Does the equations capture all the execution dynamics?
  - Non-obvious: pointers, heap structures, exceptions, high-order functions, typeless low-level hacks, etc.

```
x = readInt;
while (x ≤ 99)
    x++;
end
```

how? $\Longrightarrow$

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \ or \ x_3 \\ x_2 & = & x_1 \ and \ [-\infty, 99] \\ x_3 & = & x_2 \dot{+} 1 \\ x_4 & = & x_1 \ and \ [100, \infty] \end{array}$$

Define a Semantics Function $\hat{F}$

- Is there always a solution to the derived equations?
  - How do we compute the solution in a finite time?

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \ or \ x_3 \\ x_2 & = & x_1 \ and \ [-\infty, 99] \\ x_3 & = & x_2 \dot{+} 1 \\ x_4 & = & x_1 \ and \ [100, \infty] \end{array}$$

how? $\Longrightarrow$

$$\begin{array}{lcl} x_1 & = & [-\infty, +\infty] \end{array}$$

Compute an UB of $fix\,\hat{F} = \bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\hat{\bot})$

Trace Abstract Interpretation Framework

Semantics

$$fix(\lambda T. T_0 \cup Next\ T) \in 2^{State}$$

$$Next = \wp next$$
$$next \in State \to State$$

Abstract Semantics

$$fix(\lambda \hat{T}. \alpha(T_0) \sqcup \hat{Next}\ \hat{T}) \in \Delta \to \hat{State}$$

$$\hat{Next} = (\wp\sqcup) \circ \hat{\pi} \circ (\wp_\cup \hat{next})$$
$$\hat{next} \in \hat{State} \to 2^{\hat{State}}$$

$$2^{State} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\Delta \to \hat{State})$$

Static Analysis Engineering: global analysis of million lines of C

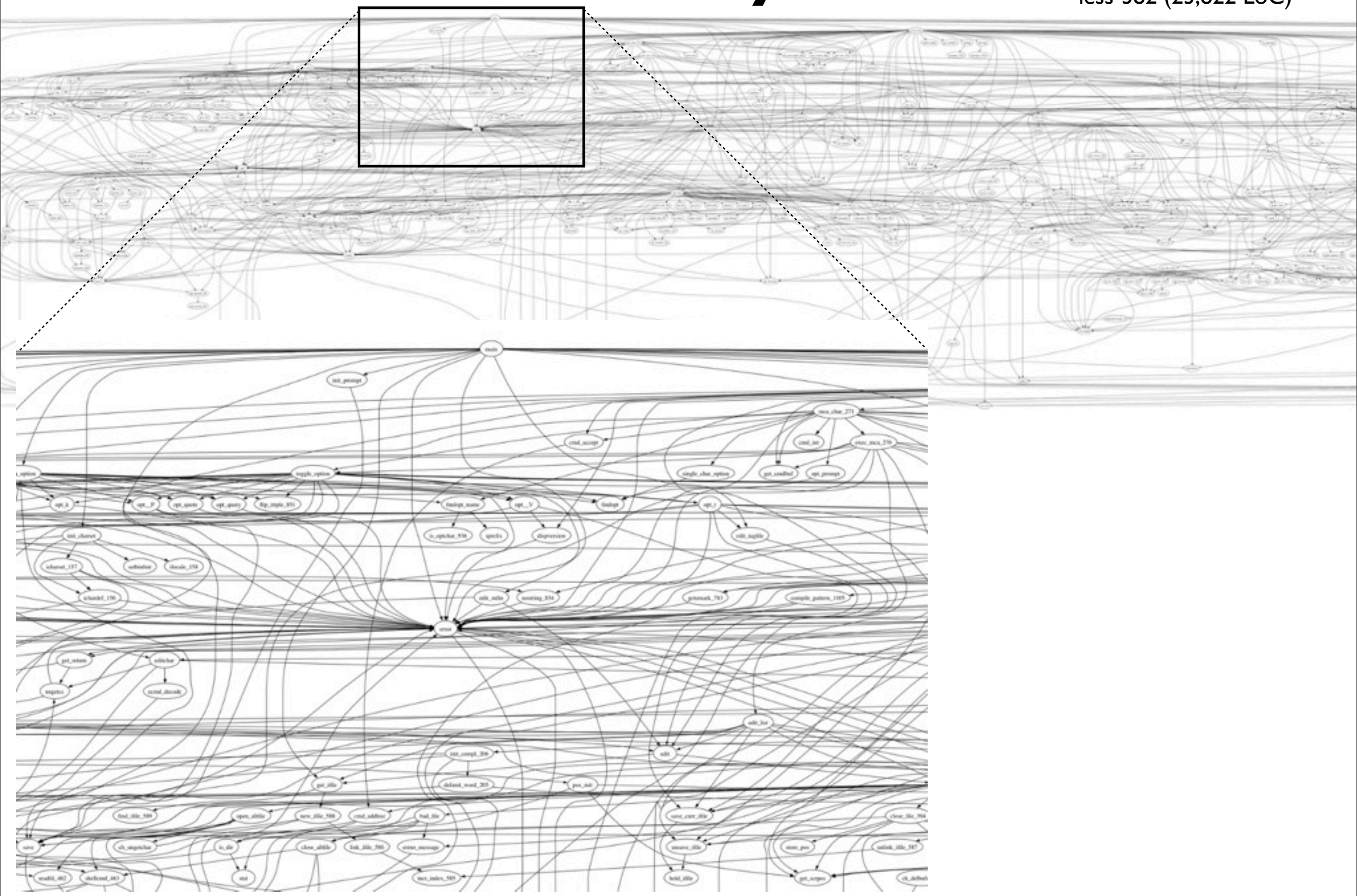
Kwangkeun Yi
Seoul National University, Korea
http://ropas.snu.ac.kr/~kwang

2/26/2012 - 3/2/2012
17th Estonian Winter School in Computer Science,
Palmse, Estonia

(Co-work with Hakjoo Oh, Kihong Heo, Wonchan Lee, Woosuk Lee)

Reality

less-382 (23,822 LoC)



Static Analysis in Reality (3/3)

Sound, "semantically-deep", global analyzer for 1MLoC C program: a scalablility barrier knocked down (as of 2011)

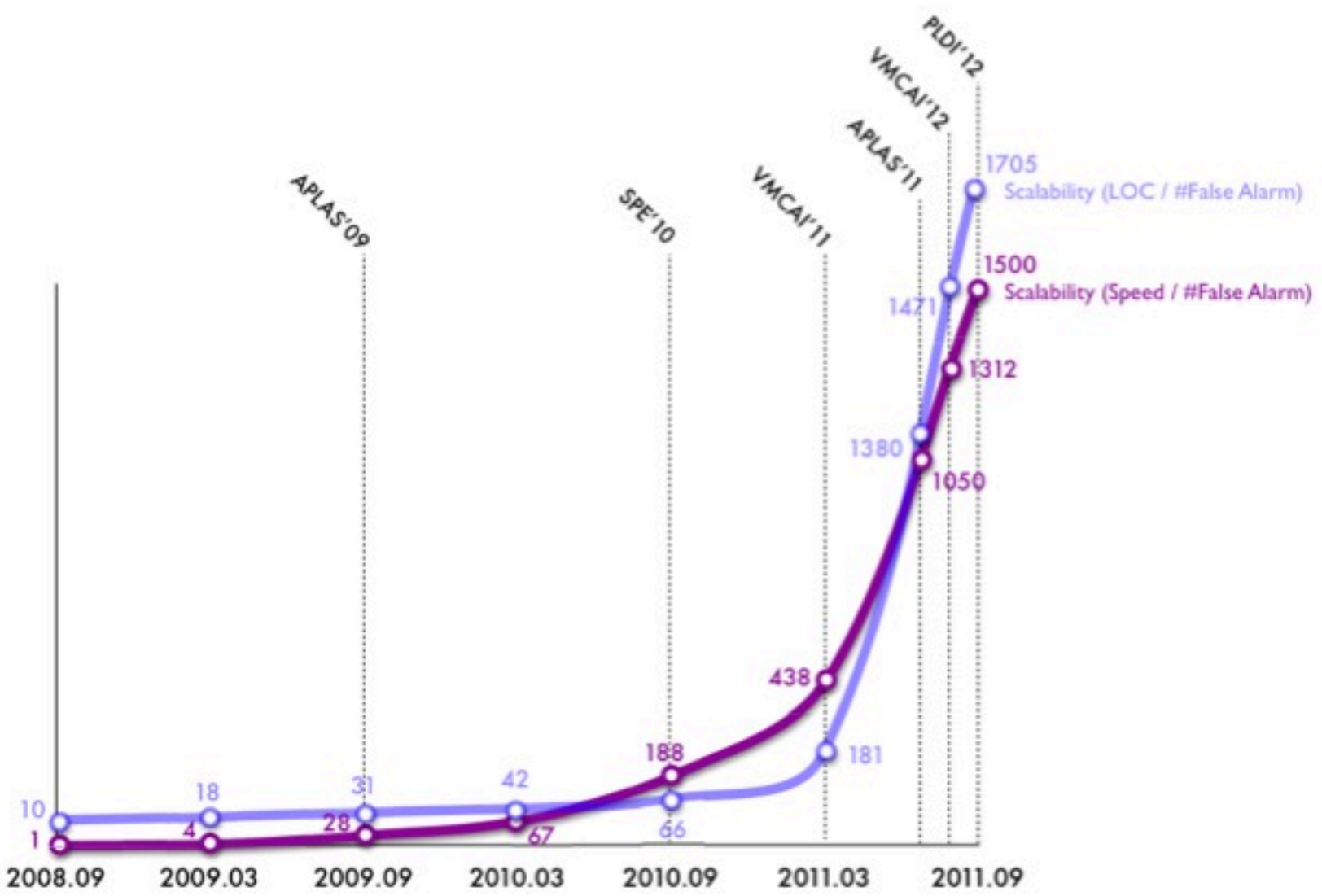
Sound-&-global analyzer class:

Sparrow
The Early Bird

Program	LOC	Baseline		Localize		Spd↑	Mem↓
		Time	Mem	Time	Mem		
gzip-1.2.4a	7 K	772	240	3	63	257 x	74 %
bc-1.06	13 K	1,270	276	7	75	181 x	73 %
less-382	23 K	9,561	1,113	33	127	289 x	86 %
make-3.76.1	27 K	24,240	1,391	21	114	1,154 x	92 %
wget-1.9	35 K	44,092	2,546	11	85	4,008 x	97 %
a2ps-4.14	64 K	∞	N/A	40	353	N/A	N/A
sendmail-8.13.6	130 K	∞	N/A	744	678	N/A	N/A
nethack-3.3.0	211 K	∞	N/A	16,373	5,298	N/A	N/A
emacs-22.1	399 K	∞	N/A	37,830	7,795	N/A	N/A
python-2.5.1	435 K	∞	N/A	11,039	5,535	N/A	N/A
linux-3.0	710 K	∞	N/A	33,618	20,529	N/A	N/A
gimp-2.6	959 K	∞	N/A	3,874	3,602	N/A	N/A
ghostscript-9.00	1,363 K	∞	N/A	14,814	6,384	N/A	N/A

Static Analysis Scalability Improvement

Sound-&-global analyzer class



Contents

- Designing Sparrow
  - in the abstract interpretation framework
- Engineering Sparrow
  - by localizations in space and time
  - [PLDI'12, VMCAI'11, APLAS'11,'09]

Design of Sparrow

The Early Bird

- Designed in the abstract interpretation framework
- To find memory safety violations in C
  - such as buffer-overrun, memory leak, null dereference, etc.
  - for the full set of C

Program

$\langle \mathbb{C}, \hookrightarrow \rangle$

- $\mathbb{C}$: set of program points
- $\hookrightarrow \subseteq \mathbb{C} \times \mathbb{C}$: control flow relation

$c' \hookrightarrow c$ (c is a next control point of c')

Commands

$$lv := e \mid lv := \mathsf{alloc}(a) \mid \mathsf{assume}(x < e) \mid \mathsf{call}(f_x, e) \mid \mathsf{return}_f$$

$$\begin{array}{llll} \text{expression} & e & \rightarrow & n \mid e \ \texttt{+} \ e \mid lv \mid \&lv \\ \text{l-value} & lv & \rightarrow & x \mid \texttt{*}e \mid e\texttt{[}e\texttt{]} \mid e\texttt{.}x \\ \text{allocation} & a & \rightarrow & \texttt{[}e\texttt{]}_l \mid \texttt{\{}x\texttt{\}}_l \end{array}$$

Semantics

- The set of all reachable states at each program point

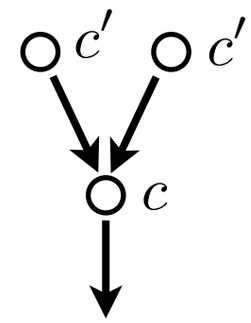
$$[\![P]\!] \in \mathbb{C} \to 2^{\mathbb{S}} = \mathrm{lfp} F$$

$$\mathbb{S} = \mathbb{L} \to \mathbb{V}$$

- Semantic function

$$F \in (\mathbb{C} \to 2^{\mathbb{S}}) \to (\mathbb{C} \to 2^{\mathbb{S}})$$

$$F(X) = \lambda c \in \mathbb{C}. f_c(\bigcup_{c' \hookrightarrow c} X(c')).$$



$f_c \in 2^{\mathbb{S}} \to 2^{\mathbb{S}}$: semantic function at control point c

Abstract Semantics

- abstract domain

$$\mathbb{C} \to 2^{\mathbb{S}} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{C} \to \hat{\mathbb{S}} \qquad 2^{\mathbb{S}} \underset{\alpha_{\mathbb{S}}}{\overset{\gamma_{\mathbb{S}}}{\leftrightarrows}} \hat{\mathbb{S}}$$

$$\hat{\mathbb{S}} = \hat{\mathbb{L}} \to \hat{\mathbb{V}}$$

$$\begin{aligned}
\hat{\mathbb{L}} &= Var + AllocSite + AllocSite \times FieldName \\
\hat{\mathbb{V}} &= \hat{\mathbb{Z}} \times 2^{\hat{\mathbb{L}}} \times 2^{AllocSite \times \hat{\mathbb{Z}} \times \hat{\mathbb{Z}}} \times 2^{AllocSite \times 2^{FieldName}} \\
\hat{\mathbb{Z}} &= \{[l,u] \mid l,u \in \mathbb{Z} \cup \{-\infty,+\infty\} \wedge l \leq u\} \cup \{\bot\}
\end{aligned}$$

Abstract Semantics

- abstract semantic function

$$\hat{F} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$$

$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')).$$

$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$: abstract semantics at each program point

Abstract Semantics

$$\hat{f}_c(\hat{s}) = \begin{cases} \hat{s}[\hat{\mathcal{L}}(lv)(\hat{s}) \overset{w}{\mapsto} \hat{\mathcal{V}}(e)(\hat{s})] & \mathsf{cmd}(c) = lv := e \\ \hat{s}[\hat{\mathcal{L}}(lv)(\hat{s}) \overset{w}{\mapsto} \langle \bot, \bot, \{\langle l, [0,0], \hat{\mathcal{V}}(e)(\hat{s}).1\rangle\}, \bot\rangle] & \mathsf{cmd}(c) = lv := \mathsf{alloc}(\texttt{[}e\texttt{]}_l) \\ \hat{s}[\hat{\mathcal{L}}(lv)(\hat{s}) \overset{w}{\mapsto} \langle \bot, \bot, \bot, \{\langle l, \{x\}\rangle\}\rangle] & \mathsf{cmd}(c) = lv := \mathsf{alloc}(\{x\}_l) \\ \hat{s}[x \mapsto \langle \hat{s}(x).1 \sqcap [-\infty, \mathsf{u}(\hat{\mathcal{V}}(e)(\hat{s}).1)], \hat{s}(x).2, \hat{s}(x).3, \hat{s}(x).4\rangle] & \mathsf{cmd}(c) = \mathsf{assume}(x < e) \\ \hat{s}[x \mapsto \hat{\mathcal{V}}(e)(\hat{s})] & \mathsf{cmd}(c) = \mathsf{call}(f_x, e) \\ \hat{s} & \mathsf{cmd}(c) = \mathsf{return}_f \end{cases}$$

$$\begin{array}{rcl} \hat{\mathcal{V}}(e) & \in & \hat{\mathbb{S}} \to \hat{\mathbb{V}} \\ \hat{\mathcal{V}}(n)(\hat{s}) & = & \langle [n,n], \bot, \bot, \bot\rangle \\ \hat{\mathcal{V}}(e_1 \texttt{ + } e_2)(\hat{s}) & = & \hat{\mathcal{V}}(e_1)(\hat{s}) \hat{+} \hat{\mathcal{V}}(e_2)(\hat{s}) \\ \hat{\mathcal{V}}(lv)(\hat{s}) & = & \bigsqcup\{\hat{s}(a) \mid a \in \hat{\mathcal{L}}(lv)(\hat{s})\} \\ \hat{\mathcal{V}}(\&lv)(\hat{s}) & = & \langle \bot, \hat{\mathcal{L}}(lv)(\hat{s}), \bot, \bot\rangle \end{array}$$

$$\begin{array}{rcl} \hat{\mathcal{L}}(lv) & \in & \hat{\mathbb{S}} \to 2^{\hat{\mathbb{L}}} \\ \hat{\mathcal{L}}(x)(\hat{s}) & = & \{x\} \\ \hat{\mathcal{L}}(*e)(\hat{s}) & = & \hat{\mathcal{V}}(e)(\hat{s}).2 \cup \{l \mid \langle l, o, s\rangle \in \hat{\mathcal{V}}(e)(\hat{s}).3\} \\ & & \cup \{\langle l, x\rangle \mid \langle l, \{x\}\rangle \in \hat{\mathcal{V}}(e)(\hat{s}).4\} \\ \hat{\mathcal{L}}(e_1\texttt{[}e_2\texttt{]})(\hat{s}) & = & \{l \mid \langle l, o, s\rangle \in \hat{\mathcal{V}}(e_1)(\hat{s}).3\} \\ \hat{\mathcal{L}}(e\,.\,x)(\hat{s}) & = & \{\langle l, x\rangle \mid \langle l, \{x\}\rangle \in \hat{\mathcal{V}}(e)(\hat{s}).4\} \end{array}$$

Computing $\bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\hat{\bot})$

$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}. \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')).$$

$\hat{X}, \hat{X}' \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$\hat{X} := \hat{X}' := \lambda c. \bot$
repeat
  $\hat{X}' := \hat{X}$
  for all $c \in \mathbb{C}$ **do**
    $\hat{X}(c) := \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$
until $\hat{X} \sqsubseteq \hat{X}'$

Naive fixpoint algorithm

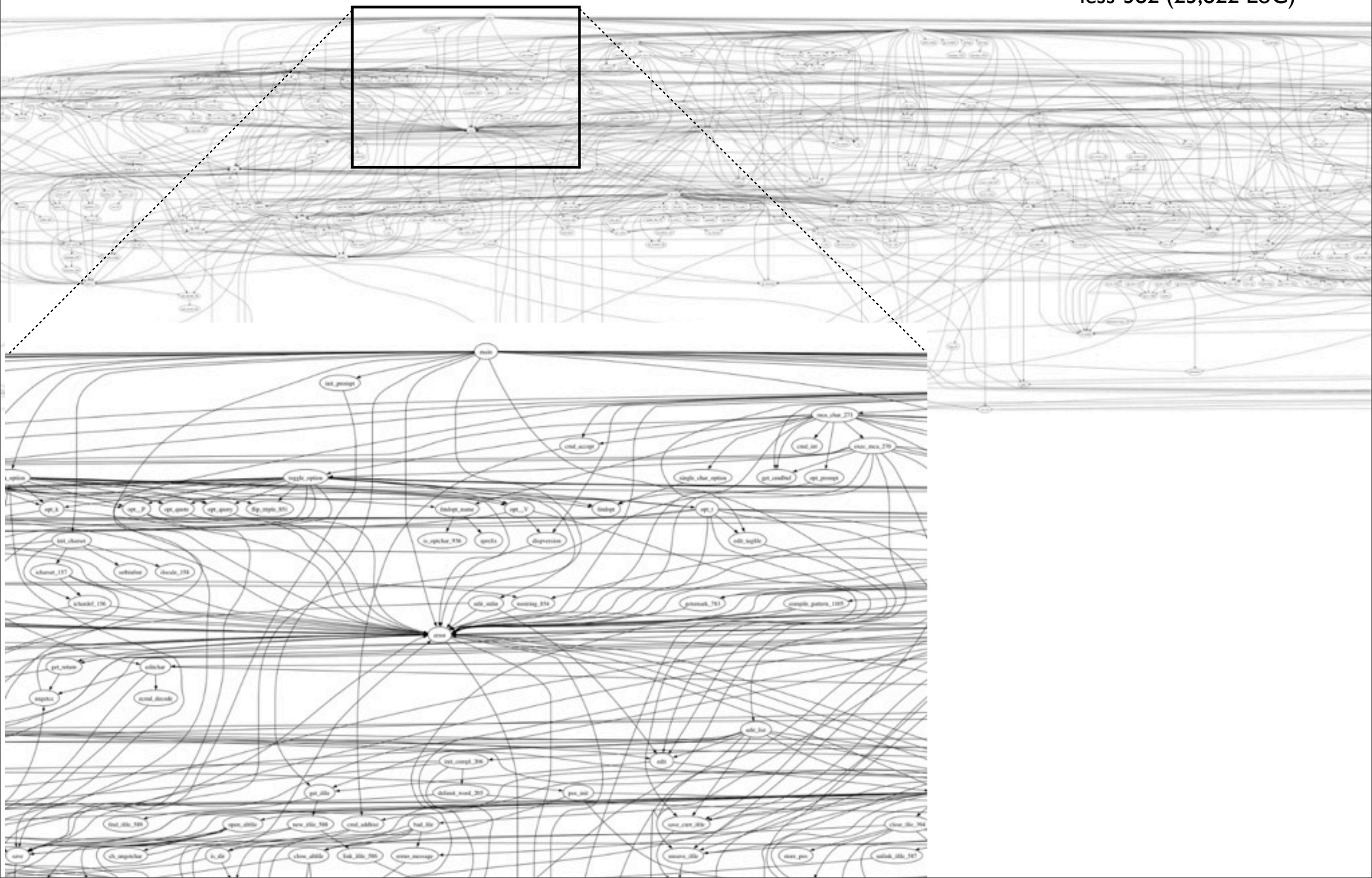
$W \in \mathit{Worklist} = 2^{\mathbb{C}}$
$\hat{X} \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$W := \mathbb{C}$
$\hat{X} := \lambda c. \bot$
repeat
  $c := \mathsf{choose}(W)$
  $\hat{s} := \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$
  if $\hat{s} \not\sqsubseteq \hat{X}(c)$
    $W := W \cup \{c' \in \mathbb{C} \mid c \hookrightarrow c'\}$
    $\hat{X}(c) := \hat{X}(c) \sqcup \hat{s}$
until $W = \emptyset$

Worklist algorithm

The Algorithms Too Weak in Reality

less-382 (23,822 LoC)



Improving Scalability

Key Idea: Localization

"Right Part at Right Moment"

- Spatial localization [VMCAI'11,APLAS'11]
- Temporal localization [PLDI'12]

Computing $\bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\hat{\bot})$

$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}. \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')).$$

$\hat{X}, \hat{X}' \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$\hat{X} := \hat{X}' := \lambda c. \bot$
repeat
  $\hat{X}' := \hat{X}$
  for all $c \in \mathbb{C}$ **do**
    $\hat{X}(c) := \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$
until $\hat{X} \sqsubseteq \hat{X}'$

Naive fixpoint algorithm

$W \in \mathit{Worklist} = 2^{\mathbb{C}}$
$\hat{X} \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$W := \mathbb{C}$
$\hat{X} := \lambda c. \bot$
repeat
  $c := \mathsf{choose}(W)$
  $\hat{s} := \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$
  if $\hat{s} \not\sqsubseteq \hat{X}(c)$
    $W := W \cup \{c' \in \mathbb{C} \mid c \hookrightarrow c'\}$
    $\hat{X}(c) := \hat{X}(c) \sqcup \hat{s}$
until $W = \emptyset$

Worklist algorithm

Performance of sound & global Sparrow The Early Bird benchmark programs

Program	LOC	Functions	Statements	Blocks	maxSCC	AbsLocs
gzip-1.2.4a	7K	132	6,446	4,152	2	1,784
bc-1.06	13K	132	10,368	4,731	1	1,619
tar-1.13	20K	221	12,199	8,586	13	3,245
less-382	23K	382	23,367	9,207	46	3,658
make-3.76.1	27K	190	14,010	9,094	57	4,527
wget-1.9	35K	433	28,958	14,537	13	6,675
screen-4.0.2	45K	588	39,693	29,498	65	12,566
a2ps-4.14	64K	980	86,867	27,565	6	17,684
bash-2.05a	105K	955	107,774	27,669	4	17,443
lsh-2.0.4	111K	1,524	137,511	27,896	13	31,164
sendmail-8.13.6	130K	756	76,630	52,505	60	19,135
nethack-3.3.0	211K	2,207	237,427	157,645	997	54,989
vim60	227K	2,770	150,950	107,629	1,668	40,979
emacs-22.1	399K	3,388	204,865	161,118	1,554	66,413
python-2.5.1	435K	2,996	241,511	99,014	723	51,859
linux-3.0	710K	13,856	345,407	300,203	493	139,667
gimp-2.6	959K	11,728	1,482,230	286,588	2	190,806
ghostscript-9.00	1,363K	12,993	2,891,500	342,293	39	201,161

Performance of sound & global

Programs	LOC	$\text{Interval}_{\text{vanilla}}$		$\text{Interval}_{\text{base}}$		Spd↑$_1$	Mem↓$_1$	$\text{Interval}_{\text{sparse}}$						Spd↑$_2$	Mem↓$_2$
		Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	7K	772	240	14	65	55 x	73 %	2	1	3	63	2.4	2.5	5 x	3 %
bc-1.06	13K	1,270	276	96	126	13 x	54 %	4	3	7	75	4.6	4.9	14 x	40 %
tar-1.13	20K	12,947	881	338	177	38 x	80 %	6	2	8	93	2.9	2.9	42 x	47 %
less-382	23K	9,561	1,113	1,211	378	8 x	66 %	27	6	33	127	11.9	11.9	37 x	66 %
make-3.76.1	27K	24,240	1,391	1,893	443	13 x	68 %	16	5	21	114	5.8	5.8	90 x	74 %
wget-1.9	35K	44,092	2,546	1,214	378	36 x	85 %	8	3	11	85	2.4	2.4	110 x	78 %
screen-4.0.2	45K	∞	N/A	31,324	3,996	N/A	N/A	724	43	767	303	53.0	54.0	41 x	92 %
a2ps-4.14	64K	∞	N/A	3,200	1,392	N/A	N/A	31	9	40	353	2.6	2.8	80 x	75 %
bash-2.05a	105K	∞	N/A	1,683	1,386	N/A	N/A	45	22	67	220	3.0	3.0	25 x	84 %
lsh-2.0.4	111K	∞	N/A	45,522	5,266	N/A	N/A	391	80	471	577	21.1	21.2	97 x	89 %
sendmail-8.13.6	130K	∞	N/A	∞	N/A	N/A	N/A	517	227	744	678	20.7	20.7	N/A	N/A
nethack-3.3.0	211K	∞	N/A	∞	N/A	N/A	N/A	14,126	2,247	16,373	5,298	72.4	72.4	N/A	N/A
vim60	227K	∞	N/A	∞	N/A	N/A	N/A	17,518	6,280	23,798	5,190	180.2	180.3	N/A	N/A
emacs-22.1	399K	∞	N/A	∞	N/A	N/A	N/A	29,552	8,278	37,830	7,795	285.3	285.5	N/A	N/A
python-2.5.1	435K	∞	N/A	∞	N/A	N/A	N/A	9,677	1,362	11,039	5,535	108.1	108.1	N/A	N/A
linux-3.0	710K	∞	N/A	∞	N/A	N/A	N/A	26,669	6,949	33,618	20,529	76.2	74.8	N/A	N/A
gimp-2.6	959K	∞	N/A	∞	N/A	N/A	N/A	3,751	123	3,874	3,602	4.1	3.9	N/A	N/A
ghostscript-9.00	1,363K	∞	N/A	∞	N/A	N/A	N/A	14,116	698	14,814	6,384	9.7	9.7	N/A	N/A

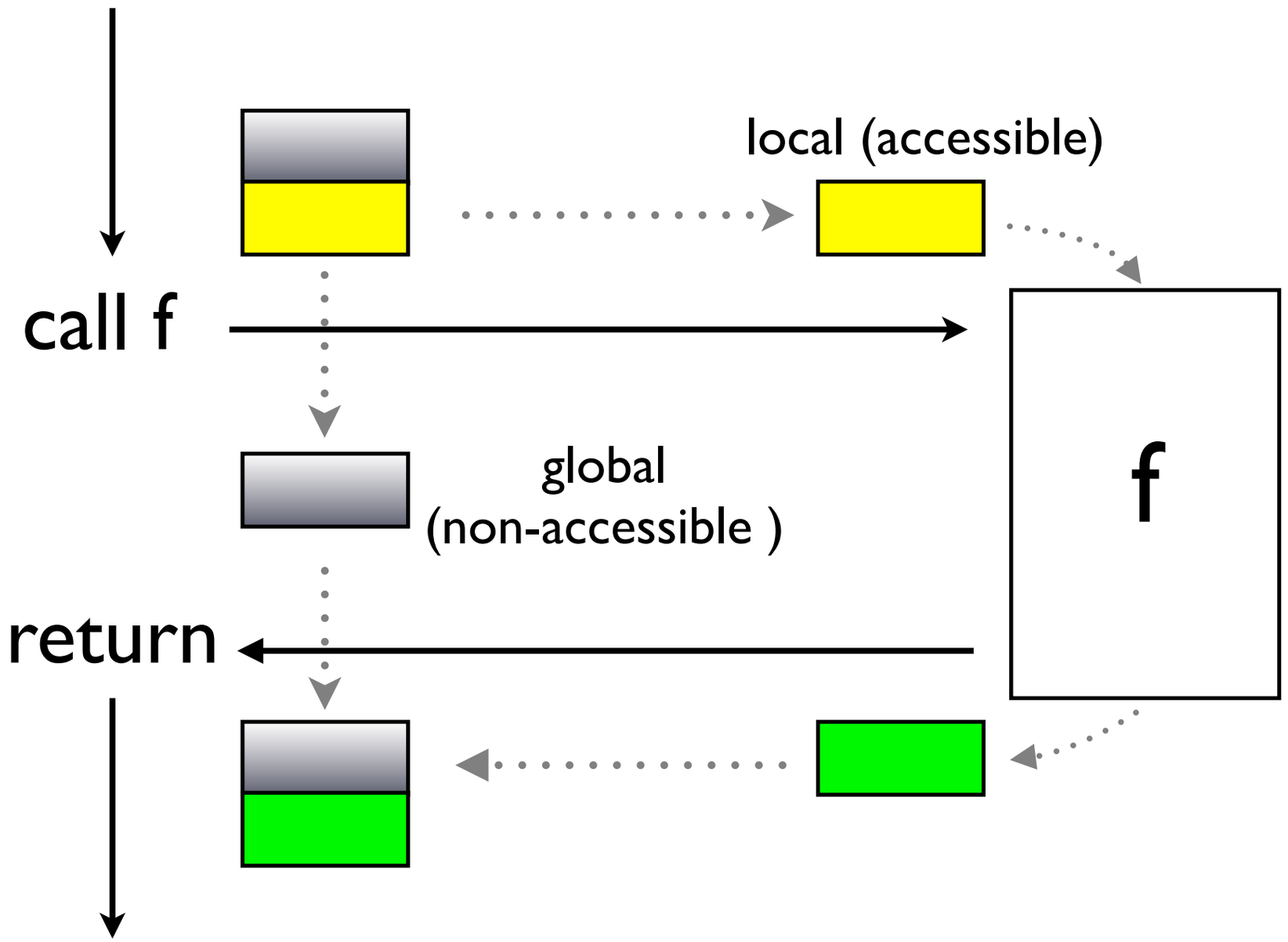
none

spatial localization

spatial+temporal localization

Spatial Localization

Memory Localization
(spatial localization)



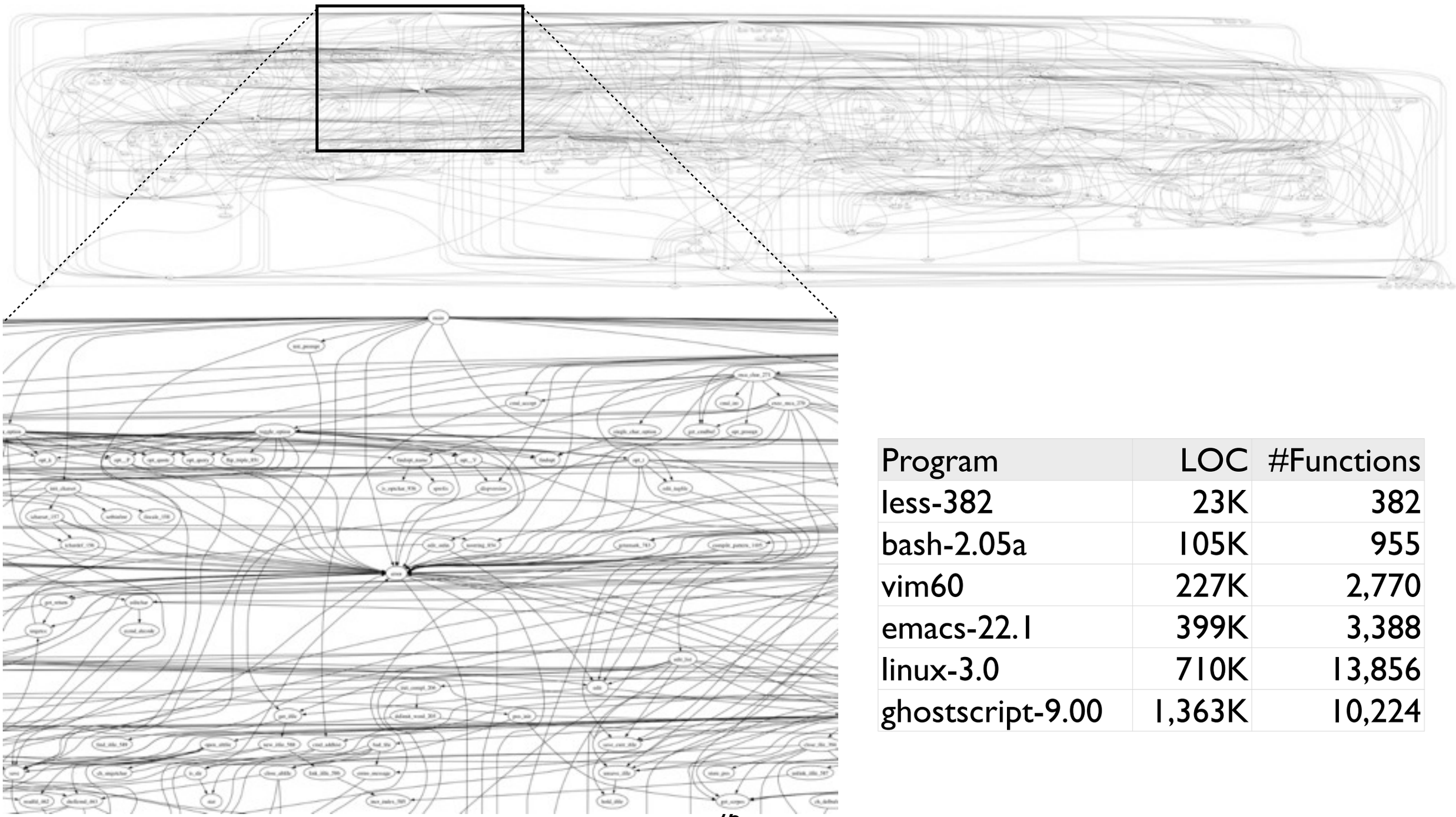
Benefits

```
int g;

int f() {...}

int main() {
  g = 0;
  f();

  g = 1;
  f();
}
```

f does not access g

Vital in Practice

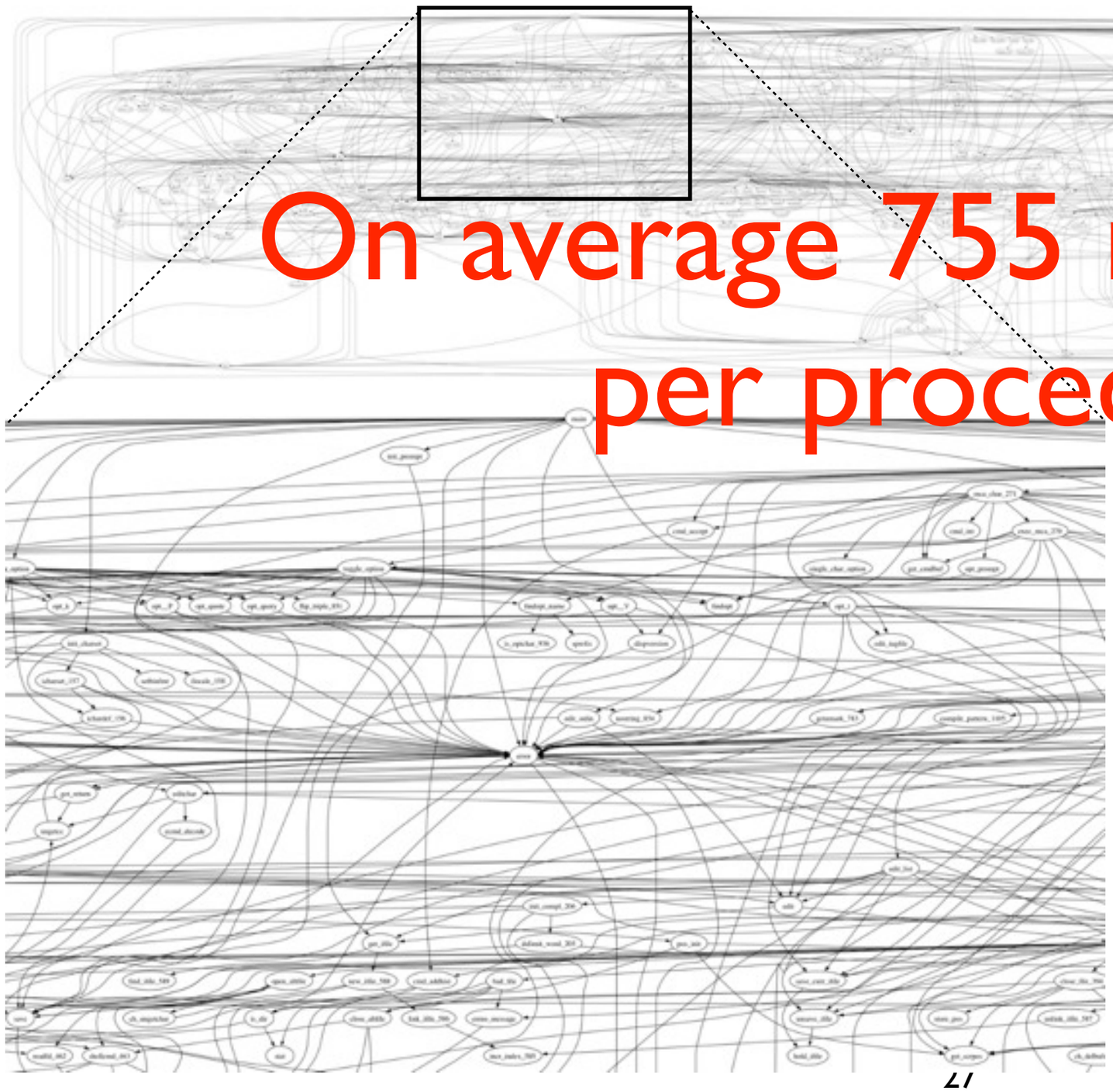
less-382 (23,822 LOC)



Program	LOC	#Functions
less-382	23K	382
bash-2.05a	105K	955
vim60	227K	2,770
emacs-22.1	399K	3,388
linux-3.0	710K	13,856
ghostscript-9.00	1,363K	10,224

Vital in Practice

less-382 (23,822 LOC)



On average 755 re-analyses per procedure

Program	LOC	#Functions
less-382	23K	382
bash-2.05a	105K	955
vim60	227K	2,770
emacs-22.1	399K	3,388
linux-3.0	710K	13,856
ghostscript-9.00	1,363K	10,224

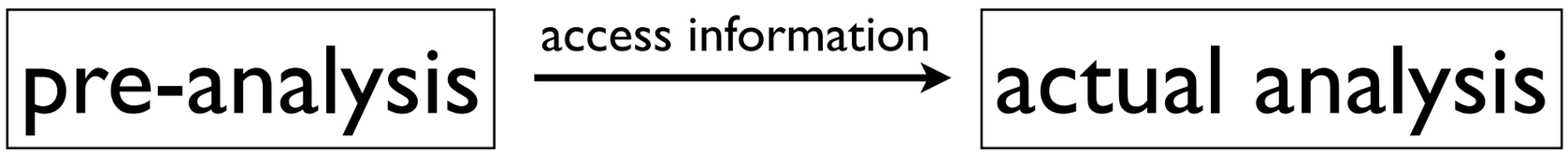
Huge Room for Localization

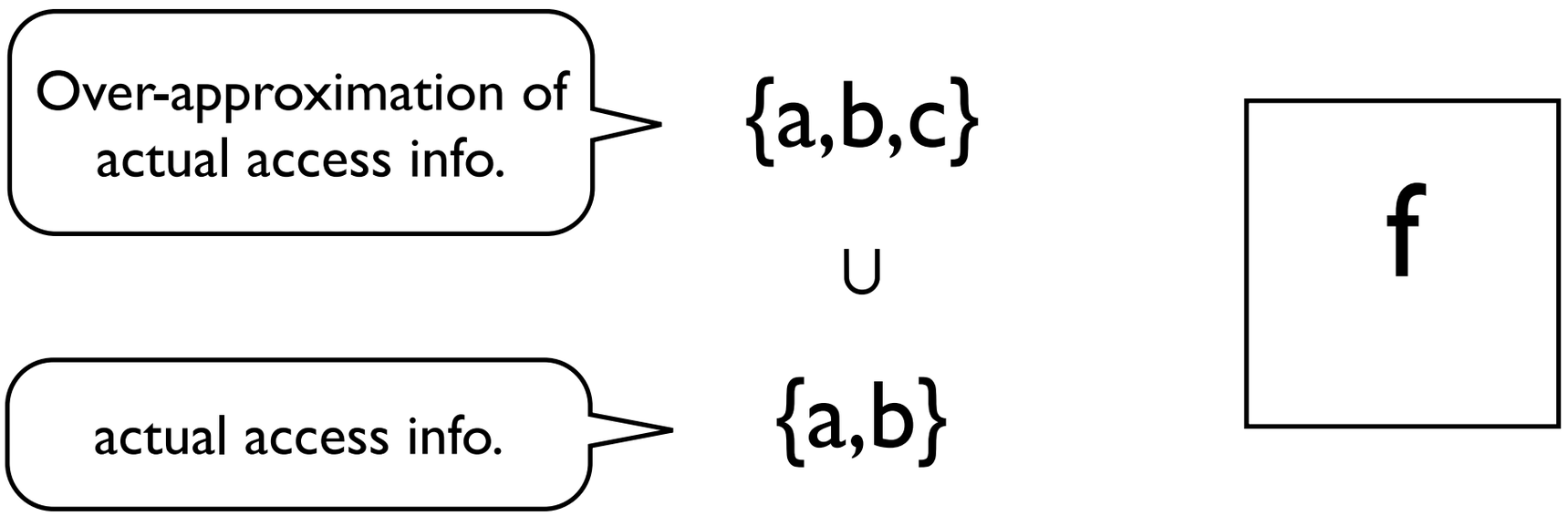
conventional reachability-based technique is too conservative

Program	LOC	accessed memory / reachable memory
spell-1.0	2,213	5 / 453 (1.1%)
barcode-0.96	4,460	19 / 1175 (1.6%)
httptunnel-3.3	6,174	10 / 673 (1.5%)
gzip-1.2.4a	7,327	22 / 1002 (2.2%)
jwhois-3.0.1	9,344	28 / 830 (3.4%)
parser	10,900	75 / 1787 (4.2%)
bc-1.06	13,093	24 / 824 (2.9%)
less-290	18,449	86 / 1546 (5.6%)

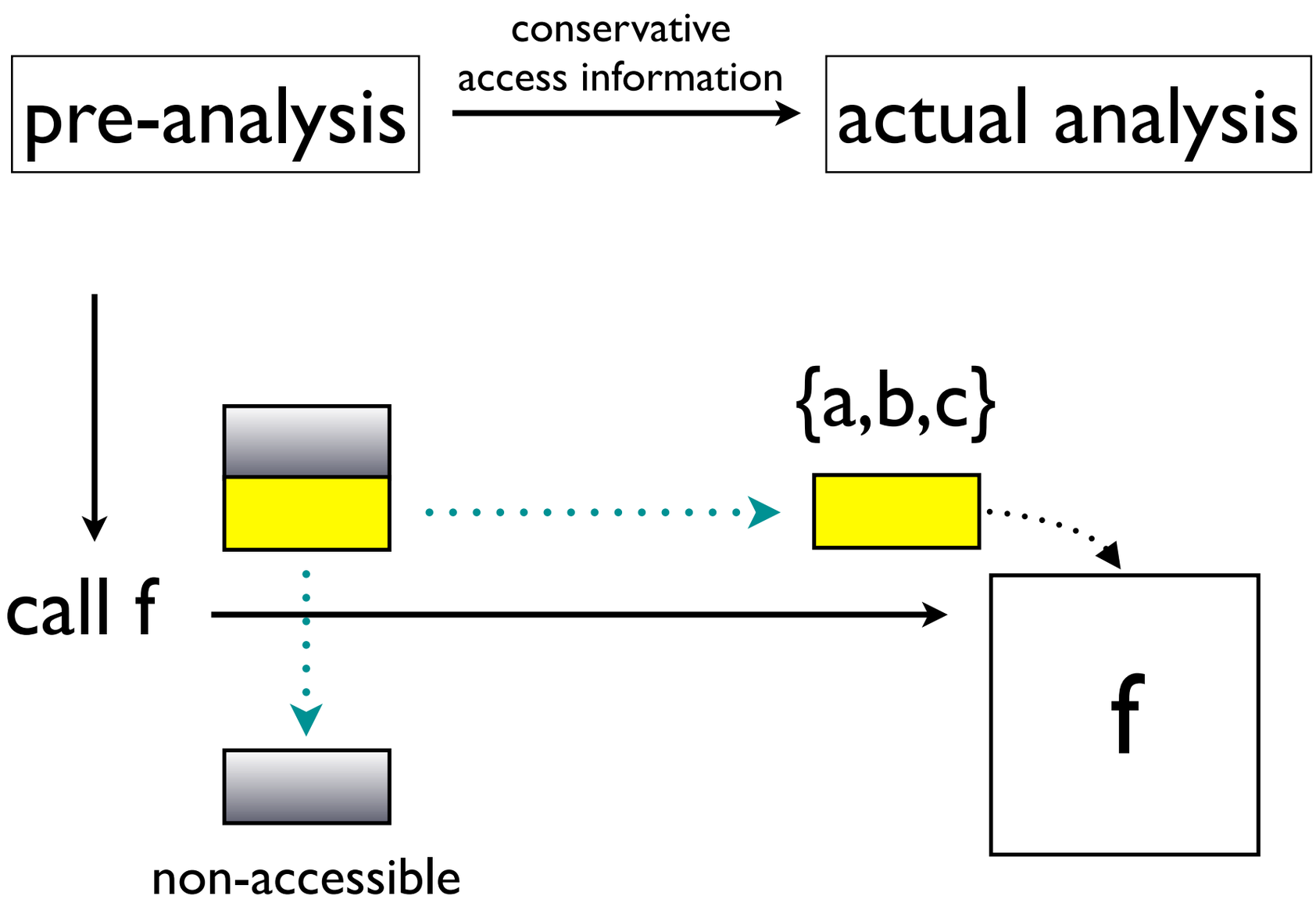
average : 4%

Access-based Localization





Access-based Localization



Our Pre-analysis

- abstract domain

$$\mathbb{C} \to \hat{\mathbb{S}} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \hat{\mathbb{S}}$$

$$\alpha = \lambda \hat{X}. \bigsqcup_{c \in \mathbb{C}} \hat{X}(c)$$

$$\gamma = \lambda \hat{s}. \lambda c \in \mathbb{C}. \hat{s}$$

- abstract semantic function

$$\hat{F}_p = \lambda \hat{s}. (\bigsqcup_{c \in \mathbb{C}} \hat{f}_c(\hat{s}))$$

$$\hat{s}_{pre} = \mathbf{fix} \hat{F}_p$$

Implementation on Sparrow

(modifying semantic function)

$$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$$

$$\hat{f}_c(\hat{s}) = \begin{cases} \hat{s}[\hat{\mathcal{L}}(lv)(\hat{s}) \overset{w}{\mapsto} \hat{\mathcal{V}}(e)(\hat{s})] & \mathsf{cmd}(c) = lv := e \\ \hat{s}[\hat{\mathcal{L}}(lv)(\hat{s}) \overset{w}{\mapsto} \langle \bot, \bot, \{\langle l, [0,0], \hat{\mathcal{V}}(e)(\hat{s}).1 \rangle\}, \bot \rangle] & \mathsf{cmd}(c) = lv := \mathsf{alloc}([e]_l) \\ \hat{s}[\hat{\mathcal{L}}(lv)(\hat{s}) \overset{w}{\mapsto} \langle \bot, \bot, \bot, \{\langle l, \{x\} \rangle\} \rangle] & \mathsf{cmd}(c) = lv := \mathsf{alloc}(\{x\}_l) \\ \hat{s}[x \mapsto \langle \hat{s}(x).1 \sqcap [-\infty, \mathsf{u}(\hat{\mathcal{V}}(e)(\hat{s}).1)], \hat{s}(x).2, \hat{s}(x).3, \hat{s}(x).4 \rangle] & \mathsf{cmd}(c) = \mathsf{assume}(x < e) \\ \hat{s}[x \mapsto \hat{\mathcal{V}}(e)(\hat{s})] & \mathsf{cmd}(c) = \mathsf{call}(f_x, e) \\ \hat{s} & \mathsf{cmd}(c) = \mathsf{return}_f \end{cases}$$

$$\hat{f}_c(\hat{s}) = \hat{s}'|_{\mathsf{access}(f)} \text{ where } \hat{s}' = \hat{s}[x \mapsto \hat{\mathcal{V}}(e)(\hat{s})]$$

$$\mathsf{access}(f) = \bigcup_{g \in \mathsf{callees}(f)} (\bigcup_{c \in \mathsf{control}(g)} \mathcal{A}(c)(\hat{s}_{pre}))$$

Performance of sound & global

Programs	LOC	$\text{Interval}_{\text{vanilla}}$		$\text{Interval}_{\text{base}}$		Spd↑$_1$	Mem↓$_1$	$\text{Interval}_{\text{sparse}}$						Spd↑$_2$	Mem↓$_2$
		Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	7K	772	240	14	65	55 x	73 %	2	1	3	63	2.4	2.5	5 x	3 %
bc-1.06	13K	1,270	276	96	126	13 x	54 %	4	3	7	75	4.6	4.9	14 x	40 %
tar-1.13	20K	12,947	881	338	177	38 x	80 %	6	2	8	93	2.9	2.9	42 x	47 %
less-382	23K	9,561	1,113	1,211	378	8 x	66 %	27	6	33	127	11.9	11.9	37 x	66 %
make-3.76.1	27K	24,240	1,391	1,893	443	13 x	68 %	16	5	21	114	5.8	5.8	90 x	74 %
wget-1.9	35K	44,092	2,546	1,214	378	36 x	85 %	8	3	11	85	2.4	2.4	110 x	78 %
screen-4.0.2	45K	∞	N/A	31,324	3,996	N/A	N/A	724	43	767	303	53.0	54.0	41 x	92 %
a2ps-4.14	64K	∞	N/A	3,200	1,392	N/A	N/A	31	9	40	353	2.6	2.8	80 x	75 %
bash-2.05a	105K	∞	N/A	1,683	1,386	N/A	N/A	45	22	67	220	3.0	3.0	25 x	84 %
lsh-2.0.4	111K	∞	N/A	45,522	5,266	N/A	N/A	391	80	471	577	21.1	21.2	97 x	89 %
sendmail-8.13.6	130K	∞	N/A	∞	N/A	N/A	N/A	517	227	744	678	20.7	20.7	N/A	N/A
nethack-3.3.0	211K	∞	N/A	∞	N/A	N/A	N/A	14,126	2,247	16,373	5,298	72.4	72.4	N/A	N/A
vim60	227K	∞	N/A	∞	N/A	N/A	N/A	17,518	6,280	23,798	5,190	180.2	180.3	N/A	N/A
emacs-22.1	399K	∞	N/A	∞	N/A	N/A	N/A	29,552	8,278	37,830	7,795	285.3	285.5	N/A	N/A
python-2.5.1	435K	∞	N/A	∞	N/A	N/A	N/A	9,677	1,362	11,039	5,535	108.1	108.1	N/A	N/A
linux-3.0	710K	∞	N/A	∞	N/A	N/A	N/A	26,669	6,949	33,618	20,529	76.2	74.8	N/A	N/A
gimp-2.6	959K	∞	N/A	∞	N/A	N/A	N/A	3,751	123	3,874	3,602	4.1	3.9	N/A	N/A
ghostscript-9.00	1,363K	∞	N/A	∞	N/A	N/A	N/A	14,116	698	14,814	6,384	9.7	9.7	N/A	N/A

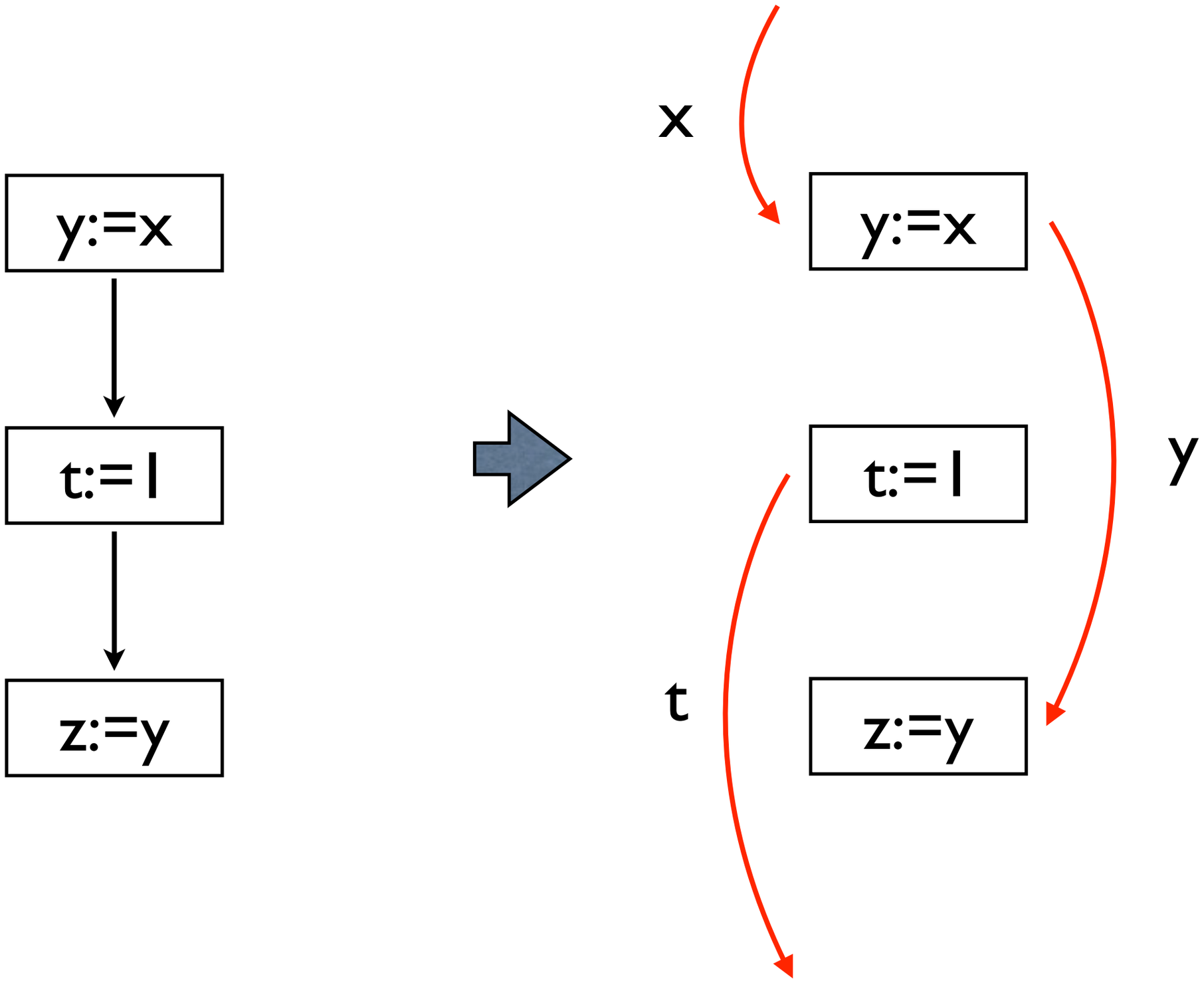
none

spatial localization

spatial+temporal localization

Temporal Localization

Temporal Localization



Sparse Analysis Framework

For a general class of abstract interpretation,

$$\hat{F} : \hat{D} \to \hat{D} \quad \Rightarrow \quad \hat{F}_s : \hat{D} \to \hat{D}$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$\mathrm{lfp}\hat{F} \qquad = \qquad \mathrm{lfp}\hat{F}_s$$

Baseline Non-sparse Analyzer

- abstract domain

$$\mathbb{C} \to 2^{\mathbb{S}} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{C} \to \hat{\mathbb{S}}$$

$$\hat{\mathbb{S}} \quad = \quad \hat{\mathbb{L}} \to \hat{\mathbb{V}}$$

- analyzer computes the fixpoint of

$$\hat{F} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$$

$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')).$$

"Obvious" Sparse Version

$$\hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \overset{l}{\rightsquigarrow} c} \hat{X}(c')|_l).$$

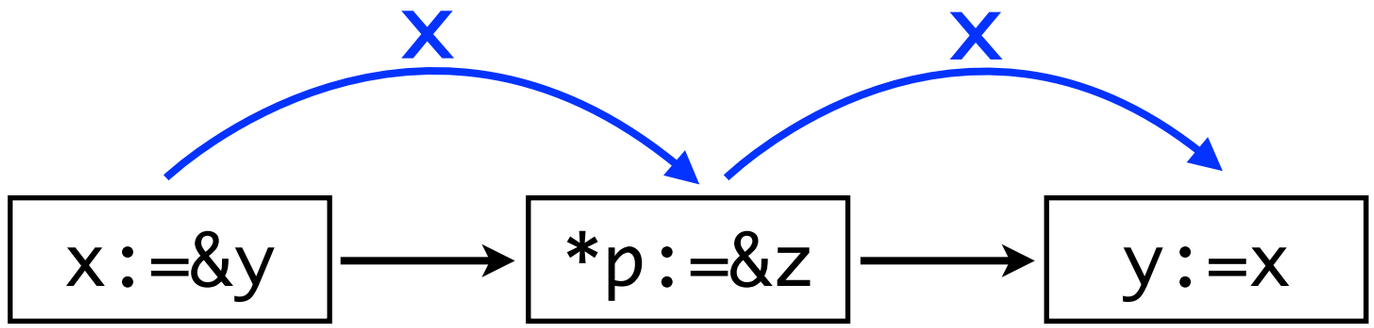
Lemma 2 (Correctness)**.** *Let* $\mathcal{S}$ *and* $\mathcal{S}_s$ *be* $\mathbf{lfp}\hat{F}$ *and* $\mathbf{lfp}\hat{F}_s$*. Then,*

$$\forall c \in \mathbb{C}.\forall l \in \mathsf{dom}(\mathcal{S}_s(c)).\mathcal{S}_s(c)(l) = \mathcal{S}(c)(l).$$

"Obvious" Data Dependency

Definition 3 (Data dependency)**.** *Let c_d and c_u be control points and l be an abstract location. Data dependency is ternary relation $\rightsquigarrow$ defined as follows:*

$$
\begin{array}{lll}
c_d \overset{l}{\rightsquigarrow} c_u & \triangleq & c_d \hookrightarrow^+ c_u \\
& \wedge & l \in \mathsf{D}(c_d) \cap \mathsf{U}(c_u) \\
& \wedge & \forall c_i \in \mathbb{C}. c_d \hookrightarrow^+ c_i \hookrightarrow^+ c_u \implies l \notin \mathsf{D}(c_i).
\end{array}
$$



Def	{x}	{x,y}	{y}
Use	ϕ	{p,x,y}	{x}

Realizable Sparse Version

And still as precise as the original

$$\hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \overset{l}{\rightsquigarrow}_a c} \hat{X}(c')|_l).$$

Lemma 3 (Correctness of Safe Approximation)**.** *Suppose sparse abstract semantic function* $\hat{F}_a$ *is derived by the safe approximation* $\hat{\mathsf{D}}$ *and* $\hat{\mathsf{U}}$*. Let* $\mathcal{S}$ *and* $\mathcal{S}_a$ *be* $\mathbf{lfp}\hat{F}$ *and* $\mathbf{lfp}\hat{F}_a$*. Then,*

$$\forall c \in \mathbb{C}.\forall l \in \mathsf{dom}(\mathcal{S}_a(c)).\mathcal{S}_a(c)(l) = \mathcal{S}(c)(l).$$

Realizable Data Dependency

Prepare def/use set using
yet another safe pre-analysis

Definition 4 (Approximated Data Dependency)**.** *Let c_d and c_u be control points and l be an abstract location. Approximated data dependency is ternary relation $\rightsquigarrow_a$ defined as follows:*

$$
\begin{array}{lcl}
c_d \overset{l}{\rightsquigarrow}_a c_u & \triangleq & c_d \hookrightarrow^+ c_u \\
& \wedge & l \in \hat{\mathsf{D}}(c_d) \cap \hat{\mathsf{U}}(c_u) \\
& \wedge & \forall c_i . c_d \hookrightarrow^+ c_i \hookrightarrow^+ c_u \implies l \notin \hat{\mathsf{D}}(c_i)
\end{array}
$$

Condition

To be safe, approximated def/use should satisfy two conditions:

- over-approximation
- spurious definitions should be also included in uses

Definition 5. *Set* $\hat{\mathsf{D}}(c)$ *and* $\hat{\mathsf{U}}(c)$ *are a safe approximation of definition set* $\mathsf{D}(c)$ *and use set* $\mathsf{U}(c)$*, respectively, if and only if*

(1) $\hat{\mathsf{D}}(c) \supseteq \mathsf{D}(c) \wedge \hat{\mathsf{U}}(c) \supseteq \mathsf{U}(c)$*; and*
(2) $\hat{\mathsf{D}}(c) - \mathsf{D}(c) \subseteq \hat{\mathsf{U}}(c)$.

Performance of sound & global

Programs	LOC	$\text{Interval}_{\text{vanilla}}$		$\text{Interval}_{\text{base}}$		Spd↑$_1$	Mem↓$_1$	$\text{Interval}_{\text{sparse}}$						Spd↑$_2$	Mem↓$_2$
		Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	7K	772	240	14	65	55 x	73 %	2	1	3	63	2.4	2.5	5 x	3 %
bc-1.06	13K	1,270	276	96	126	13 x	54 %	4	3	7	75	4.6	4.9	14 x	40 %
tar-1.13	20K	12,947	881	338	177	38 x	80 %	6	2	8	93	2.9	2.9	42 x	47 %
less-382	23K	9,561	1,113	1,211	378	8 x	66 %	27	6	33	127	11.9	11.9	37 x	66 %
make-3.76.1	27K	24,240	1,391	1,893	443	13 x	68 %	16	5	21	114	5.8	5.8	90 x	74 %
wget-1.9	35K	44,092	2,546	1,214	378	36 x	85 %	8	3	11	85	2.4	2.4	110 x	78 %
screen-4.0.2	45K	∞	N/A	31,324	3,996	N/A	N/A	724	43	767	303	53.0	54.0	41 x	92 %
a2ps-4.14	64K	∞	N/A	3,200	1,392	N/A	N/A	31	9	40	353	2.6	2.8	80 x	75 %
bash-2.05a	105K	∞	N/A	1,683	1,386	N/A	N/A	45	22	67	220	3.0	3.0	25 x	84 %
lsh-2.0.4	111K	∞	N/A	45,522	5,266	N/A	N/A	391	80	471	577	21.1	21.2	97 x	89 %
sendmail-8.13.6	130K	∞	N/A	∞	N/A	N/A	N/A	517	227	744	678	20.7	20.7	N/A	N/A
nethack-3.3.0	211K	∞	N/A	∞	N/A	N/A	N/A	14,126	2,247	16,373	5,298	72.4	72.4	N/A	N/A
vim60	227K	∞	N/A	∞	N/A	N/A	N/A	17,518	6,280	23,798	5,190	180.2	180.3	N/A	N/A
emacs-22.1	399K	∞	N/A	∞	N/A	N/A	N/A	29,552	8,278	37,830	7,795	285.3	285.5	N/A	N/A
python-2.5.1	435K	∞	N/A	∞	N/A	N/A	N/A	9,677	1,362	11,039	5,535	108.1	108.1	N/A	N/A
linux-3.0	710K	∞	N/A	∞	N/A	N/A	N/A	26,669	6,949	33,618	20,529	76.2	74.8	N/A	N/A
gimp-2.6	959K	∞	N/A	∞	N/A	N/A	N/A	3,751	123	3,874	3,602	4.1	3.9	N/A	N/A
ghostscript-9.00	1,363K	∞	N/A	∞	N/A	N/A	N/A	14,116	698	14,814	6,384	9.7	9.7	N/A	N/A

none

spatial localization

spatial+temporal localization

Conclusion

Localization techniques enables detailed, sound, and also scalable global static analysis for million lines of C.