Sparse Analysis Framework

Hakjoo Oh

Programming Research Laboratory
Seoul National University

18/04/2013 @ Dagstuhl, Germany

Co-work with Kihong Heo, Wonchan Lee, Woosuk Lee, and Kwangkeun Yi

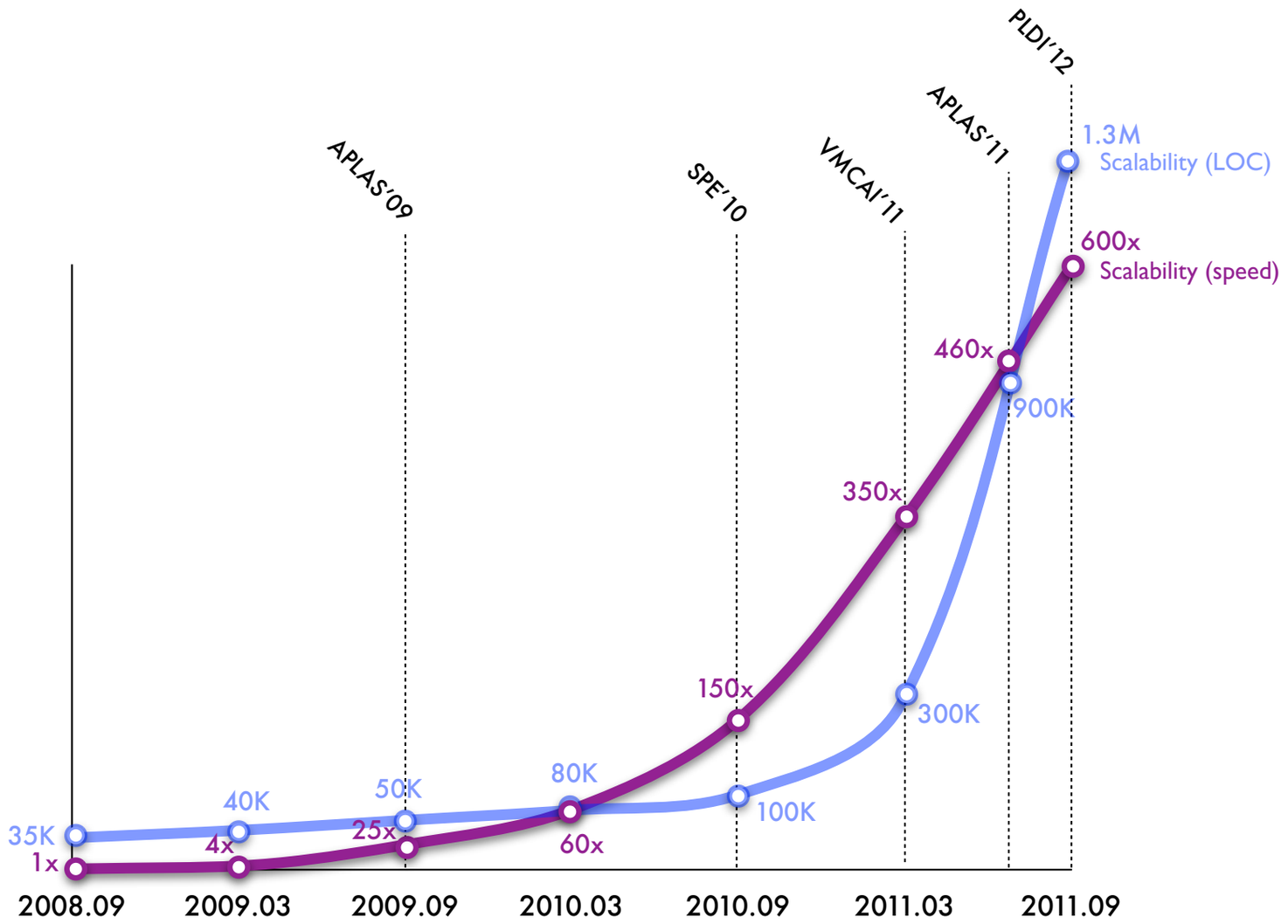
Motivation

- In 2007, we commercialized Sparrow The Early Bird
  - memory-bug-finding tool for full C
    - non domain-specific, flow-sensitive analysis for int & ptrs
    - sound(y) in design, unsound yet scalable in reality
- Realistic workbench available
  - “let’s try to scale-up its sound & global analysis version”

Scalability Improvement

Sparrow
The Early Bird

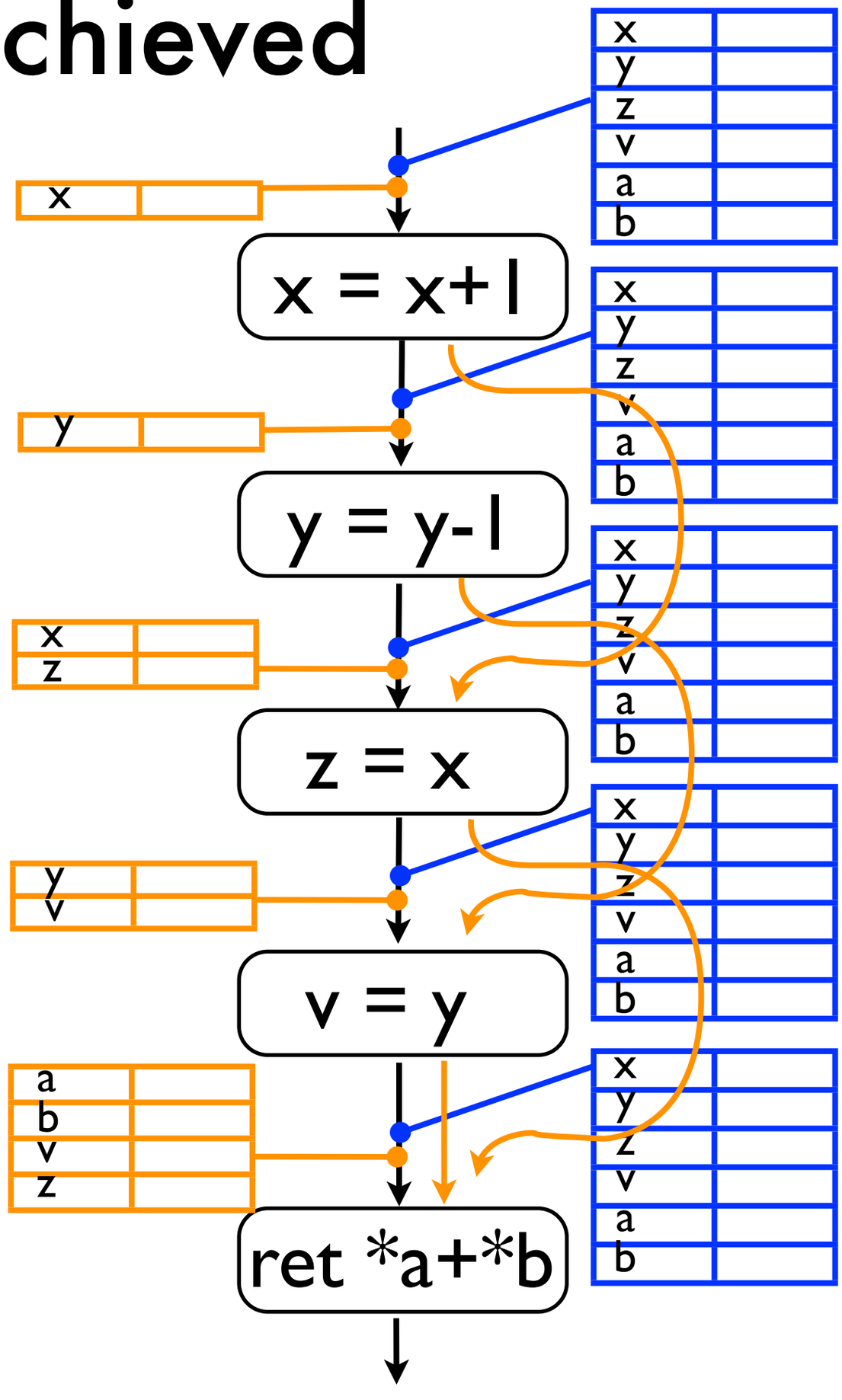
sound & global analysis version



- < 1.4M in 10hr with intervals
- < 0.14M in 20hrs with octagons

This Talk: How we achieved

Key: Sparse Analysis



Needs for Sparse Analysis Theory

- *abstract interpretation*
  - design theory for provably correct static analysis
    - the resulting analysis is "dense" and unscalable
- *sparse analysis*
  - algorithmic technique for achieving scalability*
  - no design theory like abstract interpretation

---

* Hardekopf and Lin. Semi-sparse flow-sensitive pointer analysis. POPL'09
  Hardekopf and Lin. Flow-sensitive pointer-analysis for millions of lines of code. CGO'11

PLDI'12

Sparse Analysis Framework

design theory for sparse analysis

baseline analysis

$$\hat{F} : \hat{D} \to \hat{D}$$

$$fix\,\hat{F}$$

sparsify

$\Longrightarrow$

still

$=$

“sparse” version

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

$$fix\,\hat{F}_s$$

The Framework is General

- arbitrary programming languages
  - imperative, functional, oo, etc
- arbitrary trace partitioning strategies
  - flow-/context-/path-sensitivity, etc

Sparse Analysis Framework

for simplicity, assume

- C-like programs
- flow-sensitive & context-insensitive analysis

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 the next program point to c')

Baseline Analysis

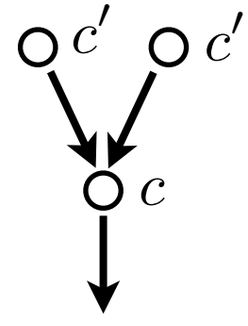
- Abstract domain

$$[\![\hat{P}]\!] \in \mathbb{C} \to \hat{\mathbb{S}} = fix\hat{F}$$
$$\hat{\mathbb{S}} = \hat{\mathbb{L}} \to \hat{\mathbb{V}}$$

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

- Abstract semantic function

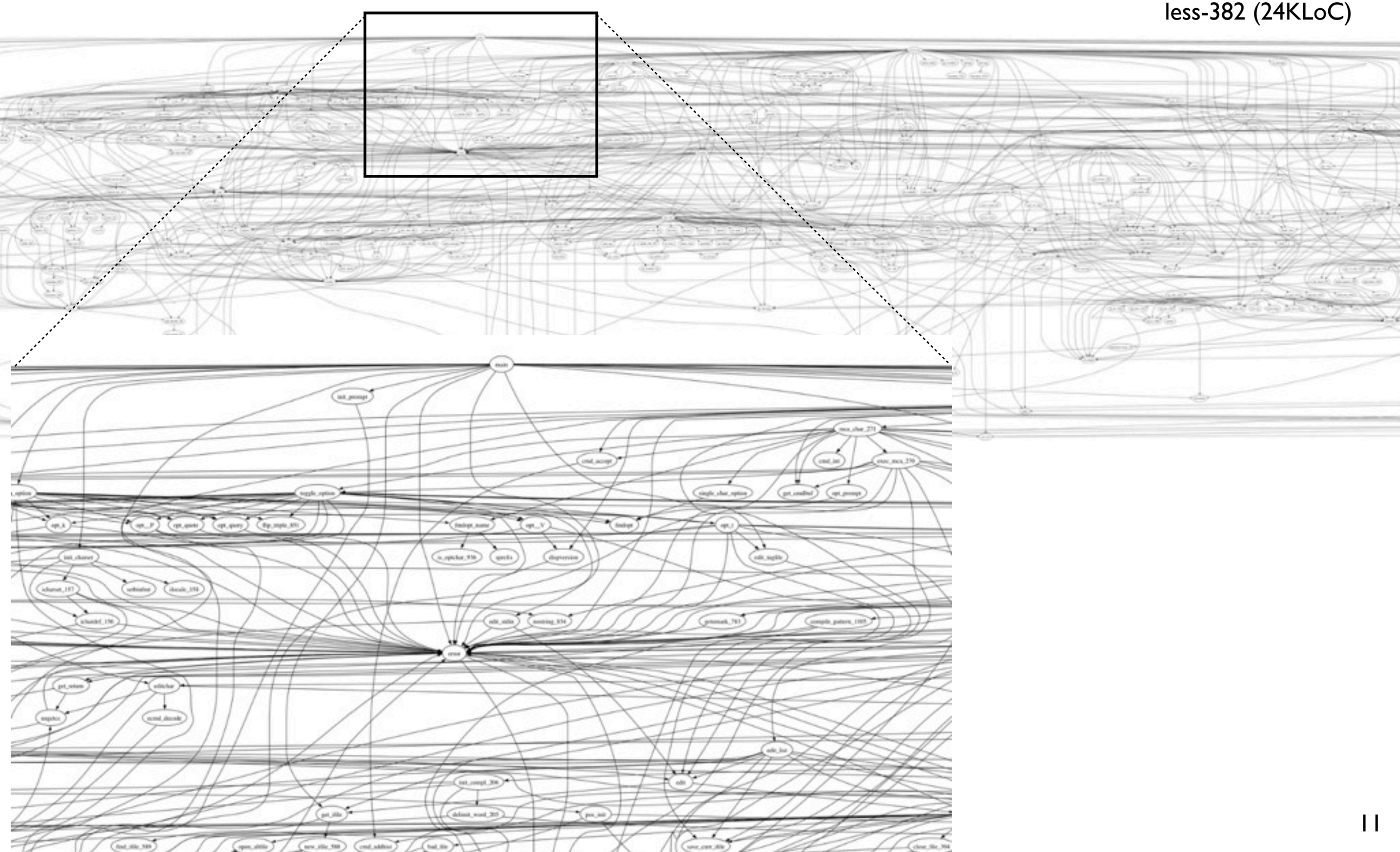
$$\hat{F} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$$
$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c'))$$



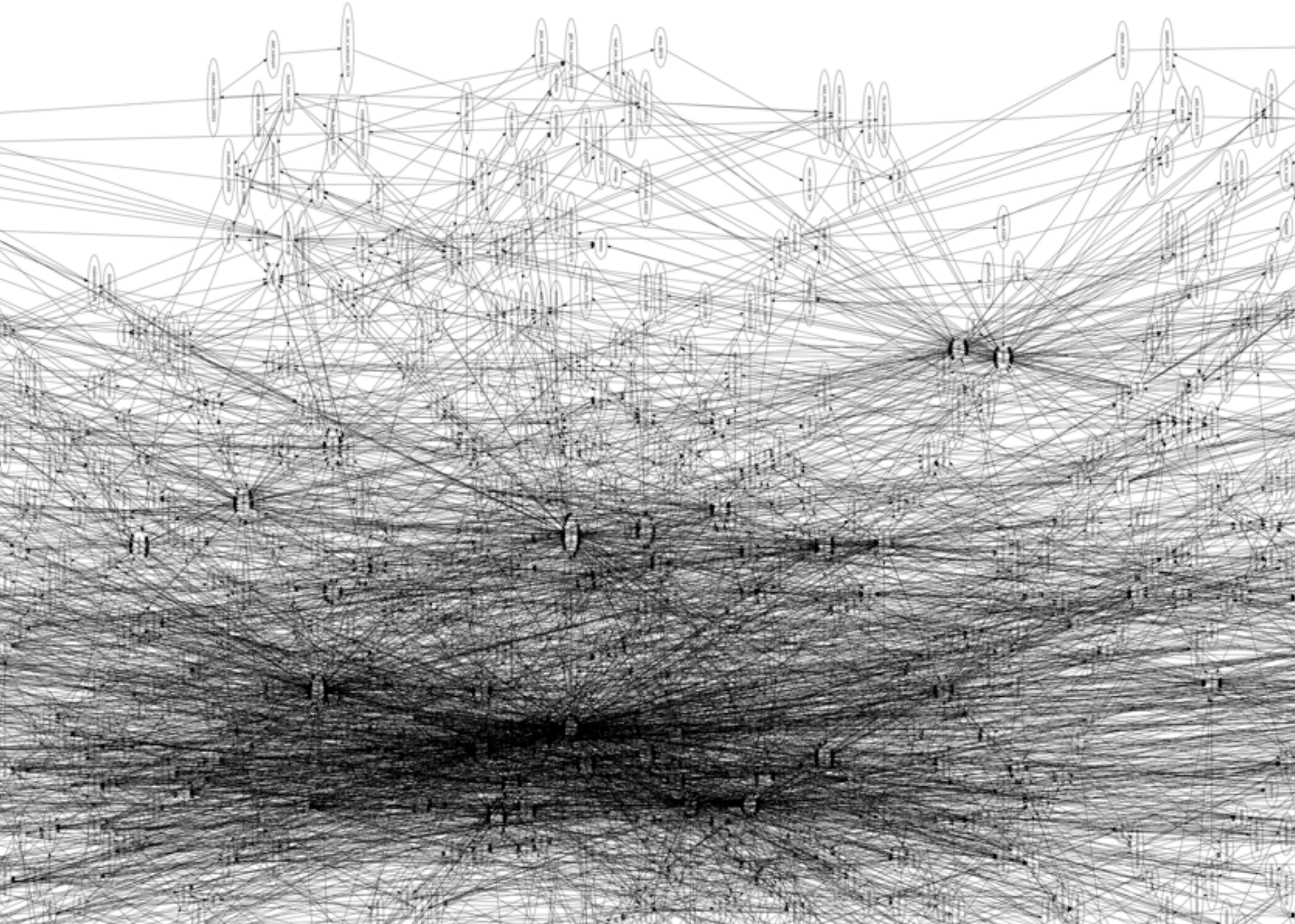
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$: abstract semantics at point c

Direct Implementation (convention) Too Weak To Scale

less-382 (24KLoC)



nethack-3.3.0 (211KLoC)



Towards Sparse Version

Analyzer computes the fixpoint of $\hat{F}_{\_} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$

- baseline non-sparse one

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

Towards Sparse Version

Analyzer computes the fixpoint of $\hat{F}_{\_} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$

- baseline non-sparse one

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

- unrealizable sparse version

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

"data dependency"

Towards Sparse Version

Analyzer computes the fixpoint of $\hat{F}_{\_} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$

- baseline non-sparse one

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

- unrealizable sparse version

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

"data dependency"

- realizable sparse version

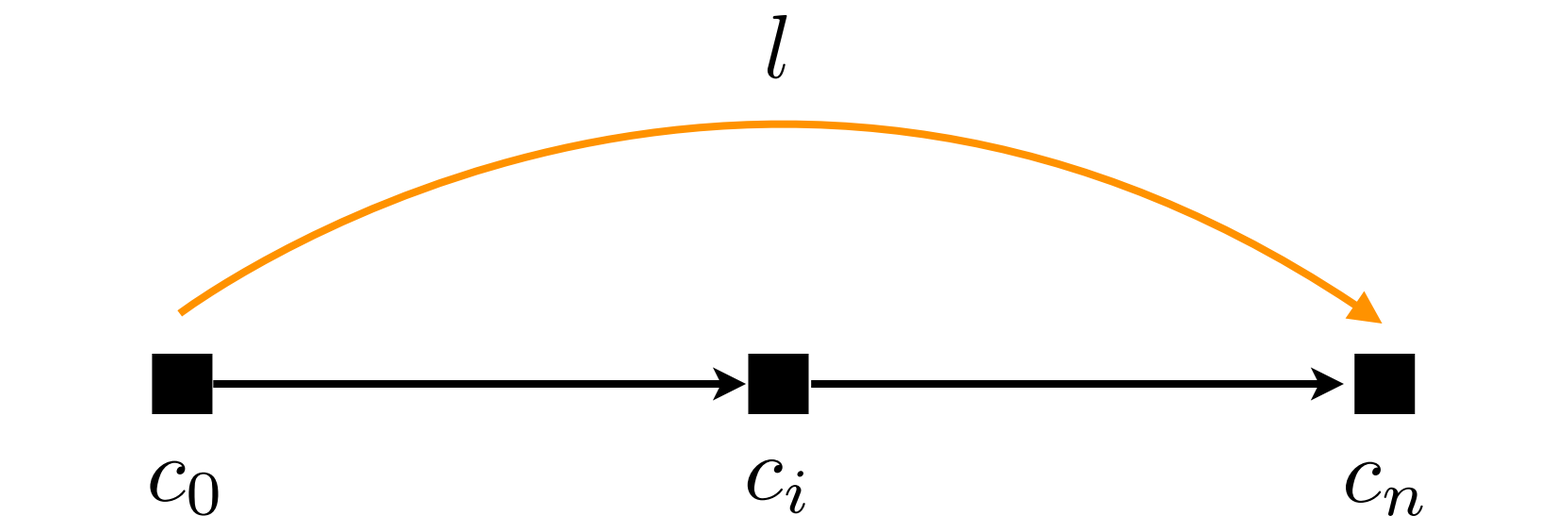
$$\hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \overset{l}{\rightsquigarrow}_a c} \hat{X}(c')|_l).$$

Unrealizable Sparse One

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

Data Dependency

$$c_0 \overset{l}{\rightsquigarrow} c_n \triangleq \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}.$$
$$l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0,n).l \notin \mathsf{D}(c_i)$$



Unrealizable Sparse One

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

Data Dependency

$$c_0 \overset{l}{\rightsquigarrow} c_n \triangleq \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}. \\ l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0,n).l \notin \mathsf{D}(c_i)$$

Def-Use Sets

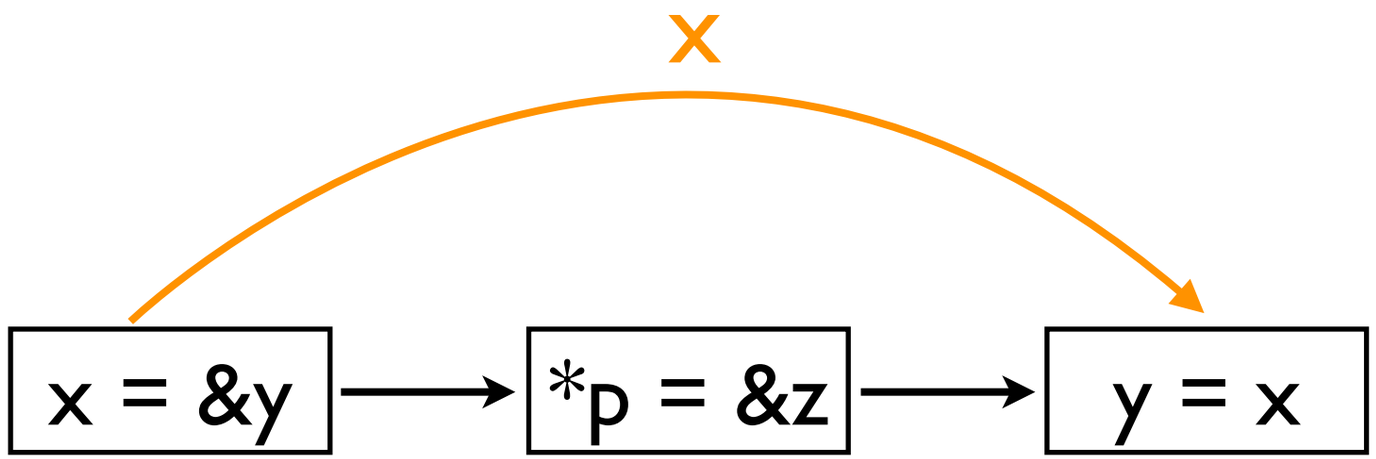
$$\mathsf{D}(c) \triangleq \{l \in \hat{\mathbb{L}} \mid \exists \hat{s} \sqsubseteq \bigsqcup_{c' \hookrightarrow c} (\mathit{fix}\hat{F})(c').\hat{f}_c(\hat{s})(l) \neq \hat{s}(l)\}.$$

$$\mathsf{U}(c) \triangleq \{l \in \hat{\mathbb{L}} \mid \exists \hat{s} \sqsubseteq \bigsqcup_{c' \hookrightarrow c} (\mathit{fix}\hat{F})(c').\hat{f}_c(\hat{s})|_{\mathsf{D}(c)} \neq \hat{f}_c(\hat{s}\backslash_l)|_{\mathsf{D}(c)}\}.$$

Preserving

$$\mathit{fix}\hat{F} = \mathit{fix}\hat{F}_s \quad \text{modulo D}$$

Data Dependency Example



Def	{x}	{a, b}	{y}
Use	ϕ	{p, a, b}	{x}

Realizable Sparse One

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

Realizable Data Dependency

$$c_0 \overset{l}{\rightsquigarrow}_a c_n \triangleq \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}.$$
$$l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0,n).l \notin \mathsf{D}(c_i)$$

Preserving

$$fix\hat{F} \overset{\text{still}}{=} fix\hat{F}_a \quad \text{modulo } \hat{\mathsf{D}}$$

If the following two conditions hold

Conditions of $\hat{\mathsf{D}}$ & $\hat{\mathsf{U}}$

- over-approximation

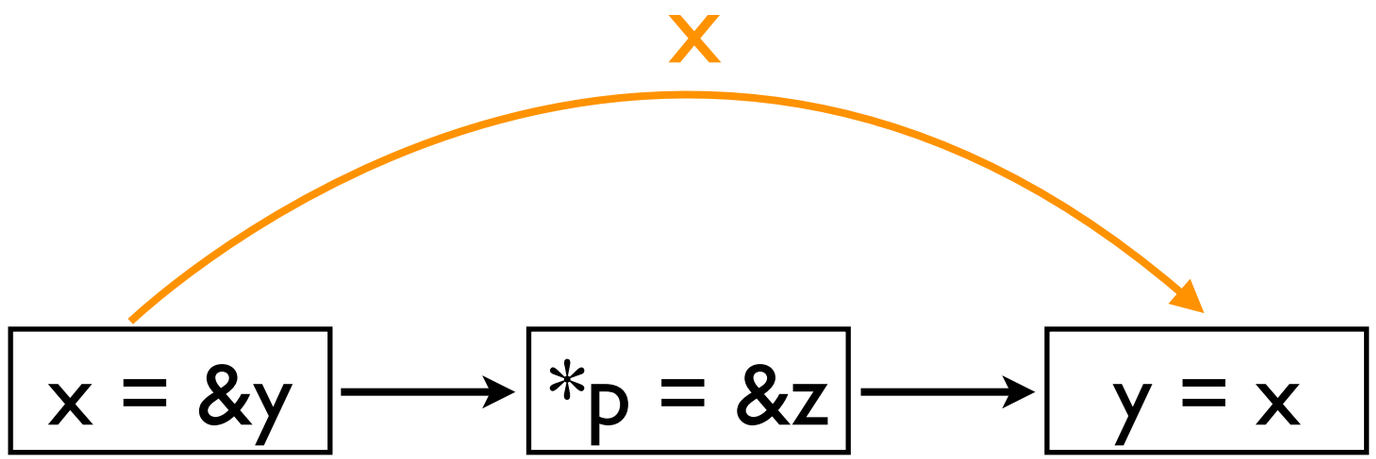
$$\hat{\mathsf{D}}(c) \supseteq \mathsf{D}(c) \wedge \hat{\mathsf{U}}(c) \supseteq \mathsf{U}(c)$$

- spurious definitions should be also included in uses

$$\underline{\hat{\mathsf{D}}(c) - \mathsf{D}(c)} \subseteq \hat{\mathsf{U}}(c)$$

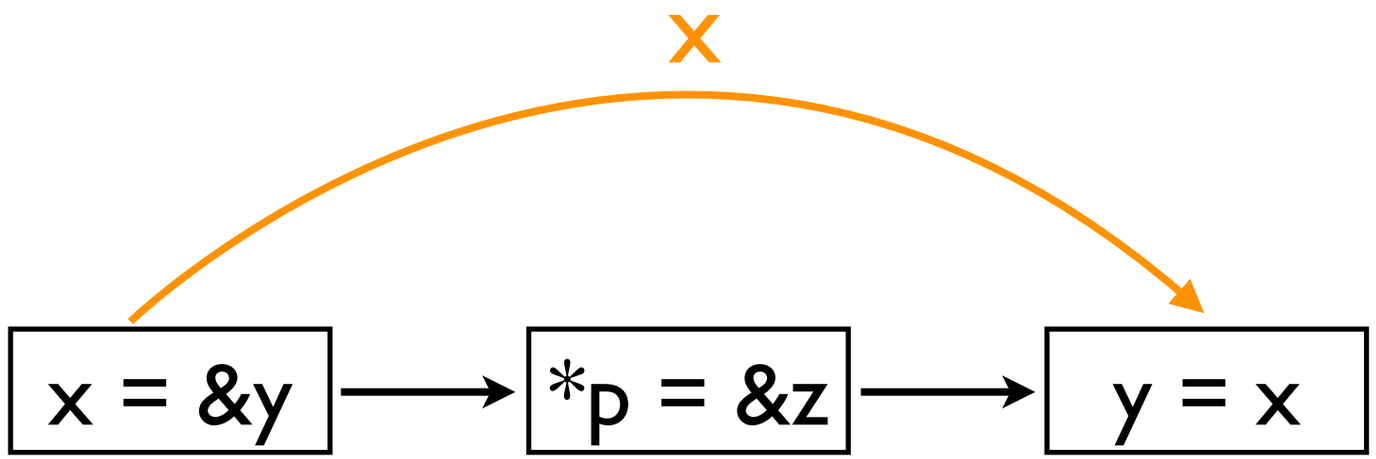
spurious definitions

Why the Conditions of $\hat{D}$ & $\hat{U}$



Def	{x}	{a, b}	{y}
Use	ϕ	{p, a, b}	{x}

Why the Conditions of $\hat{D}$ & $\hat{U}$

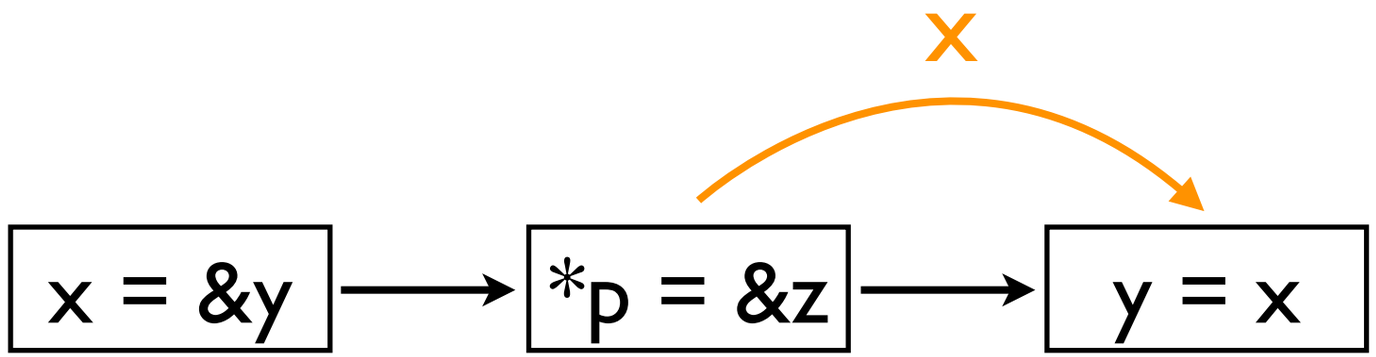


Approx. Def	{x}	{a, b, x}	{y}
Approx. Use	ϕ	{p, a, b}	{x}

$$\underline{\hat{D}(c) - D(c)} \not\subseteq \hat{U}(c)$$

{x}

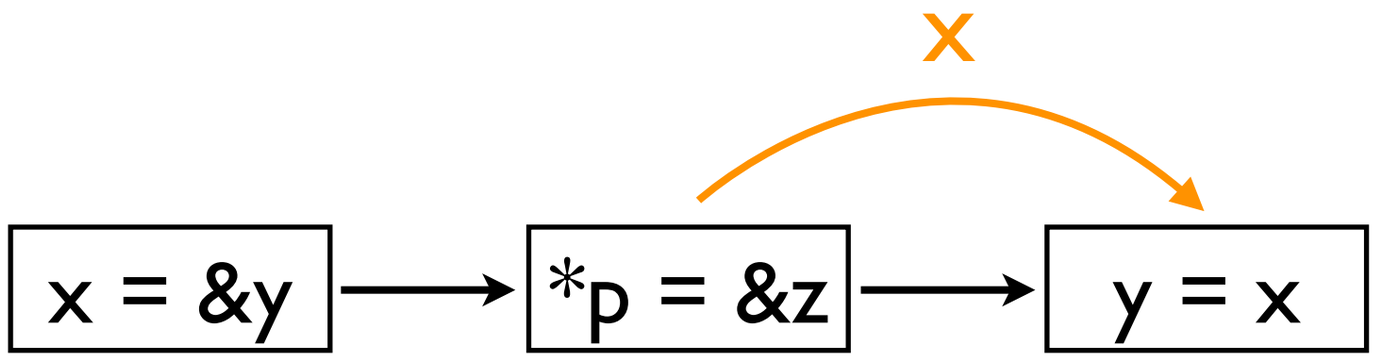
Why the Conditions of $\hat{D}$ & $\hat{U}$



Approx. Def	{x}	{a, b, x}	{y}
Approx. Use	ϕ	{p, a, b}	{x}

$$\frac{\hat{D}(c) - D(c)}{\{x\}} \not\subseteq \hat{U}(c)$$

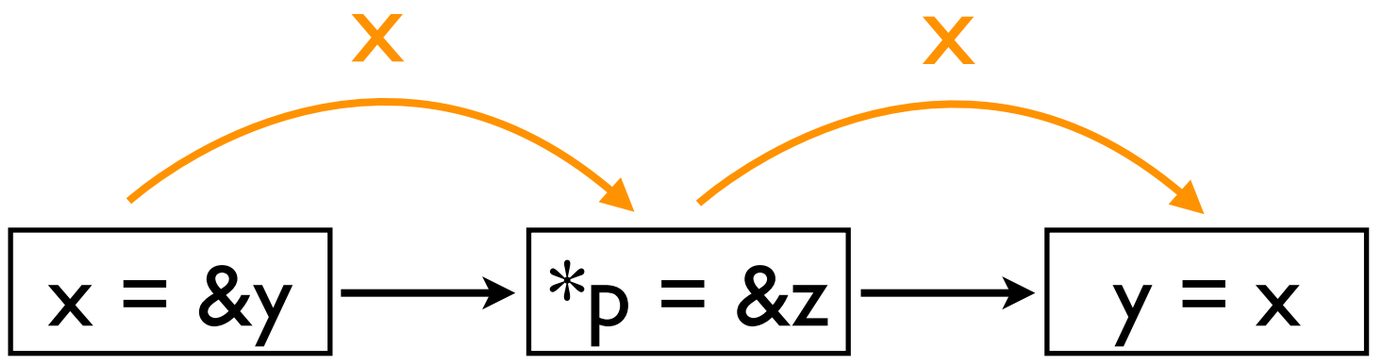
Why the Conditions of $\hat{D}$ & $\hat{U}$



Approx. Def	{x}	{a, b, x}	{y}
Approx. Use	ϕ	{p, a, b, x}	{x}

$$\frac{\hat{D}(c) - D(c) \subseteq \hat{U}(c)}{\{x\}}$$

Why the Conditions of $\hat{D}$ & $\hat{U}$



Approx. Def	{x}	{a, b, x}	{y}
Approx. Use	ϕ	{p, a, b, x}	{x}

$$\frac{\hat{D}(c) - D(c) \subseteq \hat{U}(c)}{\{x\}}$$

Hurdle: $\hat{D}$ & $\hat{U}$ Before Analysis?

- Yes, by yet another analysis with further abstraction
  - e.g., flow-insensitive abstraction

$$\mathbb{C} \to \hat{\mathbb{S}} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \hat{\mathbb{S}} \qquad \hat{F}_p = \lambda \hat{s}.(\bigsqcup_{c \in \mathbb{C}} \hat{f}_c(\hat{s}))$$

- In implementation, $\hat{U}$ includes $\hat{D}$

$$\hat{\mathsf{D}}(c) - \mathsf{D}(c) \subseteq \hat{\mathsf{U}}(c)$$

For More Details

See the extended version of our PLDI paper:

ropas.snu.ac.kr/~pronto/sparse.pdf

- details and full correctness proofs
- descriptions in more general setting
  - various languages (ftnl, oo, etc)
  - arbitrary trace partitioning (ctx-sens, path-sens, etc)

Experiments

- On top of Sparrow The Early Bird
  - Sparse non-relational analysis with interval domain

$$\hat{\mathbb{S}} = AbsLoc \rightarrow Interval$$

- Sparse relational analysis with octagon domain

$$\hat{\mathbb{S}} = Packs \rightarrow Octagon$$

Benchmarks

GNU open-source C 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
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

Interval & Pointer Analysis

Programs	LOC	Interval$_{vanilla}$		Interval$_{base}$		Spd↑$_1$	Mem↓$_1$	Interval$_{sparse}$						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

access-based localization

sparse analysis

Octagon & Pointer Analysis

Programs	$\text{Octagon}_{\text{vanilla}}$		$\text{Octagon}_{\text{base}}$		$\text{Spd}\uparrow_1$	$\text{Mem}\downarrow_1$	$\text{Octagon}_{\text{sparse}}$						$\text{Spd}\uparrow_2$	$\text{Mem}\downarrow_2$
	Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	2,078	2,832	273	1,072	8 x	62 %	7	14	21	269	13.8	14.5	13 x	75 %
bc-1.06	9,536	6,987	1,065	3,230	9 x	54 %	20	35	55	358	25.2	31.7	19 x	89 %
tar-1.13	∞	N/A	9,566	5,963	N/A	N/A	55	133	188	526	38.3	39.3	51 x	91 %
less-382	∞	N/A	16,121	8,410	N/A	N/A	92	340	432	458	42.6	45.4	37 x	95 %
make-3.76.1	∞	N/A	17,724	12,771	N/A	N/A	91	240	331	666	51.4	55.7	53 x	95 %
wget-1.9	∞	N/A	15,998	9,363	N/A	N/A	107	181	288	646	31.9	32.9	56 x	93 %
screen-4.0.2	∞	N/A	∞	N/A	N/A	N/A	2,452	13,981	16,433	9,199	372.4	376.1	N/A	N/A
a2ps-4.14	∞	N/A	∞	N/A	N/A	N/A	296	8,271	8,566	1,996	97.7	99.0	N/A	N/A
sendmail-8.13.6	∞	N/A	∞	N/A	N/A	N/A	7,256	57,552	64,808	29,658	467.6	492.3	N/A	N/A

none

access-based localization

sparse analysis

Summary

Our Sparse Framework

A recipe for precise, sound, and scalable static analysis

- Define a global safe abstract interpreter
- Make it sparse with our framework
- Resulting sparse one scales with the same precision

Thank you

Backup Slides

Data Dependency vs. Def-Use Chains

- Different notion of data dependency

$$c_0 \overset{l}{\rightsquigarrow}_{\mathsf{du}} c_n \triangleq \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}.$$
$$l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0,n).l \notin \mathsf{D}_{\mathsf{always}}(c_i)$$

- fail to preserve the original accuracy

Existing Sparse Techniques
(developed mostly in dfa/pointer-analysis community)

- Fine-grained sparse techniques in particular settings
  - e.g. sparse pointer analysis algorithms
    - tightly coupled with particular analyses
- Coarse-grained sparse techniques in general settings
  - “sparse evaluation”
  - too weak to be useful for detailed analyses

Ours: Fine-grained sparse analysis in general setting