Design and Implementation of Sparse Global Analyses for C-like Languages

Hakjoo Oh, Kihong Heo, Wonchan Lee, Woosuk Lee, and Kwangkeun Yi

Programming Research Laboratory
Seoul National University

PLDI 2012 @ Beijing, China
Dichotomy in Static Analysis

- Scalable
  - Bug-finders
- Unscalable
  - Verifiers

- Unsound
- Sound
Dichotomy in Static Analysis

![Diagram showing the dichotomy between scalable and unsound bug-finders on one axis, and scalable and unsound verifiers on another axis. The question mark indicates an area of uncertainty or exploration.](image-url)
Dichotomy in Static Analysis

scalable

unsound

unsound

bug-finders

sound

verifiers

our contribution

?
Our Story

- In 2007, we commercialized Sparrow
  - memory-bug-finding tool for full C, non domain-specific
  - designed in abstract interpretation framework
  - sound in design, unsound yet scalable in reality

- Realistic workbench available
  - “let’s try to scale-up its sound & global analysis version”
Scalability Improvement

sound & global analysis version

- < 1.4M in 10hr with intervals
- < 0.14M in 20hrs with octagons
Precision-Preserving Sparse Analysis Framework

baseline analysis

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

sparsify

“sparse” version

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

$$\text{fix } \hat{F} \quad = \quad \text{still } \quad \text{fix } \hat{F}_s$$

General for AI-based analyzers for C-like languages
Sparse Analysis Framework

- “Right Part at Right Moment”
- “Full Exploitation”
- enabled by Abstract Interpretation theory
Program

\[ \langle C, \rightarrow \rangle \]

- \( C \): set of program points
- \( \rightarrow \subseteq C \times C \): control flow relation

\[ c' \rightarrow c \quad (c \text{ is the next program point to } c') \]
Baseline Analysis

• One abstract state $\in \hat{\mathcal{S}}$ that subsumes all reachable states at each program point

$$[\hat{P}] \in \mathcal{C} \rightarrow \hat{\mathcal{S}} = \text{fix} \hat{F}$$

$$\hat{\mathcal{S}} = \hat{\mathcal{L}} \rightarrow \hat{\mathcal{V}}$$

• Abstract semantic function

$$\hat{F} \in (\mathcal{C} \rightarrow \hat{\mathcal{S}}) \rightarrow (\mathcal{C} \rightarrow \hat{\mathcal{S}})$$

$$\hat{F}(\hat{X}) = \lambda c \in \mathcal{C}. \hat{f}_c(\bigcup_{c' \leftarrow c} \hat{X}(c'))$$

$$\hat{f}_c \in \hat{\mathcal{S}} \rightarrow \hat{\mathcal{S}} : \text{abstract semantics at point } c$$
Direct Implementation (convention)
Too Weak To Scale

less-382 (23,822 LoC)
“Sparsifying” the Analysis

“Right Part at Right Moment”

\[
\begin{align*}
x &= x + 1 \\
y &= y - 1 \\
z &= x \\
v &= y \\
\text{ret} &= *a + *b
\end{align*}
\]
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”

\[
\begin{align*}
x &= x + 1 \\
y &= y - 1 \\
z &= x \\
v &= y \\
\text{ret} &= \ast a + \ast b
\end{align*}
\]
“Sparsifying” the Analysis

“Right Part at Right Moment”

\[
\begin{align*}
x &= x + 1 \\
y &= y - 1 \\
z &= x \\
v &= y \\
\text{ret} &= *a + *b
\end{align*}
\]
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”

\[
x = x + 1
\]

\[
y = y - 1
\]

\[
z = x
\]

\[
v = y
\]

\[
\text{ret} \ast a + \ast b
\]
“Sparsifying” the Analysis

“Right Part at Right Moment”

\[ x = x + 1 \]
\[ y = y - 1 \]
\[ z = x \]
\[ v = y \]
\[ \text{ret } a + b \]
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”

```plaintext
x = x+1
y = y-1
z = x
v = y
ret *a+*b
```
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”
"Sparsifying" the Analysis

"Right Part at Right Moment"
"Sparsifying" the Analysis

"Right Part at Right Moment"

x = x+1

y = y-1

z = x

v = y

ret *a+*b
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”

```
x = x + 1
y = y - 1
z = x
v = y
ret a+b
```
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”
“Sparsifying” the Analysis

“Right Part at Right Moment”
"Sparsifying" the Analysis

"Right Part at Right Moment"

\[
\hat{F}(\hat{X}) = \lambda c \in C. \hat{f}_c(\underbrace{\hat{X}(c')}_{c' \mapsto c}).
\]

replace syntactic dependency by semantic dependency (data dependency)

```
x = x+1
y = y-1
z = x
v = y
ret *a+*b
```
Towards Sparse Version

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

- baseline non-sparse one
  \[
  \hat{F}(\hat{X}) = \lambda c \in \mathbb{C}. f_c( \bigsqcup_{c' \leftarrow c} \hat{X}(c')) .
  \]

- unrealizable sparse version
  \[
  \hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}. f_c( \bigsqcup_{c' \leftarrow c} \hat{X}(c')|_l) .
  \]

- realizable sparse version
  \[
  \hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}. f_c( \bigsqcup_{c' \leftarrow a,c} \hat{X}(c')|_l) .
  \]
Unrealizable Sparse One

\[ \hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}. \hat{f}_c( \bigsqcup_{c' \leadsto c} \hat{X}(c')|_l). \]

Data Dependency

\[ c_0 \xrightarrow{l} c_n \triangleq \exists c_0 \ldots c_n \in \text{Paths}, l \in \hat{L}. \]
\[ l \in D(c_0) \cap U(c_n) \land \forall i \in (0, n). l \notin D(c_i) \]

\[ l \in D(c_0) \quad l \notin D(c_i) \quad l \in U(c_n) \]
Unrealizable Sparse One

\[ \hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}. \hat{f}_c(\bigsqcup_{c' \sim c} \hat{X}(c')|_l). \]

**Data Dependency**

\[ c_0 \xrightarrow{l} c_n \triangleq \exists c_0 \ldots c_n \in \text{Paths}, l \in \hat{L}. \]

\[ l \in D(c_0) \cap U(c_n) \land \forall i \in (0, n). l \notin D(c_i) \]

**Def-Use Sets**

\[ D(c) \triangleq \{ l \in \hat{L} | \exists \hat{s} \subseteq \bigsqcup_{c' \sim c} (\text{fix } \hat{F}')(c'). \hat{f}_c(\hat{s})(l) \neq \hat{s}(l) \}. \]

\[ U(c) \triangleq \{ l \in \hat{L} | \exists \hat{s} \subseteq \bigsqcup_{c' \sim c} (\text{fix } \hat{F}')(c'). \hat{f}_c(\hat{s})|_{D(c)} \neq \hat{f}_c(\hat{s}\backslash l)|_{D(c)} \}. \]

**Preserving**

\[ \text{fix } \hat{F} = \text{fix } \hat{F}_s \quad \text{modulo } D \]
Data Dependency Example

\[ x = \&y \]
\[ \ast p = \&z \]
\[ y = x \]

Def: \{x\} \quad \{a, b\} \quad \{y\}
Use: \emptyset \quad \{p, a, b\} \quad \{x\}
Realizable Sparse One

\[ \hat{F}_a(X) = \lambda c \in \mathbb{C}. \hat{f}_c( \bigcup_{c' \sim_a c} \hat{X}(c')|_l). \]

Realizable Data Dependency

\[ c_0 \overset{l}{\sim_a} c_n \triangleq \exists c_0 \ldots c_n \in \text{Paths}, l \in \hat{l}. \]
\[ \text{ } l \in \hat{D}(c_0) \cap \hat{U}(c_n) \land \forall i \in (0, n). l \notin \hat{D}(c_i) \]
Realizable Sparse One

\[ \hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}. \hat{f}_c( \bigsqcup_{c' \sim a c} \hat{X}(c'))|_l). \]

Realizable Data Dependency

\[ c_0 \overset{l}{\rightarrow}_a c_n \triangleq \exists c_0 \ldots c_n \in \text{Paths}, l \in \hat{\mathbb{L}}. \]
\[ l \in \hat{D}(c_0) \cap \hat{U}(c_n) \land \forall i \in (0, n). l \not\in \hat{D}(c_i) \]

Preserving

\[ \text{fix } \hat{F} = \text{fix } \hat{F}_a \quad \text{modulo } \hat{D} \]

If the following two conditions hold
Conditions of $\hat{D} & \hat{U}$

- over-approximation

$$\hat{D}(c) \supseteq D(c) \land \hat{U}(c) \supseteq U(c)$$

- spurious definitions should be also included in uses

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

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

Def  \{x\}       \{a, b\}       \{y\}

Use  $\phi$      \{p, a, b\}     \{x\}

\[
\begin{align*}
x &= & y \\
*p &= & z \\
y &= & x
\end{align*}
\]
Why the Conditions of $\hat{D}$ & $\hat{U}$

Approx. Def

$x = &y$

{a, b, x}  

{a, b, x}  

{y}

Approx. Use

$\phi$

{p, a, b}

{x}

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

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

$x = \&y \rightarrow \star p = \&z \rightarrow y = x$

Approx. Def  \{x\} \{a, b, x\} \{y\}

Approx. Use  $\emptyset$ \{p, a, b\} \{x\}

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

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

Approx. Def
- $\{x\}$
- $\{a, b, x\}$
- $\{y\}$

Approx. Use
- $\emptyset$
- $\{p, a, b, x\}$
- $\{x\}$

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

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

Approx. Def $\{x\} \rightarrow \{a, b, x\}$ \rightarrow \{y\}

Approx. Use $\phi \rightarrow \{p, a, b, x\}$ \rightarrow \{x\}

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

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

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

\[ C \rightarrow \hat{S} \xleftarrow{\gamma} \hat{S} \xrightarrow{\alpha} \hat{S} \quad \hat{F}_p = \lambda \hat{s}. ( \bigcup_{c \in C} \hat{f}_c(\hat{s}) ) \]

- In implementation, \( \hat{U} \) includes \( \hat{D} \)

\[ \hat{D}(c) - D(c) \subseteq \hat{U}(c) \]
Existing Sparse Techniques
(developed mostly in dfa community)

- Different notion of data dependency

\[ x \in D \rightarrow x \in D_{\text{may}} \rightarrow x \in U \]

def-use chains fail to preserve original precision
Existing Sparse Techniques
(developed mostly in dfa community)

• Different notion of data dependency

\[ x \in D \rightarrow x \in D_{\text{may}} \rightarrow x \in D \]

our data dependency preserves original precision
Existing Sparse Techniques
(developed mostly in dfa community)

- Different notion of data dependency

\[ x \in D \rightarrow x \in D_{may} \rightarrow x \in D \]

- Existing sparse analyses are not general
  - tightly coupled with particular analysis, or
  - limited to a particular target language
Performance
Experiments

• On top of \textit{Sparrow}

  • \textbf{Sparse non-relational analysis} with interval domain
    $$\hat{\mathcal{S}} = \text{AbsLoc} \rightarrow \text{Interval}$$

  • \textbf{Sparse relational analysis} with octagon domain
    $$\hat{\mathcal{S}} = \text{Packs} \rightarrow \text{Octagon}$$
## Performance

**Sparse Interval Analysis**

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

Spd↑ is the speedup of Airac Local over Airac Base.
Mem↓ shows the memory savings of Airac Local over Airac Base.

All experiments were done on a Linux system running on a single core of Intel GHz box with GB of main memory.
## Performance

**Sparse Interval Analysis**

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Non-sparse</th>
<th>Sparse</th>
<th>Spd↑</th>
<th>Mem↓</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Mem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzip-1.2.4a</td>
<td>7 K</td>
<td>772</td>
<td>240</td>
<td>3</td>
<td>63</td>
</tr>
<tr>
<td>bc-1.06</td>
<td>13 K</td>
<td>1,270</td>
<td>276</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>less-382</td>
<td>23 K</td>
<td>9,561</td>
<td>1,113</td>
<td>33</td>
<td>127</td>
</tr>
<tr>
<td>make-3.76.1</td>
<td>27 K</td>
<td>24,240</td>
<td>1,391</td>
<td>21</td>
<td>114</td>
</tr>
<tr>
<td>wget-1.9</td>
<td>35 K</td>
<td>44,092</td>
<td>2,546</td>
<td>11</td>
<td>85</td>
</tr>
<tr>
<td>a2ps-4.14</td>
<td>64 K</td>
<td>∞</td>
<td>N/A</td>
<td>40</td>
<td>353</td>
</tr>
<tr>
<td>sendmail-8.13.6</td>
<td>130 K</td>
<td>∞</td>
<td>N/A</td>
<td>744</td>
<td>678</td>
</tr>
<tr>
<td>nethack-3.3.0</td>
<td>211 K</td>
<td>∞</td>
<td>N/A</td>
<td>16,373</td>
<td>5,298</td>
</tr>
<tr>
<td>emacs-22.1</td>
<td>399 K</td>
<td>∞</td>
<td>N/A</td>
<td>37,830</td>
<td>7,795</td>
</tr>
<tr>
<td>python-2.5.1</td>
<td>435 K</td>
<td>∞</td>
<td>N/A</td>
<td>11,039</td>
<td>5,535</td>
</tr>
<tr>
<td>linux-3.0</td>
<td>710 K</td>
<td>∞</td>
<td>N/A</td>
<td>33,618</td>
<td>20,529</td>
</tr>
<tr>
<td>gimp-2.6</td>
<td>959 K</td>
<td>∞</td>
<td>N/A</td>
<td>3,874</td>
<td>3,602</td>
</tr>
<tr>
<td>ghostscript-9.00</td>
<td>1,363 K</td>
<td>∞</td>
<td>N/A</td>
<td>14,814</td>
<td>6,384</td>
</tr>
</tbody>
</table>

Spd↑ means the speedup of **Airac Local** over **Airac Base**. Mem↓ shows the memory savings of **Airac Local** over **Airac Base**. All experiments were done on a Linux system running on a single core of Intel GHz box with GB of main memory.
## Performance

### Sparse Octagon Analysis

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Non-sparse</th>
<th>Sparse</th>
<th>Spd↑</th>
<th>Mem↓</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Mem</td>
<td>Time</td>
<td>Mem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzip-1.2.4a</td>
<td>7 K</td>
<td>2,078</td>
<td>2,832</td>
<td>21</td>
<td>269</td>
</tr>
<tr>
<td></td>
<td></td>
<td>98 x</td>
<td>91 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bc-1.06</td>
<td>13 K</td>
<td>9,536</td>
<td>6,987</td>
<td>55</td>
<td>358</td>
</tr>
<tr>
<td></td>
<td></td>
<td>173 x</td>
<td>95 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tar-1.13</td>
<td>20 K</td>
<td>∞</td>
<td>N/A</td>
<td>188</td>
<td>526</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>less-382</td>
<td>23 K</td>
<td>∞</td>
<td>N/A</td>
<td>432</td>
<td>458</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>make-3.76.1</td>
<td>27 K</td>
<td>∞</td>
<td>N/A</td>
<td>331</td>
<td>666</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget-1.9</td>
<td>35 K</td>
<td>∞</td>
<td>N/A</td>
<td>288</td>
<td>646</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>screen-4.0.2</td>
<td>45 K</td>
<td>∞</td>
<td>N/A</td>
<td>16,433</td>
<td>9,199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a2ps-4.14</td>
<td>64 K</td>
<td>∞</td>
<td>N/A</td>
<td>8,546</td>
<td>1,996</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sendmail-8.13.6</td>
<td>130 K</td>
<td>∞</td>
<td>N/A</td>
<td>64,808</td>
<td>29,658</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

Our Sparse Framework

For precise, sound, and scalable static analysis

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

Thank you