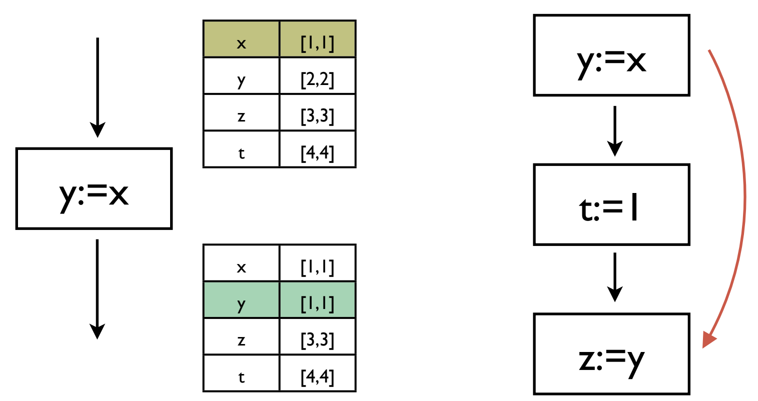
오학주, 허기홍, 이원찬, 이우석, 이광근 (서울대학교 프로그래밍 연구실)

Global analysis of million lines of code

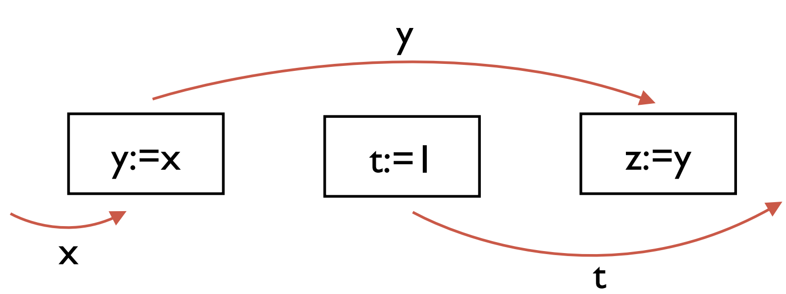
(Sparse Analysis Framework for C-like Languages)

1 Key Idea: Sparse Analysis

In global analysis of imperative programs, the analysis is typically sparse in space and time.



Sparse analysis directly follows the actual semantic dependences.



2 Sparse Analysis Framework

Setting $\mathbb{C} \to 2^{\mathbb{S}} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{C} \to \hat{\mathbb{S}} \qquad \hat{F}(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')).$

Def/Use

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

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

Safe Approximation

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

Data Dependency

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

Sparse Abstract Semantic Function

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

Theorem (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,*

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

3 Performance

Programs	LOC	$\text{Interval}_{\text{vanilla}}$		$\text{Interval}_{\text{base}}$		$\text{Spd}\uparrow_1$	$\text{Mem}\downarrow_1$	$\text{Interval}_{\text{sparse}}$						$\text{Spd}\uparrow_2$	$\text{Mem}\downarrow_2$
		Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{\mathsf{D}}(c)$	$\hat{\mathsf{U}}(c)$		
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

Table 3: Performance of interval analysis: time (in seconds) and peak memory consumption (in megabytes) of the various versions of analyses. ∞ means the analysis ran out of time (exceeded 24 hour time limit). **Dep** and **Fix** reports the time spent during data dependency analysis and actual analysis steps, respectively, of the sparse analysis. $\textbf{Spd}\uparrow_1$ is the speed-up of $\text{Interval}_{\text{base}}$ over $\text{Interval}_{\text{vanilla}}$. $\textbf{Mem}\downarrow_1$ shows the memory savings of $\text{Interval}_{\text{base}}$ over $\text{Interval}_{\text{vanilla}}$. $\textbf{Spd}\uparrow_2$ is the speed-up of $\text{Interval}_{\text{sparse}}$ over $\text{Interval}_{\text{base}}$. $\textbf{Mem}\downarrow_2$ shows the memory savings of $\text{Interval}_{\text{sparse}}$ over $\text{Interval}_{\text{base}}$. $\hat{\mathsf{D}}(c)$ and $\hat{\mathsf{U}}(c)$ show the average size of $\hat{\mathsf{D}}(c)$ and $\hat{\mathsf{U}}(c)$, respectively.

Programming Research Laboratory

SEOUL NATIONAL UNIVERSITY