Efficient Alias Set Analysis Using SSA Form

Ondřej Lhoták Nomair Naeem

University of Waterloo

Aliases and SSA Form

How can aliases be represented in SSA form?

Not this talk.

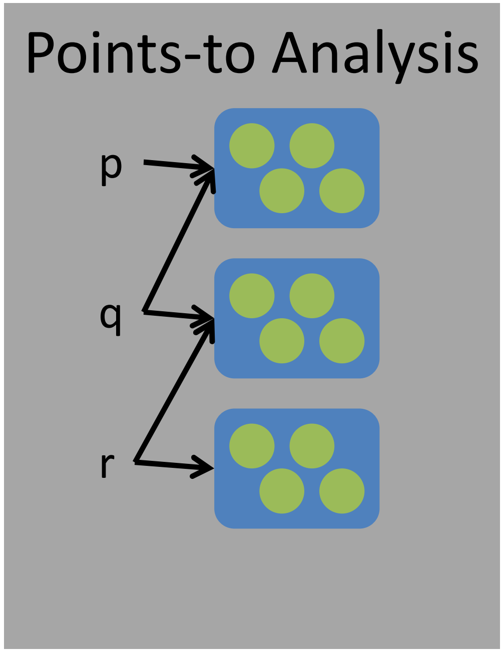
How can SSA form help alias analysis?

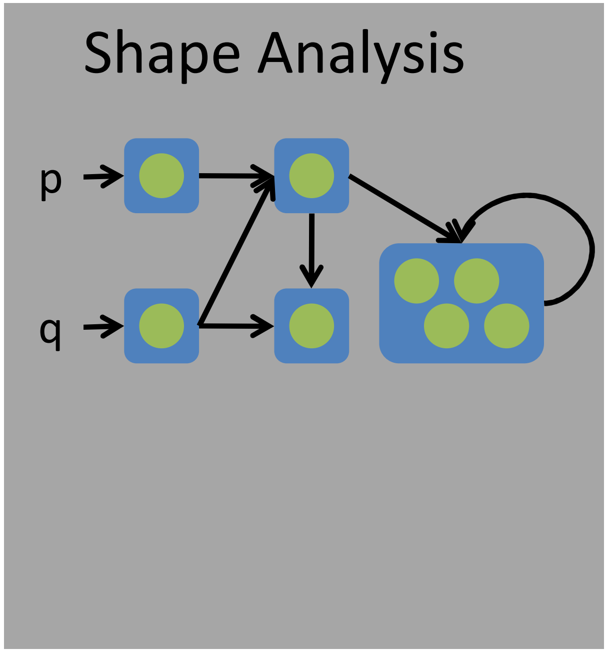
This talk.

Observe some interesting SSA properties along the way…

Range of Pointer Analyses

Legend:  Concrete (run-time) object
 Abstract (static) object

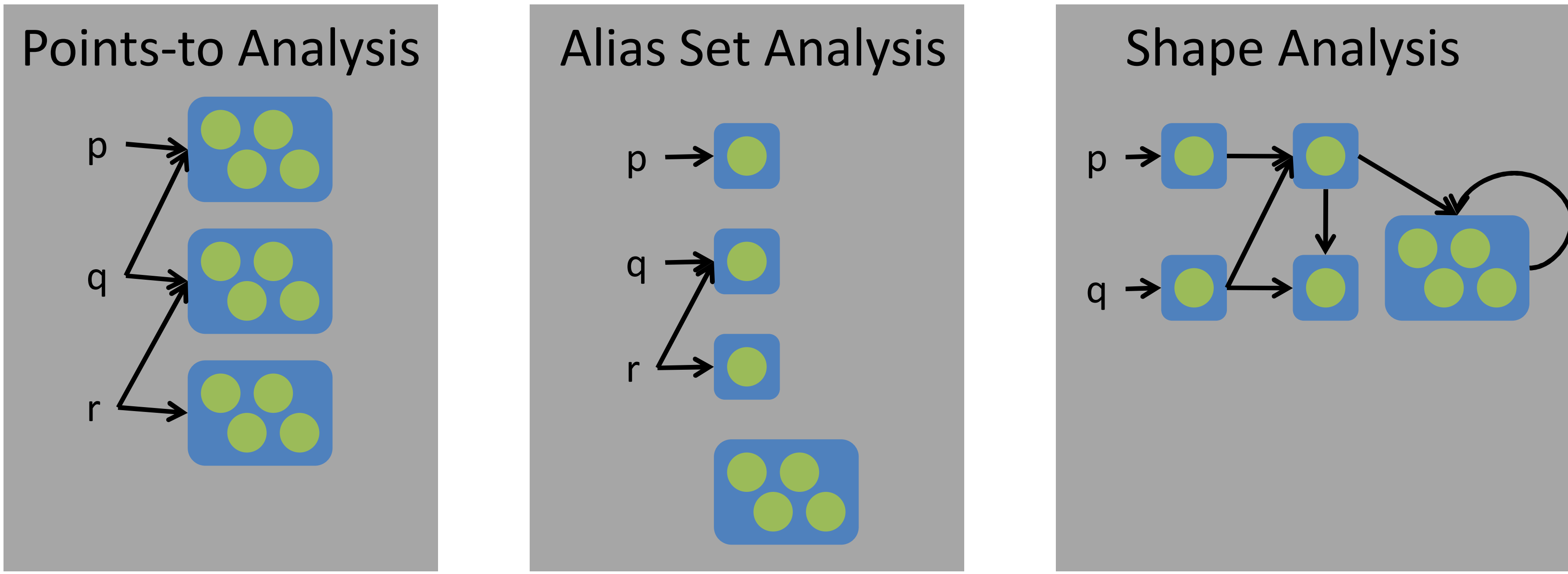




Efficient analysis  Precise analysis

Range of Pointer Analyses

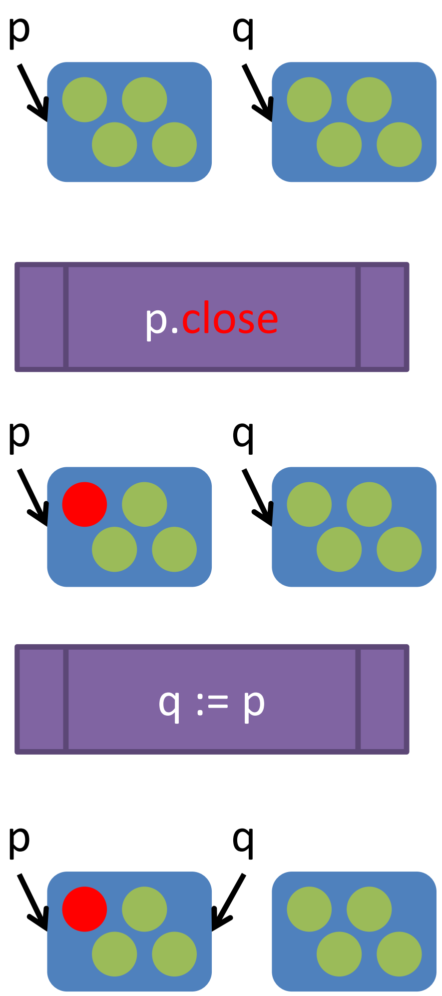
Legend:
Concrete (run-time) object
Abstract (static) object

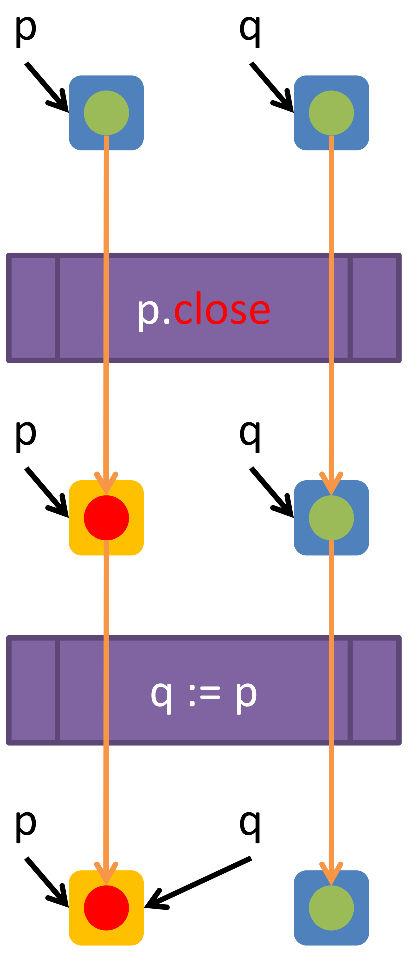


Efficient analysis

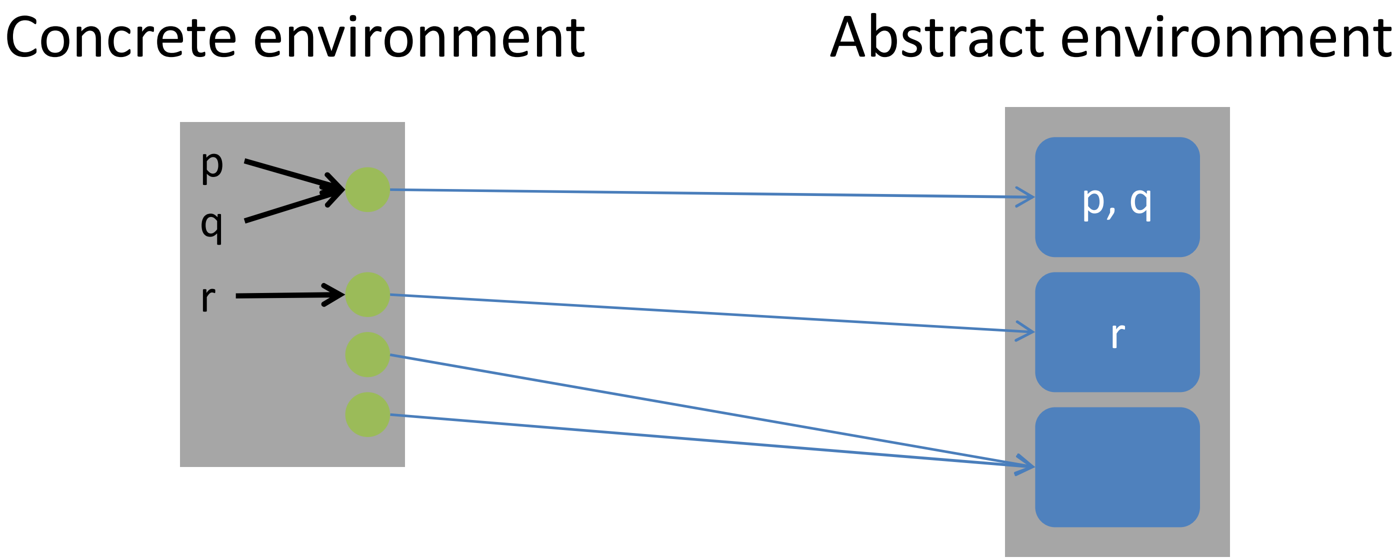
Precise analysis

Why Alias Sets: Object Tracking





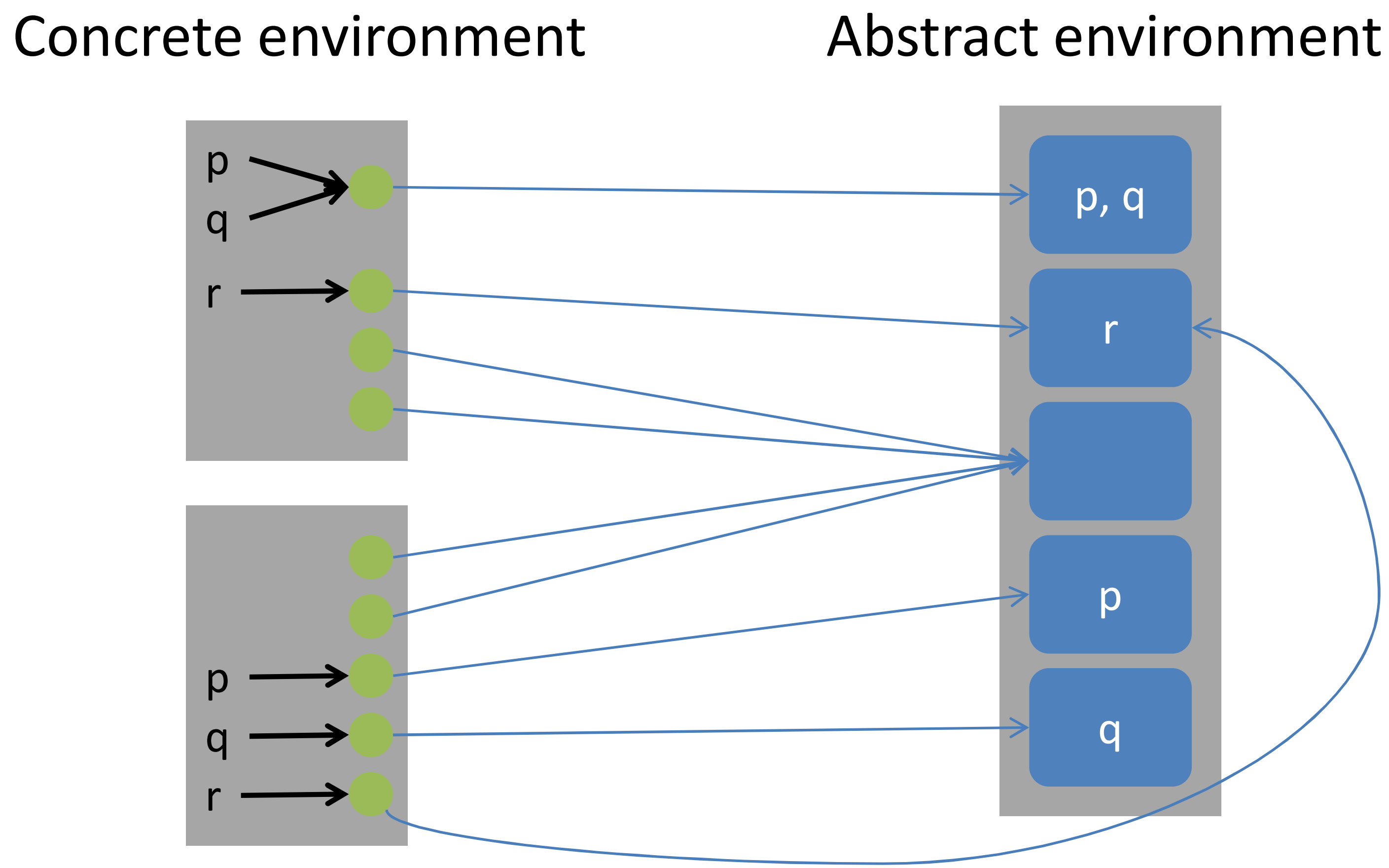
The Alias Set Abstraction



Each element of the abstract domain is a set of abstract objects.
Each abstract object is a set of pointer variables.

represents the object (if any) pointed by p and q and no other local variables.

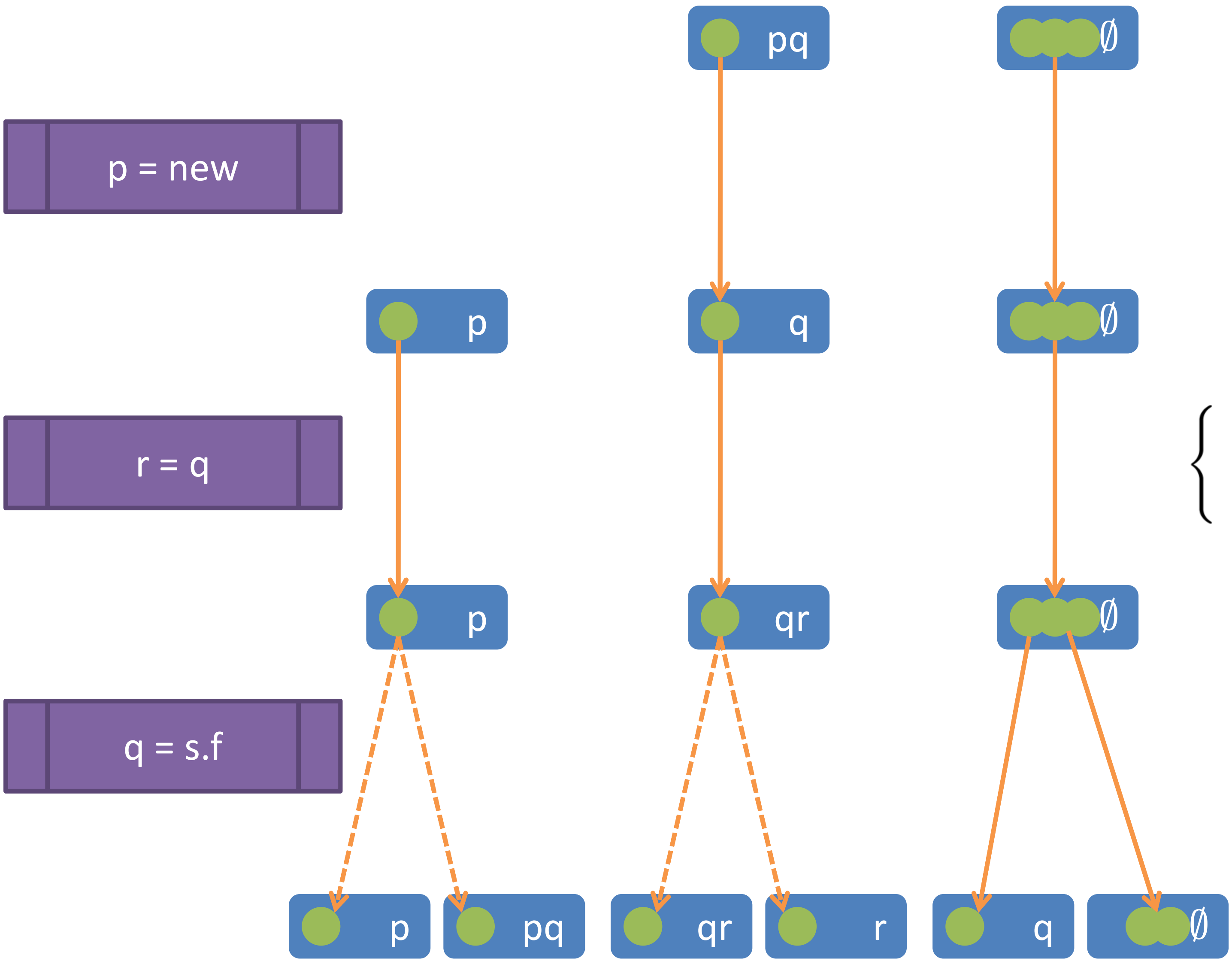
The Alias Set Abstraction



Each element of the abstract domain is a set of abstract objects.
Each abstract object is a set of pointer variables.

represents the object (if any) pointed by p and q and no other local variables.

Transfer Functions



$$\{p\}, o^\sharp \setminus \{p\}$$

$$\begin{cases} o^\sharp \setminus \{r\} & \text{if } q \notin o^\sharp \\ o^\sharp \cup \{r\} & \text{if } q \in o^\sharp \end{cases}$$

$$o^\sharp \setminus \{q\}, o^\sharp \cup \{q\}$$

Benefits of SSA Form for Alias Set Analysis

IF

- Convert code to SSA form
- Represent each alias set by sorted list, ordered by dominance of (unique) definitions

THEN

☺ All inserts into set are at head of list

☺ All removals from set are at head of list

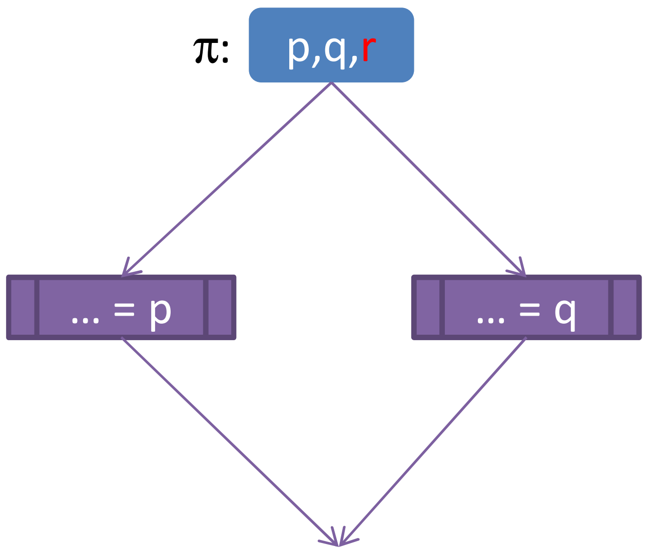
☺ All removals are at ϕ nodes

☺ Tails of lists can be shared (hash consing)

Filtering by Liveness

- Since r is not live at π, it is irrelevant.

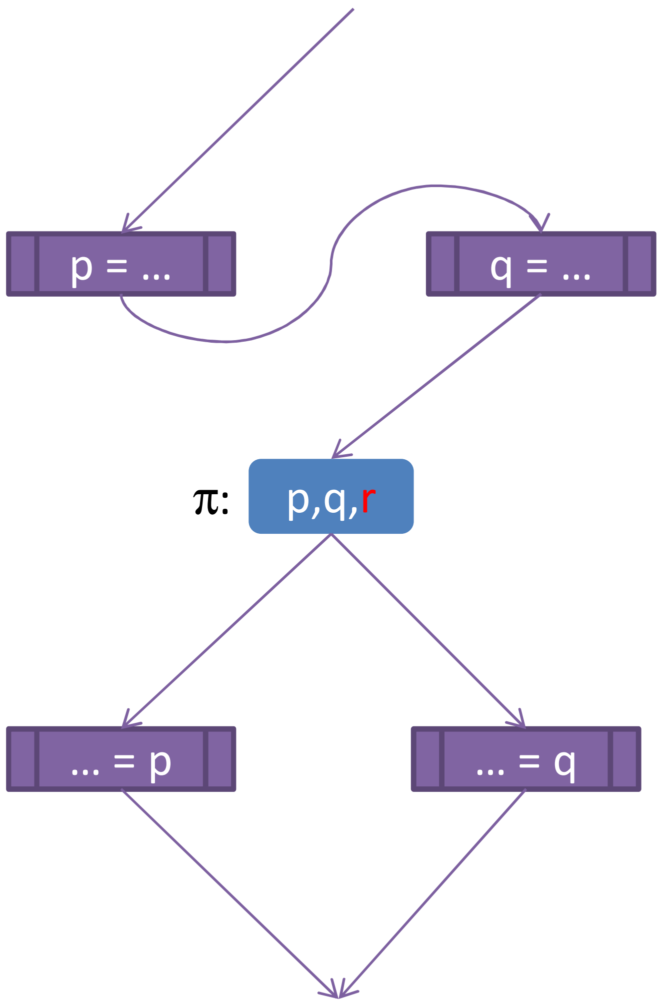
$$[\![s]\!]^2(o^\sharp) = [\![s]\!]^1(o^\sharp) \cap \text{live-out}(s)$$



dom-vars

Definition: Given statement π, dom-vars(π) is the set of all variables whose (unique) definition dominates π.

Filtering by Liveness



- Since r is not live at π, it is irrelevant.
- Since p and q are live at π, their defs must dominate π.

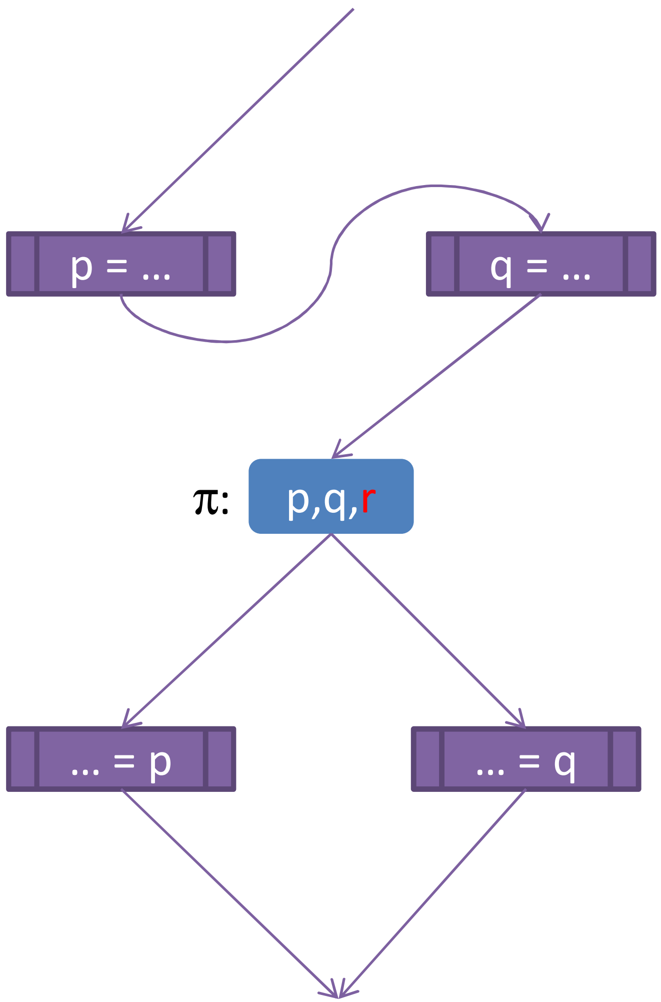
$$[\![s]\!]^2(o^\sharp) = [\![s]\!]^1(o^\sharp) \cap \text{live-out}(s)$$

$$[\![s]\!]^3(o^\sharp) = [\![s]\!]^1(o^\sharp) \cap \text{dom-vars}(s)$$

$$[\![s]\!]^1(o^\sharp) \supseteq [\![s]\!]^3(o^\sharp) \supseteq [\![s]\!]^2(o^\sharp)$$

SSA Property 1:
live-out(s) $\subseteq$ dom-vars(s).

Filtering by Liveness



- Since r is not live at π, it is irrelevant.
- Since p and q are live at π, their defs must dominate π.
- Since defs of p and q dominate π, one must dominate the other.

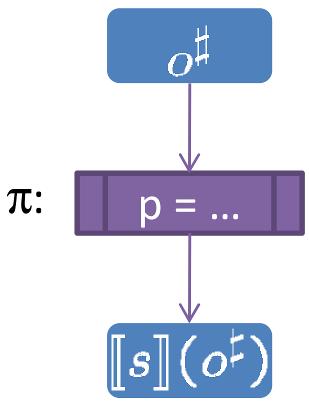
$$[\![s]\!]^2(o^\sharp) = [\![s]\!]^1(o^\sharp) \cap \text{live-out}(s)$$

$$[\![s]\!]^3(o^\sharp) = [\![s]\!]^1(o^\sharp) \cap \text{dom-vars}(s)$$

$$[\![s]\!]^1(o^\sharp) \supseteq [\![s]\!]^3(o^\sharp) \supseteq [\![s]\!]^2(o^\sharp)$$

SSA Property 2:
If $\{p_1, p_2, \ldots\}$ are simultaneously live, then the p_i are totally ordered by dominance of their definitions.

Insertion at Head of List

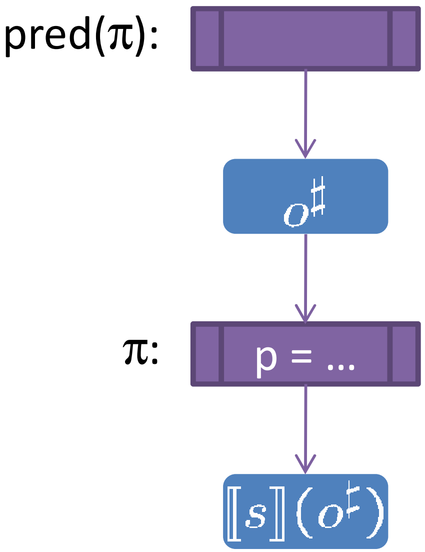


Fact: $[\![s]\!]^1(o^\sharp) \subseteq o^\sharp \cup \mathsf{def}(s)$ for all s.

Therefore, if $o^\sharp \subseteq \mathsf{dom\text{-}vars}(\pi)$, then def of p is dominated by defs of all variables in $o^\sharp$. Thus, insertion of p occurs at head of list.

SSA Property 3:
If a transfer function adds only the variable being defined to a set S, it preserves the property that S $\subseteq$ dom-vars.

Removal from Head of List



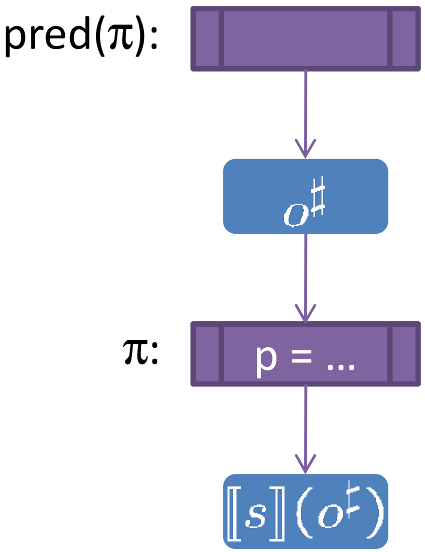
Fact: $[\![s]\!]^1(o^\sharp) \supseteq o^\sharp \setminus \mathsf{def}(s)$ for all s.

Thus, if $o^\sharp \subseteq$ dom-vars(pred(π)), then $p \notin o^\sharp$.
Thus, the $\setminus$ operation in $[\![s]\!]^1$ is unnecessary.

The only removal necessary is intersection with dom-vars(π).

SSA Property 4:
If S ⊆ dom-vars(pred(π)), then the variable defined at π is not in S.

Removal from Head of List

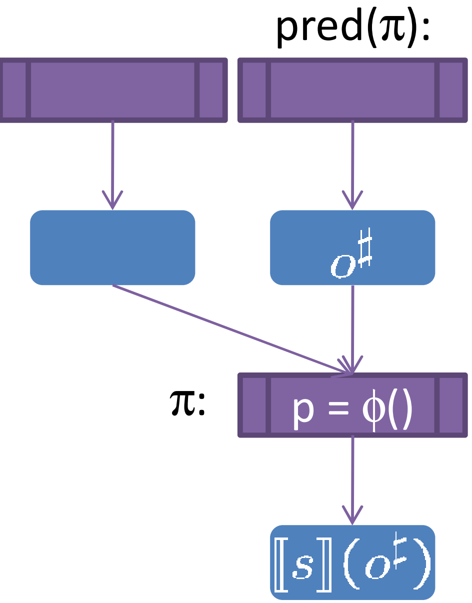


If pred(π) is the *only* predecessor of π,
then dom-vars(π) $=$ dom-vars(pred(π)) $\cup \{p\}$.

If $o^\sharp \subseteq$ dom-vars(pred(π)),
and $[\![s]\!]^1(o^\sharp) \subseteq o^\sharp \cup \{p\}$,
then $[\![s]\!]^1(o^\sharp) \subseteq$ dom-vars(π).

So no intersection is necessary.

Removal from Head of List



If π has multiple predecessors,
then dom-vars(π) = dom-vars(idom(π)) $\cup \{p\}$.

Every var in $o^\sharp \setminus$ dom-vars(idom(π)) is dominated by every var in dom-vars(idom(π)). Therefore, the variables to be removed are at the head of the list $o^\sharp$.

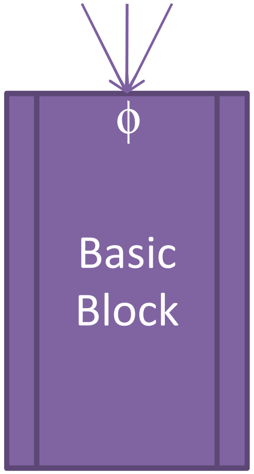
Thus, $[\![\phi]\!]^6(o^\sharp) = [\![\phi]\!]^1(\text{prune}(o^\sharp))$, where prune removes vars from the head of the list until the def of the head of the list strictly dominates π.

So intersection is removal from head of list.

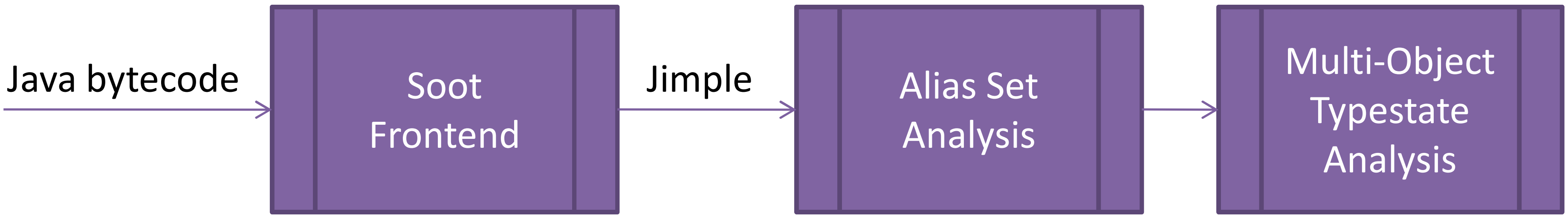
Removal from Head of List

SSA Property 5:
To maintain the property that $S \subseteq$ dom-vars(π), it suffices to intersect S with dom-vars(idom(π)) only at control flow merge points.

It is convenient to arrange for all control flow merge points to be (possibly vacuous) ϕ nodes.



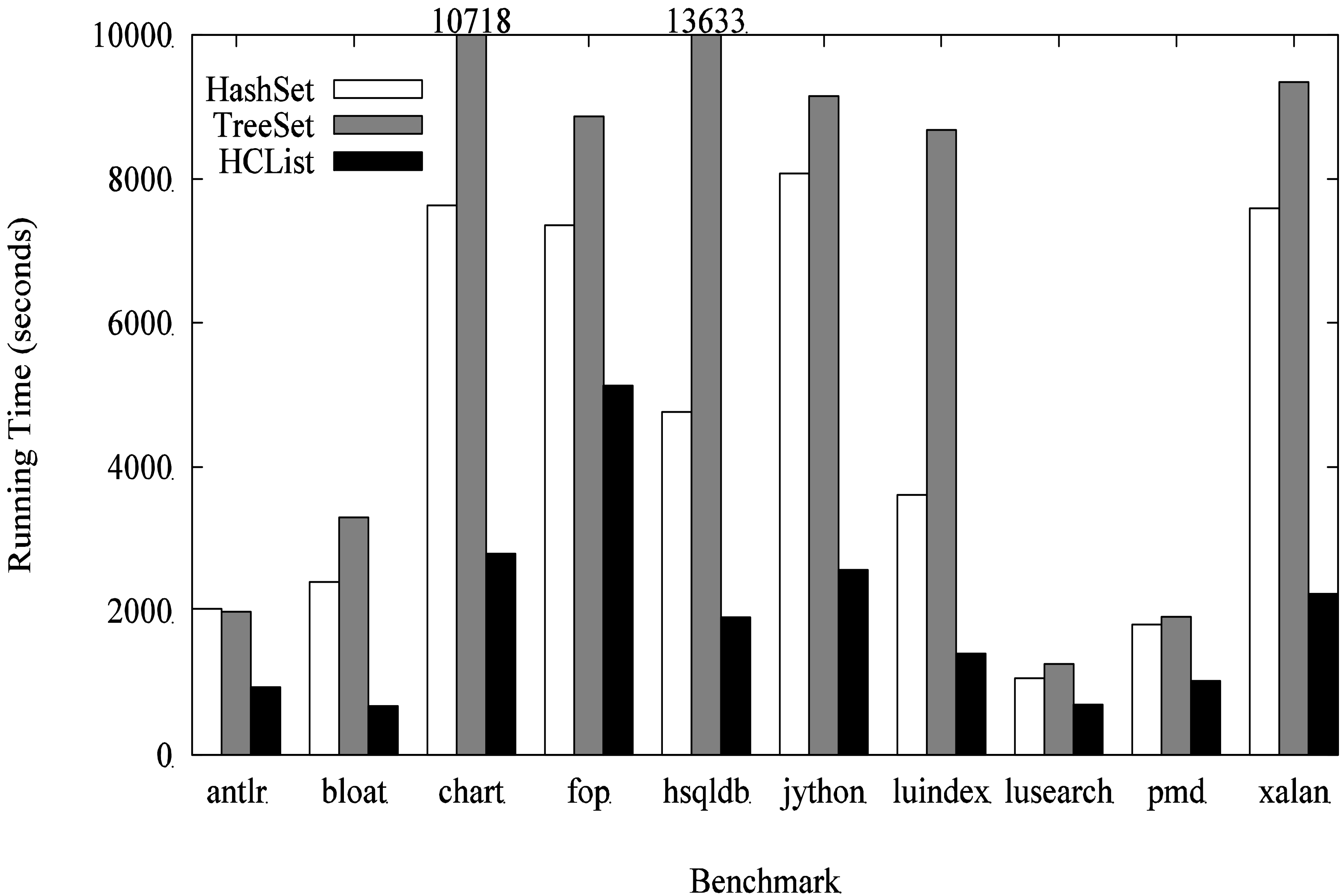
Implementation Overview



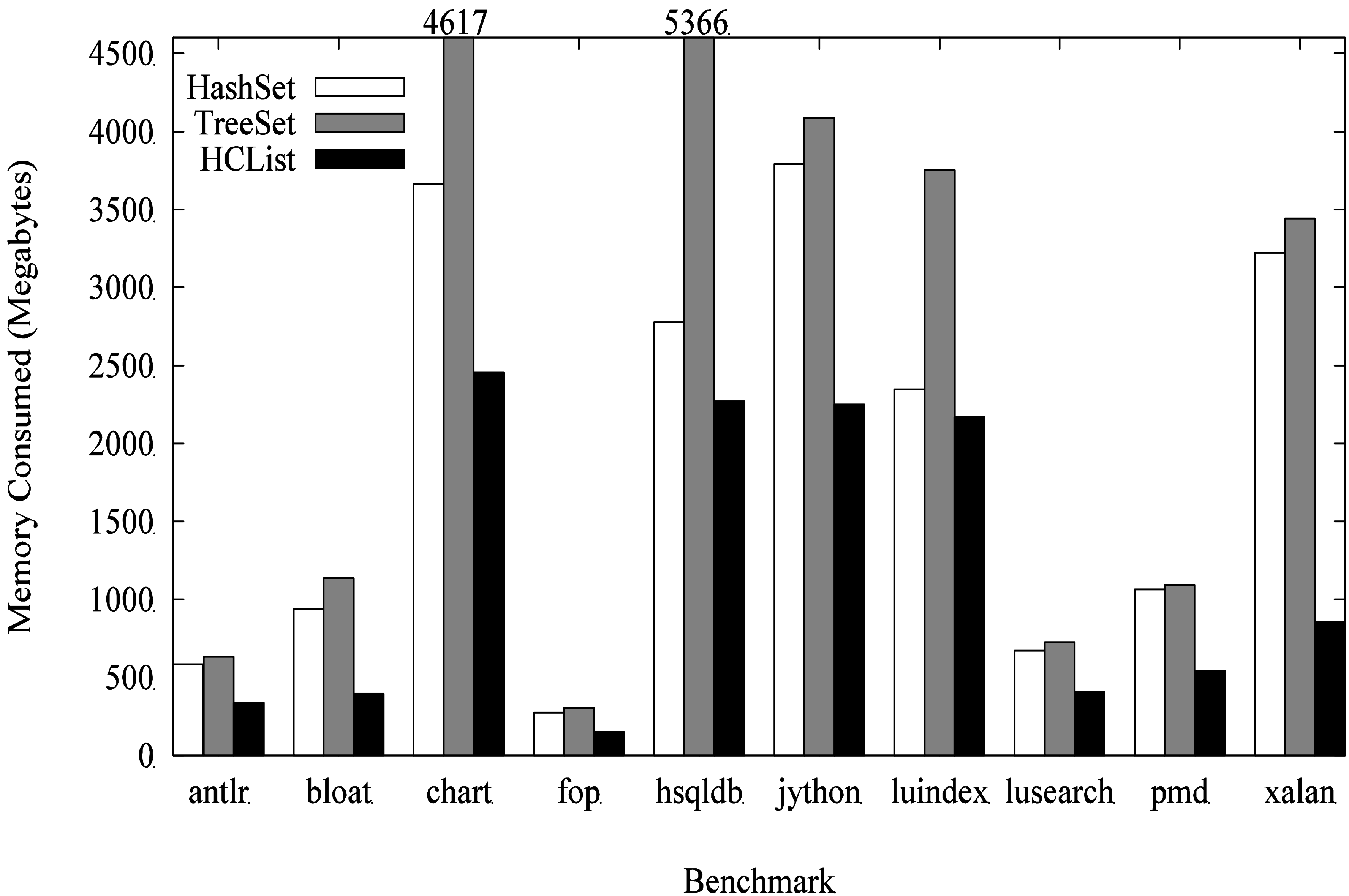
The Alias Set Analysis and the Typestate Analysis are each an instantiation of the IFDS algorithm:

- Interprocedural
- Context-Sensitive
- Precise
- Expensive

Analysis Performance Improvement



Analysis Performance Improvement



Summary of SSA Properties

1. live-out(s) $\subseteq$ dom-vars(s).
2. If $\{p_1, p_2, \ldots\}$ are simultaneously live, then the p_i are totally ordered by dominance of their definitions.
3. If a transfer function adds only the variable being defined to set S, it preserves the property that S $\subseteq$ dom-vars.
4. If S $\subseteq$ dom-vars(pred(π)), then the variable defined at π is not in S.
5. To maintain the property that S $\subseteq$ dom-vars(π), it suffices to intersect S with dom-vars(idom(π)) only at control flow merge points.