A Proofs of Lemmas and Theorems

This section provides detailed proofs for the lemmas and theorems stated in the main paper. For clarity, some of the definitions, lemmas, and theorems are restated as needed.

THEOREM 4.1. (***Equivalence between Field and Variable Pointers under Object-Sensitive Analysis***). Under k-object-sensitive pointer analysis, a field f of a heap object h and a local variable v in a method m invoked on h exhibit structural equivalence in their context-sensitive representations.

PROOF. Let us assume a k-object-sensitive pointer analysis.

Context Representation for Field Pointers. For a heap object h allocated under context $[o_1, \dots, o_l]$, where $l \le k-1$, the field pointer $h.f$ is represented by as:

$$([o_1, \dots, o_l], h.f)$$

This indicates that the access to the field f of object h is resolved under the context in which h was created, and the pointer name is tied to the field label f and the base object h.

Context Representation for Local Variables. Suppose the method m is invoked on the receiver object h allocated under context $[o_1, \dots, o_l]$, where $l \le k-1$. Then, under object-sensitive analysis, the context of method m becomes:

$$c_m = [o_1, \dots, o_{k-1}, h]$$

Let v be a local variable within m. Its context-sensitive representation is then:

$$([o_1, \dots, o_{k-1}, h], v)$$

Structural Equivalence. Now, consider rewriting the field pointer representation. Note that the context-sensitive representation of $h.f$ can be rewritten by pushing the object h into the context and separating the field label:

$$([o_1, \dots, o_{k-1}], h.f) \quad \equiv \quad ([o_1, \dots, o_{k-1}, h], f)$$

This reinterpretation is valid because object-sensitive pointer analysis uses abstract heap objects to construct contexts. Here, we merely shift the object h from being the "base" of a field label to being part of the context sequence, while preserving the essential identity of the field.

Conclusion. Since both representations are of the same structural form, we conclude that *field pointers and local variables are structurally equivalent in their contextual representation* under k-object-sensitive analysis. □

THEOREM 4.2. The ground truth of selective context-sensitive objects must be precision-relevant: Precision $\uparrow (h) \implies \mathrm{PR}(h, k-1)$, where $k-1$ is the length limit of object contexts.

To facilitate the proof of Theorem 4.2, we first introduce the following lemma, whose proof is given afterwards to maintain the coherence of presentation.

LEMMA A.1. *Given the context-sensitive PFG $G_{\mathbb{O}}$, and a field f of object h, the following formula always holds:*

$$\neg\textsc{Partial}(h, f) \implies \forall x \in \textsc{Pre}(h, f), \forall y \in \textsc{Suc}(h, f) : x \bowtie y$$

We now present the proof for Theorem 4.2 below.

PROOF FOR THEOREM 4.2. We prove by contraposition. That is, we assume $\neg\mathrm{PR}(h, k-1)$ and show that this implies $\neg$Precision $\uparrow (h)$, i.e., h cannot contribute to precision improvement under context sensitivity.

From Definition 4.6, an object h is *not* precision-relevant if for each field f of h, one of the two cases below is satisfied: it is not partial or its corresponding flows are not distinguishable. We examine each case separately:

- **Not Partial:** If each field f of h is not partial, it is formalized as

$$\forall f \in \mathbb{F} \text{ of } h : \neg\textsc{Partial}(h, f)$$

By Lemma A.1, we have:

$$\neg\textsc{Partial}(h, f) \implies \forall x \in \textsc{Pre}(h, f), \forall y \in \textsc{Suc}(h, f) : x \bowtie y$$

By the nature of pointer analysis, we have in PFG $G_{\mathbb{O}\setminus\{h\}}$, for field f of h, there is only one field pointer $(\emptyset, h.f)$, since all the contexts of h are removed in $G_{\mathbb{O}\setminus\{h\}}$, hence we have $\forall x \in \textsc{Pre}(h, f), \forall y \in \textsc{Suc}(h, f) : x \bowtie y$ satisfied in $G_{\mathbb{O}\setminus\{h\}}$. This implies that $\forall x \in \textsc{Pre}(h, f), \forall y \in \textsc{Suc}(h, f) : x \bowtie z$ holds in both $G_{\mathbb{O}}$ and $G_{\mathbb{O}\setminus\{h\}}$. Consequently, the reachability related to the flows through $h.f$ remain the same in $G_{\mathbb{O}}$ and $G_{\mathbb{O}\setminus\{h\}}$, thus we have:

$$\forall v \in \mathbb{V}, \forall o \in \{o \mid o \rightsquigarrow_{G_{\mathbb{O}\setminus\{h\}}} v\} : o \rightsquigarrow_{G_{\mathbb{O}}} v$$

Therefore, we conclude $\neg\text{Precision} \uparrow (h)$.

- **Flow Not Distinguishable:**
If the corresponding flows of each field f of h are not distinguishable, it is formalized as:

$$\forall f \in \mathbb{F} \text{ of } h : \neg\textsc{DistObj}(h, f) \land$$
$$(\forall y \in \textsc{Pre}(h, f), \forall o \in pt(y) : \neg\text{PR}(o, l') \lor \neg\textsc{DistCtx}(o, h, f) \lor l' \leq 0)$$

We discuss every condition below:

  - (1) By Definition 4.4, we have:

$$\neg\textsc{DistObj}(h, f) := \forall y \not\bowtie h.f, y' \in \textsc{Pre}(h, f), \forall c_y, c'_y \in \mathbb{C} : pt(c_y, y) = pt(c'_y, y')$$

  which means all the predecessors have the same objects, making the contexts of h redundant for distinguishing these objects.

  - (2) By Definition 4.5, we have:

$$\neg\textsc{DistCtx}(o, h, f) := \forall y \not\bowtie h.f, y' \in \textsc{Pre}(h, f), \forall c_o, c_y, c'_y \in \mathbb{C} :$$
$$(c_o, o) \in \hat{pt}(c_y, y) \land (c_o, o) \in \hat{pt}(c'_y, y')$$

  which means all the predecessors have the identical contextual objects, making the contexts of h redundant for distinguishing these contextual objects as well.

  - (3) By $l' \leq 0$, we have that the context length limit of o is exceeded and the contexts of h are invisible to o, making contexts of h redundant for distinguishing the contextual objects of o.
  - (4) By $\neg\text{PR}(o, l')$, we have the object o is not *Precision-Relevant*, which leads to a recursive situation:
    * **Base:** o is not *Precision-Relevant* due to $\neg\textsc{DistObj} \land (\neg\textsc{DistCtx} \lor l' \leq 0)$. In such scenarios, we have already shown that the contexts of o are redundant for distinguishing pointer flows, implying that there is no need to differentiate the contextual replicas of o.
    * **Recursive:** o is not *Precision-Relevant* due to $\neg\textsc{DistObj} \land \neg\text{PR}$. This results in a recursive situation. However, by unrolling this recursion, we eventually reach one of the previously analyzed cases, which all establish that the contexts of o are redundant for distinguishing pointer flows.

By the contradiction of all the conditions above, we have the contexts of h redundant for distinguishing the pointer flows through its field pointers, thus, we have:

$$\forall x \in \textsc{Pre}(h, f), \forall y \in \{y | x \rightsquigarrow_{G_{\mathbb{O}\setminus\{h\}}} y\} : x \rightsquigarrow_{G_{\mathbb{O}}} y$$

further we have:

$$\forall v \in \mathbb{V}, \forall o \in \{o \mid o \rightsquigarrow_{G_{\mathbb{O}\setminus\{h\}}} v\} : o \rightsquigarrow_{G_{\mathbb{O}}} v$$

Therefore, we conclude $\neg$Precision $\uparrow(h)$.

In all possible cases where $\neg\text{PR}(h, k-1)$, we conclude that $\neg$Precision $\uparrow(h)$. Hence, the contrapositive $\neg\text{PR}(h, k-1) \implies \neg$Precision $\uparrow(h)$ holds, and so does the original implication Precision $\uparrow(h) \implies \text{PR}(h, k-1)$. □

In the following, we present the proof of Lemma A.1.

Proof for Lemma A.1. By Definition 4.3 , we have:

$$\neg\textsc{Partial}(h, f) \iff \forall x \in \textsc{Pre}(h, f), \forall y \in \textsc{Suc}(h, f) : x \bowtie h.f \lor h.f \bowtie y.$$

We aim to prove:

$$x \bowtie h.f \lor h.f \bowtie y \implies x \bowtie y$$

We consider the two disjunctive cases separately.

Case 1: $x \bowtie h.f$. By Definition 4.2, this means:

$$\forall c_x, c_h \in \mathbb{C} : (c_x, x) \rightarrow (c_h, h.f).$$

Furthermore, by Definition 4.1, for any $y \in \textsc{Suc}(h, f)$, we have:

$$\forall c_y \in \mathbb{C}, \exists c_h \in \mathbb{C} : (c_h, h.f) \rightarrow (c_y, y).$$

By composing these two reachability relations, we derive:

$$\forall c_x, c_y \in \mathbb{C} : (c_x, x) \rightsquigarrow (c_y, y),$$

which is the definition of $x \bowtie y$.

Case 2: $h.f \bowtie y$. This is symmetric to Case 1. We have:

$$\forall c_h, c_y \in \mathbb{C} : (c_h, h.f) \rightarrow (c_y, y),$$

and by Definition 4.1, for any $x \in \textsc{Pre}(h, f)$, we have:

$$\forall c_x \in \mathbb{C}, \exists c_h \in \mathbb{C} : (c_x, x) \rightarrow (c_h, h.f).$$

Therefore, composing again yields:

$$\forall c_x, c_y \in \mathbb{C} : (c_x, x) \rightsquigarrow (c_y, y),$$

which again implies $x \bowtie y$. □

Theorem 4.3. A *Precision-Relevant* object must be an *Applied Precision-Relevant* object: $\text{PR}(h, k-1) \implies \overline{\text{PR}}(h, k-1)$, where k is the context length of methods, $k-1$ is the context length of objects.

To facilitate the proof of Theorem 4.3, we first introduce the following lemmas, whose proofs are given afterwards to maintain the coherence of presentation.

Lemma 4.1. Given a field load statement $y = x.f$, the following formula holds under context-sensitive analysis:

$$\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies \hat{pt}(c_1, x) \neq \hat{pt}(c_2, x)$$

where $c_1, c_2 \in \mathbb{C}$.

Lemma 4.2. Under object-sensitive analysis, if a variable y in method m have different points-to sets under different contexts, we have $\overline{\textsc{DistObj2}}$ in $\overline{\text{PR}}$ satisfied on VFG $\mathcal{G}$:

$$\exists c_1, c_2 \in \mathbb{C} : pt(c_1, y) \neq pt(c_2, y) \implies \hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies this_m \rightsquigarrow_{\mathcal{G}} y \vee (\exists o \in \mathbb{O} : o \rightsquigarrow_{\mathcal{G}} y)$$

Lemma 4.3. Given a field f of h, if $\textsc{Partial}(h, f)$ is satisfied on $G_{\mathbb{O}}$, we have $\overline{\textsc{Partial}}$ satisfied on VFG $\mathcal{G}$:

$$\textsc{Partial}(h, f) \implies \overline{\textsc{Partial}}$$

We now present the proof for Theorem 4.3.

Proof for Theorem 4.3. Both PR and $\overline{\text{PR}}$ defined in Definitions 4.6 and 4.8 rely on several atomic properties. Therefore, it suffices to prove each of the following atomic properties individually.

- **(1)** $\textsc{Partial}(h, f) \implies \overline{\textsc{Partial}}$
- **(2)** $\textsc{DistObj}(h, f) \vee (\exists y \in \textsc{Pre}(h, f), \exists o \in pt(y) : \text{PR}(o, l') \wedge \textsc{DistCtx}(o, h, f) \wedge l' > 0) \implies \overline{\textsc{DistObj1}} \vee \overline{\textsc{DistObj2}} \vee \overline{\textsc{DistCtx}}$

By Lemma 4.3, we have **(1)** proven, thus we only need to prove **(2)**, and we prove the two implications:

$$\textsc{DistObj}(h, f) \implies \overline{\textsc{DistObj1}} \vee \overline{\textsc{DistObj2}} \tag{3}$$

$$\begin{array}{c}\exists y \in \textsc{Pre}(h, f), \exists o \in pt(y) : \\ \text{PR}(o, l') \wedge \textsc{DistCtx}(o, h, f) \wedge l' > 0\end{array} \implies \overline{\textsc{DistObj1}} \vee \overline{\textsc{DistObj2}} \vee \overline{\textsc{DistCtx}} \tag{4}$$

(1) Implication (3): By Definition 4.4, we have:

$$\textsc{DistObj}(h, f) := \exists y \not\ltimes h.f, y' \in \textsc{Pre}(h, f), c_y, c'_y \in \mathbb{C} : pt(c_y, y) \neq pt(c'_y, y')$$

When y and y' are different variables and they have different points-to sets, i.e., $\overline{\textsc{DistObj1}}$, thus we have:

$$\textsc{DistObj}(h, f) \implies \overline{\textsc{DistObj1}}$$

When y and y' are the same variable, which means the variable have different points-to sets under different contexts, by Lemma 4.2, we have:

$$\textsc{DistObj}(h, f) \implies \overline{\textsc{DistObj2}}$$

Thus, we have implication 3 satisfied.

(2) Implication (4), where the recursion on both sides can be unrolled within same steps: By Definition 4.5, we have:

$$\textsc{DistCtx}(o, h, f) := \begin{array}{c}\exists y \not\ltimes h.f, y' \in \textsc{Pre}(h, f), \exists c_o, c_y, c'_y \in \mathbb{C} : \\ (c_o, o) \in \hat{pt}(c_y, y) \wedge (c_o, o) \notin \hat{pt}(c'_y, y')\end{array}$$

According to Definition 4.5, a difference between the contextual points-to sets of (c_y, y) and (c'_y, y') is required. This discrepancy implies either the predicate $\overline{\textsc{DistObj1}}$ holds directly, or it can be over-approximated via the reachability criteria defined by $\overline{\textsc{DistObj2}}$ or by the context-sensitive distinction $\overline{\textsc{DistCtx}}$, as established in Lemma 4.2. Thus we have:

$$\begin{array}{c}\exists y \in \textsc{Pre}(h, f), \exists o \in pt(y) : \\ \text{PR}(o, l') \wedge \textsc{DistCtx}(o, h, f) \wedge l' > 0\end{array} \implies \overline{\textsc{DistObj1}} \vee \overline{\textsc{DistObj2}} \vee \overline{\textsc{DistCtx}}$$

□

In the following, we present our proof for Lemma 4.1.

Proof for Lemma 4.1. We proceed by contraposition. Suppose that:

$$\hat{pt}(c_1, x) = \hat{pt}(c_2, x) = P$$

where P is the contextual points-to set pointed to by x in both contexts c_1 and c_2.

By the **[Field-Load]** rule defined in Figure 3, for each contextual object $(c', h) \in P$, the following holds:

$$(c', h.f) \rightarrow (c_1, y) \quad \text{and} \quad (c', h.f) \rightarrow (c_2, y)$$

Since the same contextual points-to set P is accessed under both c_1 and c_2, and each $(c', h.f)$ points to both (c_1, y) and (c_2, y) identically, it follows that the resulting contextual points-to sets for y are also equal:

$$\hat{pt}(c_1, y) = \hat{pt}(c_2, y)$$

This contradicts the assumption that $\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y)$. Hence, by contraposition:

$$\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies \hat{pt}(c_1, x) \neq \hat{pt}(c_2, x)$$

□

Before presenting the proof of Lemma 4.2, we first introduce the following lemmas, whose proofs are given afterwards.

Lemma A.2. *Under object-sensitive analysis, for any method m and its associated receiver variable $this_m$, if method m has two distinct contexts $c_1, c_2 \in \mathbb{C}$, then the context-sensitive points-to sets of $this_m$ are distinct, i.e., $\hat{pt}(c_1, this_m) \neq \hat{pt}(c_2, this_m)$.*

Lemma A.3. *Under object-sensitive analysis, given a variable y in method m, the following formula holds:*

$$\exists c_1, c_2 \in \mathbb{C} : \hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies \exists c_{n1}, c_{n2} \in \mathbb{C}, \exists n \in \mathcal{N} : \hat{pt}(c_1, n) \neq \hat{pt}(c_2, n) \land n \rightsquigarrow_{\mathcal{G}} y$$

Proof for Lemma 4.2. **Step 1.** We prove the implication:

$$\exists c_1, c_2 \in \mathbb{C} : pt(c_1, y) \neq pt(c_2, y) \implies \hat{pt}(c_1, y) \neq \hat{pt}(c_2, y)$$

By the definitions of pt and $\hat{pt}$, we have:

$$pt(c_i, y) := \{o \mid (c, o) \in \hat{pt}(c_i, y) \text{ for some } c \in \mathbb{C}\}$$

Thus, the implication is naturally satisfied.

Step 2. We prove the implication:

$$\exists c_1, c_2 \in \mathbb{C} : \hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies this_m \rightsquigarrow_{\mathcal{G}} y \ \lor \ (\exists o \in \mathbb{O} : o \rightsquigarrow_{\mathcal{G}} y)$$

By Lemma A.3, we have:

$$\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies \exists n \in \mathcal{N} : \hat{pt}(c_1, n) \neq \hat{pt}(c_2, n) \land n \rightsquigarrow_{\mathcal{G}} y$$

And the only root sources of pointer flow in $\mathcal{G}$ are:

- "this" variable $this_m$,
- some allocated heap objects $o \in \mathbb{O}$.

By Lemma A.2, $this_m$ carries distinct contextual points-to sets across different calling contexts, and so do the allocated heap objects due to the setting of context-sensitive objects. Hence, there must exist a path from either $this_m$ or some heap object o to y, i.e.,

$$this_m \rightsquigarrow_{\mathcal{G}} y \ \lor \ \exists o \in \mathbb{O} : o \rightsquigarrow_{\mathcal{G}} y$$

□

We now present the proofs for Lemmas A.2 and A.3.

Proof for Lemma A.2. Let m be a method. Under object-sensitive analysis, the context $c \in \mathbb{C}$ for a call to m is constructed from a contextual receiver object (c', o), where $o \in \mathbb{O}$ and $c' \in \mathbb{C}$, by a function cons illustrated in the rule **[Method-Call]** in Figure 3:

$$\textsc{cons}(c, o) = \lceil c :: o \rceil_k$$

Also, by rule **[Method-Call]**, we have:

$$(c', o) \to (c, this_m)$$

which implies:

$$(c', o) \in \hat{pt}(c, this_m)$$

Now suppose $c_1 \neq c_2$ are two distinct contexts for m, and both were generated by CONS from pairs (c'_1, o_1) and (c'_2, o_2), respectively:

$$c_1 = \text{CONS}(c'_1, o_1), \quad c_2 = \text{CONS}(c'_2, o_2)$$

We show that $c_1 \neq c_2$ implies $(c'_1, o_1) \neq (c'_2, o_2)$. Suppose for contradiction that $(c'_1, o_1) = (c'_2, o_2)$. Then by definition of CONS, we would have:

$$c_1 = \text{CONS}(c'_1, o_1) = \text{CONS}(c'_2, o_2) = c_2$$

which contradicts $c_1 \neq c_2$. Hence, it must be the case that:

$$(c'_1, o_1) \neq (c'_2, o_2)$$

Now using the **[Method-Call]** rule again:

$$(c'_1, o_1) \in \hat{pt}(c_1, this_m), \quad (c'_2, o_2) \in \hat{pt}(c_2, this_m)$$

Since $(c'_1, o_1) \neq (c'_2, o_2)$, we conclude:

$$\hat{pt}(c_1, this_m) \neq \hat{pt}(c_2, this_m)$$

□

PROOF FOR LEMMA A.3. Assume $\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y)$. We begin by analyzing the possible derivation rules by which an edge $n \to_{\mathcal{G}} y$ could have been constructed in the VFG $\mathcal{G}$. We consider all VFG construction rules from Figure 6.

(a) **[Assign]** $_{\mathcal{G}}$: Suppose $n \to_{\mathcal{G}} y$ is constructed due to an assignment $y = n$. Since assignment statements directly propagate pointer flows under different contexts, it follows that:

$$\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies \hat{pt}(c_1, n) \neq \hat{pt}(c_2, n)$$

(b) **[Field-Load]** $_{\mathcal{G}}$:
(1) Suppose the edge arises from a field load $y = n.f$. By Lemma 4.1, if $\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y)$, then:

$$\hat{pt}(c_1, n) \neq \hat{pt}(c_2, n)$$

(2) Suppose the edge arises from $n = y.f$, this does not affect the points-to set of y.

(c) **[Field-Store]** $_{\mathcal{G}}$: Edges from statements of the form $n.f = y$ do not affect the points-to set of y.

(d) **[Method-Call]** $_{\mathcal{G}}$: Suppose the edge $n \to_{\mathcal{G}} y$ is constructed when y receives a return value from a callee called on n. By Theorem 4.1, the same reasoning as in case **(b)** applies, and we obtain:

$$\hat{pt}(c_1, n) \neq \hat{pt}(c_2, n)$$

When the the edge $n \to_{\mathcal{G}} y$ is constructed when n is an argument passed into a callee called on y, then same reasoning as in case **(c)** applies.

(e) **[New]** $_{\mathcal{G}}$: Suppose the edge $n \to_{\mathcal{G}} y$ is constructed due to an allocation statement. Since allocation statements directly propagate pointer flows under different contexts, we have:

$$\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y) \implies \hat{pt}(\lceil c_1 \rceil_{k-1}, n) \neq \hat{pt}(\lceil c_2 \rceil_{k-1}, n)$$

In all relevant cases, we have shown that $\hat{pt}(c_1, y) \neq \hat{pt}(c_2, y)$ implies the existence of a predecessor node $n \in \mathcal{N}$ such that:

$$\exists c_{n1}, c_{n2} \in \mathbb{C}, \hat{pt}(c_{n1}, n) \neq \hat{pt}(c_{n2}, n) \quad \text{and} \quad n \rightarrow_{\mathcal{G}} y$$

We now apply this reasoning recursively: for such an n, if it is not a source node in $\mathcal{G}$, then it must itself have a predecessor under one of the VFG rules, and the same argument can be applied again, which completes the proof. □

Before presenting the proof of Lemma 4.3, we first introduce the following lemma, whose proof is given afterwards.

Lemma A.4. *Given a field store statement $x.f = y$, the following formula holds under context-sensitive analysis:*

$$\exists c_o, c_1, c_2 \in \mathbb{C}, o \in \mathbb{O}, (c_h, h) \in \hat{pt}(c_1, x), (c'_h, h') \in \hat{pt}(c_2, x) :$$
$$(c_o, o) \in \hat{pt}(c_1, y) \land (c_o, o) \notin \hat{pt}(c_2, y) \land (c_o, o) \in \hat{pt}(c_h, h.f) \land (c_o, o) \notin \hat{pt}(c'_h, h'.f)$$
$$\Longrightarrow$$
$$(c_h, h) \in \hat{pt}(c_1, x) \land (c'_h, h') \notin \hat{pt}(c_1, x)$$

Now we present the proof of Lemma 4.3.

Proof for Lemma 4.3. By Definition 4.8, we have:

$$\overline{\text{Partial}} := h \rightsquigarrow_{\mathcal{G}} this_m \lor \exists x \in \text{Base}(h, f) : h \rightsquigarrow_{\mathcal{G}} x$$

We prove the lemma by establishing the following two implications:

$$\text{Partial}(h, f) \implies \exists x \in \text{Base}(h, f), \exists c_h, c_x \in \mathbb{C} : (c_h, h) \notin \hat{pt}(c_x, x)$$
$$\exists x \in \text{Base}(h, f), \exists c_h, c_x \in \mathbb{C} : (c_h, h) \notin \hat{pt}(c_x, x) \implies h \rightsquigarrow_{\mathcal{G}} this_m \lor \exists x \in \text{Base}(h, f) : h \rightsquigarrow_{\mathcal{G}} x$$

Step 1: We prove the implication by contradiction:

$$\text{Partial}(h, f) \implies \exists x \in \text{Base}(h, f), \exists c_h, c_x \in \mathbb{C} : (c_h, h) \notin \hat{pt}(c_x, x)$$

Suppose that:

$$\text{Partial}(h, f) \quad \text{and} \quad \forall x \in \text{Base}(h, f), \forall c_x, c_h \in \mathbb{C} : (c_h, h) \in \hat{pt}(c_x, x)$$

By Definition 4.3, $\text{Partial}(h, f)$ is defined as:

$$\exists p \in \text{Pre}(h, f), \exists s \in \text{Suc}(h, f) : \ p \not\leftrightarrow h.f \land h.f \not\leftrightarrow s$$

We first consider the reachability regarding the predecessors, which is built by the rule **[Field-Store]** for statement $x.f = y$ in Figure 3, we have:

$$(c, y) \rightarrow (c_h, h.f) \iff (c_h, h) \in \hat{pt}(c, x)$$

From our assumption, we have:

$$\forall x \in \text{Base}(h, f), \forall c_x, c_h : (c_h, h) \in \hat{pt}(c_x, x)$$

This implies that for every field store $x.f = y$ and every c, we always have $(c, y) \rightarrow (c_h, h.f)$ in the PFG and no partial reachability can occur. The same rationale applies for the successors of field load statements as well. Thus, this contradicts the assumption, and we conclude:

$$\exists x \in \text{Base}(h, f), \exists c_x, c_h : (c_h, h) \notin \hat{pt}(c_x, x)$$

Step 2: We prove the implication:

$$\exists x \in \text{Base}(h, f), \exists c_h, c_x \in \mathbb{C} : (c_h, h) \notin \hat{pt}(c_x, x) \implies h \rightsquigarrow_{\mathcal{G}} this_m \lor \exists x \in \text{Base}(h, f) : h \rightsquigarrow_{\mathcal{G}} x$$

We assume $\exists x \in \textsc{Base}(h, f), \exists c_h, c_x \in \mathbb{C} : (c_h, h) \notin \hat{pt}(c_x, x)$, which means the distinction between contextual replicas of h is preserved until reaching variable x. Thus, we can derive that such distinction at least is preserved within the method m where h is allocated. Since such distinction is naturally satisfied from the allocation site of h (guaranteed by the **[New]** rule defined in Figure 3), thus, we only need such distinction either preserved at a base variable within method m, or at the exit of m, i.e., *this* variable. We then prove such distinction can be preserved along a VFG path starting from h.

The direct successor of h (say s) must be built by $\textbf{[New]}_{\mathcal{G}}$, since this rule propagates objects within same contexts, thus we naturally have $\exists c_h, c'_h, c_s \in \mathbb{C} : (c_h, h) \in \hat{pt}(c_s, s) \wedge (c'_h, h) \notin \hat{pt}(c_s, s)$. We proceed by analyzing the possible derivation rules by which the path $h \rightarrow_{\mathcal{G}} s \rightarrow_{\mathcal{G}} x$ could have been constructed in the VFG $\mathcal{G}$. We consider all VFG construction rules from Figure 6.

(a) $\textbf{[Assign]}_{\mathcal{G}}$: Suppose $s \rightarrow_{\mathcal{G}} x$ is constructed due to an assignment $x = s$. Since assignments in VFG directly propagate pointer flows, it follows that:

$$\exists c_h, c'_h, c_s \in \mathbb{C} : (c_h, h) \in \hat{pt}(c_s, s) \wedge (c'_h, h) \notin \hat{pt}(c_s, s)$$

(b) $\textbf{[Field-Load]}_{\mathcal{G}}$: Suppose the edge $s \rightarrow_{\mathcal{G}} x$ arises from a field load $s = x.f$ or $x = s.f$, both cases do not affect the points-to set of s.

(c) $\textbf{[Field-Store]}_{\mathcal{G}}$: Suppose the edge $s \rightarrow_{\mathcal{G}} x$ arises from a field store statement of the form $x.f = s$, by Lemma A.4, we have the following condition must be satisfied:

$$\exists c_1, c_2 \in \mathbb{C} : \hat{pt}(c_1, x) \neq \hat{pt}(c_2, x)$$

(d) $\textbf{[Method-Call]}_{\mathcal{G}}$: Suppose the edge $s \rightarrow_{\mathcal{G}} x$ is constructed when x receives a return value from a callee called on s. By Theorem 4.1, the same reasoning as in case **(b)** applies. Similarly, when the edge $s \rightarrow_{\mathcal{G}} x$ is constructed when s is an argument passed into a callee called on x, then same reasoning as in case **(c)** applies.

We now apply this reasoning recursively and the same argument can be applied again, and we have the distinction preserved:

- naturally, if the VFG path contains only case **(a)**.
- conditionally, if the VFG path contains cases **(c)** or **(d)**, and the end node x of VFG path must satisfy:

$$\exists c_1, c_2 \in \mathbb{C} : \hat{pt}(c_1, x) \neq \hat{pt}(c_2, x)$$

By Lemma A.2, *this* variable must have different contextual points-to sets under different contexts, thus we have either the contextual replicas of h can reach any base variable like x by assignment statements, or it must has a VFG path ending at *this* variable to reach out of the current method, which is:

$$h \rightsquigarrow_{\mathcal{G}} this_m \vee \exists x \in \textsc{Base}(h, f) : h \rightsquigarrow_{\mathcal{G}} x$$

□

And we present the proof of Lemma A.4.

Proof for Lemma A.4. By the **[Field-Store]** rule defined in Figure 3, for the field store statement $x.f = y$, we have a PFG edge $(c, y) \rightarrow (c', h.f)$ if and only if $(c', h) \in \hat{pt}(c, x)$. Since we already have:

$$(c_o, o) \in \hat{pt}(c_1, y) \wedge (c_o, o) \notin \hat{pt}(c_2, y) \wedge (c_o, o) \in \hat{pt}(c_h, h.f) \wedge (c_o, o) \notin \hat{pt}(c'_h, h'.f)$$

we naturally have:

$$(c_h, h) \in \hat{pt}(c_1, x) \wedge (c'_h, h') \notin \hat{pt}(c_1, x)$$

□

B Detailed Comparison of Entire Analysis Time under 2-object-sensitive and 3-object-sensitive analysis

Table 5 and Table 6 provide the detailed analysis results for the entire analysis (including all three stages) under 2-object-sensitivity and 3-object-sensitivity, respectively, corresponding to the speedup comparisons presented in Figure 11a and Figure 11b in the main text.

Table 5. Precision and Efficiency results of entire analysis (including all three stages) of Cut-Shortcut, Zipper, Conch, DebloaterX and Moon, whereas CI and 2OBJ are presented as baselines.

Program	Analysis	T	S	#MFC	#PCS	#RM	#CE
antlr	2obj	26.5	-	510	1631	7806	51264
	ci	4.8	5.5X	1124	1976	8190	57341
	CSC	5.7	4.6X	812	1896	8153	54460
	Z-2obj	23.3	1.1X	528	1650	7836	51391
	C-2obj	13.8	1.9X	510	1631	7806	51264
	D-2obj	17.2	1.5X	510	1631	7806	51264
	M-2obj	10.8	2.5X	510	1631	7806	51264
biojava	2obj	13.5	-	436	843	7440	34802
	ci	4.5	3.0X	1046	1236	7910	41678
	CSC	5.7	2.4X	687	1150	7853	38403
	Z-2obj	13.6	1.0X	449	876	7487	34947
	C-2obj	10.8	1.2X	436	843	7440	34802
	D-2obj	12.9	1.0X	436	843	7440	34802
	M-2obj	8.8	1.5X	436	843	7440	34802
chart	2obj	100.2	-	1337	2027	15149	72650
	ci	9.2	10.9X	2560	2703	15923	86433
	CSC	10.4	9.6X	1914	2405	15545	78847
	Z-2obj	52.9	1.9X	1376	2045	15234	72989
	C-2obj	70.5	1.4X	1337	2027	15149	72650
	D-2obj	48.5	2.1X	1337	2027	15149	72658
	M-2obj	26.5	3.8X	1337	2027	15149	72658
dcevm	2obj	374.1	-	2225	4417	20531	100557
	ci	13.5	27.7X	3593	5669	21503	124314
	CSC	15.2	24.6X	2917	4932	20958	108001
	Z-2obj	195.3	1.9X	2316	4464	20573	100824
	C-2obj	245.1	1.5X	2227	4425	20532	100594
	D-2obj	214.2	1.7X	2231	4431	20532	100608
	M-2obj	92.8	4.0X	2241	4433	20535	100629
findbugs	2obj	811.2	-	2013	3508	16264	87808
	ci	9.7	83.7X	3448	4435	16774	105576
	CSC	11.6	70.0X	2766	4212	16705	96294
	Z-2obj	164.4	4.9X	2044	3512	16276	87855
	C-2obj	76.2	10.6X	2014	3508	16264	87808
	D-2obj	60.1	13.5X	2014	3508	16264	87808
	M-2obj	35.7	22.7X	2017	3511	16268	87838
hsqldb	2obj	11.8	-	407	847	6981	34881
	ci	4.4	2.7X	922	1202	7386	41755
	CSC	5.0	2.3X	657	1119	7331	38607
	Z-2obj	12.7	0.9X	428	868	7015	35020
	C-2obj	9.4	1.3X	407	847	6981	34881
	D-2obj	11.8	1.0X	407	847	6981	34881
	M-2obj	8.6	1.4X	407	847	6981	34881
luindex	2obj	10.7	-	395	923	7019	33587
	ci	3.9	2.7X	920	1283	7409	39677
	CSC	4.5	2.4X	641	1201	7357	36882
	Z-2obj	12.1	0.9X	417	944	7048	33718
	C-2obj	8.8	1.2X	395	923	7019	33588
	D-2obj	11.1	1.0X	395	923	7019	33588
	M-2obj	7.9	1.4X	395	923	7019	33588
open-telemetry	2obj	9.2	-	351	767	6303	29976
	ci	3.9	2.4X	825	1117	6713	35944
	CSC	4.2	2.2X	581	1035	6668	33157
	Z-2obj	10.8	0.9X	365	789	6335	30086
	C-2obj	7.9	1.2X	351	767	6303	29976
	D-2obj	9.9	0.9X	351	767	6303	29976
	M-2obj	7.4	1.3X	351	767	6303	29976
sqlite-jdbc	2obj	8.7	-	351	768	6322	30094
	ci	3.8	2.3X	826	1118	6732	36062
	CSC	4.4	2.0X	581	1036	6687	33275
	Z-2obj	11.5	0.8X	365	790	6354	30204
	C-2obj	7.9	1.1X	351	768	6322	30094
	D-2obj	9.8	0.9X	351	768	6322	30094
	M-2obj	7.2	1.2X	351	768	6322	30094
tomcat	2obj	16.8	-	402	871	6978	33095
	ci	4.0	4.1X	931	1320	7388	39513
	CSC	4.6	3.6X	665	1243	7340	37125
	Z-2obj	13.8	1.2X	421	895	7022	33248
	C-2obj	10.0	1.7X	402	871	6978	33095
	D-2obj	11.3	1.5X	402	871	6978	33095
	M-2obj	8.2	2.1X	402	871	6978	33095

Program	Analysis	T	S	#MFC	#PCS	#RM	#CE
avrora	2obj	18.8	-	568	933	10940	48361
	ci	5.5	3.4X	1188	1372	11373	55476
	CSC	6.3	3.0X	882	1283	11282	52535
	Z-2obj	18.0	1.0X	595	948	10969	48466
	C-2obj	14.8	1.3X	568	933	10940	48361
	D-2obj	19.6	1.0X	568	933	10940	48361
	M-2obj	12.8	1.5X	568	933	10940	48361
bloat	2obj	352.9	-	1285	1566	9019	56577
	ci	5.5	64.5X	2082	2249	9454	67338
	CSC	7.6	46.4X	1594	2166	9414	63909
	Z-2obj	291.4	1.2X	1309	1597	9061	56784
	C-2obj	214.4	1.6X	1285	1566	9019	56577
	D-2obj	30.2	11.7X	1285	1566	9019	56577
	M-2obj	20.4	17.3X	1285	1566	9019	56577
check-style	2obj	OOM	-	-	-	-	-
	ci	8.2	>657.7X	1939	2760	12766	80169
	CSC	10.3	>521.7X	1503	2641	12713	74197
	Z-2obj	1489.6	>3.6X	1132	2249	12342	67170
	C-2obj	3822.0	>1.4X	1109	2216	12307	66970
	D-2obj	43.3	>124.7X	1109	2216	12307	66970
	M-2obj	24.9	>216.8X	1110	2216	12307	66970
ddjava	2obj	10.9	-	367	815	6840	32248
	ci	4.1	2.7X	880	1177	7249	38462
	CSC	4.8	2.3X	618	1093	7205	35797
	Z-2obj	13.0	0.8X	384	827	6855	32316
	C-2obj	8.9	1.2X	367	815	6840	32248
	D-2obj	10.9	1.0X	367	815	6840	32248
	M-2obj	8.4	1.3X	367	815	6840	32248
fop	2obj	10.2	-	395	830	7591	34364
	ci	4.0	2.5X	911	1210	7997	40423
	CSC	4.5	2.2X	653	1121	7944	37562
	Z-2obj	13.1	0.8X	417	852	7621	34496
	C-2obj	9.2	1.1X	395	830	7591	34364
	D-2obj	11.9	0.9X	395	830	7591	34364
	M-2obj	8.2	1.2X	395	830	7591	34364
jd	2obj	58.0	-	1592	2831	17000	82056
	ci	10.0	5.8X	2714	3750	17878	95602
	CSC	11.8	4.9X	2127	3278	17354	87467
	Z-2obj	41.8	1.4X	1686	2871	17045	82387
	C-2obj	62.0	0.9X	1592	2831	17000	82060
	D-2obj	52.3	1.1X	1592	2832	17000	82061
	M-2obj	29.4	2.0X	1597	2832	17000	82061
lusearch	2obj	11.8	-	409	1119	7671	36464
	ci	4.2	2.8X	1032	1490	8094	43014
	CSC	4.8	2.5X	724	1411	8053	40001
	Z-2obj	13.7	0.9X	434	1143	7700	36598
	C-2obj	9.3	1.3X	409	1119	7671	36464
	D-2obj	13.2	0.9X	409	1119	7671	36464
	M-2obj	8.4	1.4X	409	1119	7671	36464
pmd	2obj	32.5	-	1398	2354	11851	59910
	ci	6.5	5.0X	2265	2950	12365	69505
	CSC	7.6	4.3X	1748	2710	12273	64834
	Z-2obj	31.8	1.0X	1428	2375	11882	60047
	C-2obj	33.2	1.0X	1398	2354	11851	59910
	D-2obj	25.6	1.3X	1398	2354	11851	59910
	M-2obj	16.8	1.9X	1398	2354	11851	59910
sunflow	2obj	35.7	-	1172	1965	13313	61168
	ci	8.2	4.3X	2146	2553	13981	72716
	CSC	9.0	4.0X	1660	2359	13797	67237
	Z-2obj	27.2	1.3X	1215	1988	13401	61537
	C-2obj	32.1	1.1X	1186	1994	13346	61442
	D-2obj	31.2	1.1X	1172	1965	13314	61173
	M-2obj	19.1	1.9X	1172	1965	13314	61173
trade-beans	2obj	26.7	-	466	967	8021	39111
	ci	4.7	5.7X	1095	1420	8652	47885
	CSC	5.5	4.9X	786	1314	8546	44610
	Z-2obj	22.2	1.2X	501	1017	8213	40042
	C-2obj	14.9	1.8X	481	999	8065	39461
	D-2obj	15.4	1.7X	466	967	8021	39111
	M-2obj	10.5	2.6X	466	967	8021	39111

Program	Analysis	T	S	#MFC	#PCS	#RM	#CE
batik	2obj	810.7	-	2142	4956	20254	107822
	ci	13.3	61.0X	3403	5810	20994	127604
	CSC	18.6	43.6X	2846	5524	20819	121105
	Z-2obj	427.8	1.9X	2161	4968	20265	107892
	C-2obj	430.3	1.9X	2142	4956	20254	107822
	D-2obj	297.1	2.7X	2142	4958	20254	107824
	M-2obj	230.9	3.5X	2150	4962	20254	107828
bytecode-viewer	2obj	2992.2	-	4064	4626	14210	100280
	ci	9.4	316.6X	4723	5616	14637	110795
	CSC	21.7	138.0X	4366	5547	14601	106506
	Z-2obj	2146.0	1.4X	4080	4644	14246	100443
	C-2obj	1693.2	1.8X	4064	4626	14210	100280
	D-2obj	725.3	4.1X	4064	4626	14210	100280
	M-2obj	243.4	12.3X	4065	4629	14210	100283
classy-shark	2obj	22.3	-	564	1117	9393	44257
	ci	5.3	4.2X	1323	1682	10178	56231
	CSC	7.1	3.1X	876	1400	9489	48577
	Z-2obj	22.2	1.0X	581	1124	9412	44332
	C-2obj	19.2	1.2X	564	1117	9393	44257
	D-2obj	18.6	1.2X	564	1117	9393	44257
	M-2obj	12.3	1.8X	564	1117	9393	44257
eclipse	2obj	OOM	-	-	-	-	-
	ci	18.8	>287.7X	5090	10672	23386	182975
	CSC	20.7	>261.4X	4282	10250	22977	172521
	Z-2obj	OOM	-	-	-	-	-
	C-2obj	1528.0	>3.5X	3612	9703	22627	162654
	D-2obj	598.6	>9.0X	3612	9703	22627	162654
	M-2obj	291.4	>18.5X	3612	9703	22627	162659
h2	2obj	17.0	-	398	911	7559	36261
	ci	4.6	3.7X	943	1274	7959	42799
	CSC	5.2	3.3X	687	1194	7914	40294
	Z-2obj	14.7	1.2X	423	929	7591	36379
	C-2obj	11.6	1.5X	398	911	7559	36261
	D-2obj	13.5	1.3X	398	911	7559	36261
	M-2obj	8.9	1.9X	398	911	7559	36261
JPC	2obj	65.3	-	1356	4237	15553	81294
	ci	10.3	6.3X	2252	4934	16138	94701
	CSC	11.4	5.7X	1790	4763	16034	87453
	Z-2obj	40.2	1.6X	1382	4274	15582	81489
	C-2obj	56.8	1.2X	1356	4238	15553	81307
	D-2obj	57.8	1.1X	1356	4243	15554	81319
	M-2obj	31.2	2.1X	1360	4243	15554	81321
mindustry	2obj	9.3	-	378	778	6497	30844
	ci	3.8	2.4X	865	1136	6907	36808
	CSC	4.3	2.2X	622	1051	6862	33970
	Z-2obj	11.9	0.8X	398	805	6540	30984
	C-2obj	8.4	1.1X	378	778	6497	30844
	D-2obj	9.8	1.0X	378	778	6497	30844
	M-2obj	7.3	1.3X	378	778	6497	30844
recaf	2obj	475.3	-	3247	5701	27823	142265
	ci	20.4	23.3X	5127	7402	29038	187415
	CSC	34.0	14.0X	4138	6702	28300	161535
	Z-2obj	261.6	1.8X	3378	5732	27857	142450
	C-2obj	296.4	1.6X	3249	5716	27823	142326
	D-2obj	268.5	1.8X	3249	5722	27823	142332
	M-2obj	161.7	2.9X	3257	5722	27826	142365
tesseract	2obj	11.7	-	399	843	7743	37267
	ci	4.6	2.5X	1013	1291	8313	45068
	CSC	6.1	1.9X	747	1216	8270	41968
	Z-2obj	13.8	0.8X	421	867	7775	37383
	C-2obj	10.0	1.2X	399	843	7743	37267
	D-2obj	12.3	1.0X	399	843	7743	37267
	M-2obj	8.9	1.3X	399	843	7743	37267
xalan	2obj	372.9	-	592	1643	9666	46782
	ci	5.2	71.3X	1298	2084	10092	54001
	CSC	6.0	61.6X	884	1967	10034	50747
	Z-2obj	194.5	1.9X	609	1665	9697	46916
	C-2obj	218.4	1.7X	592	1643	9666	46782
	D-2obj	27.7	13.5X	592	1643	9666	46782
	M-2obj	14.3	26.0X	592	1643	9666	46782

Table 6. Precision and Efficiency results of entire analysis (including all three stages) of Cut-Shortcut, Zipper, Conch, DebloaterX and Moon, whereas CI and 3OBJ are presented as baselines.

Program	Analysis	T	S	#MFC	#PCS	#RM	#CE
antlr	3obj	526.2	-	450	1624	7805	51237
	ci	4.8	108.7X	1124	1976	8190	57341
	CSC	5.7	92.0X	812	1896	8153	54460
	Z-3obj	235.8	2.2X	473	1643	7835	51364
	C-3obj	127.1	4.1X	450	1624	7805	51237
	D-3obj	19.2	27.3X	450	1627	7805	51240
	M-3obj	11.1	47.4X	450	1626	7805	51239
biojava	3obj	1053.9	-	347	837	7439	34756
	ci	4.5	232.7X	1046	1236	7910	41678
	CSC	5.7	186.5X	687	1150	7853	38403
	Z-3obj	211.7	5.0X	369	870	7486	34901
	C-3obj	190.3	5.5X	347	837	7439	34756
	D-3obj	15.3	69.0X	347	839	7439	34758
	M-3obj	9.2	114.8X	347	838	7439	34757
chart	3obj	Oom	-	-	-	-	-
	ci	9.2	>587.6X	2560	2703	15923	86433
	CSC	10.4	>519.7X	1914	2405	15545	78847
	Z-3obj	428.9	>12.6X	1280	2021	15204	72544
	C-3obj	Oot	-	-	-	-	-
	D-3obj	183.8	>29.4X	1230	2005	15119	72228
	M-3obj	52.8	>102.2X	1230	2004	15119	72227
dcevm	3obj	Oom	-	-	-	-	-
	ci	13.5	>399.4X	3593	5669	21503	124314
	CSC	15.2	>355.0X	2917	4932	20958	108001
	Z-3obj	Oom	-	-	-	-	-
	C-3obj	Oot	-	-	-	-	-
	D-3obj	1125.9	>4.8X	2131	4396	20493	100083
	M-3obj	504.9	>10.7X	2131	4396	20493	100084
findbugs	3obj	Oom	-	-	-	-	-
	ci	9.7	>557.3X	3448	4435	16774	105576
	CSC	11.6	>465.9X	2766	4212	16705	96294
	Z-3obj	Oot	-	-	-	-	-
	C-3obj	524.3	>10.3X	1632	3477	16217	86892
	D-3obj	64.4	>83.9X	1652	3478	16217	86899
	M-3obj	37.4	>144.3X	1652	3478	16219	86915
hsqldb	3obj	588.8	-	355	840	6980	34854
	ci	4.4	134.7X	922	1202	7386	41755
	CSC	5.0	116.8X	657	1119	7331	38607
	Z-3obj	168.2	3.5X	382	861	7014	34993
	C-3obj	161.6	3.6X	355	840	6980	34854
	D-3obj	13.9	42.5X	355	843	6980	34857
	M-3obj	8.9	65.8X	355	842	6980	34856
luindex	3obj	376.2	-	341	916	7018	33560
	ci	3.9	96.5X	920	1283	7409	39677
	CSC	4.5	82.9X	641	1201	7357	36882
	Z-3obj	138.8	2.7X	364	937	7047	33691
	C-3obj	124.5	3.0X	341	916	7018	33561
	D-3obj	13.8	27.3X	341	919	7018	33564
	M-3obj	8.3	45.3X	341	918	7018	33563
open-telemetry	3obj	325.3	-	296	760	6303	29955
	ci	3.9	83.2X	825	1117	6713	35944
	CSC	4.2	77.5X	581	1035	6668	33157
	Z-3obj	104.9	3.1X	318	783	6335	30066
	C-3obj	108.6	3.0X	296	760	6303	29955
	D-3obj	11.5	28.3X	296	762	6303	29957
	M-3obj	7.7	42.4X	296	762	6303	29957
sqlite-jdbc	3obj	310.9	-	296	761	6322	30073
	ci	3.8	81.0X	826	1118	6732	36062
	CSC	4.4	70.3X	581	1036	6687	33275
	Z-3obj	102.4	3.0X	318	784	6354	30184
	C-3obj	106.2	2.9X	296	761	6322	30073
	D-3obj	11.2	27.7X	296	763	6322	30075
	M-3obj	7.5	41.2X	296	763	6322	30075
tomcat	3obj	687.6	-	345	866	6977	33068
	ci	4.0	169.8X	931	1320	7388	39513
	CSC	4.6	149.1X	665	1243	7340	37125
	Z-3obj	167.6	4.1X	374	891	7022	33226
	C-3obj	143.0	4.8X	345	866	6977	33068
	D-3obj	13.1	52.6X	345	868	6977	33070
	M-3obj	8.8	78.1X	345	868	6977	33070

Program	Analysis	T	S	#MFC	#PCS	#RM	#CE
avrora	3obj	2592.0	-	496	929	10939	48330
	ci	5.5	470.4X	1188	1372	11373	55476
	CSC	6.3	409.5X	882	1283	11282	52535
	Z-3obj	890.2	2.9X	536	944	10969	48439
	C-3obj	587.3	4.4X	496	929	10939	48330
	D-3obj	22.0	117.9X	496	931	10939	48332
	M-3obj	13.0	199.2X	496	930	10939	48336
bloat	3obj	Oot	-	-	-	-	-
	ci	5.5	>987.2X	2082	2249	9454	67338
	CSC	7.6	>710.5X	1594	2166	9414	63909
	Z-3obj	4784.5	>1.1X	1215	1583	9045	56555
	C-3obj	1458.7	>3.7X	1188	1552	9003	56348
	D-3obj	34.6	>156.3X	1188	1555	9003	56351
	M-3obj	20.4	>264.3X	1188	1554	9003	56350
check-style	3obj	Oom	-	-	-	-	-
	ci	8.2	>657.7X	1939	2760	12766	80169
	CSC	10.3	>521.7X	1503	2641	12713	74197
	Z-3obj	Oot	-	-	-	-	-
	C-3obj	Oot	-	-	-	-	-
	D-3obj	50.1	>107.9X	996	2172	12260	66188
	M-3obj	27.9	>193.3X	1000	2171	12260	66188
ddjava	3obj	398.8	-	312	810	6840	32226
	ci	4.1	98.0X	880	1177	7249	38462
	CSC	4.8	83.4X	618	1093	7205	35797
	Z-3obj	126.2	3.2X	336	823	6855	32290
	C-3obj	119.2	3.3X	312	810	6840	32226
	D-3obj	13.3	30.1X	312	812	6840	32228
	M-3obj	8.5	46.9X	312	812	6840	32229
fop	3obj	363.4	-	336	824	7591	34344
	ci	4.0	91.1X	911	1210	7997	40423
	CSC	4.5	80.0X	653	1121	7944	37562
	Z-3obj	120.3	3.0X	370	846	7621	34476
	C-3obj	108.1	3.4X	336	824	7591	34344
	D-3obj	13.0	27.9X	336	826	7591	34346
	M-3obj	8.4	43.2X	336	825	7591	34345
jd	3obj	Oom	-	-	-	-	-
	ci	10.0	>538.4X	2714	3750	17878	95602
	CSC	11.8	>459.2X	2127	3278	17354	87467
	Z-3obj	265.0	>20.4X	1600	2818	16979	81855
	C-3obj	281.9	>19.2X	1491	2751	16875	81318
	D-3obj	99.9	>54.1X	1491	2754	16875	81326
	M-3obj	39.7	>136.2X	1491	2754	16875	81327
lusearch	3obj	582.3	-	357	1112	7670	36437
	ci	4.2	139.6X	1032	1490	8094	43014
	CSC	4.8	121.6X	724	1411	8053	40001
	Z-3obj	180.8	3.2X	382	1136	7699	36571
	C-3obj	151.3	3.8X	357	1112	7670	36437
	D-3obj	14.5	40.2X	357	1115	7670	36440
	M-3obj	9.6	61.0X	357	1114	7670	36439
pmd	3obj	872.1	-	1333	2348	11851	59851
	ci	6.5	133.3X	2265	2950	12365	69505
	CSC	7.6	114.3X	1748	2710	12273	64834
	Z-3obj	287.6	3.0X	1374	2369	11882	59988
	C-3obj	170.7	5.1X	1333	2348	11851	59851
	D-3obj	27.8	31.3X	1333	2350	11851	59853
	M-3obj	19.5	44.6X	1333	2349	11851	59852
sunflow	3obj	Oom	-	-	-	-	-
	ci	8.2	>654.5X	2146	2553	13981	72716
	CSC	9.0	>600.7X	1660	2359	13797	67237
	Z-3obj	779.3	>6.9X	1116	1976	13382	61274
	C-3obj	606.2	>8.9X	1083	1976	13325	61154
	D-3obj	53.2	>101.4X	1071	1955	13294	60907
	M-3obj	20.3	>266.5X	1071	1954	13294	60919
trade-beans	3obj	4379.4	-	401	962	7989	38931
	ci	4.7	927.8X	1095	1420	8652	47885
	CSC	5.5	799.2X	786	1314	8546	44610
	Z-3obj	1685.7	2.6X	448	1015	8209	39991
	C-3obj	1152.1	3.8X	417	991	8040	39276
	D-3obj	18.1	241.6X	401	964	7989	38933
	M-3obj	11.3	387.9X	401	963	7991	38941

Program	Analysis	T	S	#MFC	#PCS	#RM	#CE
batik	3obj	Oom	-	-	-	-	-
	ci	13.3	>406.3X	3403	5810	20994	127604
	CSC	18.6	>290.2X	2846	5524	20819	121105
	Z-3obj	Oot	-	-	-	-	-
	C-3obj	Oom	-	-	-	-	-
	D-3obj	Oom	-	-	-	-	-
	M-3obj	777.5	>6.9X	2044	4930	20243	106779
bytecode-viewer	3obj	Oot	-	-	-	-	-
	ci	9.4	>571.4X	4723	5616	14637	110795
	CSC	21.7	>249.0X	4366	5547	14601	106506
	Z-3obj	Oot	-	-	-	-	-
	C-3obj	Oot	-	-	-	-	-
	D-3obj	Oot	-	-	-	-	-
	M-3obj	1858.6	>2.9X	3661	4195	14042	94460
classy-shark	3obj	3020.7	-	484	1106	9384	44166
	ci	5.3	566.7X	1323	1682	10178	56231
	CSC	7.1	425.4X	876	1400	9489	48577
	Z-3obj	1407.2	2.1X	512	1113	9407	44251
	C-3obj	1058.6	2.9X	484	1106	9384	44166
	D-3obj	21.9	137.9X	484	1108	9384	44168
	M-3obj	13.3	226.9X	484	1107	9386	44176
eclipse	3obj	Oom	-	-	-	-	-
	ci	18.8	>287.7X	5090	10672	23386	182975
	CSC	20.7	>261.4X	4282	10250	22977	172521
	Z-3obj	Oot	-	-	-	-	-
	C-3obj	Oom	-	-	-	-	-
	D-3obj	Oom	-	-	-	-	-
	M-3obj	3739.3	>1.4X	3484	9661	22572	161716
h2	3obj	1942.4	-	336	907	7557	36232
	ci	4.6	423.2X	943	1274	7959	42799
	CSC	5.2	376.4X	687	1194	7914	40294
	Z-3obj	650.3	3.0X	373	925	7590	36354
	C-3obj	427.4	4.5X	336	907	7557	36232
	D-3obj	17.0	114.5X	336	909	7558	36237
	M-3obj	10.0	193.9X	336	908	7558	36236
JPC	3obj	1438.2	-	1204	4101	15206	79620
	ci	10.3	139.5X	2252	4934	16138	94701
	CSC	11.4	125.9X	1790	4763	16034	87453
	Z-3obj	194.6	7.4X	1234	4139	15233	79803
	C-3obj	168.0	8.6X	1204	4102	15206	79633
	D-3obj	67.9	21.2X	1204	4105	15207	79641
	M-3obj	25.4	56.7X	1204	4119	15209	79684
mindustry	3obj	325.0	-	321	771	6495	30815
	ci	3.8	85.1X	865	1136	6907	36808
	CSC	4.3	75.2X	622	1051	6862	33970
	Z-3obj	111.3	2.9X	352	799	6540	30962
	C-3obj	109.1	3.0X	321	771	6495	30815
	D-3obj	11.5	28.2X	321	773	6495	30817
	M-3obj	7.7	42.4X	321	773	6495	30817
recaf	3obj	Oom	-	-	-	-	-
	ci	20.4	>265.1X	5127	7402	29038	187415
	CSC	34.0	>158.6X	4138	6702	28300	161535
	Z-3obj	Oom	-	-	-	-	-
	C-3obj	Oom	-	-	-	-	-
	D-3obj	1172.3	>4.6X	3037	5651	27467	140236
	M-3obj	498.0	>10.8X	3038	5646	27468	140249
tesseract	3obj	733.4	-	341	836	7743	37221
	ci	4.6	159.1X	1013	1291	8313	45068
	CSC	6.1	120.0X	747	1216	8270	41968
	Z-3obj	198.5	3.7X	372	861	7775	37339
	C-3obj	185.6	4.0X	341	836	7743	37221
	D-3obj	14.1	52.1X	341	838	7743	37223
	M-3obj	9.4	77.8X	341	838	7743	37223
xalan	3obj	Oom	-	-	-	-	-
	ci	5.2	>1032.5X	1298	2084	10092	54001
	CSC	6.0	>892.6X	884	1967	10034	50747
	Z-3obj	1545.4	>3.5X	561	1658	9696	46887
	C-3obj	464.5	>11.6X	539	1636	9665	46753
	D-3obj	31.7	>170.2X	539	1639	9665	46756
	M-3obj	15.7	>343.5X	539	1638	9665	46755