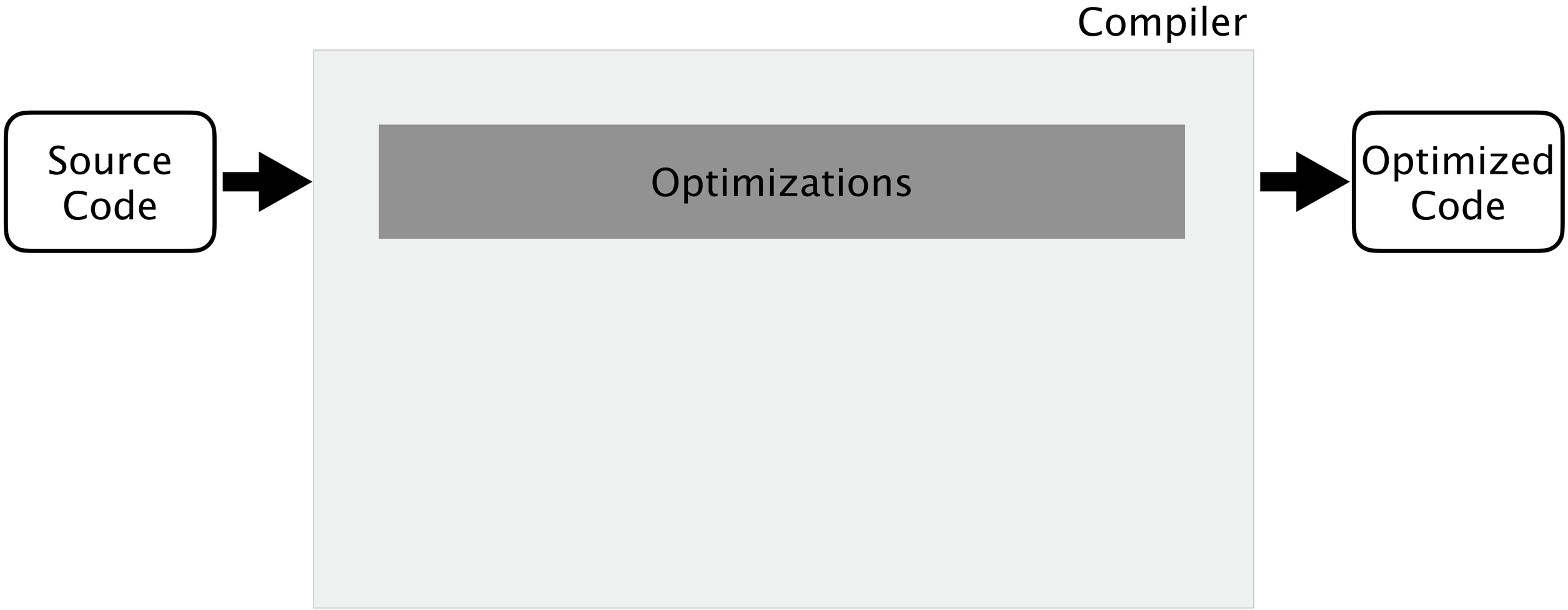
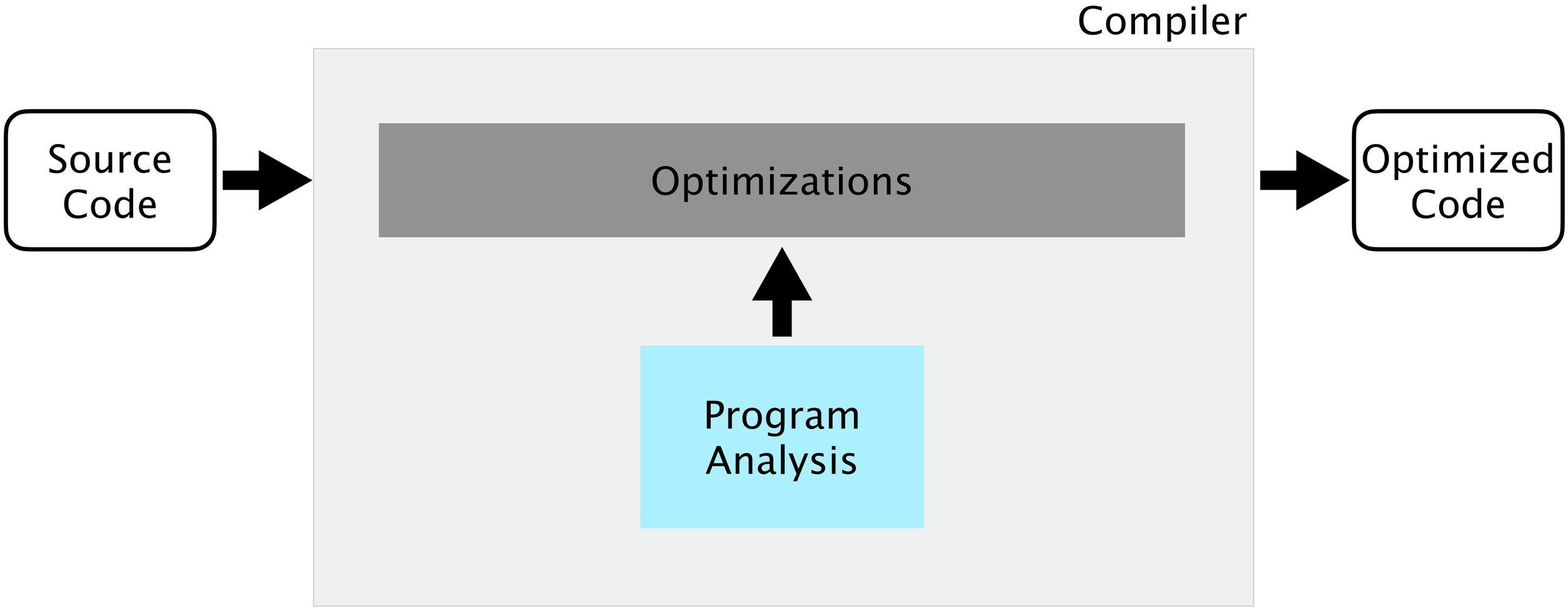
SCAF: A Speculation-Aware Collaborative Dependence Analysis Framework

Sotiris Apostolakis, Ziyang Xu, Zujun Tan, Greg Chan, Simone Campanoni[†], and David I. August

PLDI 2020

[†]Northwestern University


Compiler
Source Code
Optimizations
Optimized Code


Compiler
Source Code
Optimizations
Program Analysis
Optimized Code

Memory Analysis limits applicability of optimizations

Memory Analysis limits applicability of optimizations

- undecidable in theory [Landi, LPLS'92]
  For any fixed analysis algorithm, there is a counter-example input for which the algorithm is imprecise.

Memory Analysis limits applicability of optimizations

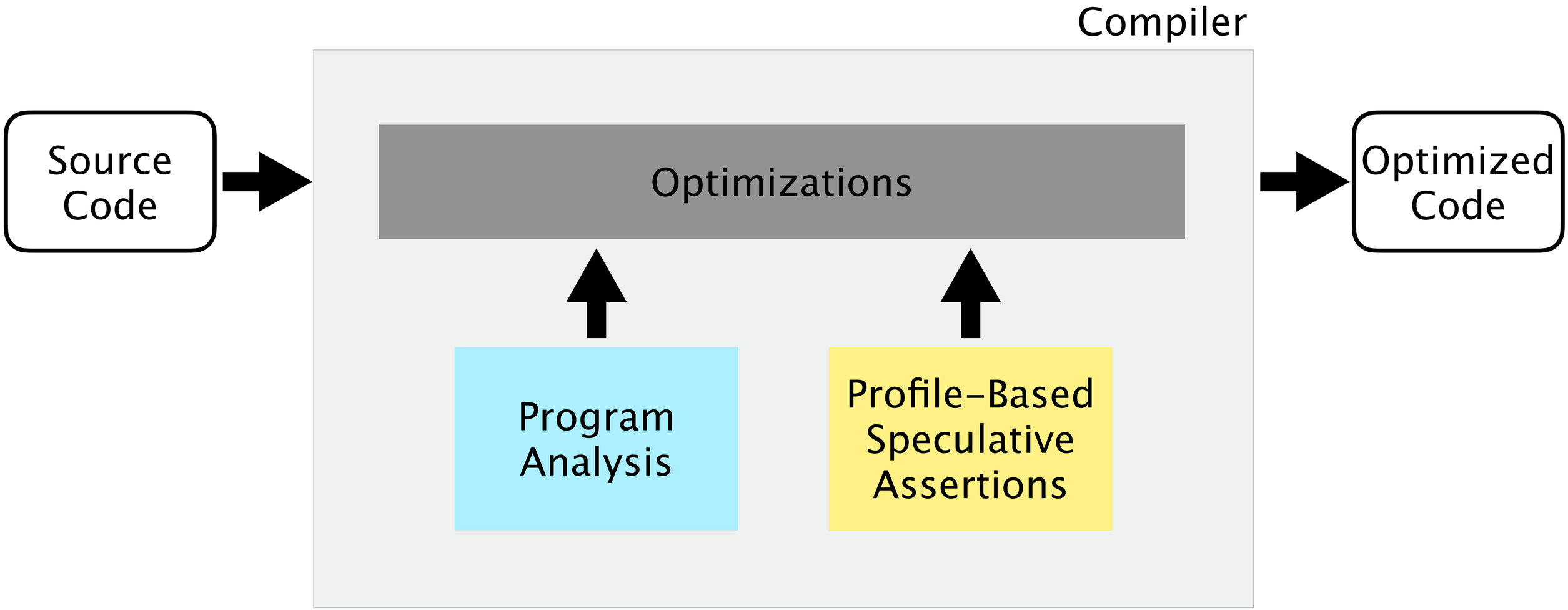
- undecidable in theory [Landi, LPLS'92]
  For any fixed analysis algorithm, there is a counter-example input for which the algorithm is imprecise.
- insufficiently precise in practice [Hind, PASTE'01]
  especially for languages like C/C++.

Memory Analysis limits applicability of optimizations

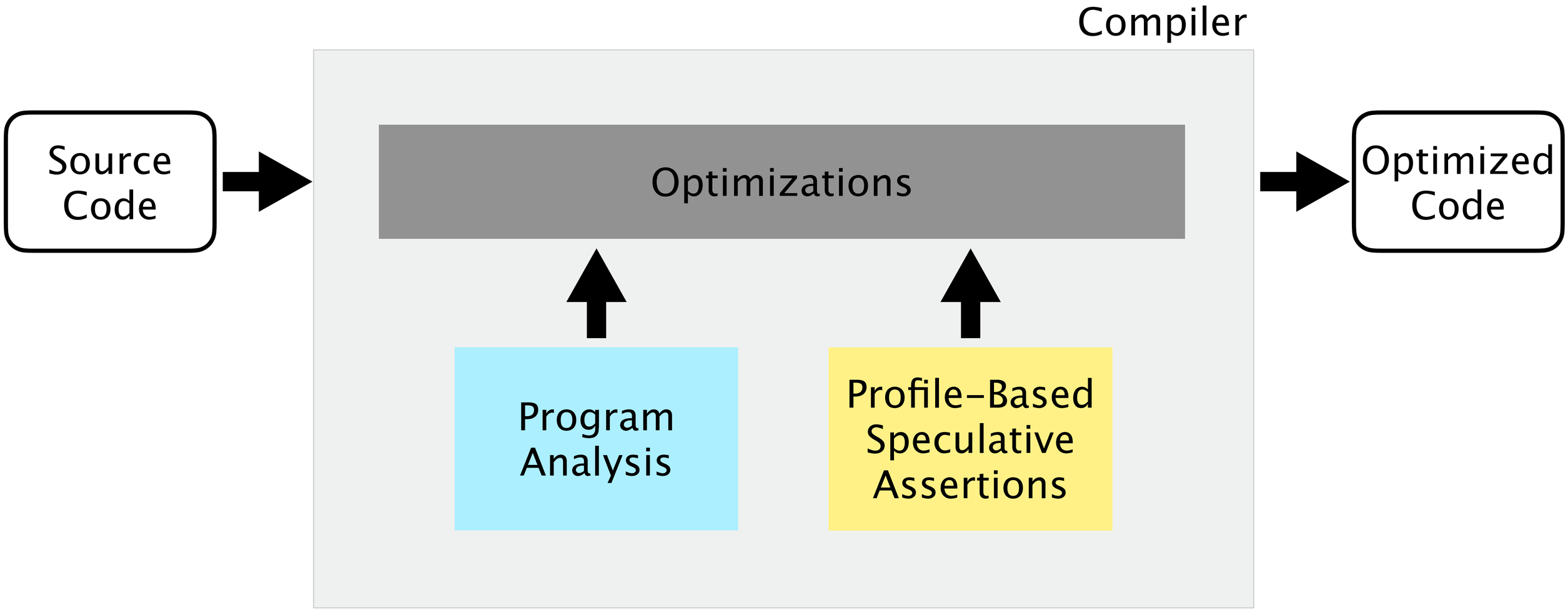
- undecidable in theory [Landi, LPLS'92]
  For any fixed analysis algorithm, there is a counter-example input for which the algorithm is imprecise.
- insufficiently precise in practice [Hind, PASTE'01]
  especially for languages like C/C++.
- conservatively respects all possible inputs
  Many real dependences rarely occur in practice.

Speculation enables optimization of the expected case

Speculation enables optimization of the expected case

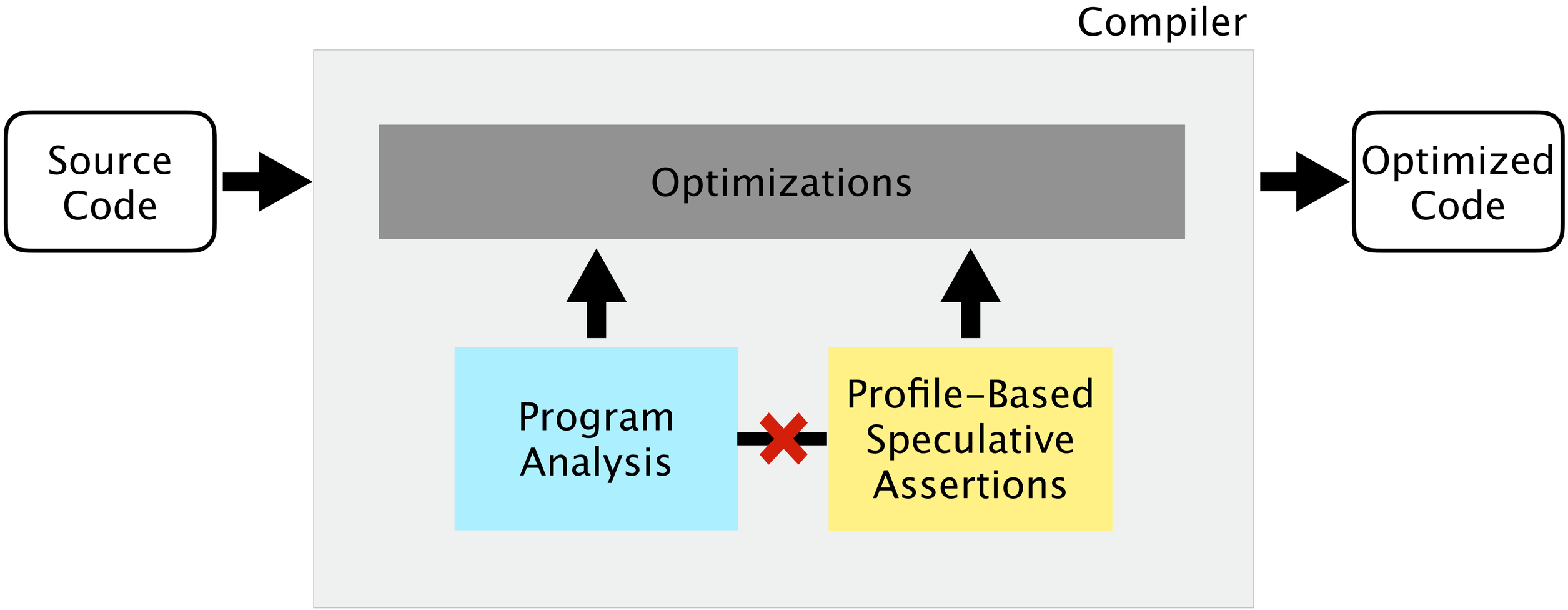


State-of-the-art [1,2,3,4] does not fully leverage speculation



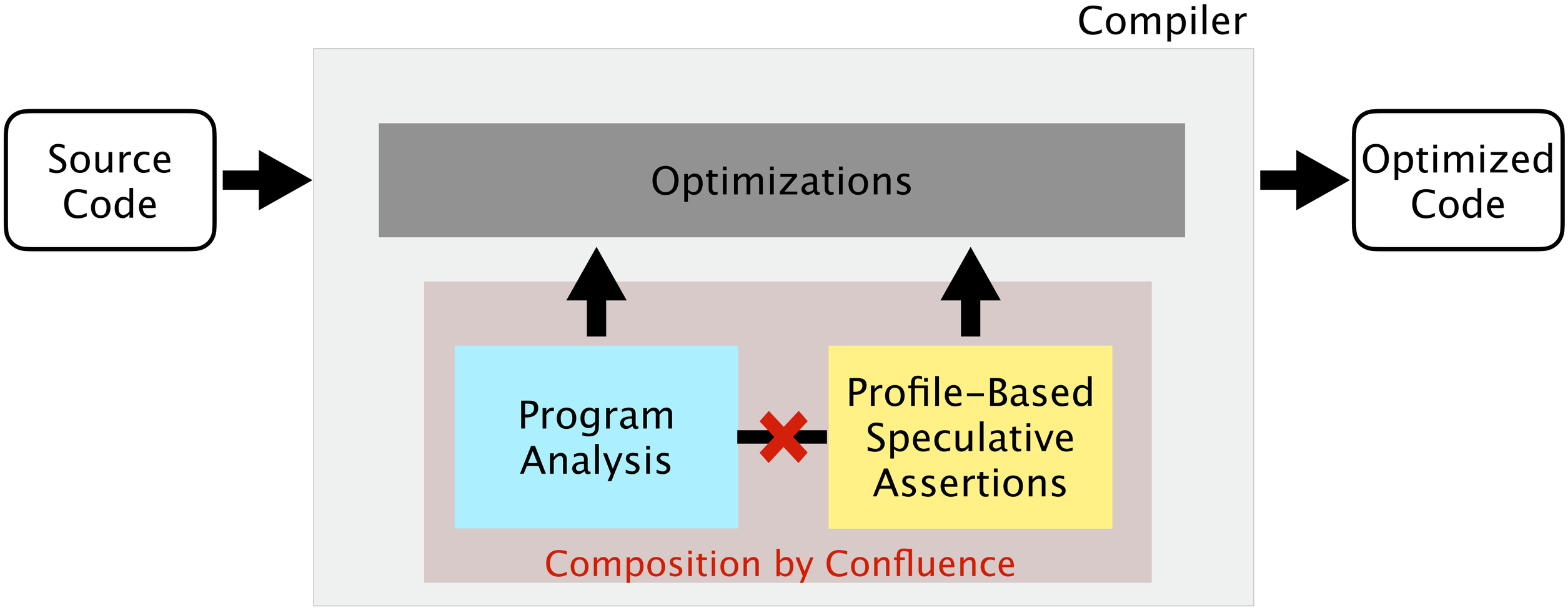
[1] Johnson et al., PLDI '12 [2] Kim et al., CGO '12 [3] Mehrara et al., PLDI '09 4 Vachharajani et al., PACT '07

State-of-the-art [1,2,3,4] does not fully leverage speculation



[1] Johnson et al., PLDI '12 [2] Kim et al., CGO '12 [3] Mehrara et al., PLDI '09 4 Vachharajani et al., PACT '07

State-of-the-art [1,2,3,4] does not fully leverage speculation



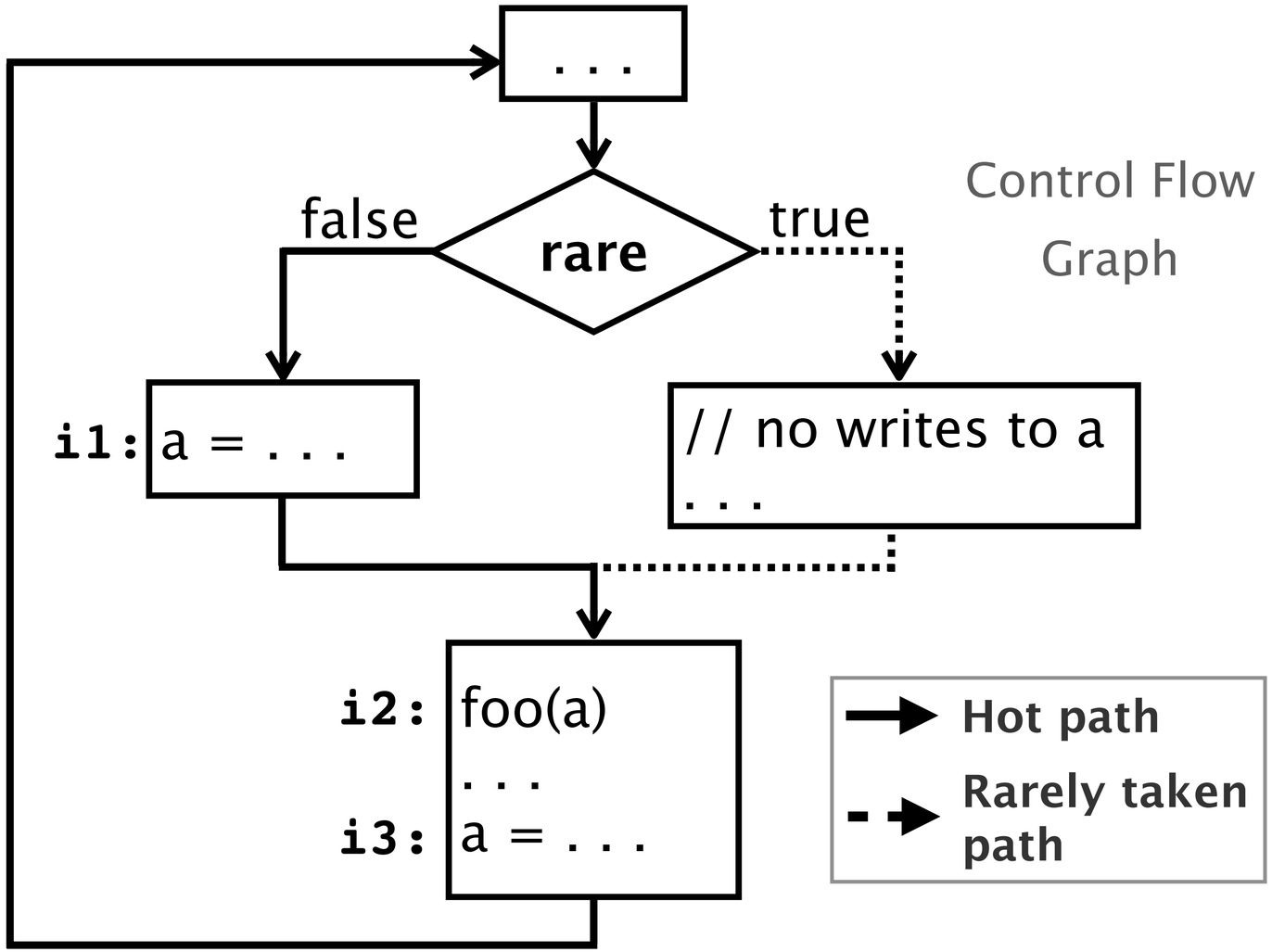
[1] Johnson et al., PLDI '12 [2] Kim et al., CGO '12 [3] Mehrara et al., PLDI '09 4 Vachharajani et al., PACT '07

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

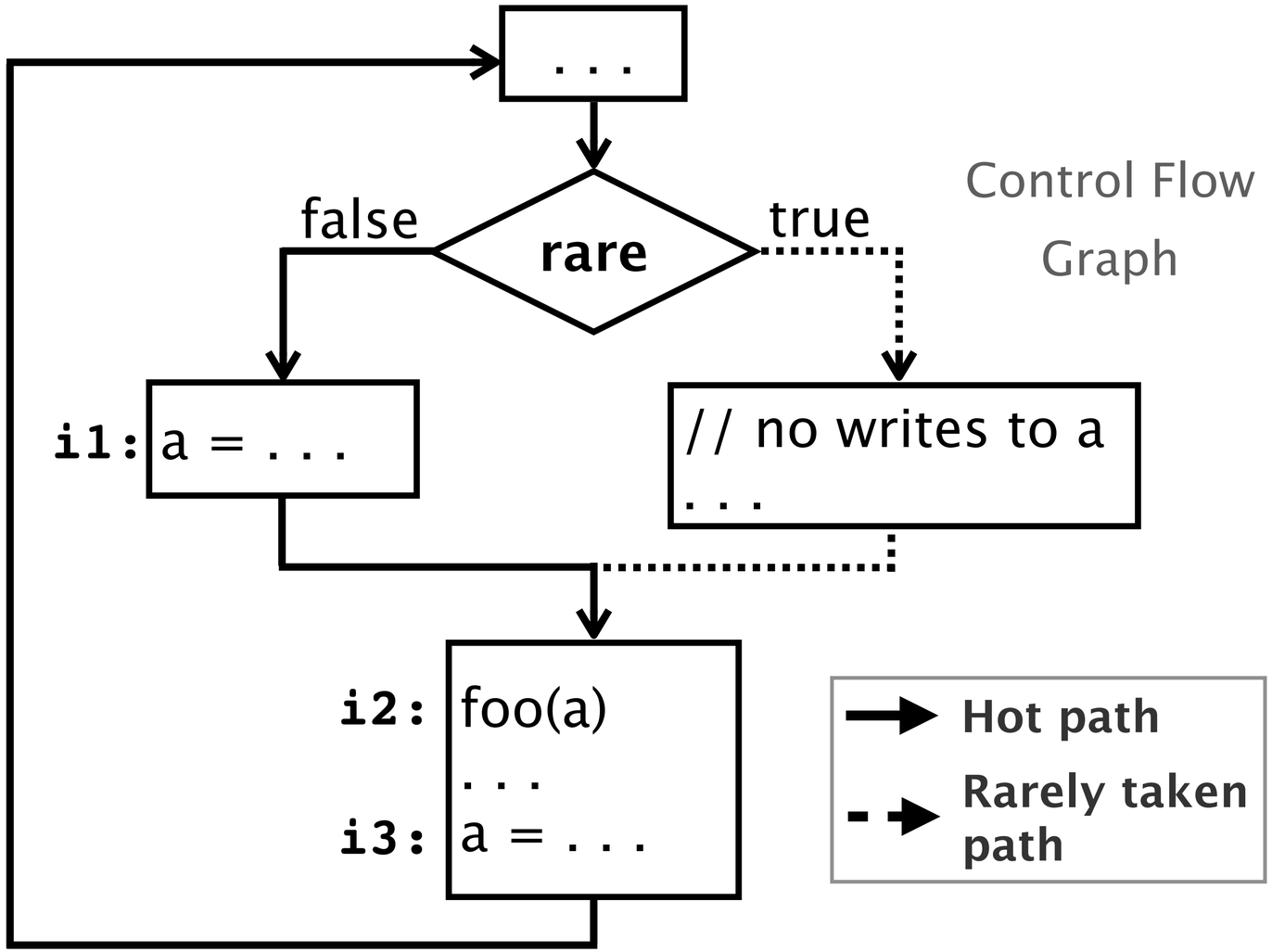


Control Flow Graph

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Is there a cross-iteration data flow from i3 to i2?

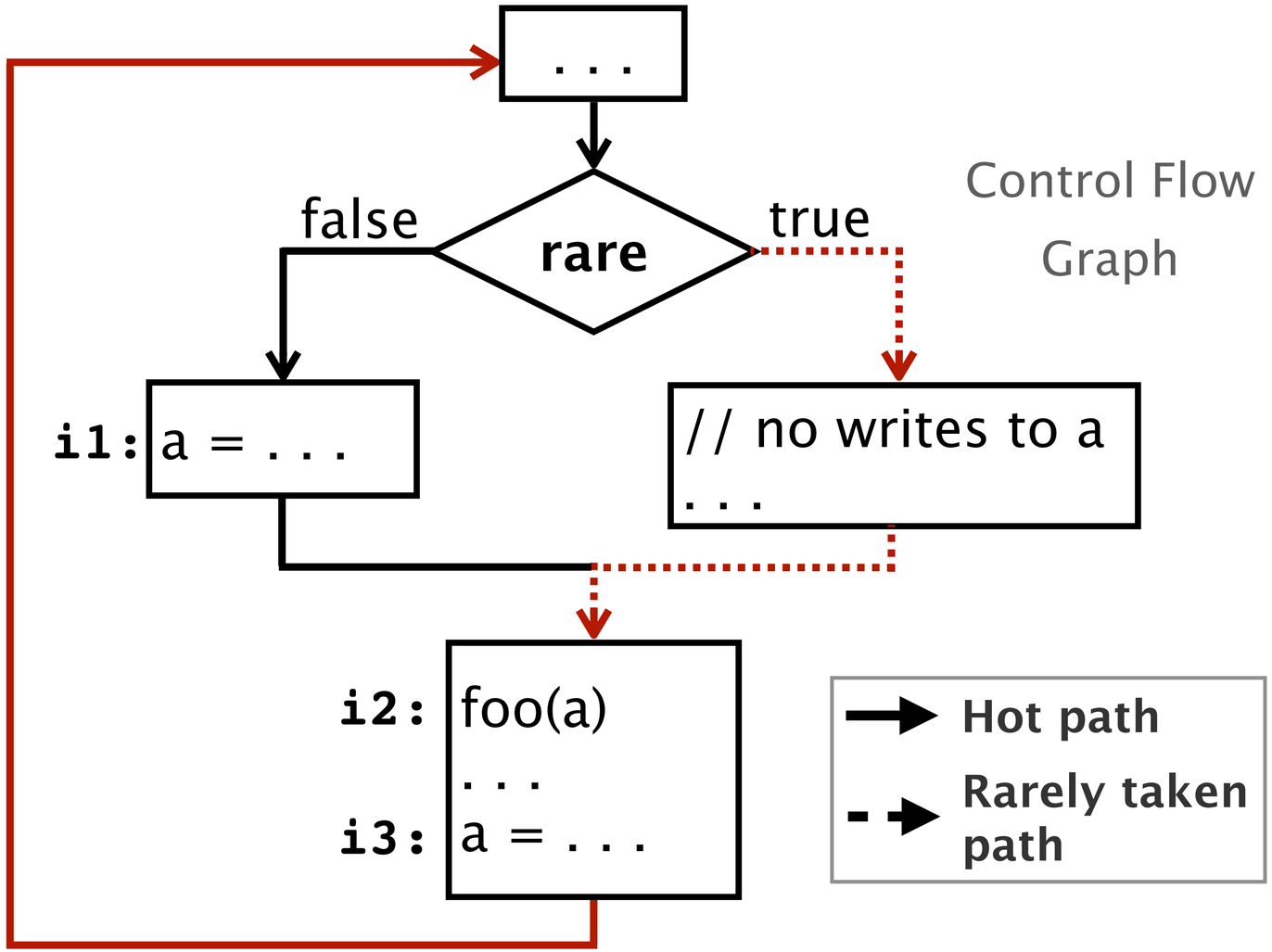


Control Flow Graph

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Is there a cross-iteration data flow from i3 to i2?



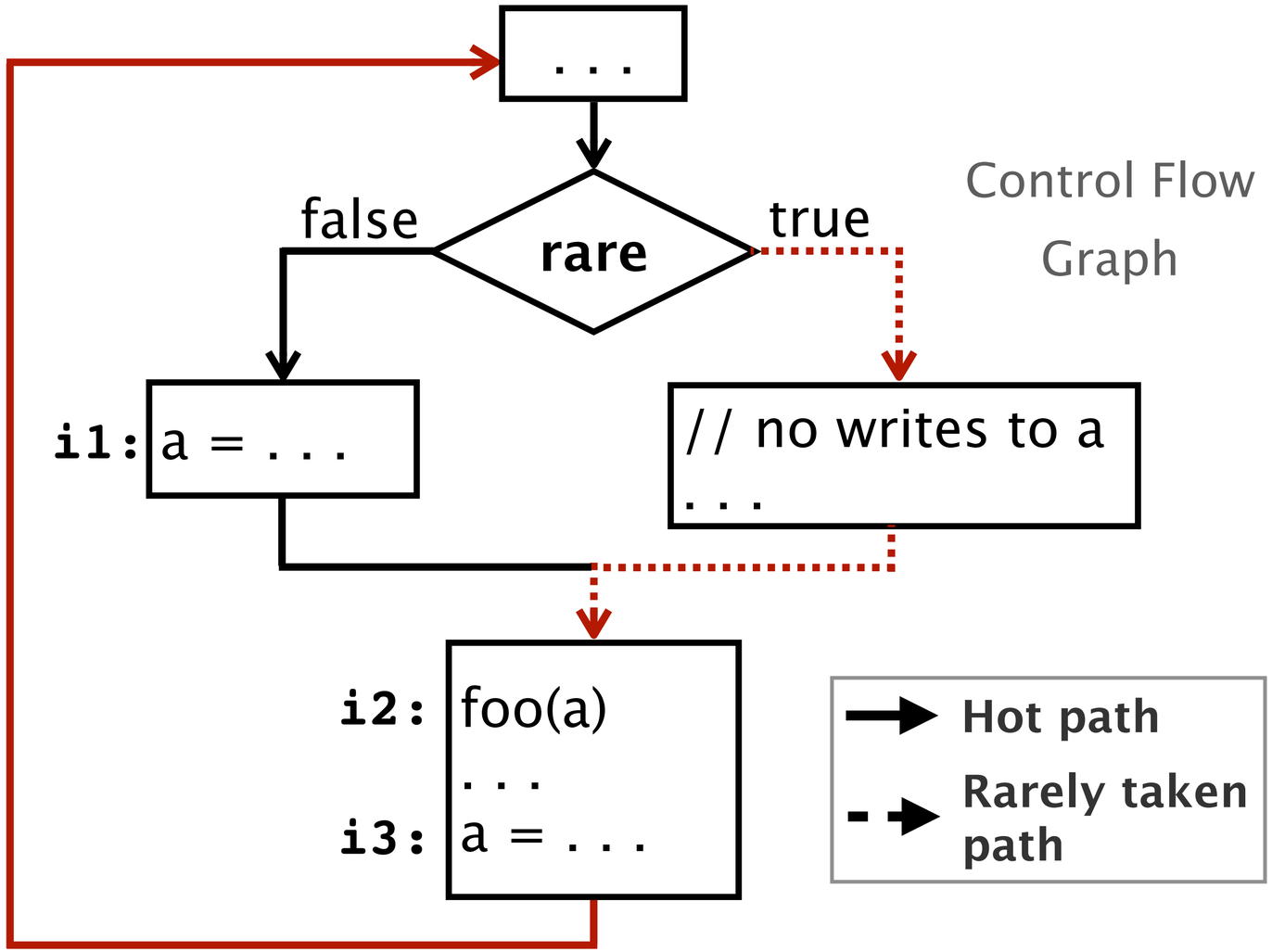
Control Flow Graph

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Is there a
cross-iteration
data flow
from i3 to i2?

Composition by Confluence
cannot assert its absence.

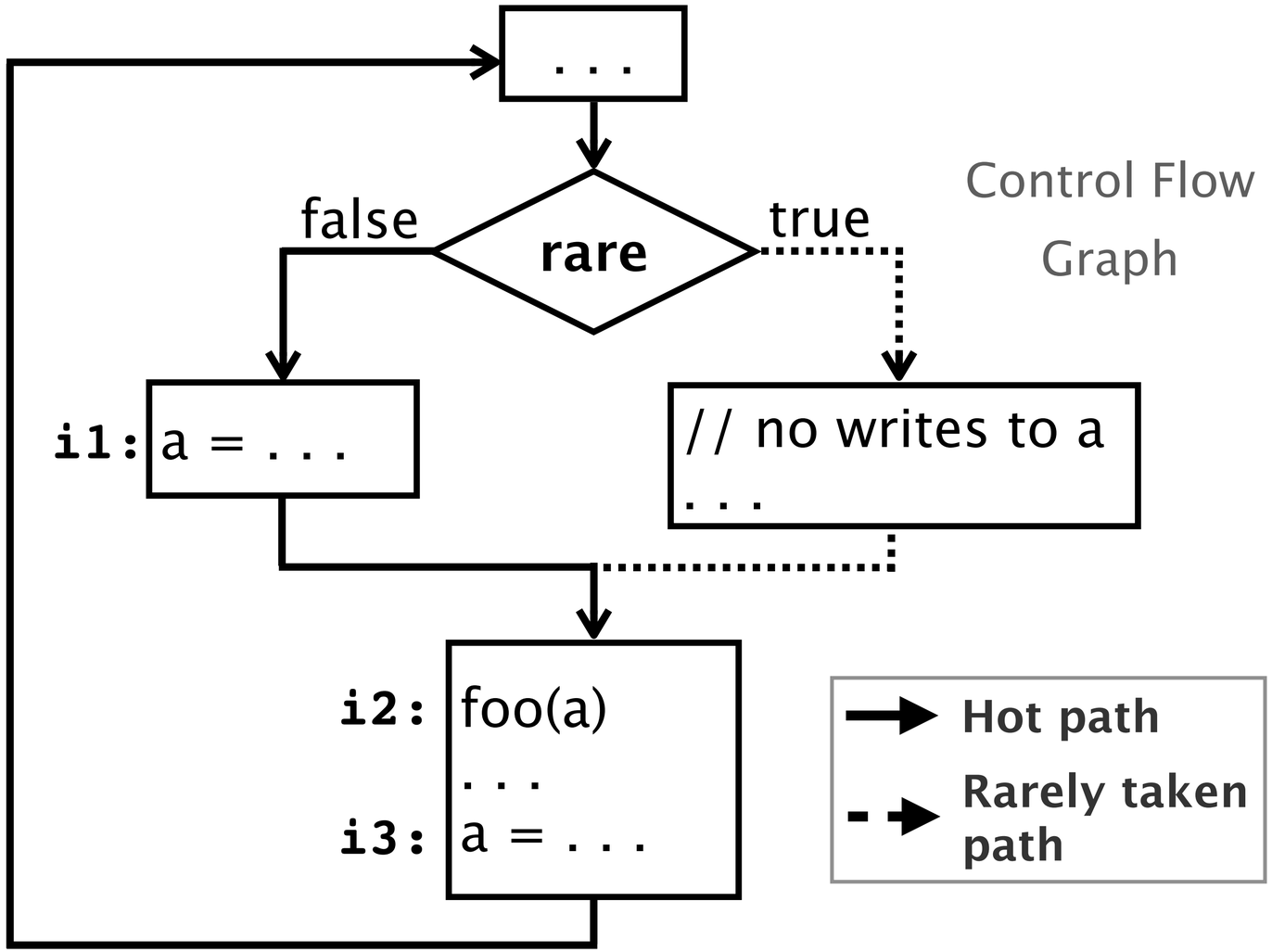


Control Flow Graph

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Is there a cross-iteration data flow from i3 to i2?

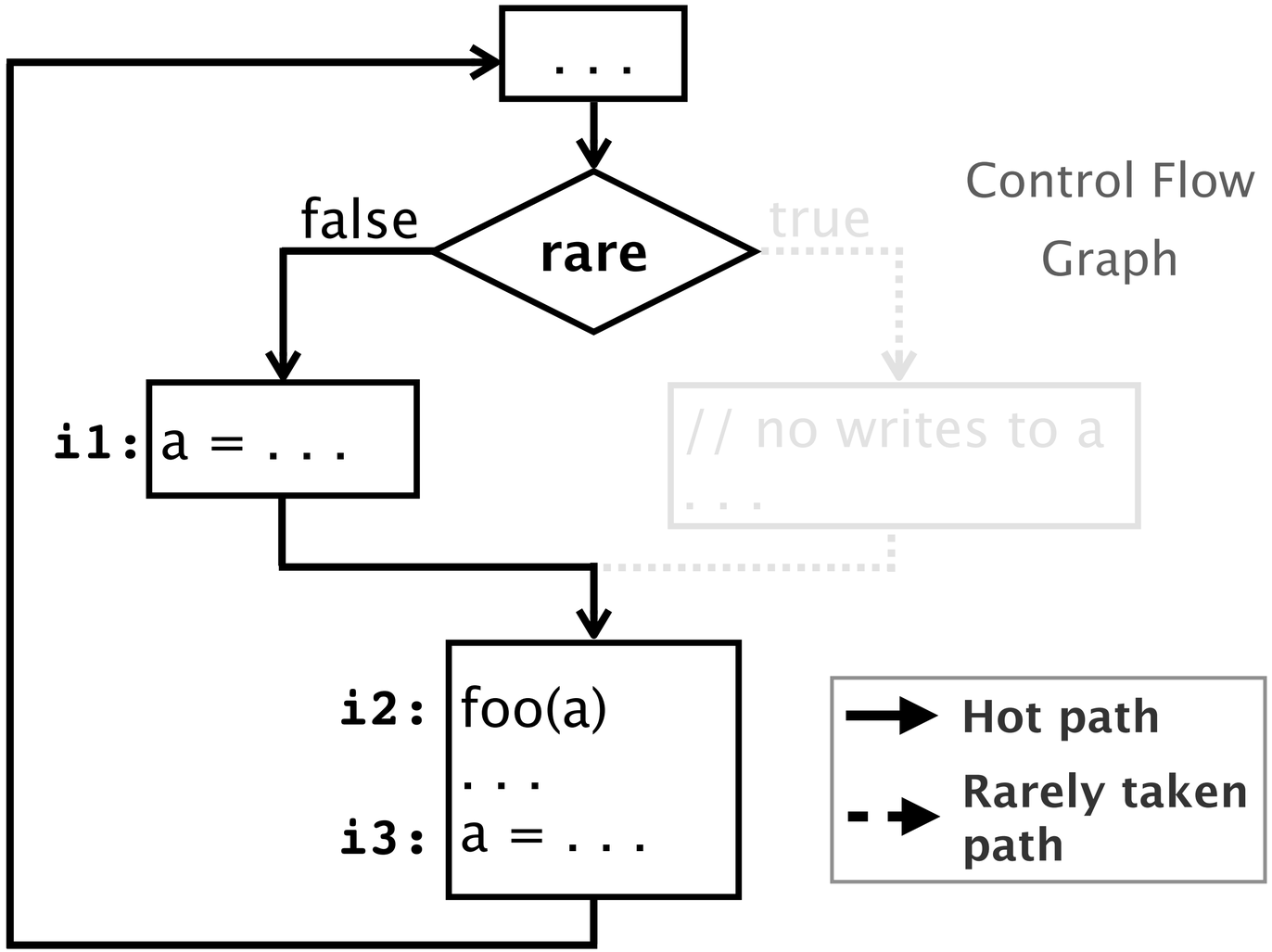


Control Flow Graph

Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

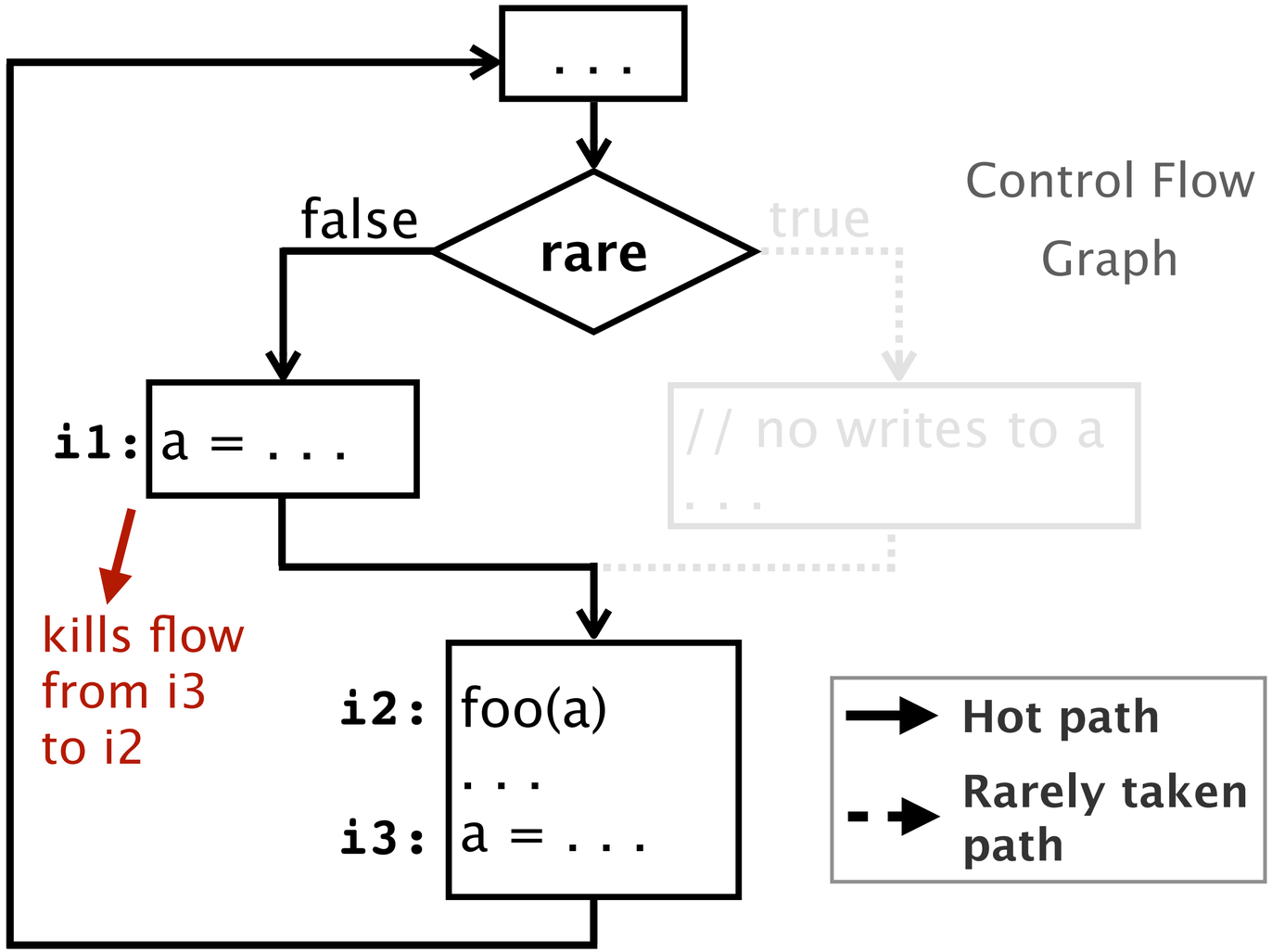
Is there a cross-iteration data flow from i3 to i2?



Motivating Example

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Is there a cross-iteration data flow from i3 to i2?

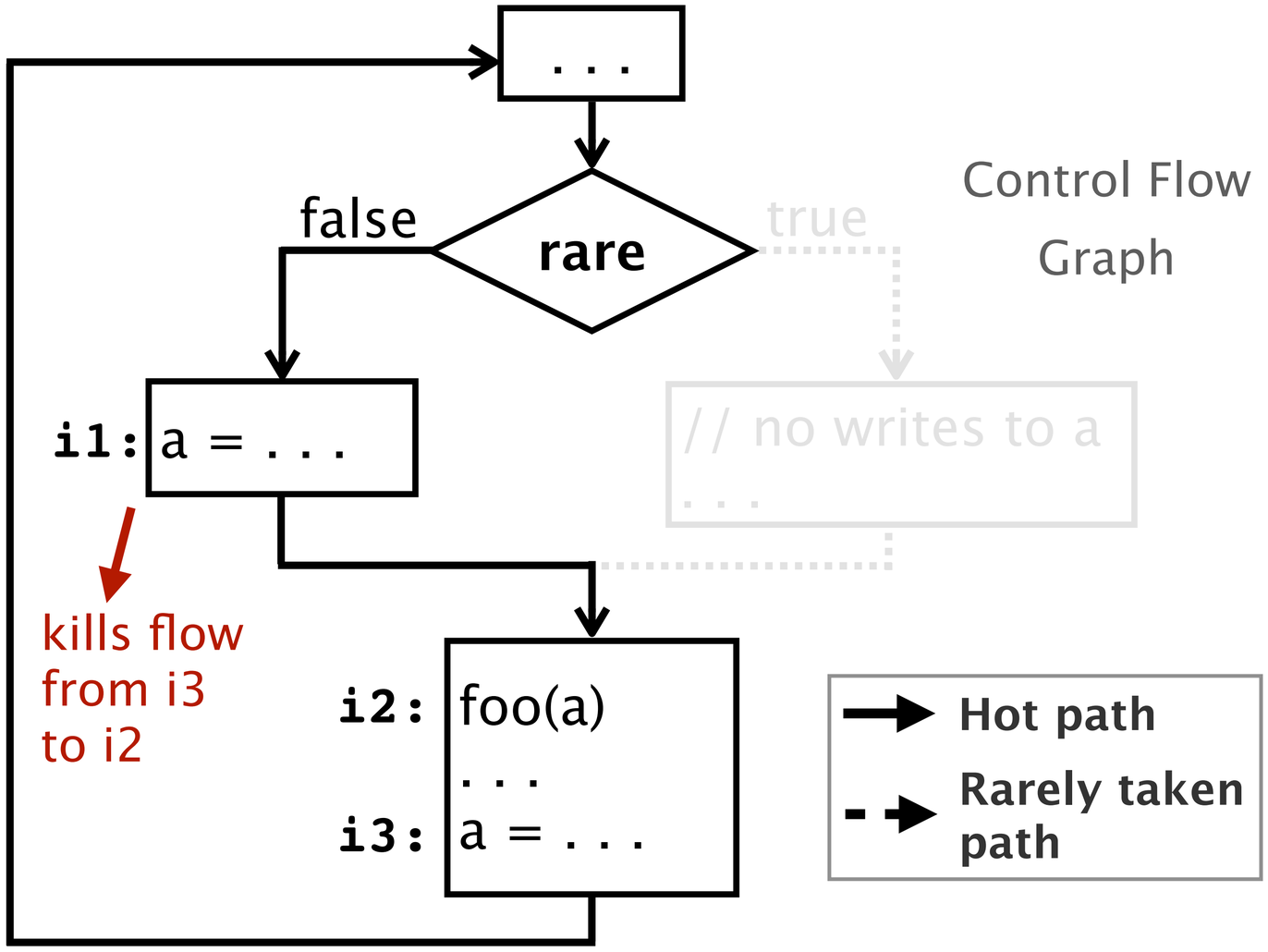


Motivating Example

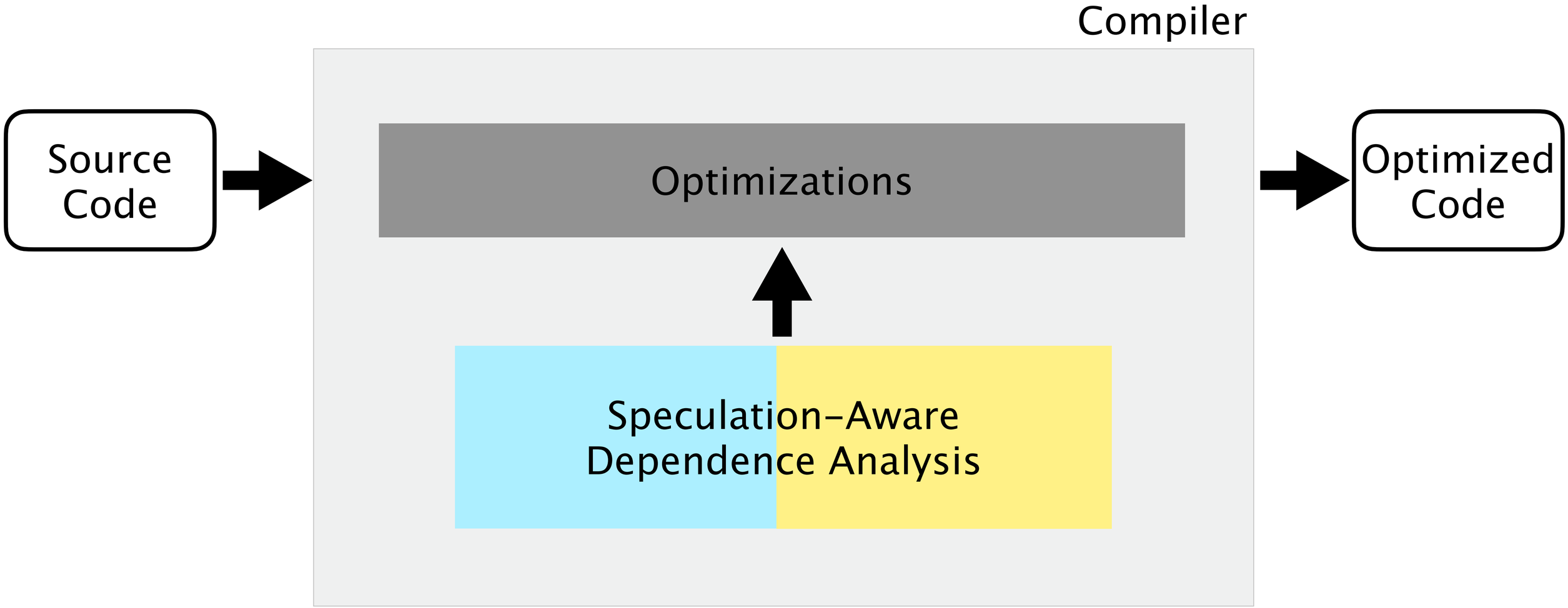
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Is there a cross-iteration data flow from i3 to i2?

Memory analysis and speculation combined can assert its absence.



Monolithic Integration [1,2,3]



[1] Apostolakis et al., ASPLOS '20 [2] Devecsery et al., ASPLOS '18 [3] Fernandez et al., PACT '02

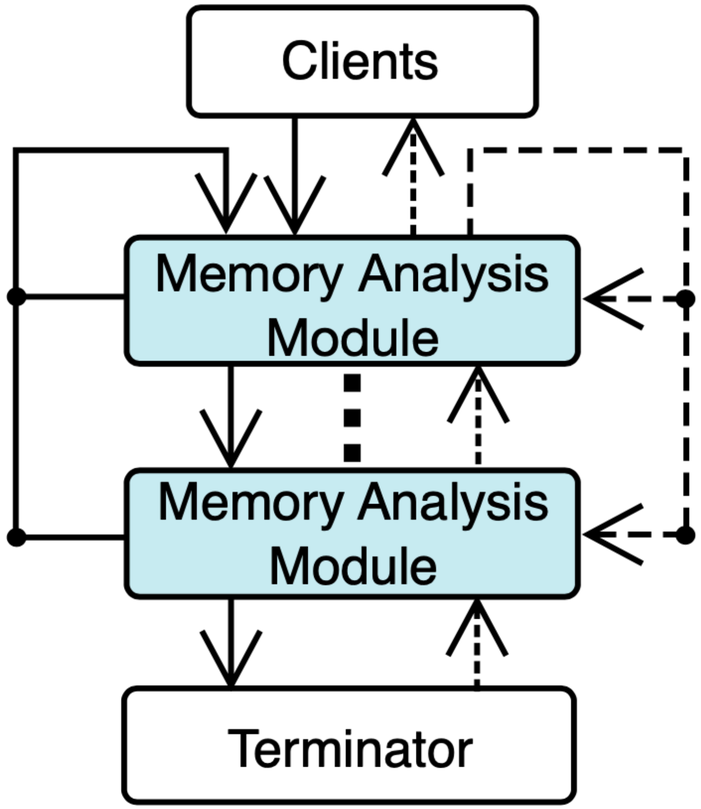
Proposed Approach: Composition by Collaboration

Proposed Approach is both Modular & Collaborative

Approaches	Supported Forms of Collaboration		Memory Analysis Decoupled from Speculation
	Among Speculative Techniques	Between Memory Analysis and Speculative Techniques	
Monolithic Integration [1,2,3]	✗	✓	✗
Composition by Confluence [4,5,6,7]	✗	✗	✓
Composition by Collaboration (**This Work**)	✓	✓	✓

[1] Apostolakis et al., ASPLOS '20 [2] Devecsery et al., ASPLOS '18 [3] Fernandez et al., PACT '02
[4] Johnson et al., PLDI '12 [5] Kim et al., CGO '12 [6] Mehrara et al., PLDI '09 [7] Vachharajani et al., PACT '07

CAF*: Collaborative Dependence Analysis Framework

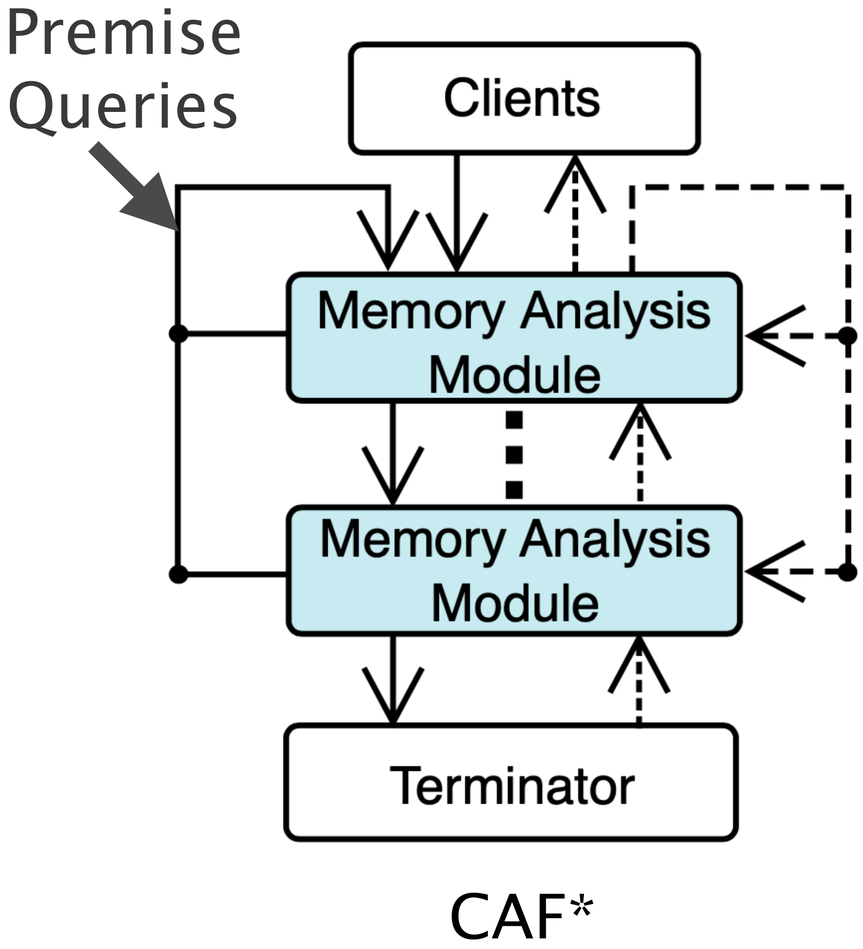


CAF*

Collaborative resolution of analysis queries by simple analysis algorithms

* Nick P. Johnson et al., Collaborative Dependence Analysis Framework in CGO '17

CAF*: Collaborative Dependence Analysis Framework



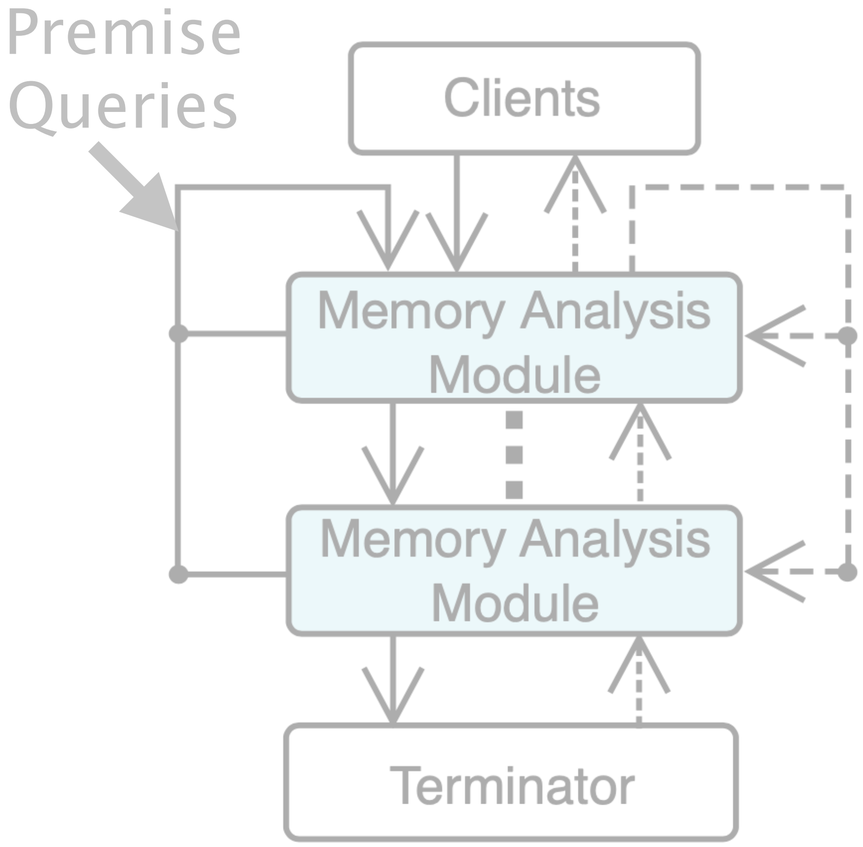
CAF*

Collaborative resolution of analysis queries by simple analysis algorithms

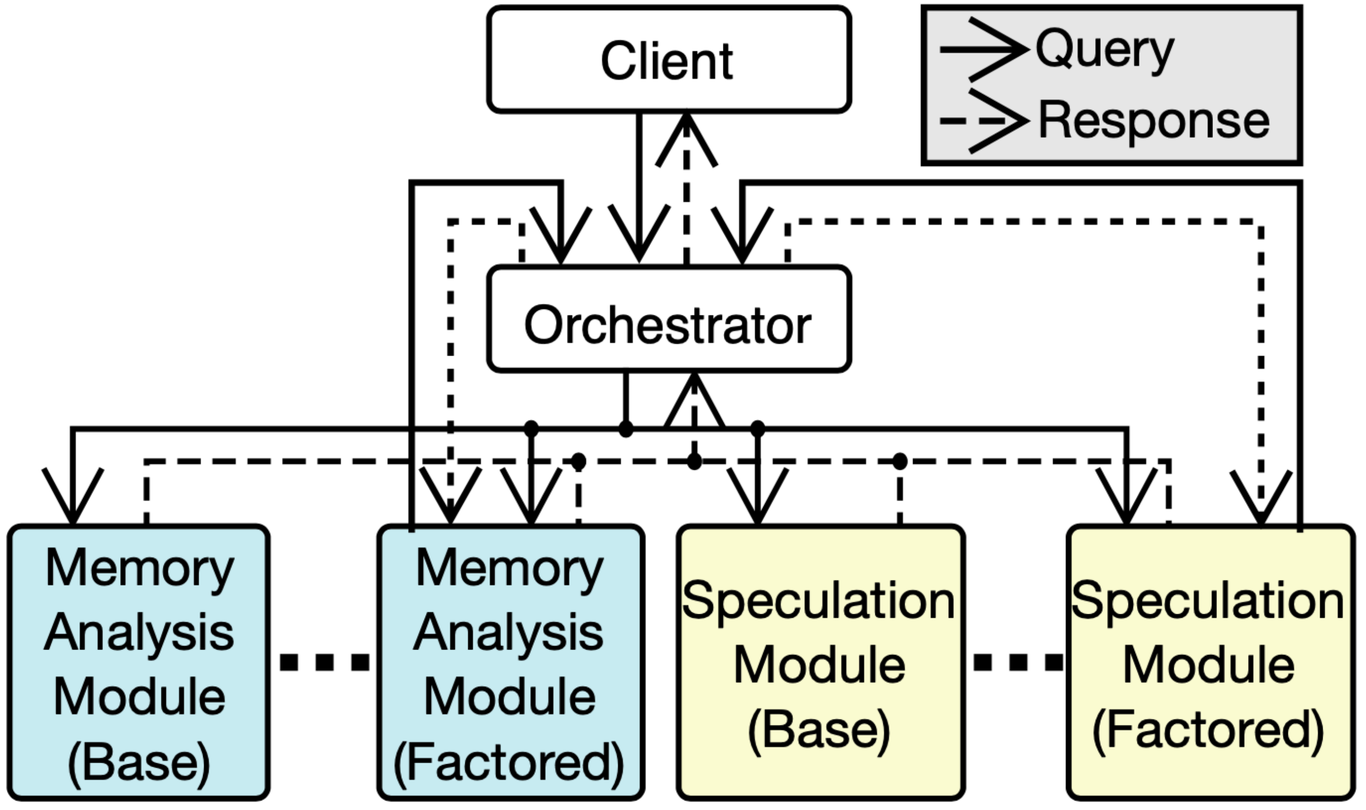
Isolate propositions beyond ones module's logic as premise queries.

* Nick P. Johnson et al., Collaborative Dependence Analysis Framework in CGO '17

SCAF: Speculation-Aware Collaborative Dependence Analysis Framework



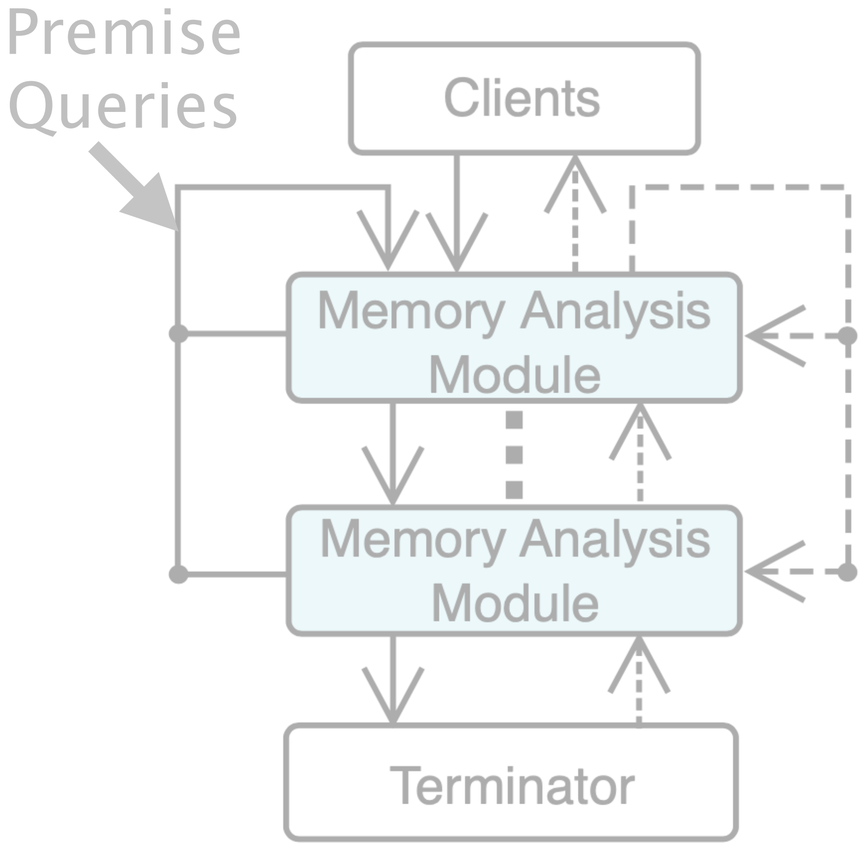
CAF*



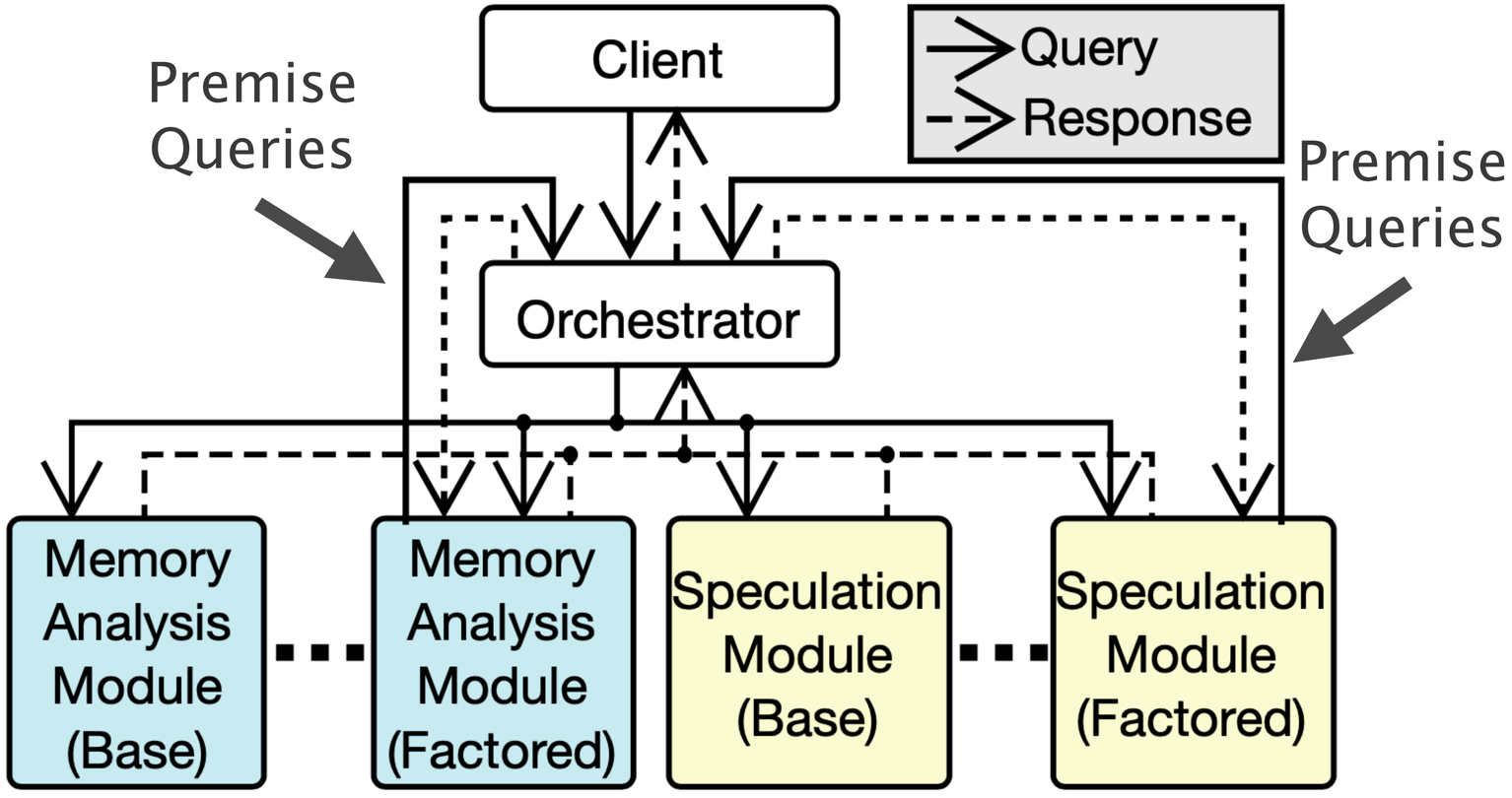
SCAF

* Nick P. Johnson et al., Collaborative Dependence Analysis Framework in CGO '17

SCAF: Speculation-Aware Collaborative Dependence Analysis Framework



CAF*



SCAF

* Nick P. Johnson et al., Collaborative Dependence Analysis Framework in CGO '17

SCAF’s Query Language

SCAF's Query Language

Query Response:

Analysis result might be predicated on speculative assertions

SCAF's Query Language

Query Response:
    Analysis result might be predicated on speculative assertions

New Query Parameters:
    Control-flow parameter in the form of dominance information

SCAF's Query Language

Query Response:

Analysis result might be predicated on speculative assertions

New Query Parameters:

Control-flow parameter in the form of dominance information

Desired result parameter for quick bail-out

SCAF in action

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

SCAF in action

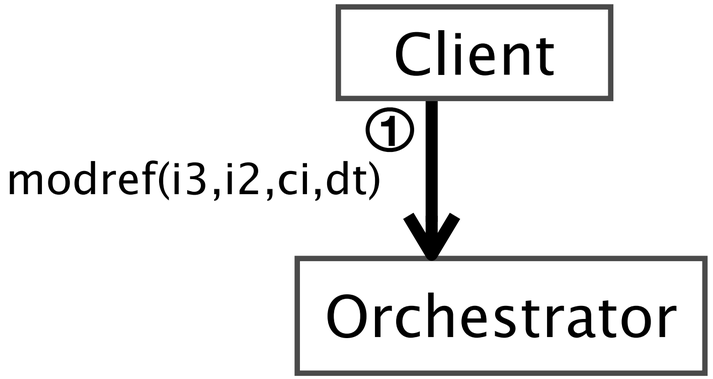
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

Client

Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

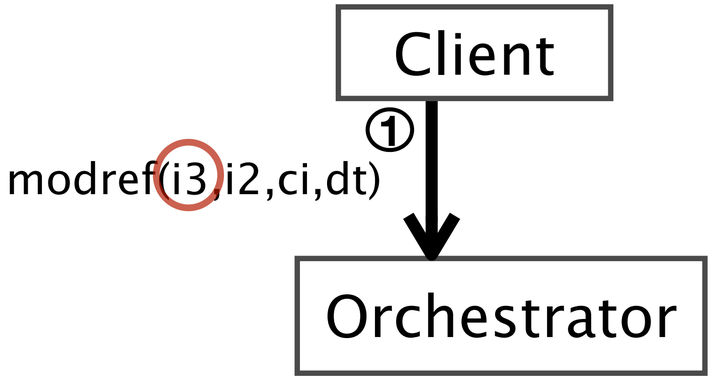
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

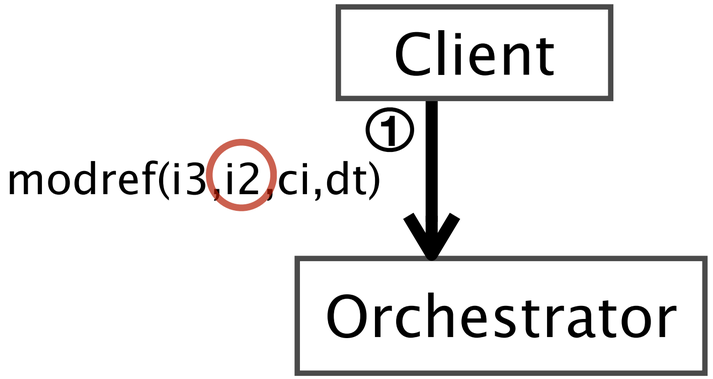
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

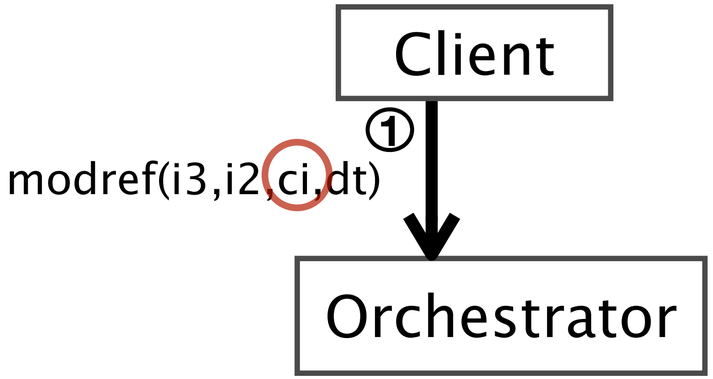
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

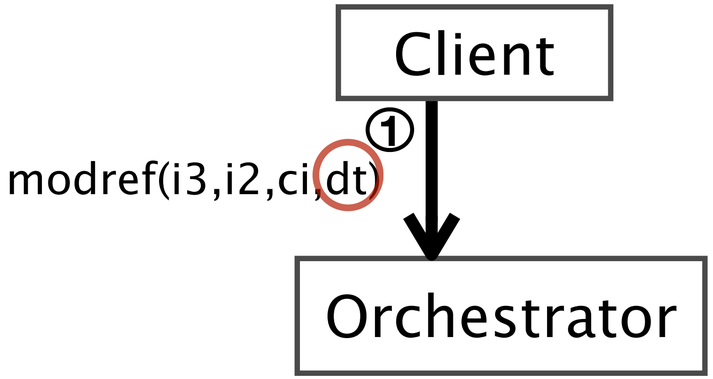
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

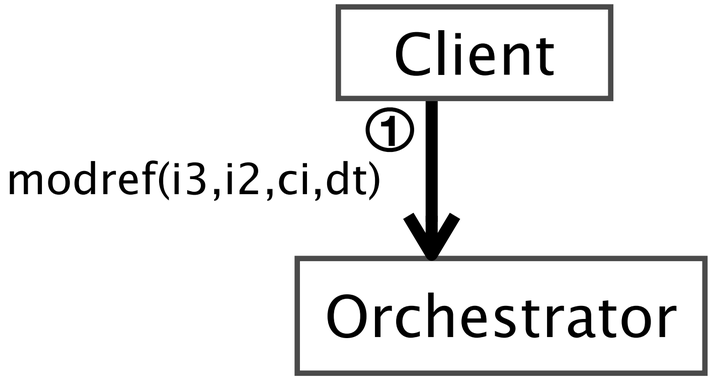
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

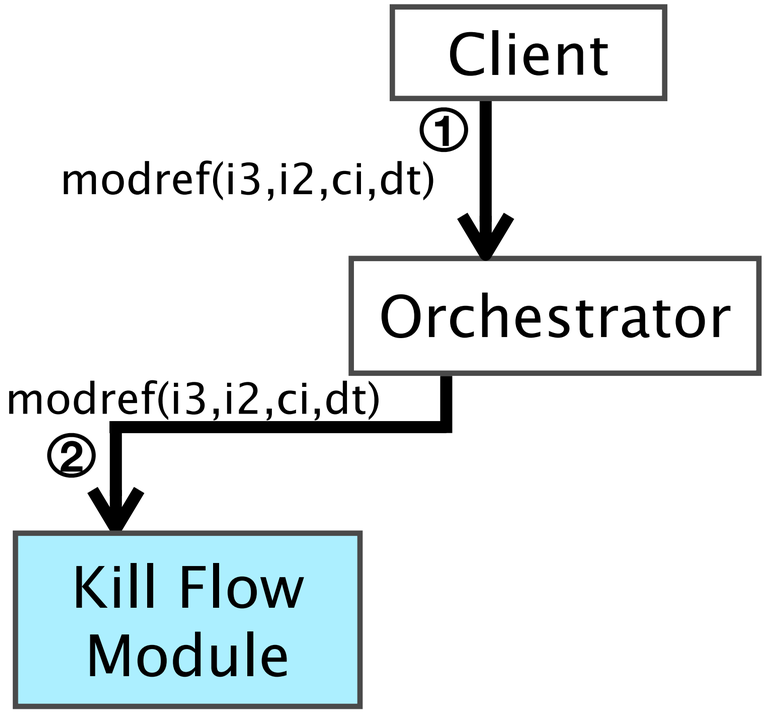
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

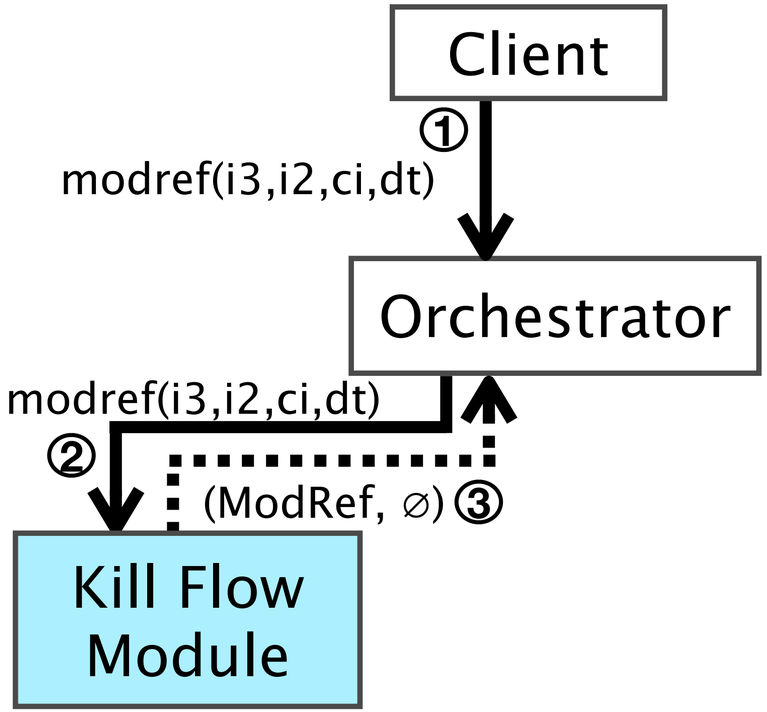
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

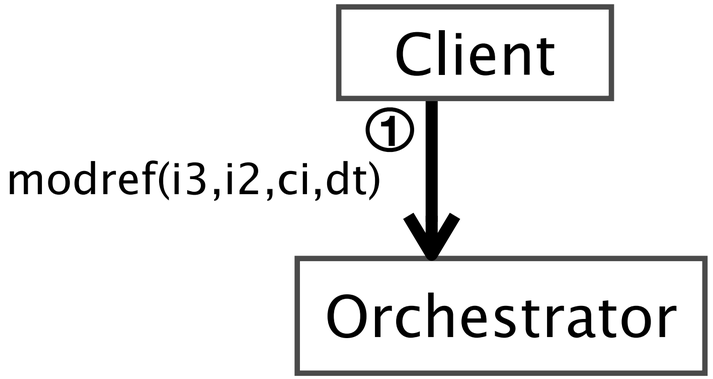
```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



Is there a
cross-iter
data flow
from i3 to i2?

SCAF in action

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```



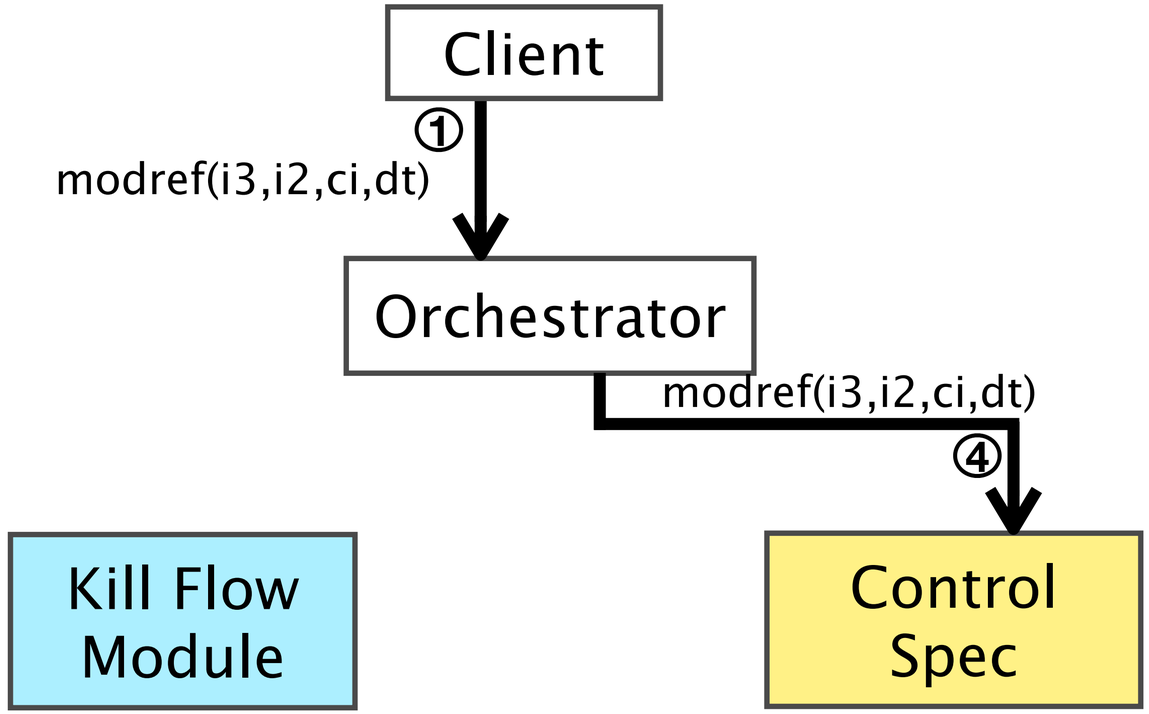
Kill Flow Module

Is there a cross-iter data flow from i3 to i2?

SCAF in action

```
   loop L:
     if (rare)
       // no writes to a
       ...
     else
i1:    a = ...
i2:  foo(a)
     ...
i3:  a = ...
```

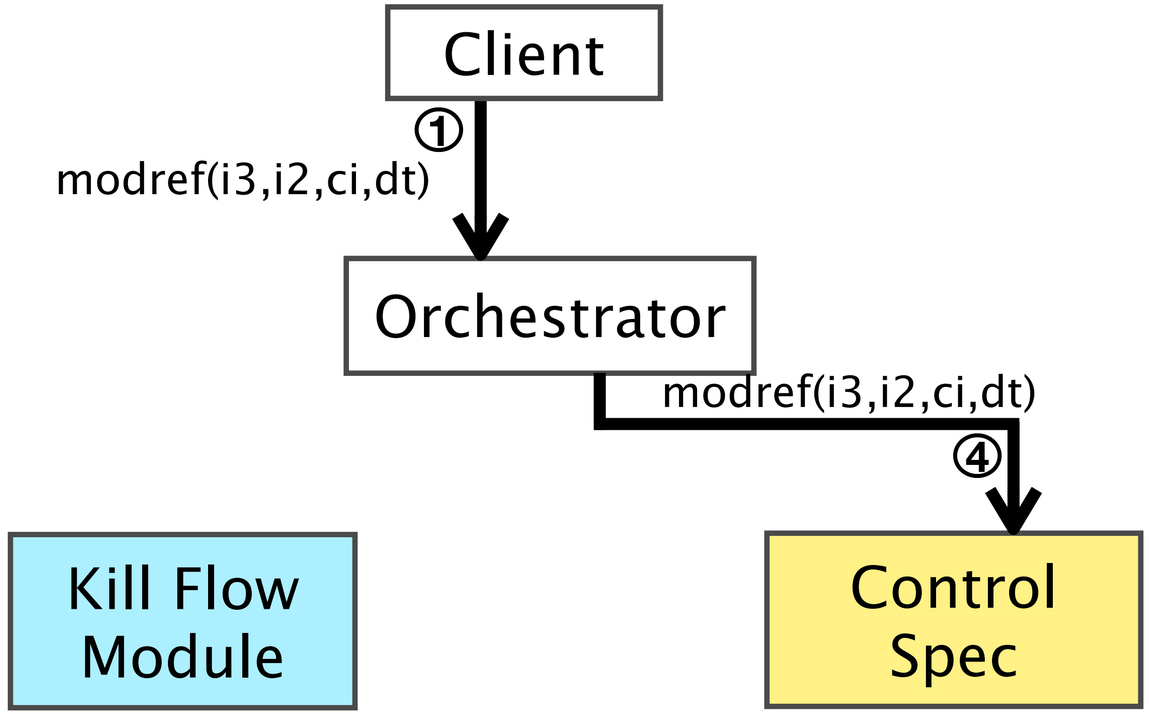
Is there a
cross-iter
data flow
from i3 to i2?



SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

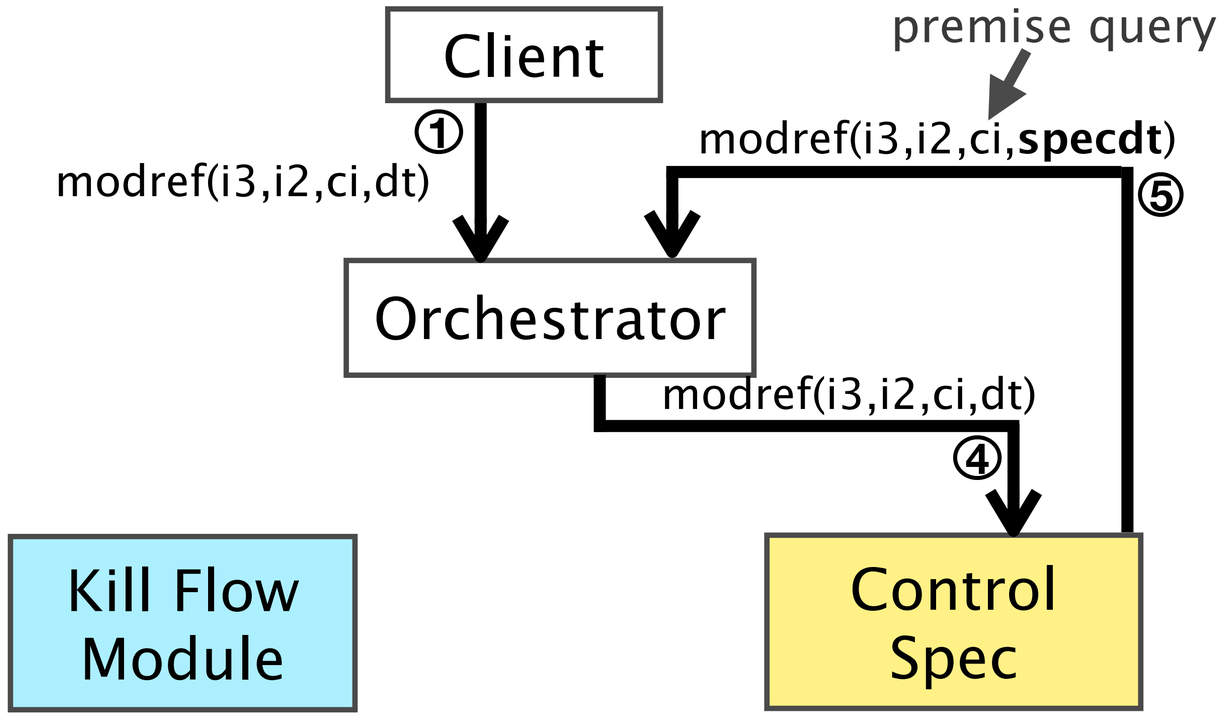
Is there a
cross-iter
data flow
from i3 to i2?



SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a
cross-iter
data flow
from i3 to i2?

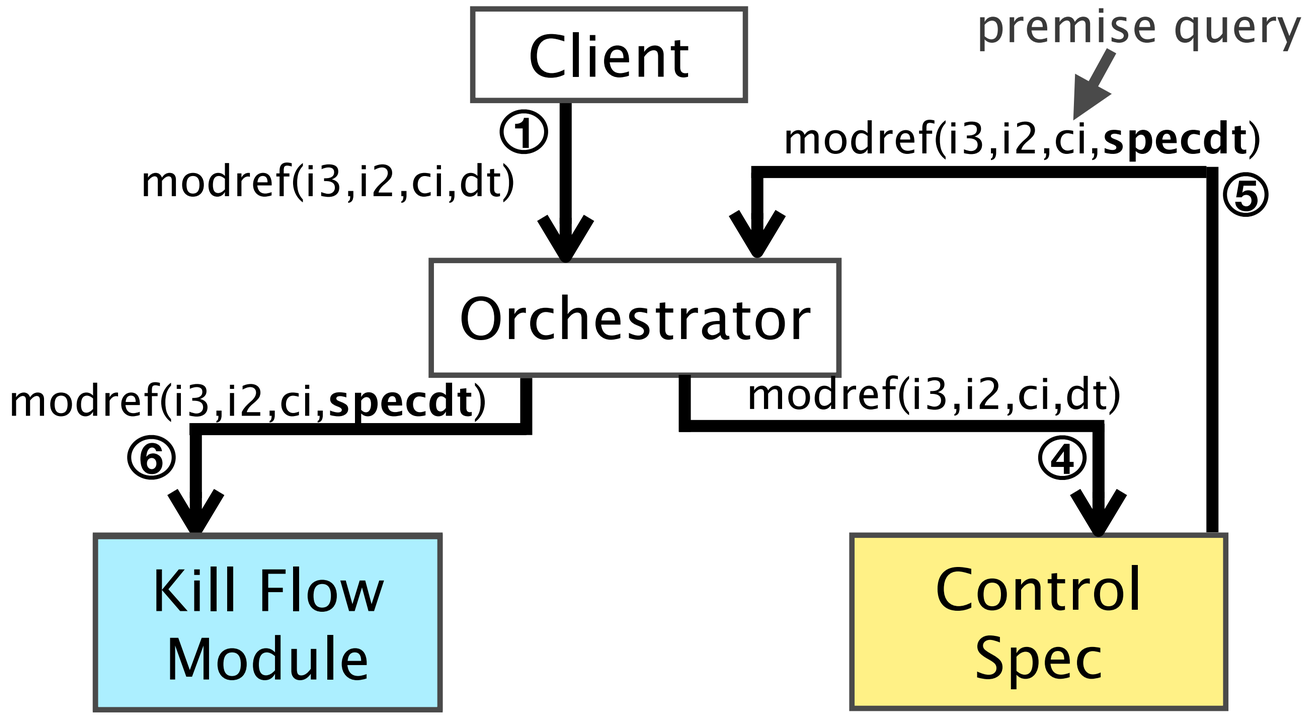


spec-dt encodes
assertion that
branch never taken

SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a
cross-iter
data flow
from i3 to i2?

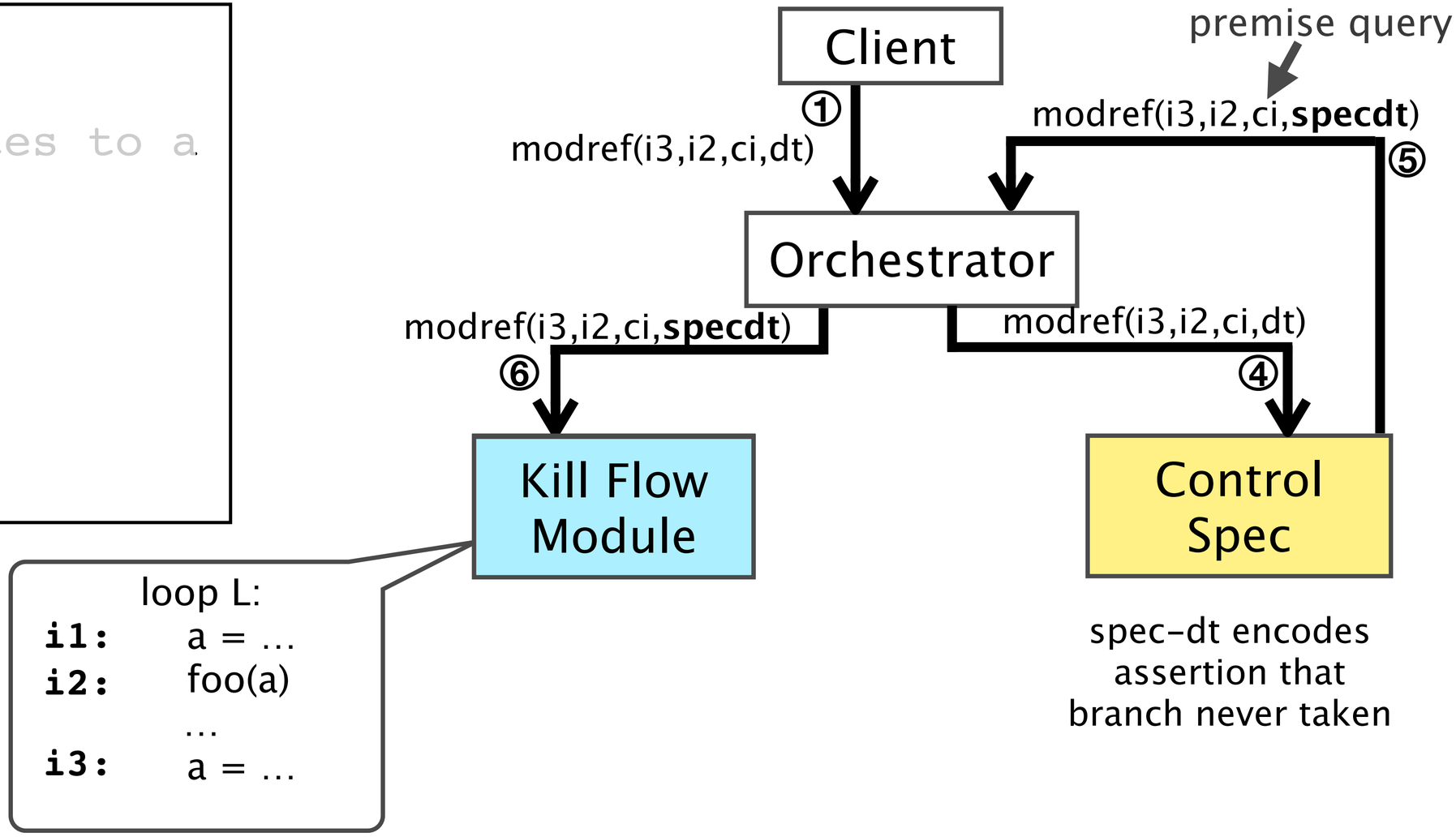


spec-dt encodes
assertion that
branch never taken

SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a cross-iter data flow from i3 to i2?

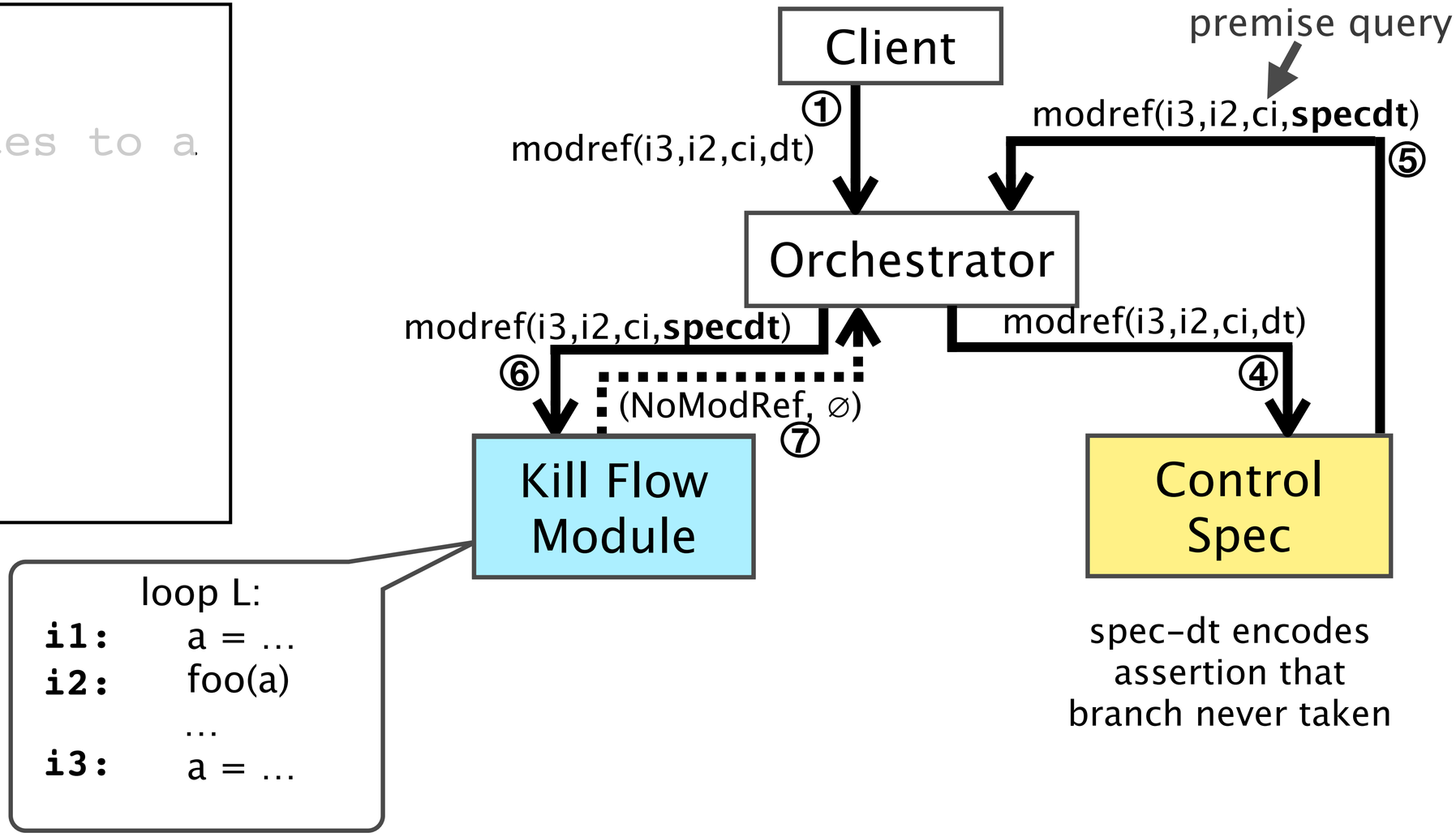


View of loop based on spec-dt

SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a cross-iter data flow from i3 to i2?

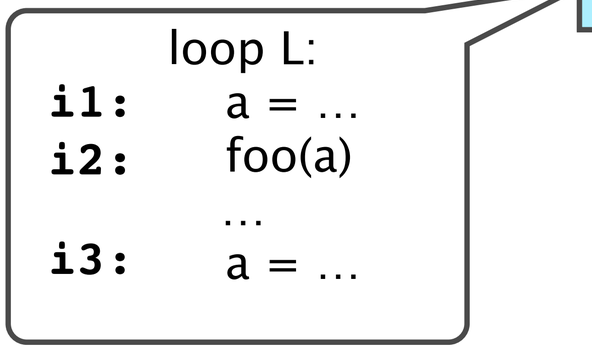


View of loop based on spec-dt

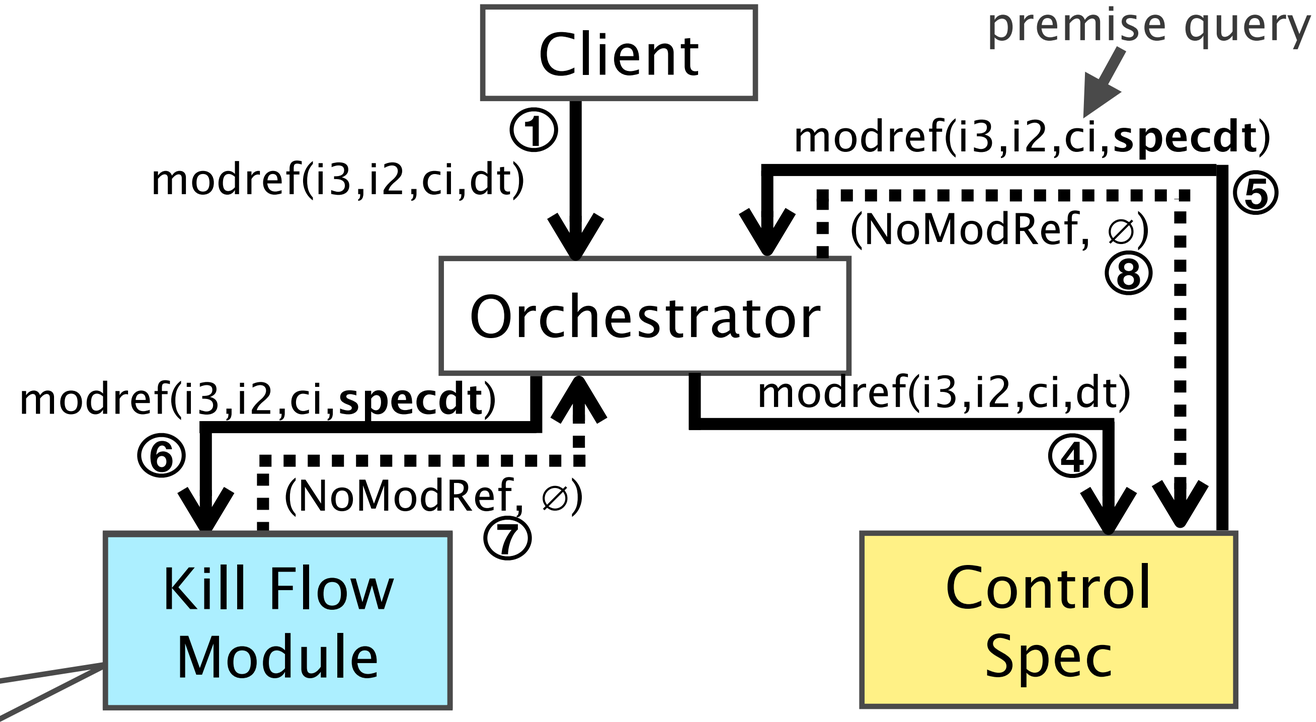
SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a cross-iter data flow from i3 to i2?



View of loop based on spec-dt

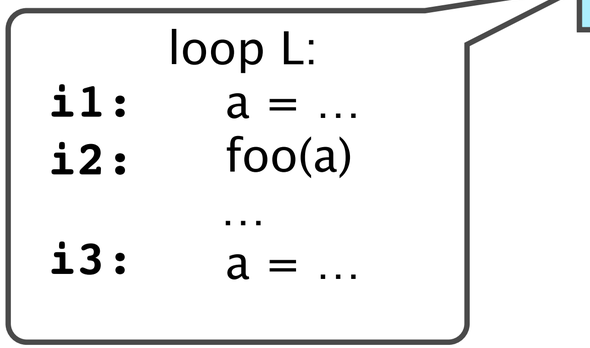


spec-dt encodes assertion that branch never taken

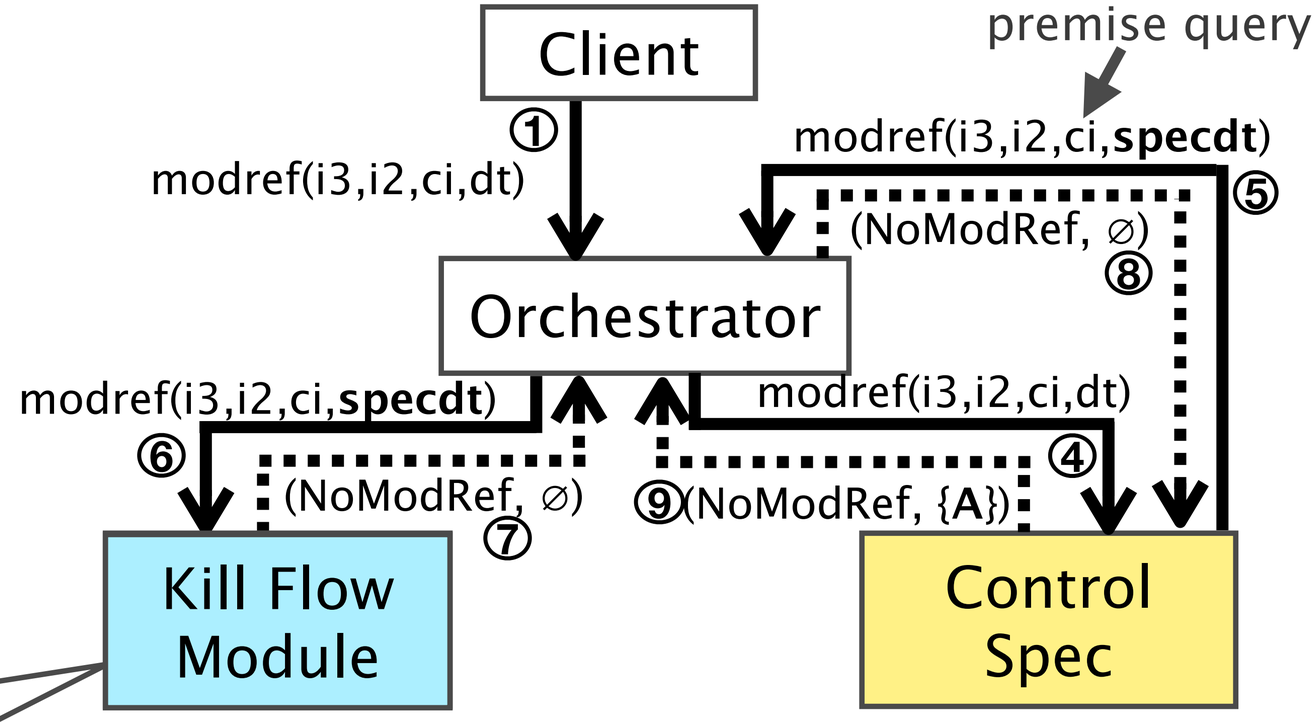
SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a cross-iter data flow from i3 to i2?



View of loop based on spec-dt



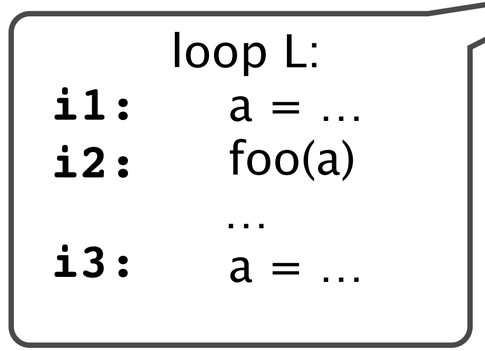
spec-dt encodes assertion that branch never taken

Speculation assertion **A:** branch never taken

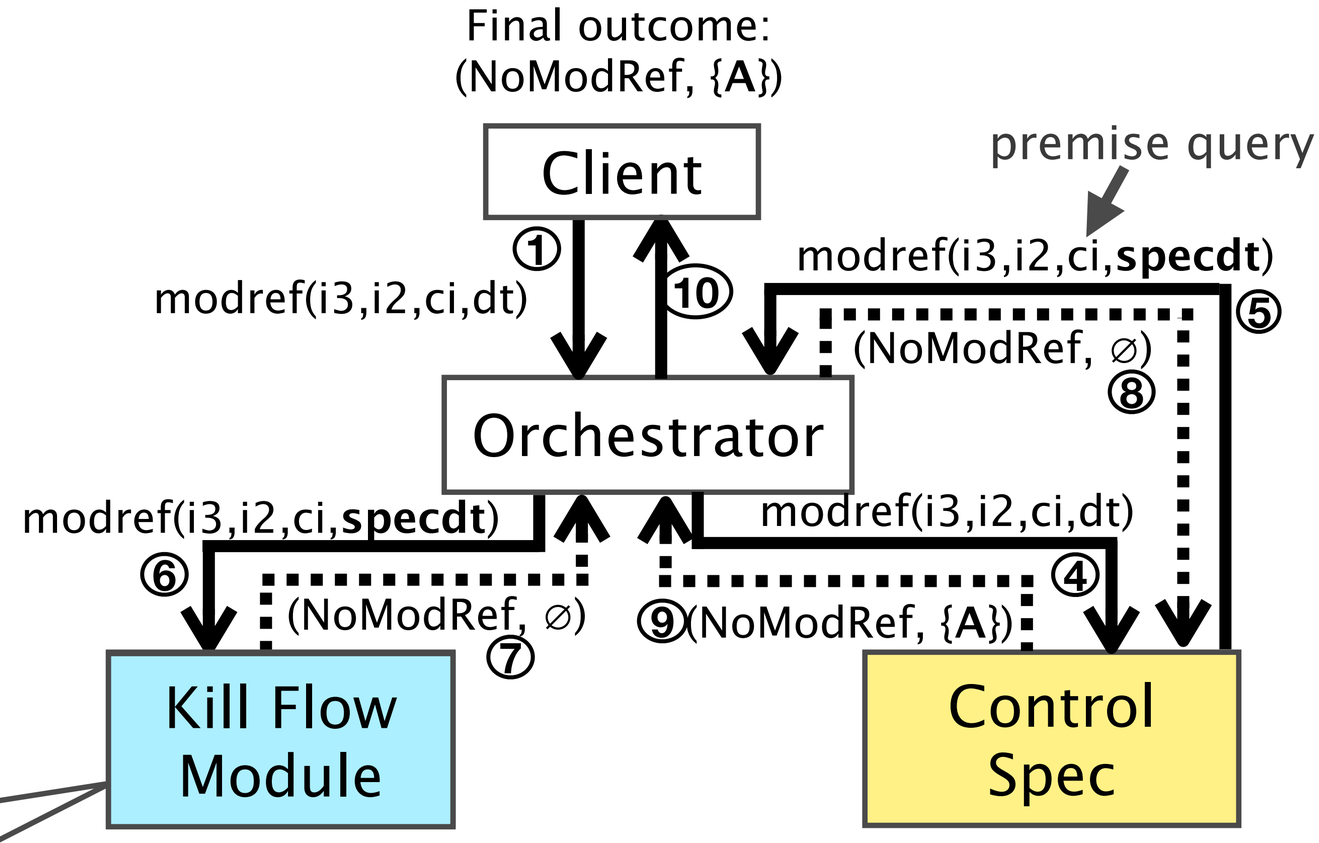
SCAF in action

```
   loop L:
      if (rare)
        // no writes to a
        ...
      else
i1:     a = ...
i2:   foo(a)
      ...
i3:   a = ...
```

Is there a cross-iter data flow from i3 to i2?



View of loop based on spec-dt



spec-dt encodes assertion that branch never taken

Speculation assertion **A:** branch never taken

SCAF’s Evaluation Methodology

SCAF's Evaluation Methodology

Empirically Evaluated Claim

SCAF reduces the need for memory speculation

SCAF's Evaluation Methodology

Empirically Evaluated Claim

SCAF reduces the need for memory speculation

Benchmarks

16 C/C++ benchmarks from SPEC CPU

SCAF's Evaluation Methodology

Empirically Evaluated Claim

SCAF reduces the need for memory speculation

Benchmarks

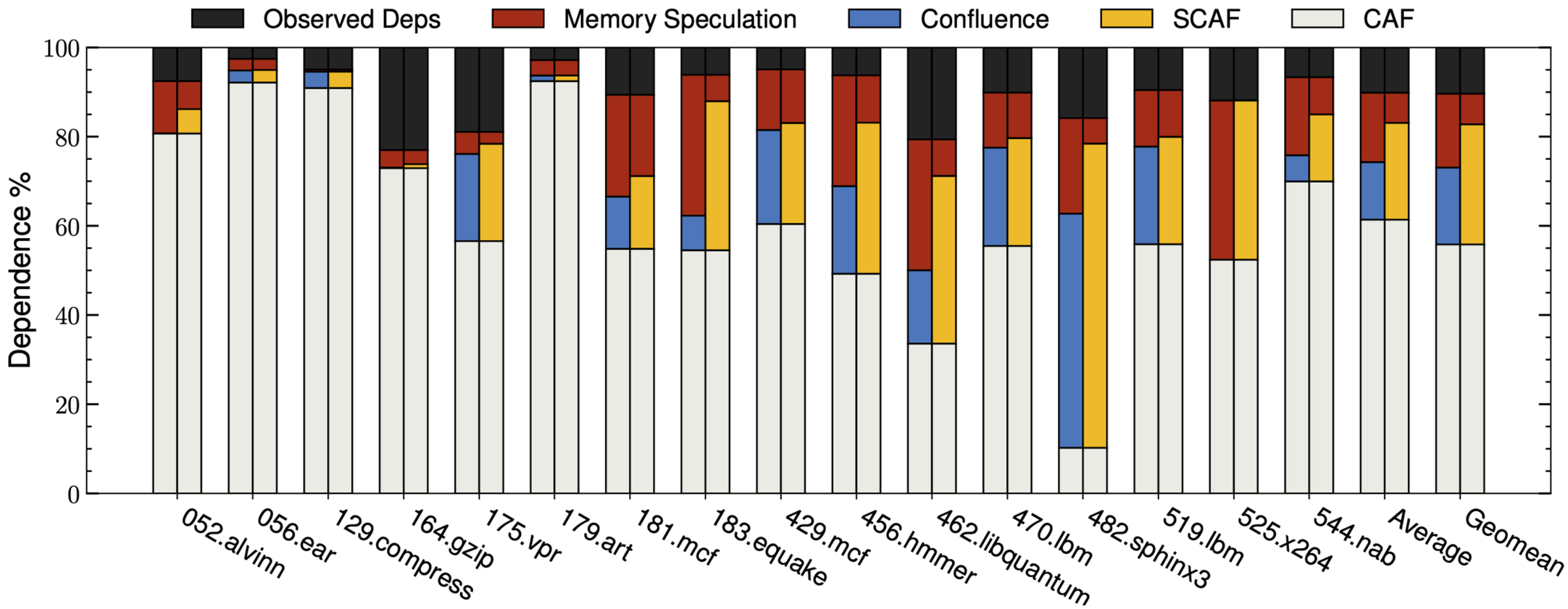
16 C/C++ benchmarks from SPEC CPU

State-of-art Baseline

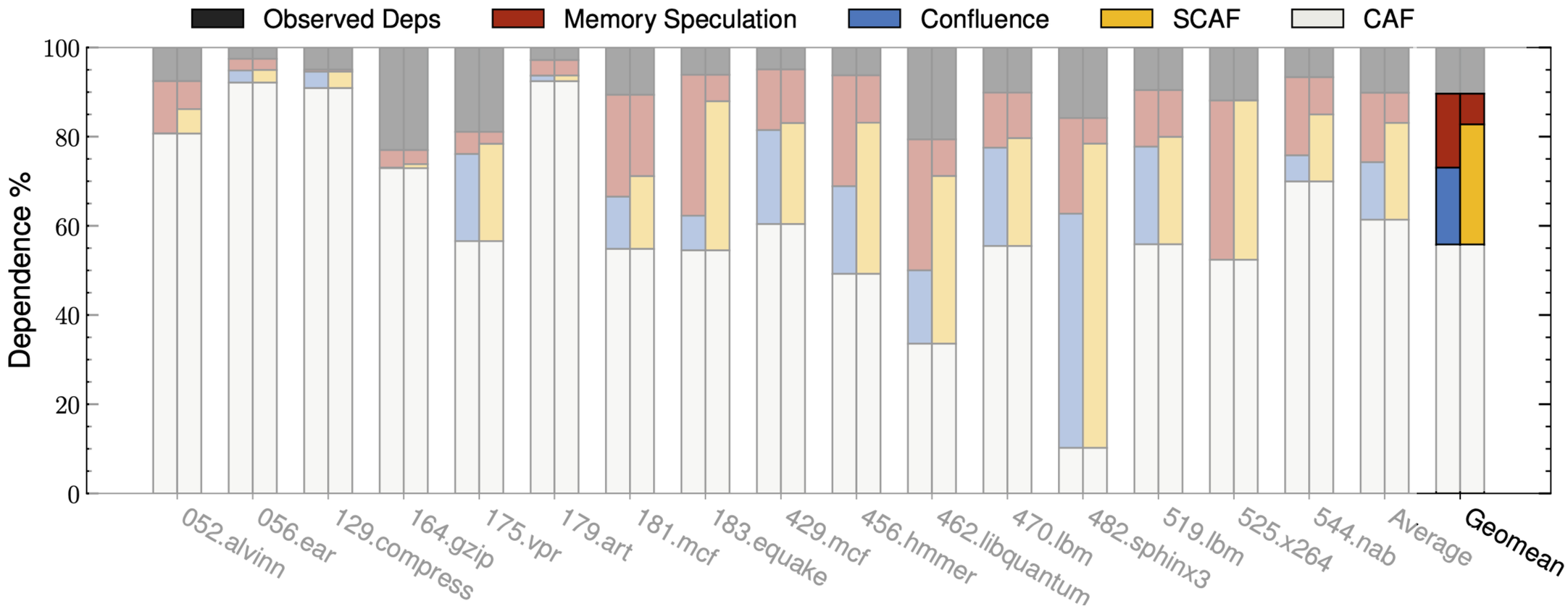
Composition by Confluence: analysis results are the confluence of results of individual components [1,2,3,4]

[1]Johnson et al., PLDI '12 [2]Kim et al., CGO '12 [3]Mehrara et al., PLDI '09 [4]Vachharajani et al., PACT '07

SCAF reduces need for expensive memory speculation



SCAF reduces need for expensive memory speculation



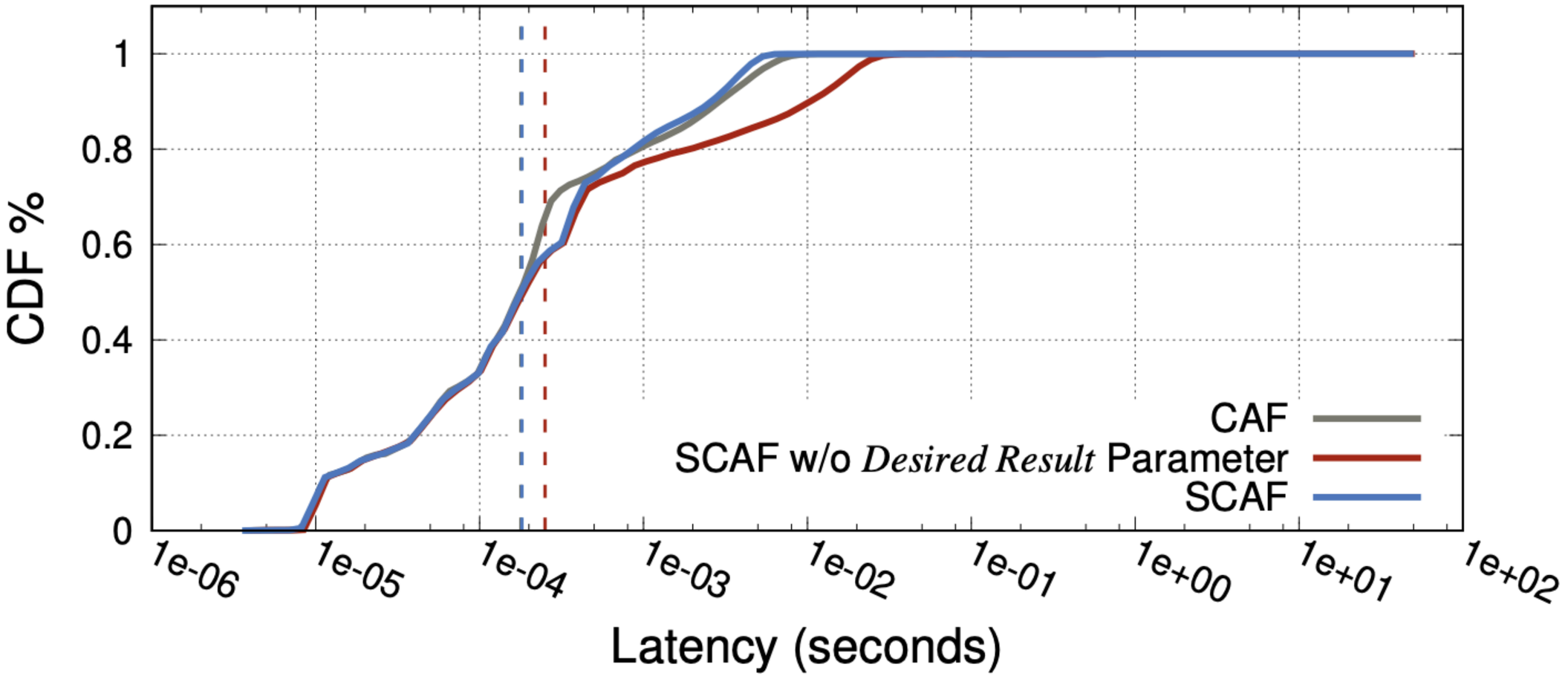
SCAF enables various Forms of Beneficial Collaboration

	Benchmark Coverage	Loop Coverage
Among Speculation Modules	87.5%	53.6%
Between Memory Analysis and Speculation Modules	93.8%	42.9%
All	100.0%	66.1%

Beneficial Collaboration: two or more modules collaboratively resolve more queries than in isolation

New Desired Result Parameter reduces Query Latency

28 % geomean reduction



Conclusion

- SCAF is a modular and collaborative dependence analysis framework that computes the full impact of speculation on memory dependence analysis
- SCAF dramatically reduces, compared to the state-of-the-art, the need for expensive-to-validate memory speculation.
- SCAF is essential for memory analysis sensitive clients and a necessary step toward robust automatic parallelization

Artifact available: https://doi.org/10.5281/zenodo.3751586