

Paper Review: Demanded Abstract Interpretation

Stein et al., PLDI '21.

Jung Hyun Kim

*SoftSec Lab., KAIST
IS661 Spring, 2024*

Abstract Interpretation

Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.

Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.

```
int a;  
if (cond) {  
    a = 0;  
} else {  
    a = 1;  
}  
print(a);
```

Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.

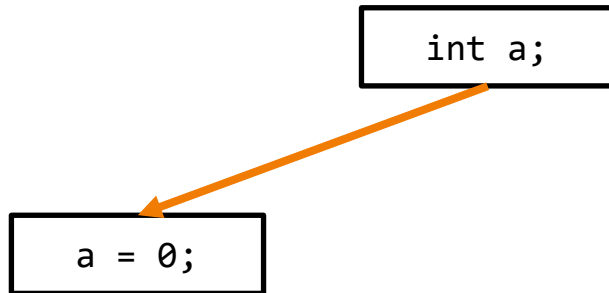
```
int a;  
if (cond) {  
    a = 0;  
} else {  
    a = 1;  
}  
print(a);
```

```
int a;
```

Abstract Interpretation

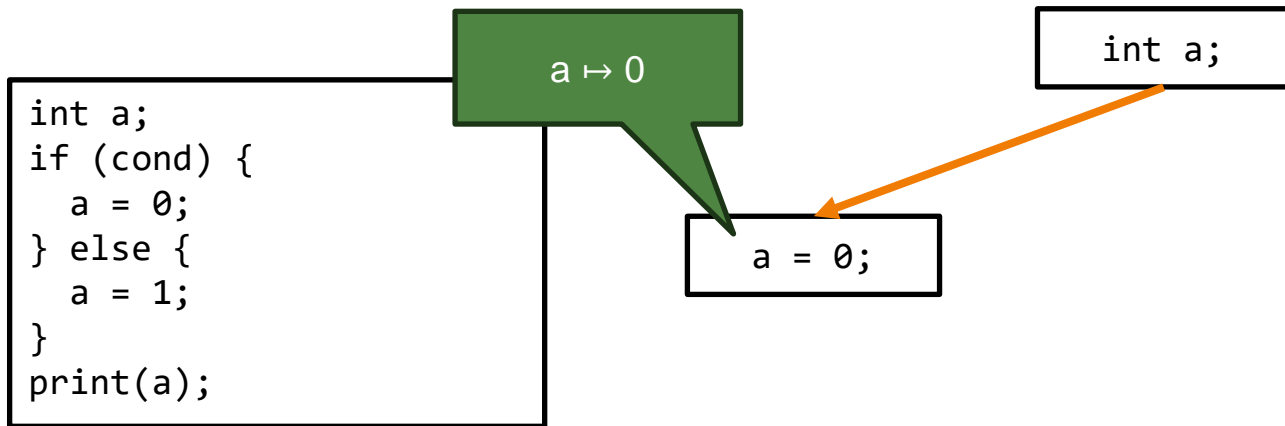
- Abstract Interpretation (AI): a *sound approximation* of a program.

```
int a;  
if (cond) {  
    a = 0;  
} else {  
    a = 1;  
}  
print(a);
```



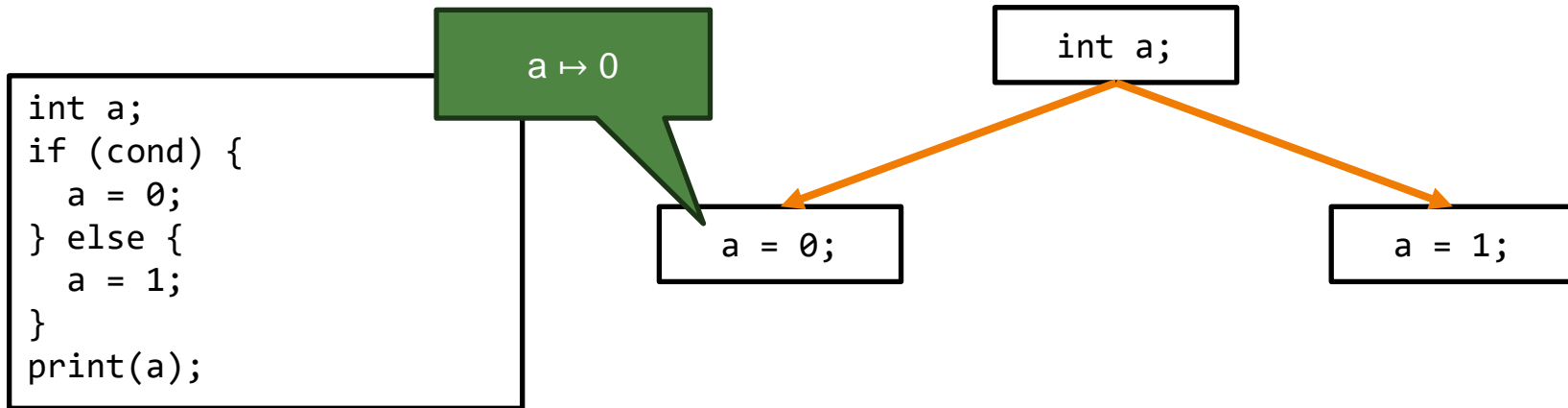
Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.



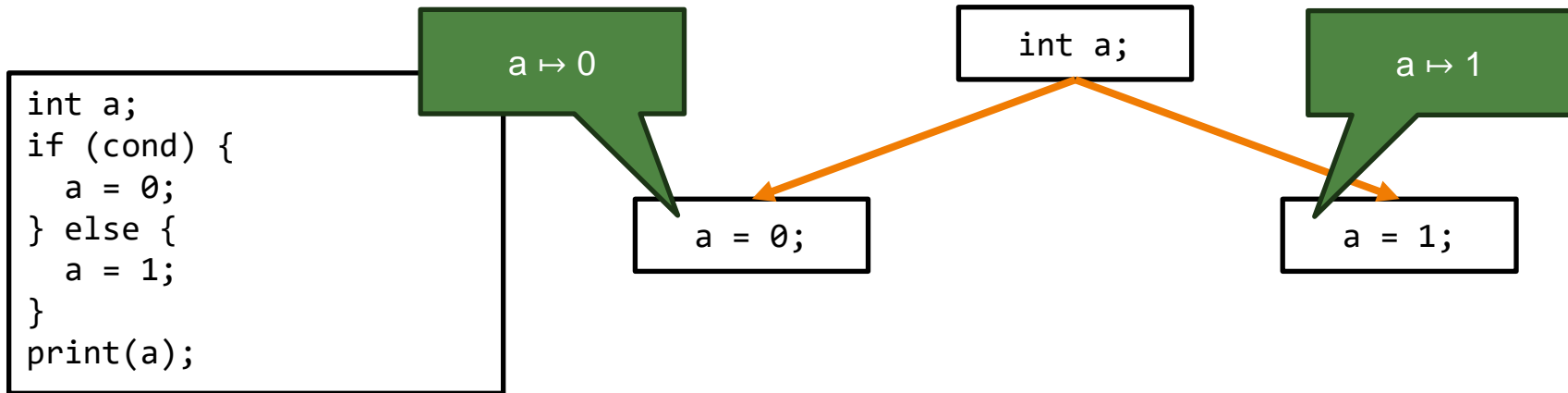
Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.



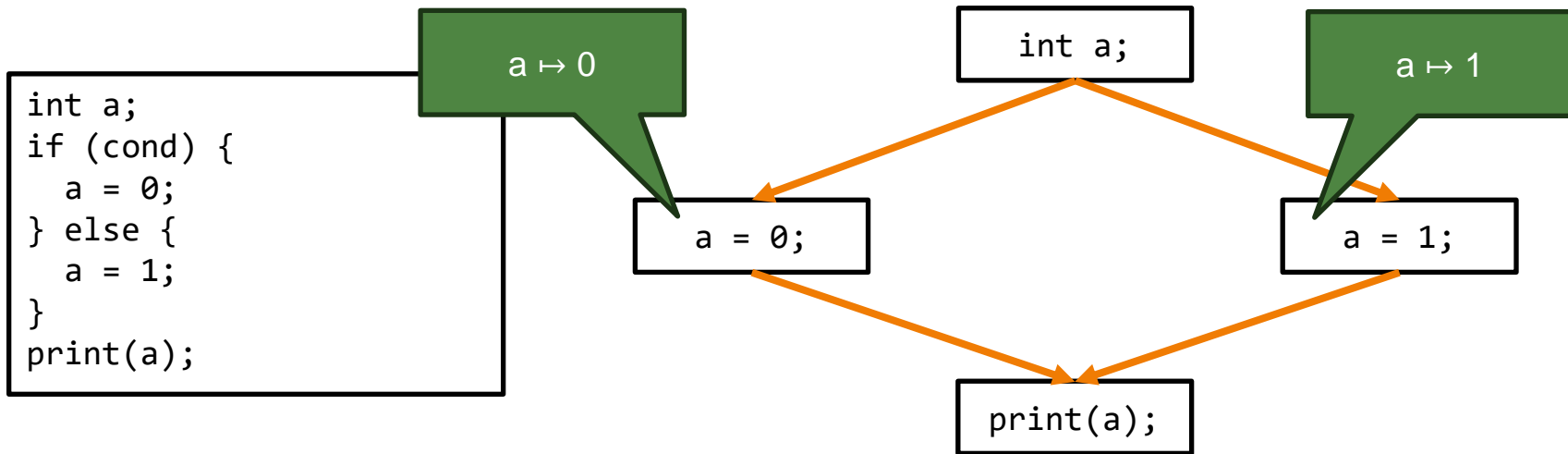
Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.



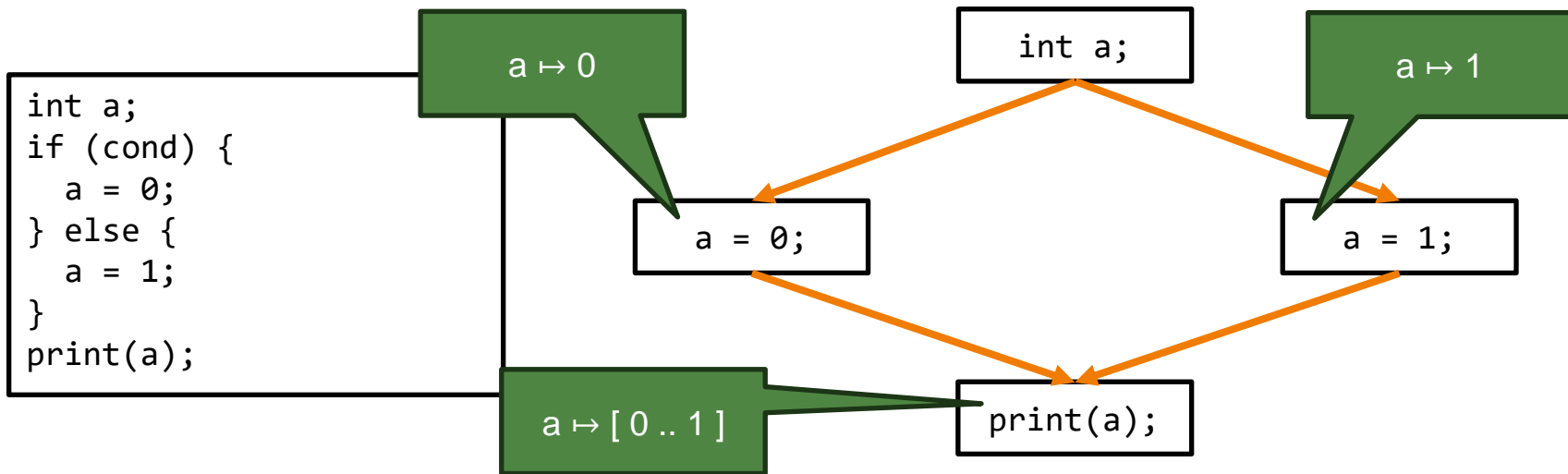
Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.



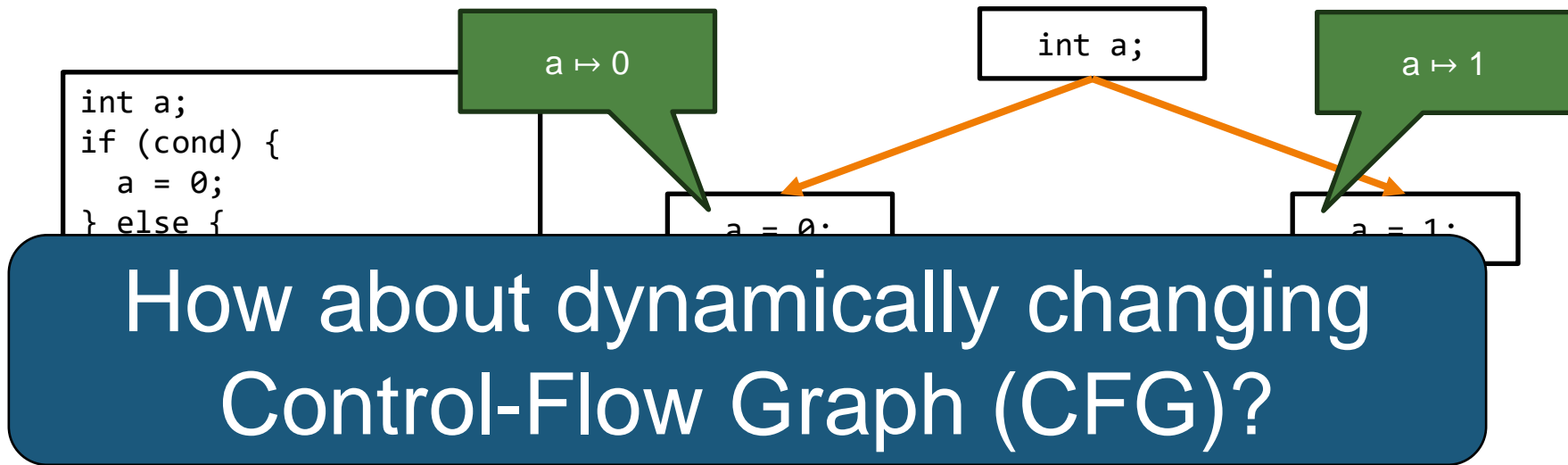
Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.



Abstract Interpretation

- Abstract Interpretation (AI): a *sound approximation* of a program.



Challenges in AI with Dynamic CFG

Challenges in AI with Dynamic CFG

- We can use *offline* AI algorithms every time a CFG changes.

Challenges in AI with Dynamic CFG

- We can use *offline* AI algorithms every time a CFG changes.
 - An *offline* algorithm will run by a single edit *from scratch*!

Challenges in AI with Dynamic CFG

- We can use *offline* AI algorithms every time a CFG changes.
 - An *offline* algorithm will run by a single edit *from scratch*!
- Two challenges driving *inefficiency* arise:

Challenges in AI with Dynamic CFG

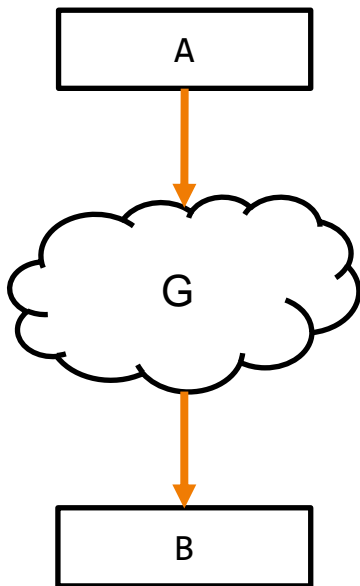
- We can use *offline* AI algorithms every time a CFG changes.
 - An *offline* algorithm will run by a single edit *from scratch*!
- Two challenges driving *inefficiency* arise:
 - C1) We may *recalculate* the states that will *never be affected*.

Challenges in AI with Dynamic CFG

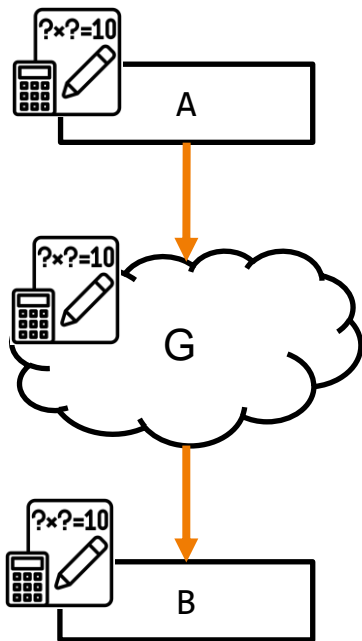
- We can use *offline* AI algorithms every time a CFG changes.
 - An *offline* algorithm will run by a single edit *from scratch*!
- Two challenges driving *inefficiency* arise:
 - C1) We may *recalculate* the states that will *never be affected*.
 - C2) We may *recalculate* the states that will *never be queried*.

C1) Recalculate Unaffected Nodes

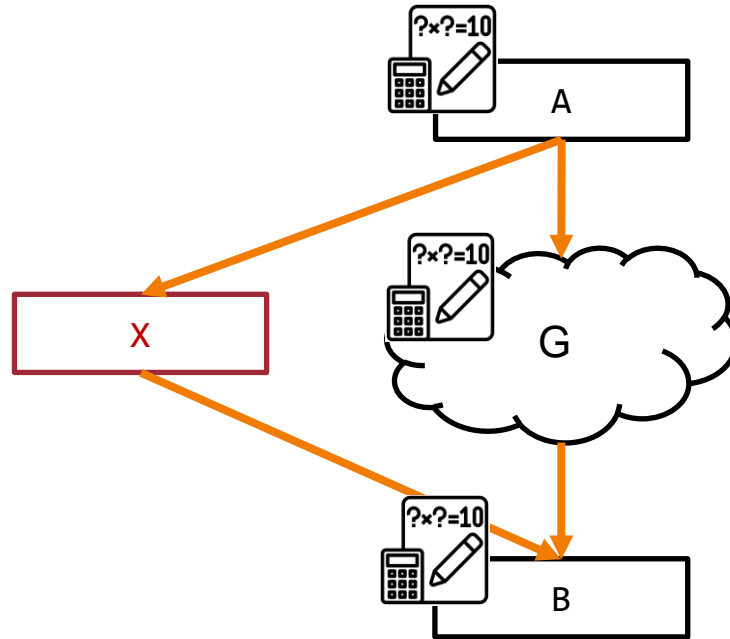
C1) Recalculate Unaffected Nodes



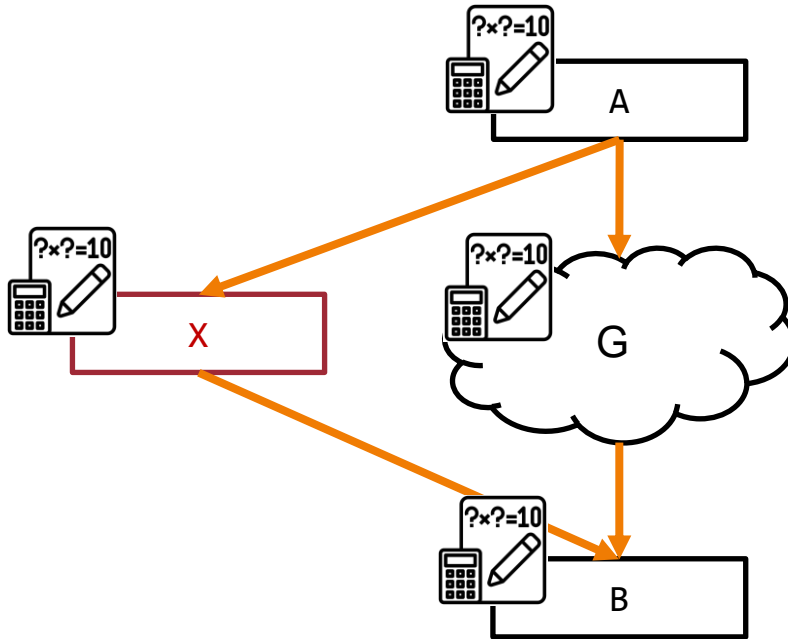
C1) Recalculate Unaffected Nodes



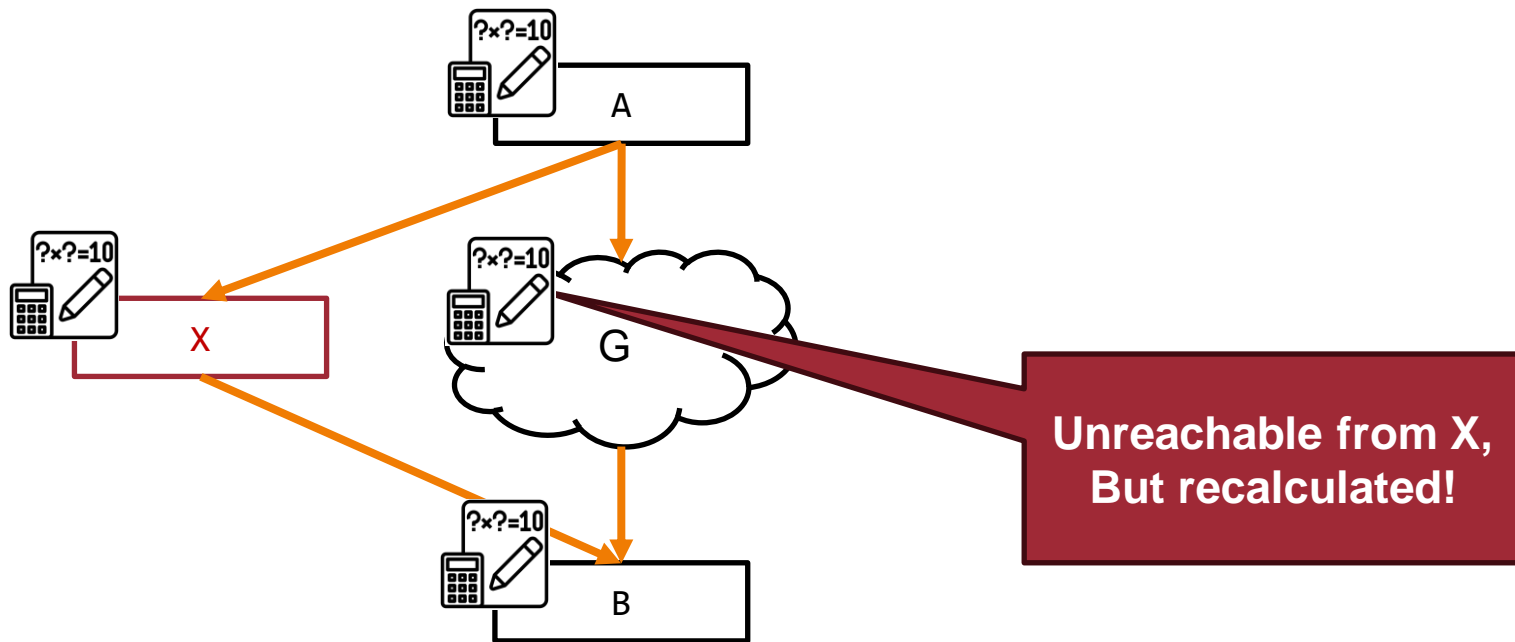
C1) Recalculate Unaffected Nodes



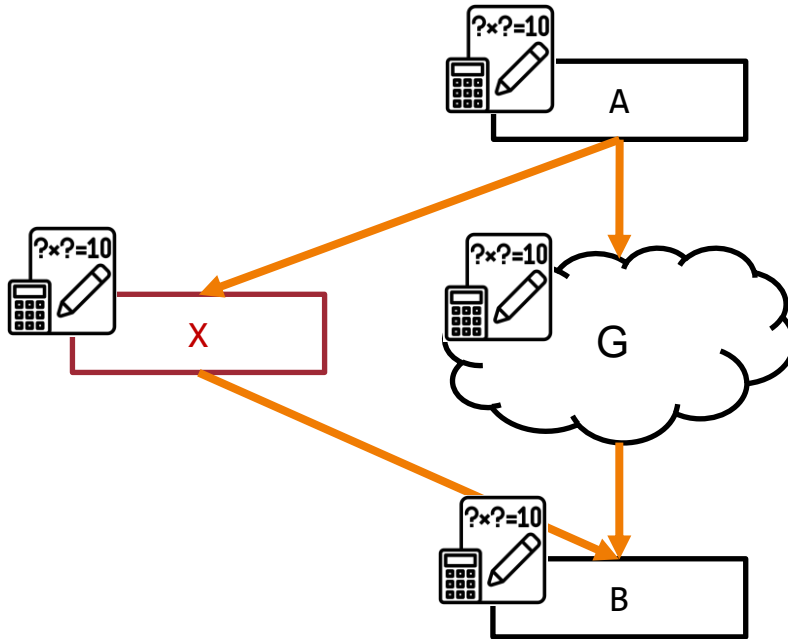
C1) Recalculate Unaffected Nodes



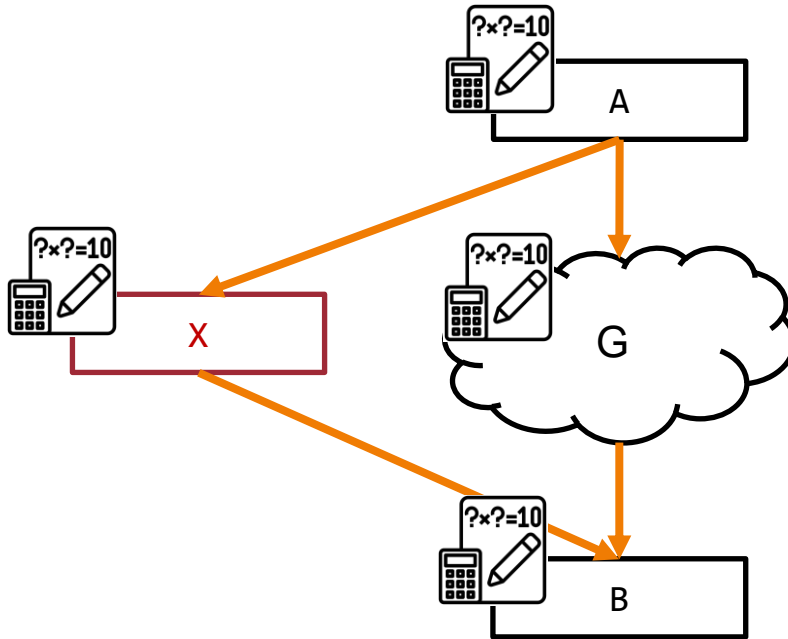
C1) Recalculate Unaffected Nodes



C2) Recalculate Not-Interesting Nodes

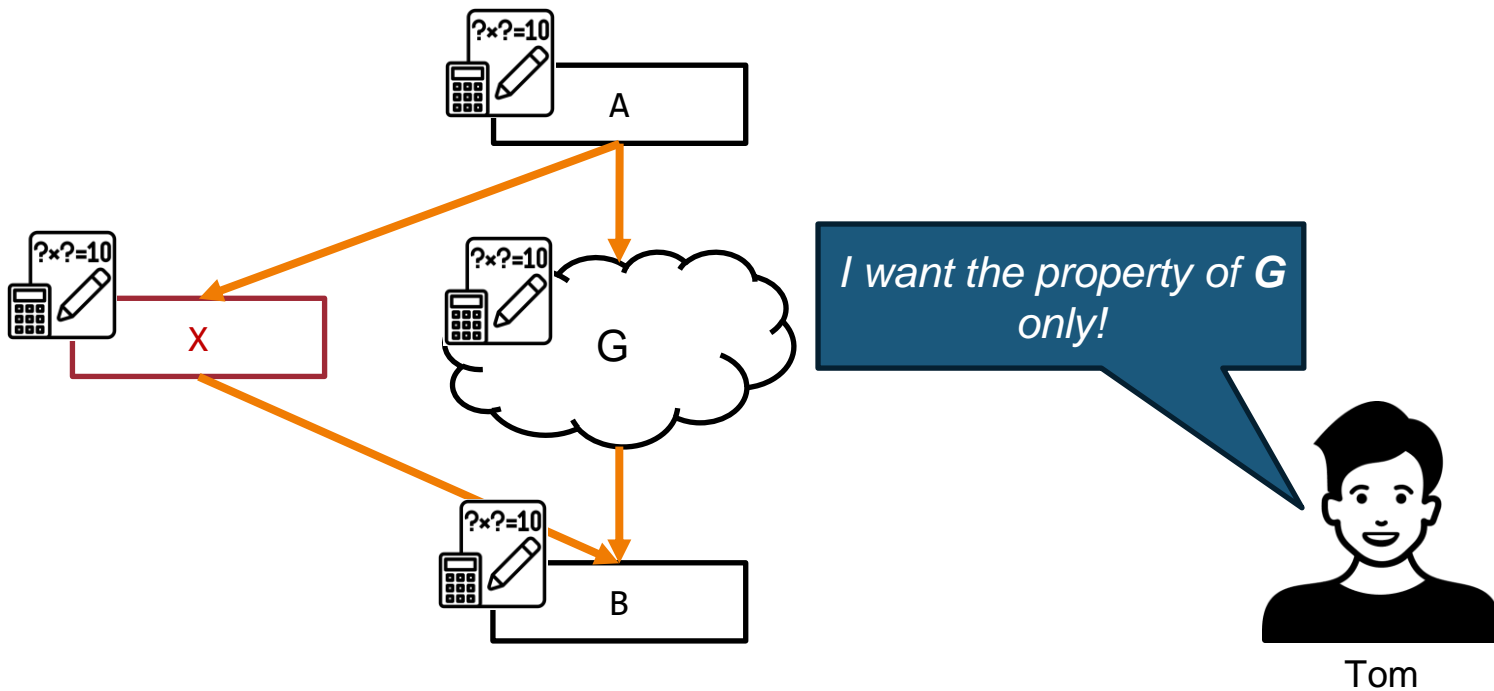


C2) Recalculate Not-Interesting Nodes

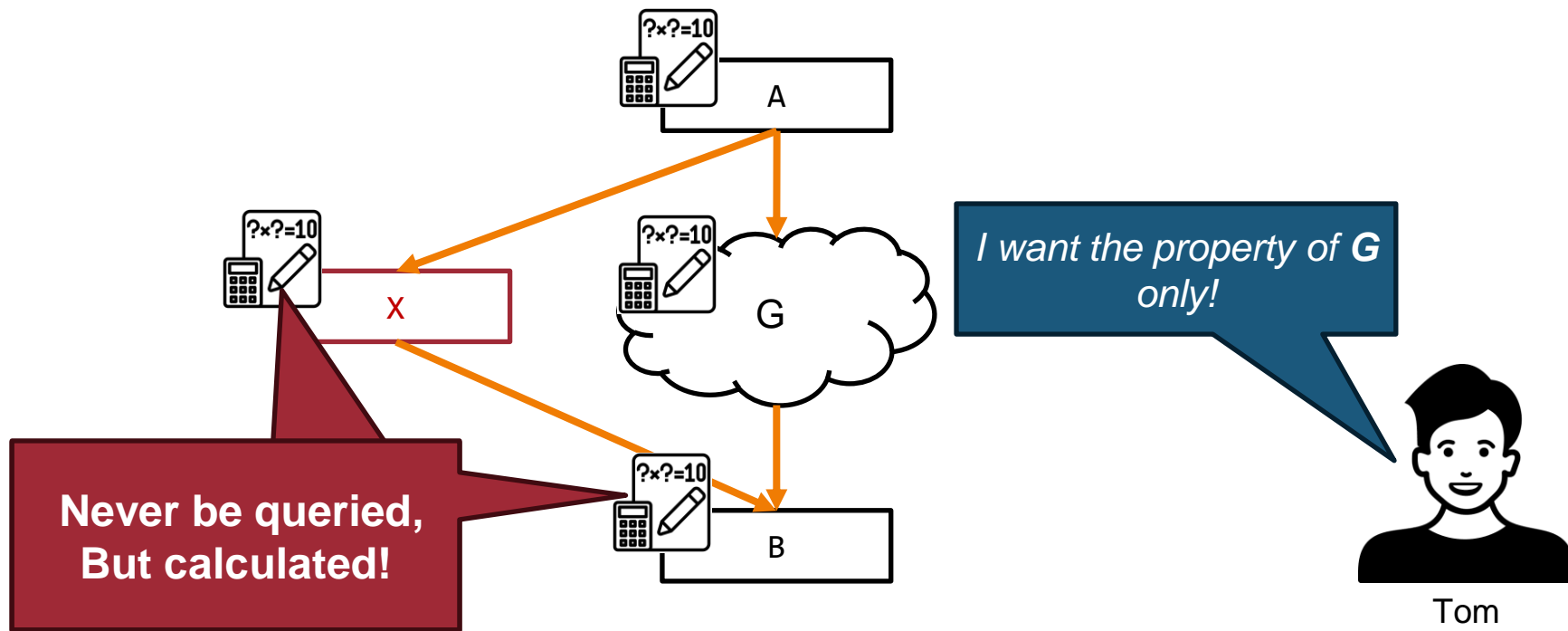


Tom

C2) Recalculate Not-Interesting Nodes



C2) Recalculate Not-Interesting Nodes



Demanded Abstract Interpretation

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*
- Demanded Abstract Interpretation Graph (DAIG):

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*
- Demanded Abstract Interpretation Graph (DAIG):
 - A directed *acyclic* graph that is transformed from a CFG.

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*
- Demanded Abstract Interpretation Graph (DAIG):
 - A directed *acyclic* graph that is transformed from a CFG.
 - Just propagate the state changes forward!

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*
- Demanded Abstract Interpretation Graph (DAIG):
 - A directed *acyclic* graph that is transformed from a CFG.
 - Just propagate the state changes forward!
 - Recursion in AI is unrolled *dynamically* when running DAIGs.

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*
- Demanded Abstract Interpretation Graph (DAIG):
 - A directed *acyclic* graph that is transformed from a CFG.
 - Just propagate the state changes forward!
 - Recursion in AI is unrolled *dynamically* when running DAIGs.
 - Once transformed, we can *incrementally* run an AI on top of it!

Demanded Abstract Interpretation

- “*Demanded*”: Calculate the information when we *need* to.
 1. Calculate the necessary information *on demand*. → *Demand-driven*
 2. Calculate the information that *changes only*. → *Incremental*
- Demanded Abstract Interpretation Graph (DAIG):
 - A directed *acyclic* graph that is transformed from a CFG.
 - Just propagate the state changes forward!
 - Recursion in AI is unrolled *dynamically* when running DAIGs.
 - Once transformed, we can *incrementally* run an AI on top of it!
 - This is *30 times* faster than using offline algorithm *from scratch*.

Overview

Overview

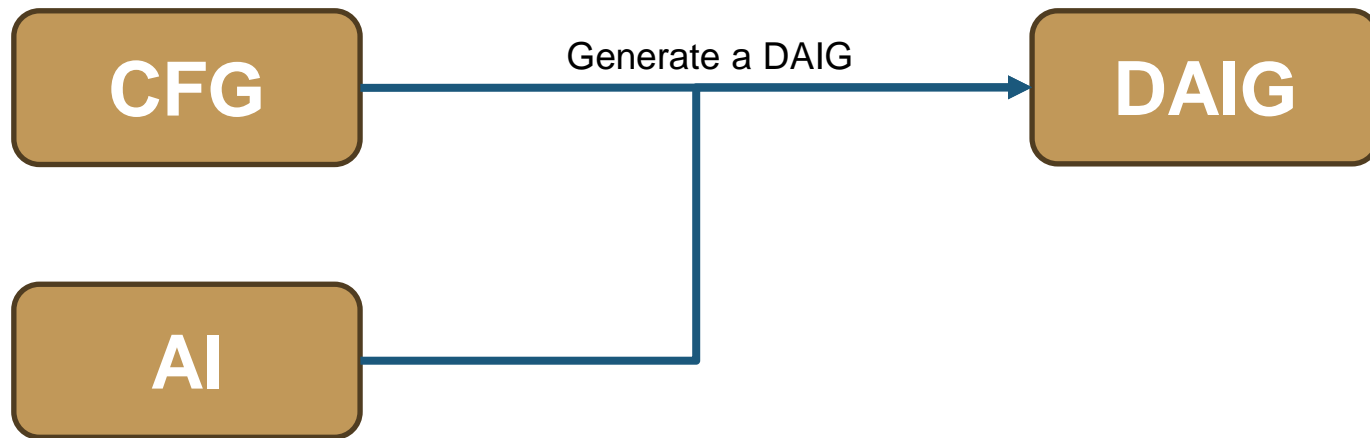
CFG

Overview

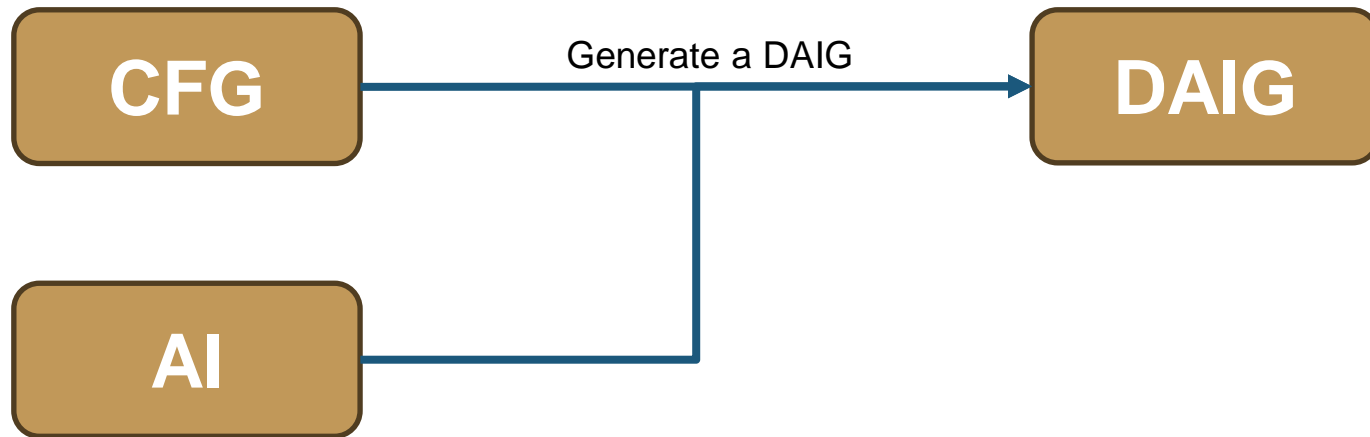
CFG

AI

Overview

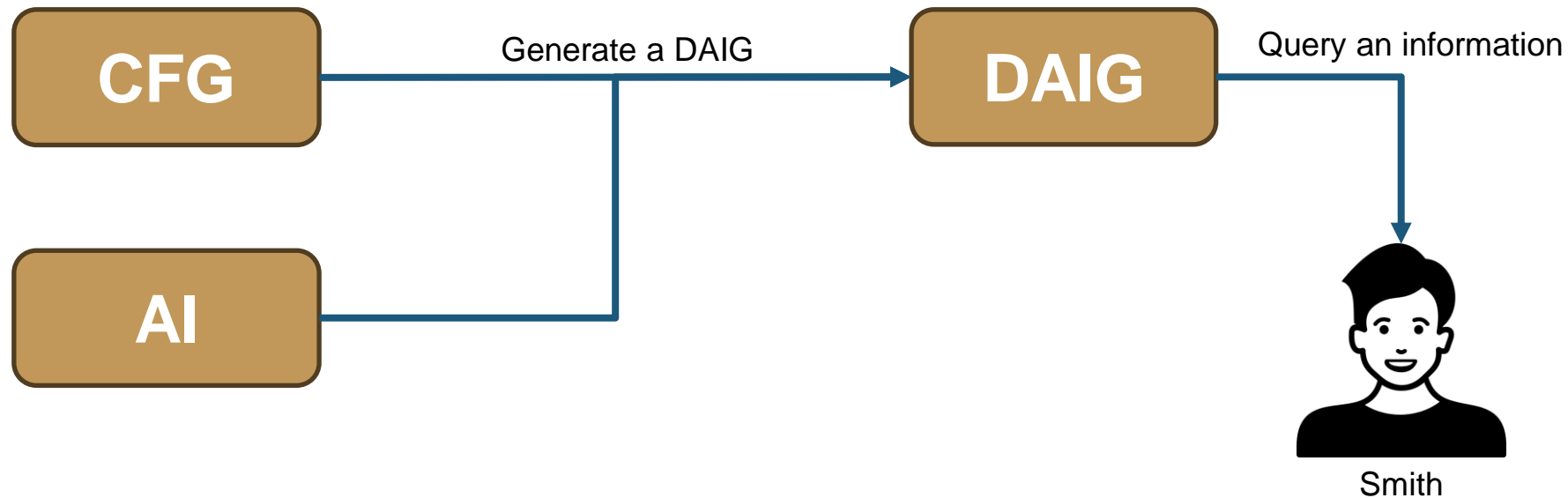


Overview

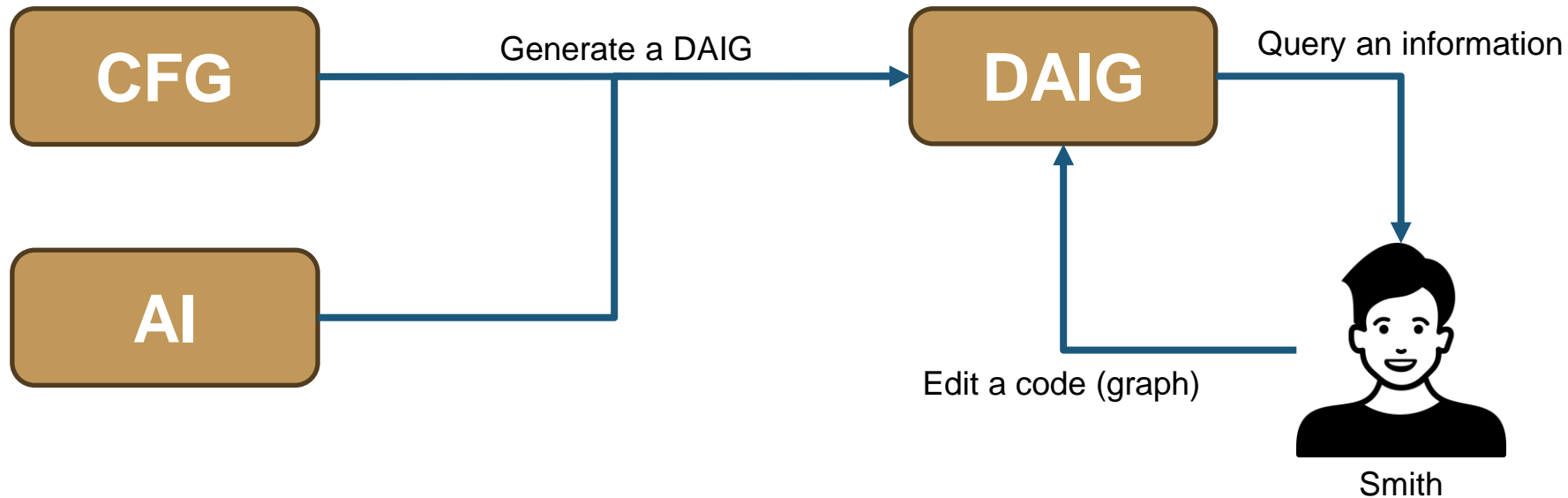


Smith

Overview



Overview



DAIG

DAIG

```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```


DAIG

```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

* We ignore the conditional branches for brevity.

DAIG

```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

 : State cell


 : Statement cell

* We ignore the conditional branches for brevity.

DAIG

`i = 0;`

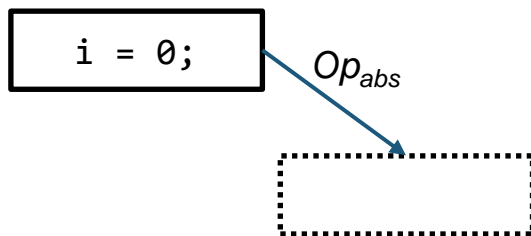
```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

 : State cell


 : Statement cell

* We ignore the conditional branches for brevity.

DAIG



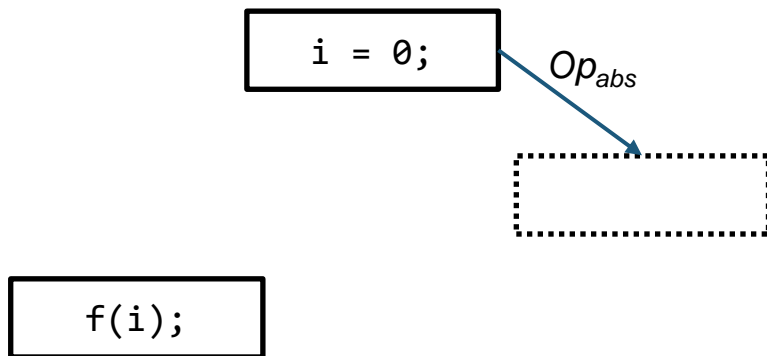
```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

 : State cell


 : Statement cell

* We ignore the conditional branches for brevity.

DAIG



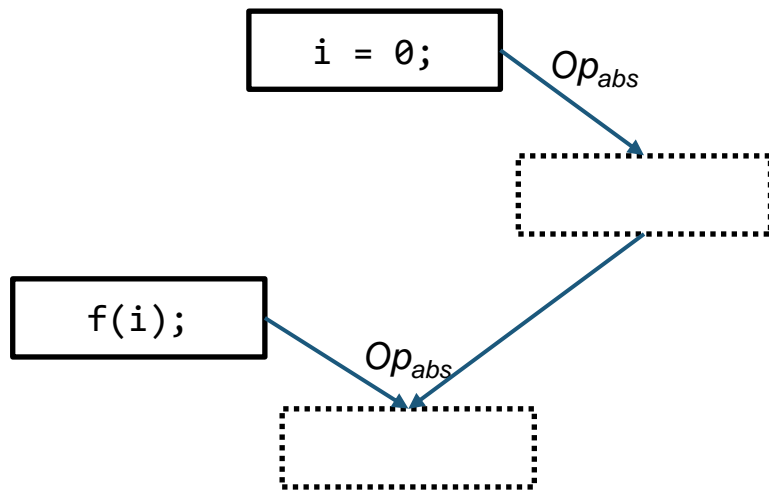
```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

 : State cell


 : Statement cell

* We ignore the conditional branches for brevity.

DAIG



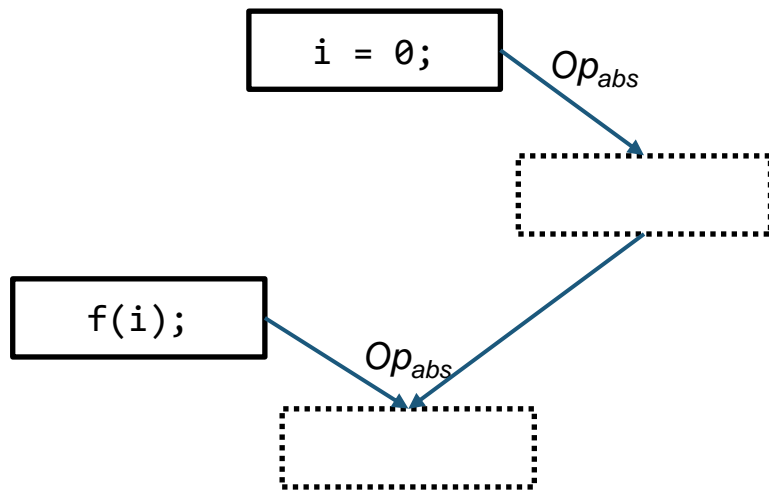
```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

 : State cell

 : Statement cell


* We ignore the conditional branches for brevity.

DAIG



```
int i;  
i = 0;  
if (...) {  
    f(i);  
}  
else {  
    g(i);  
}
```

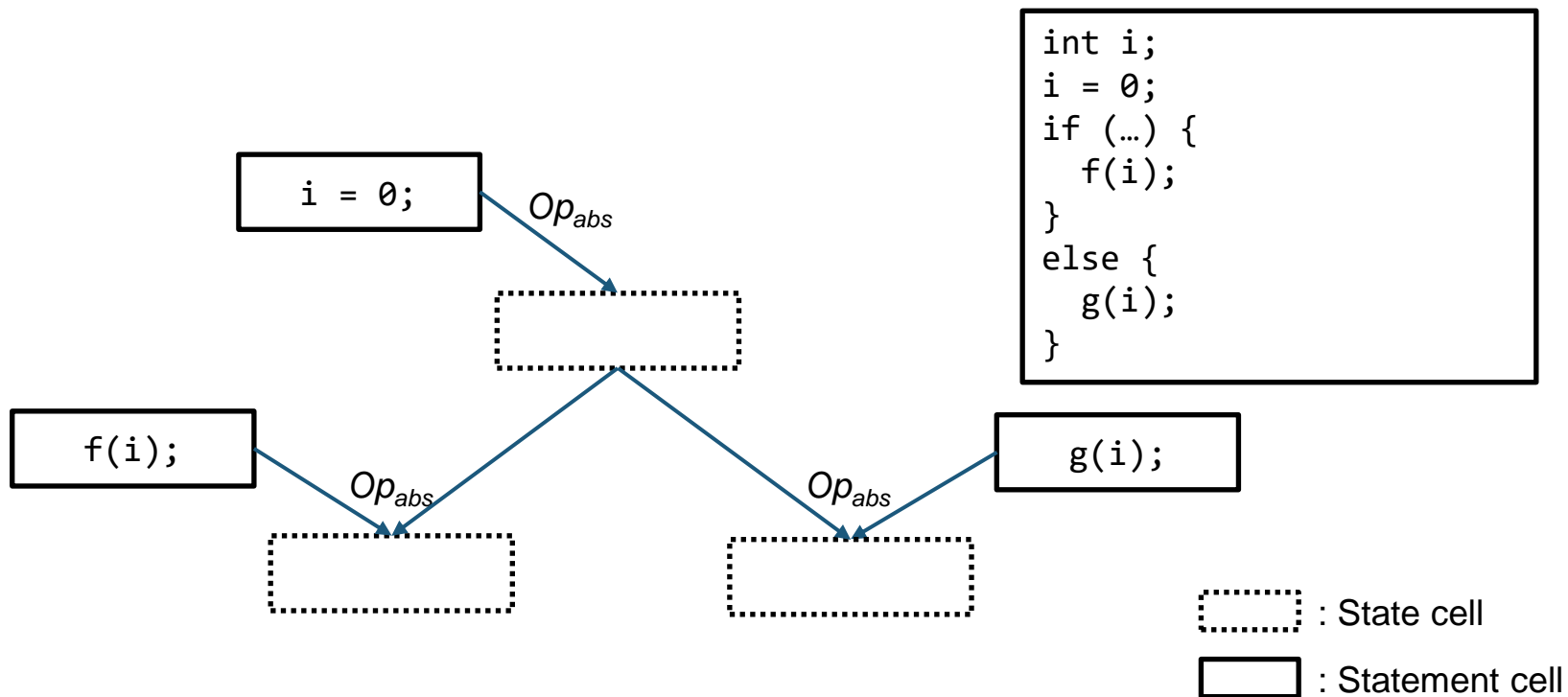
`g(i);`

 : State cell

 : Statement cell

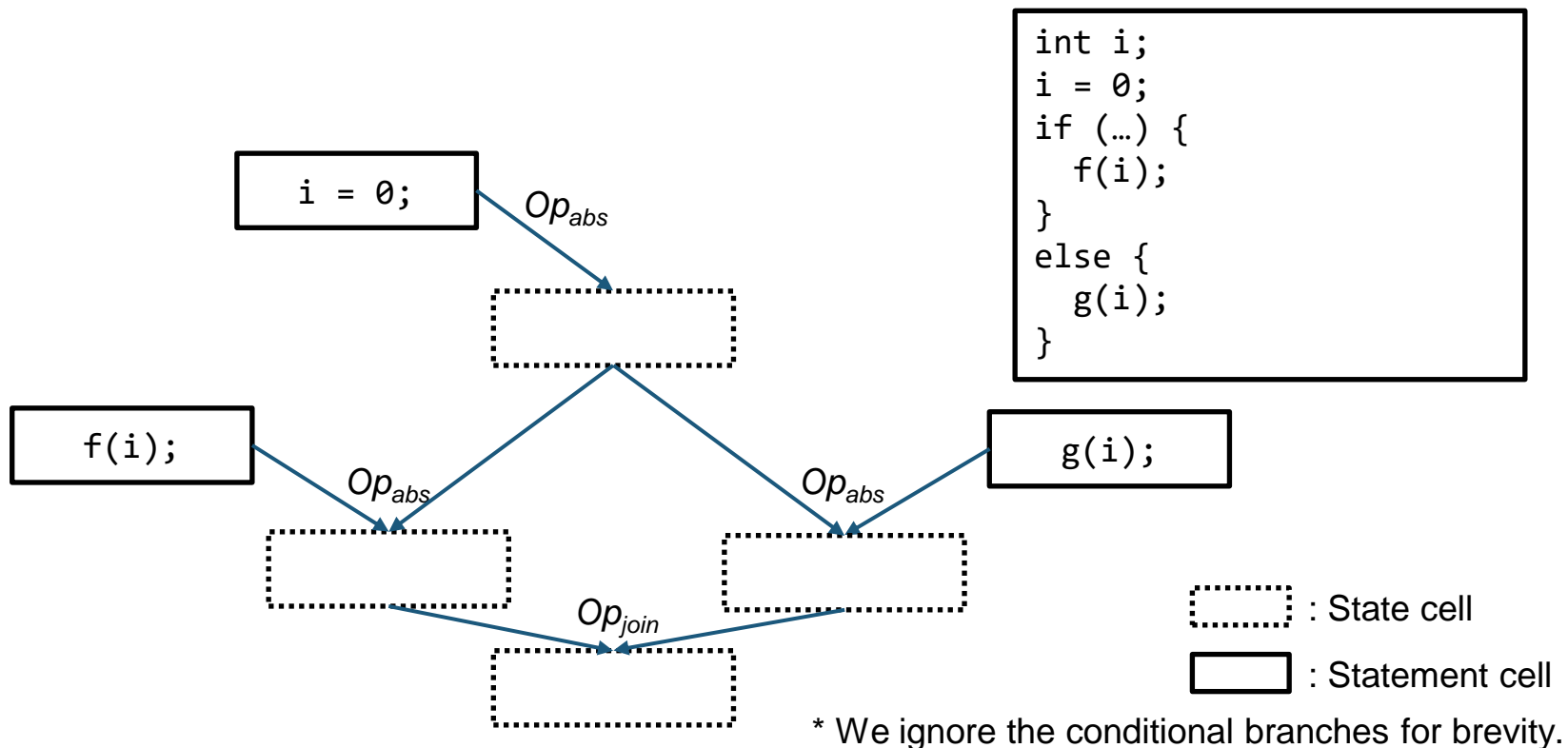
* We ignore the conditional branches for brevity.

DAIG

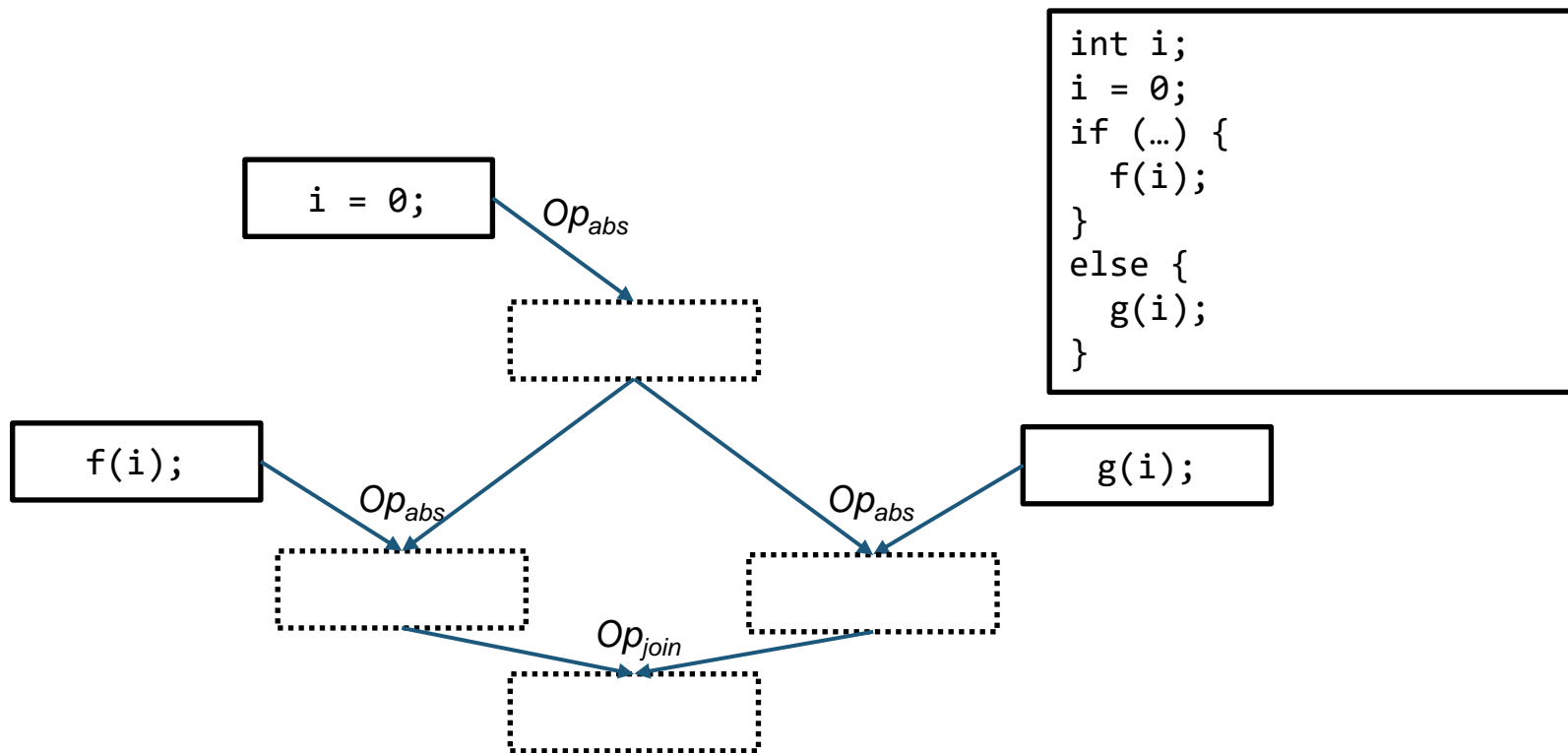


* We ignore the conditional branches for brevity.

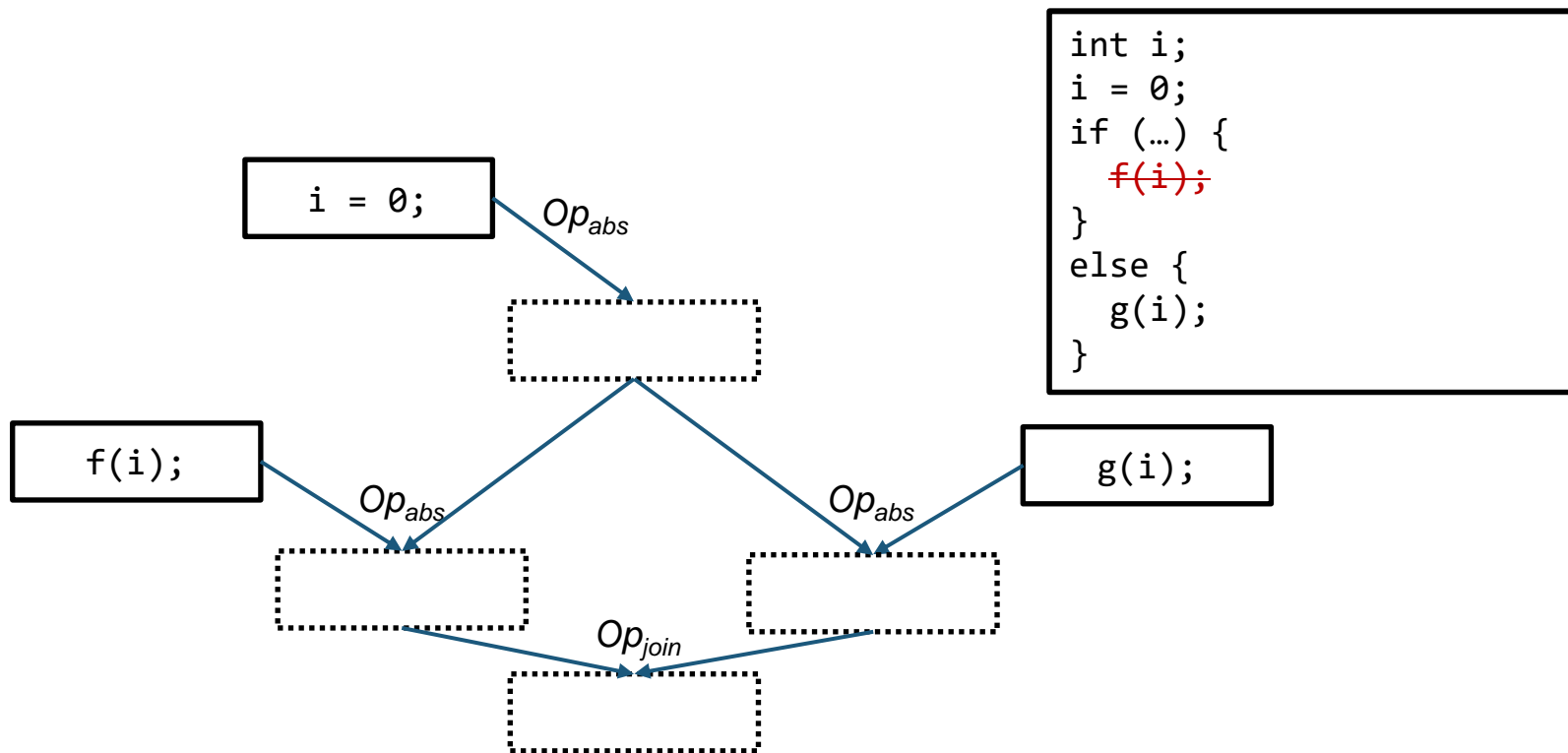
DAIG



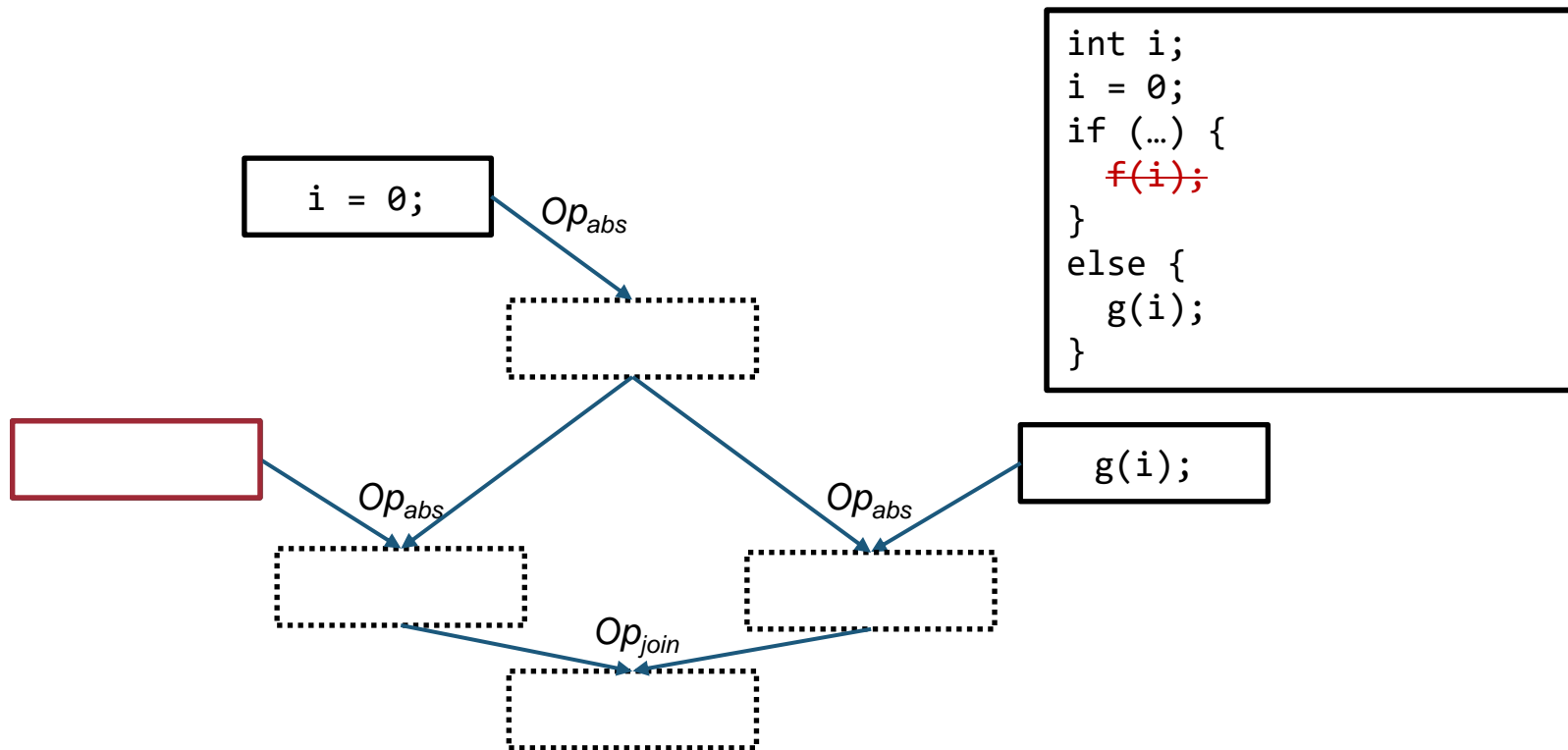
DAIG with Incremental Update



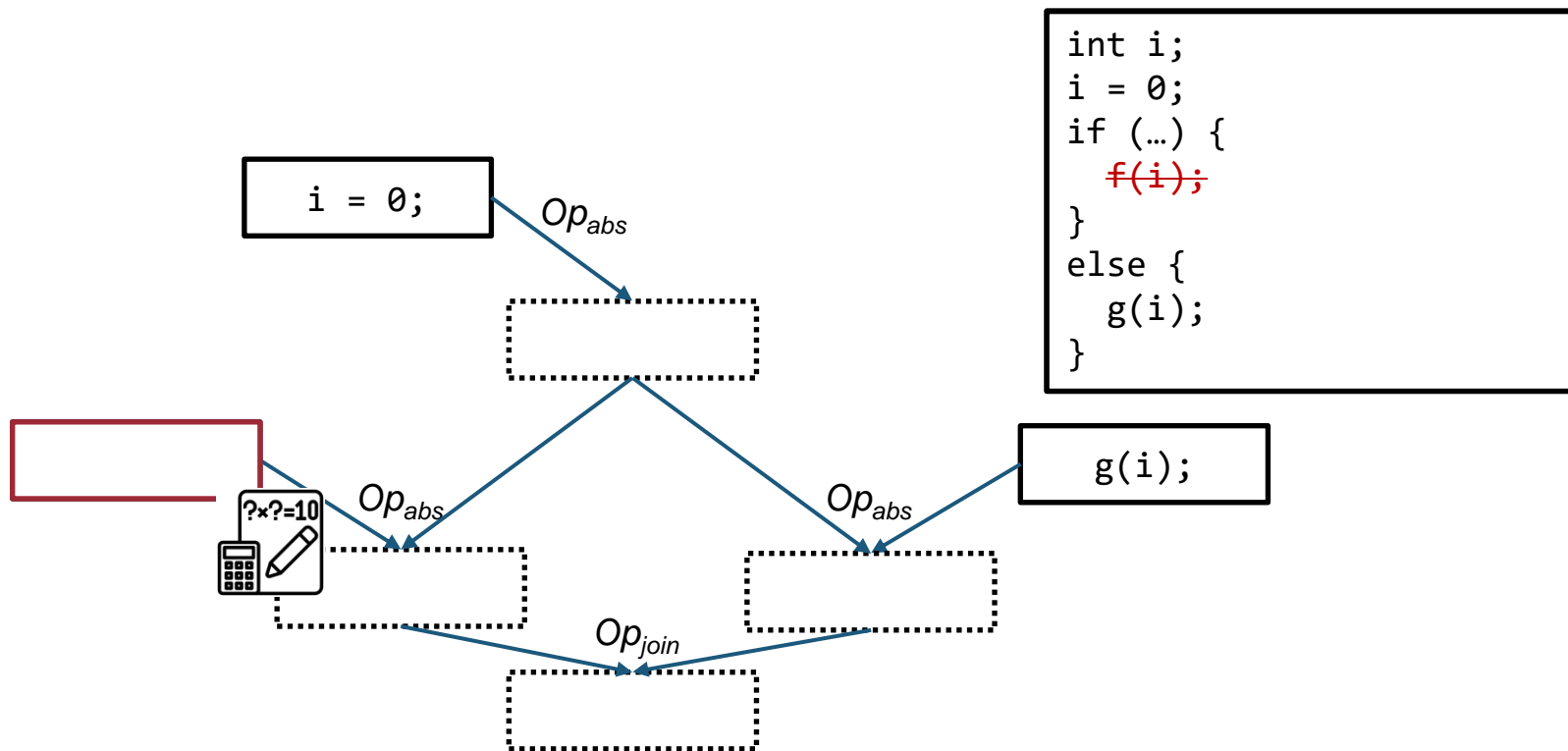
DAIG with Incremental Update



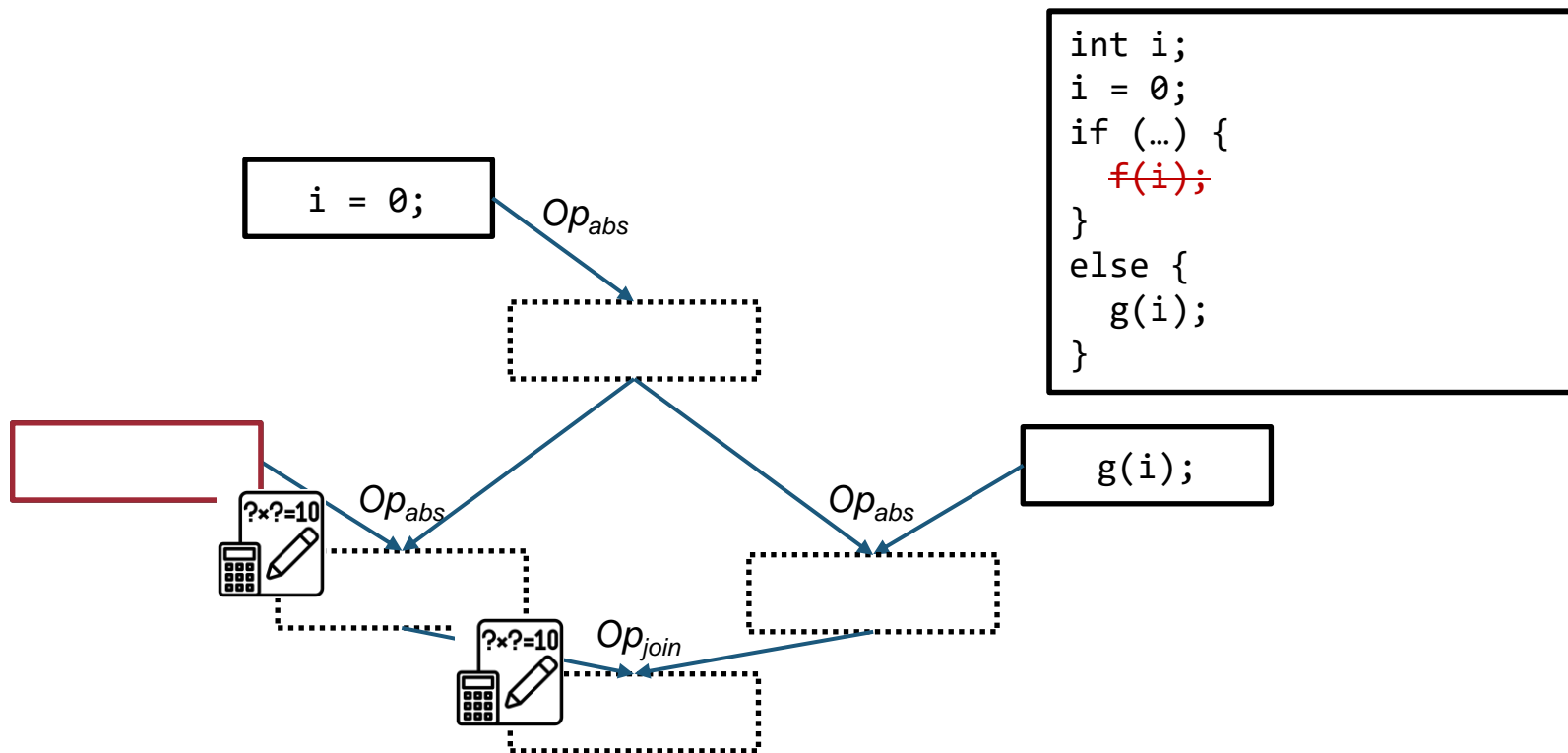
DAIG with Incremental Update



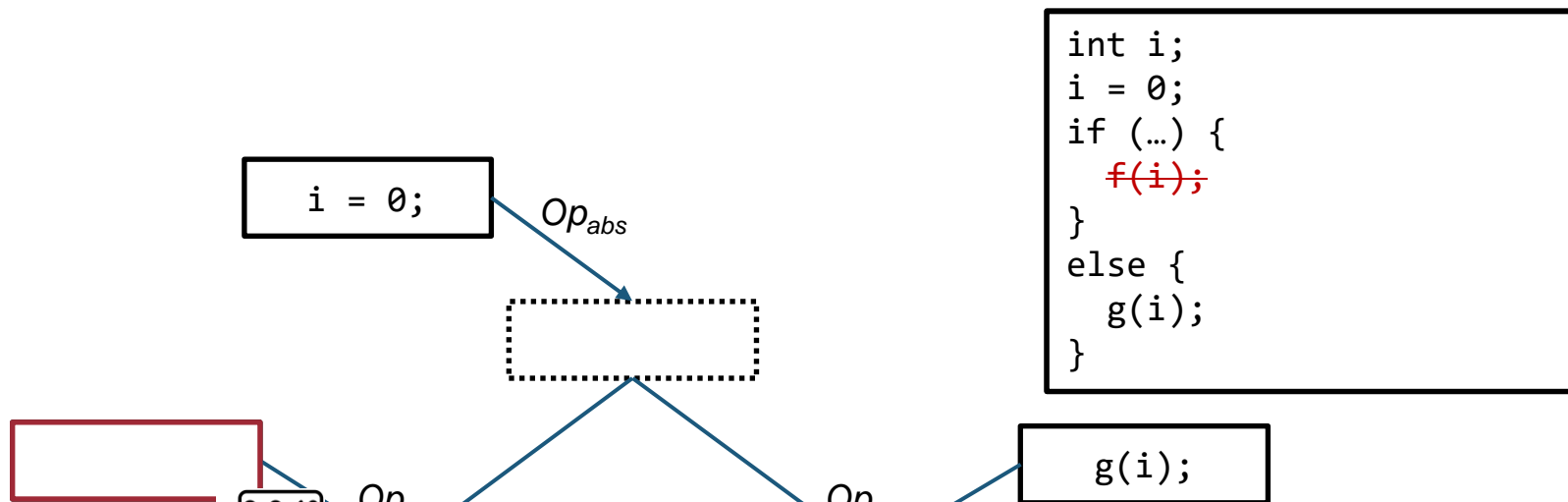
DAIG with Incremental Update



DAIG with Incremental Update

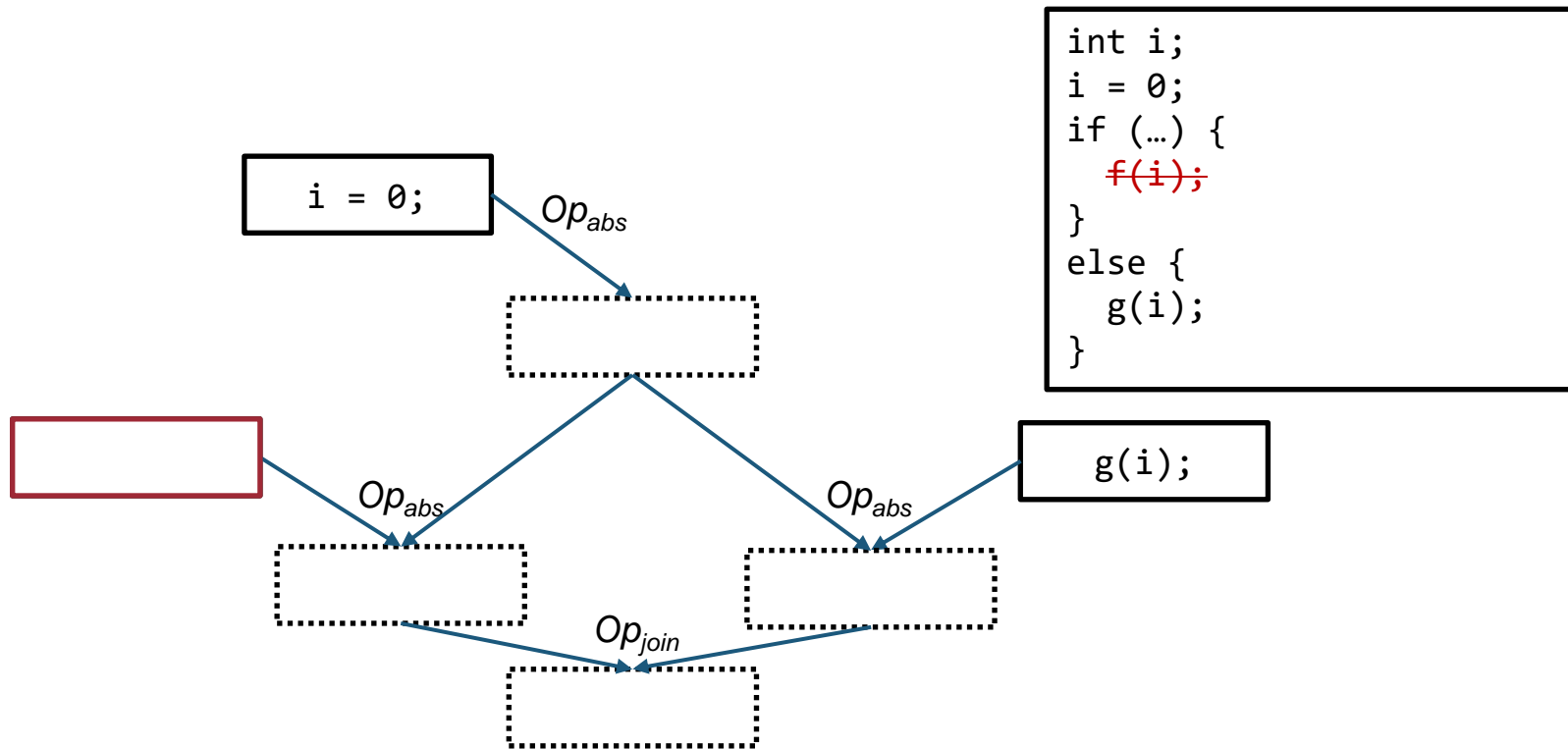


DAIG with Incremental Update

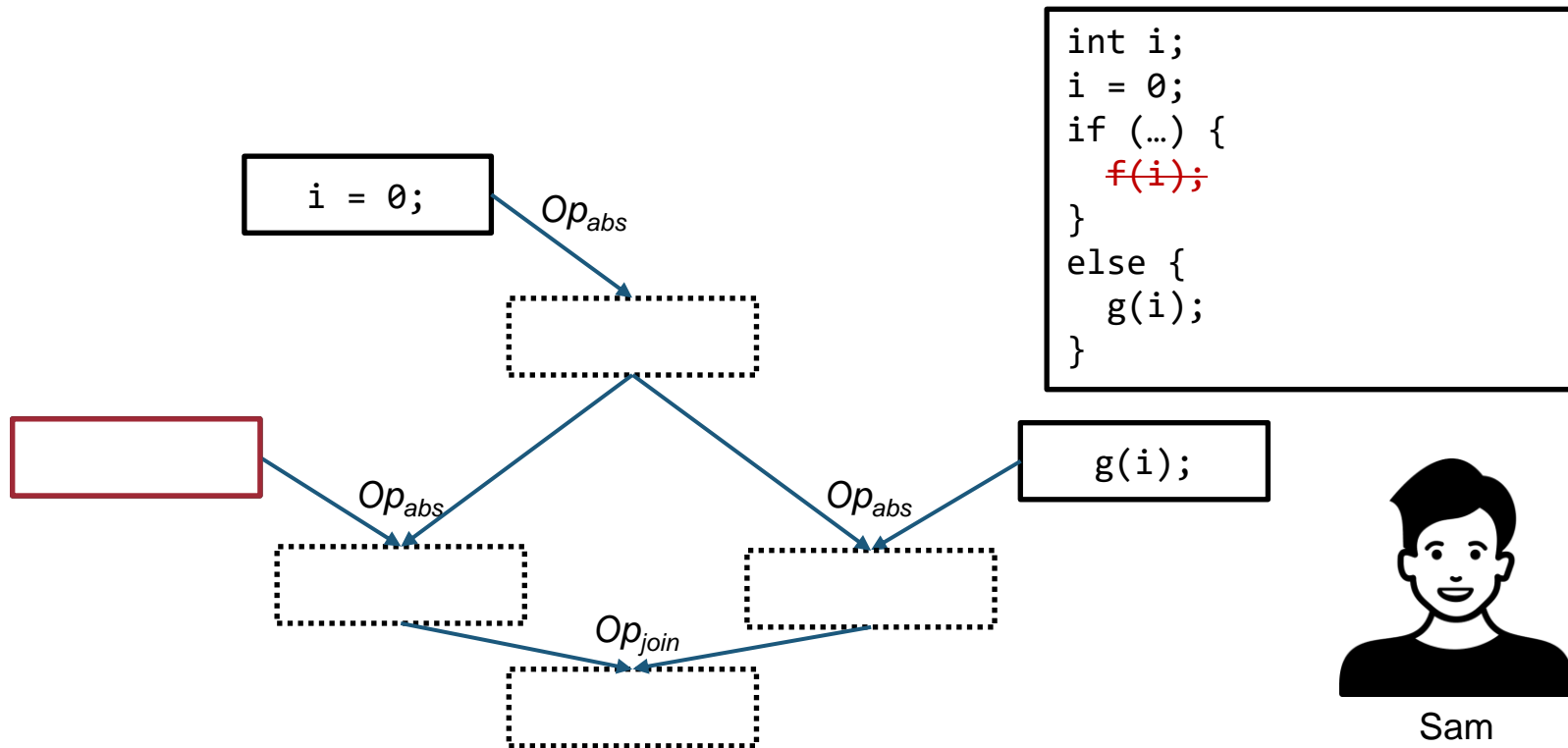


Evaluate the affected states only!

DAIG with Demand-Driven Evaluation

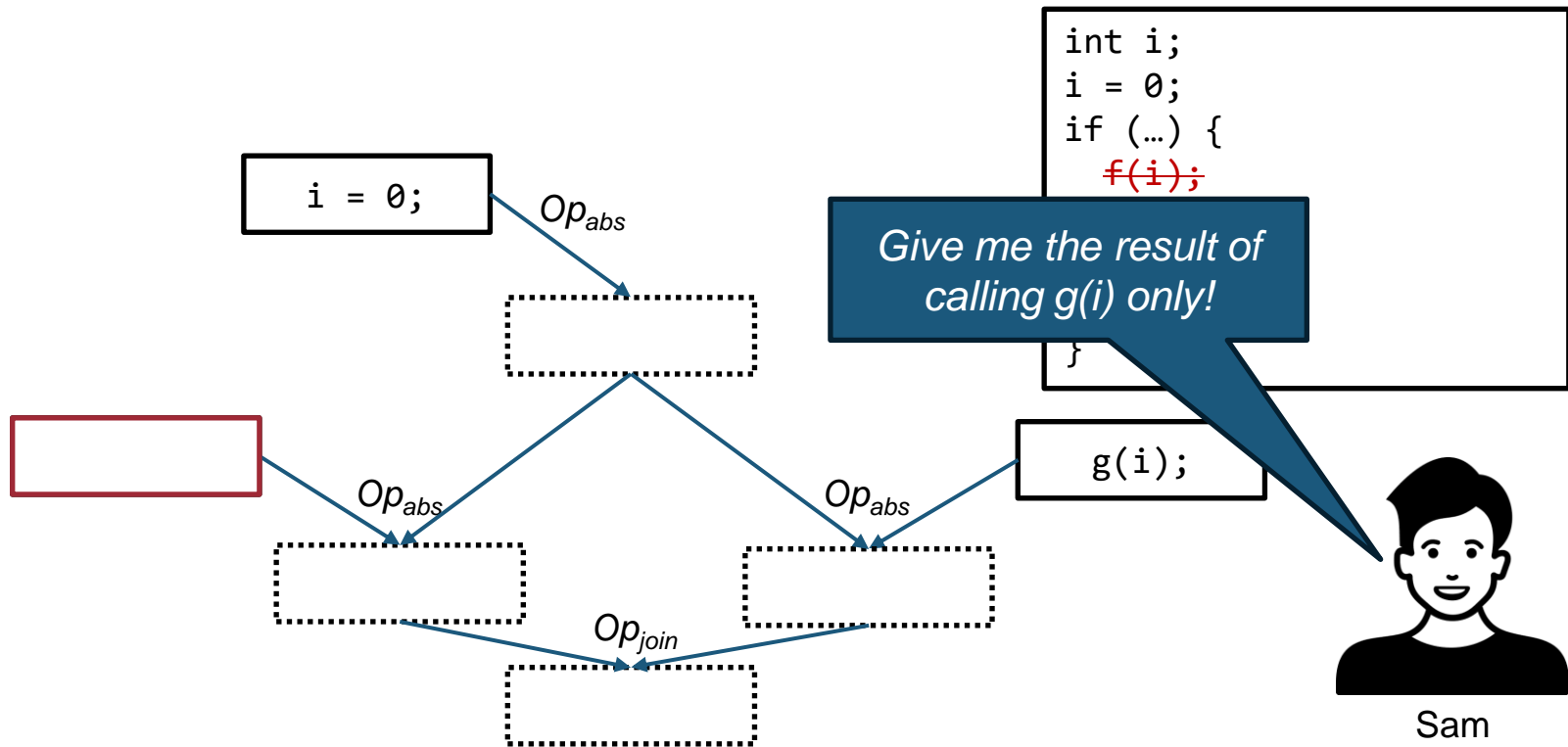


DAIG with Demand-Driven Evaluation

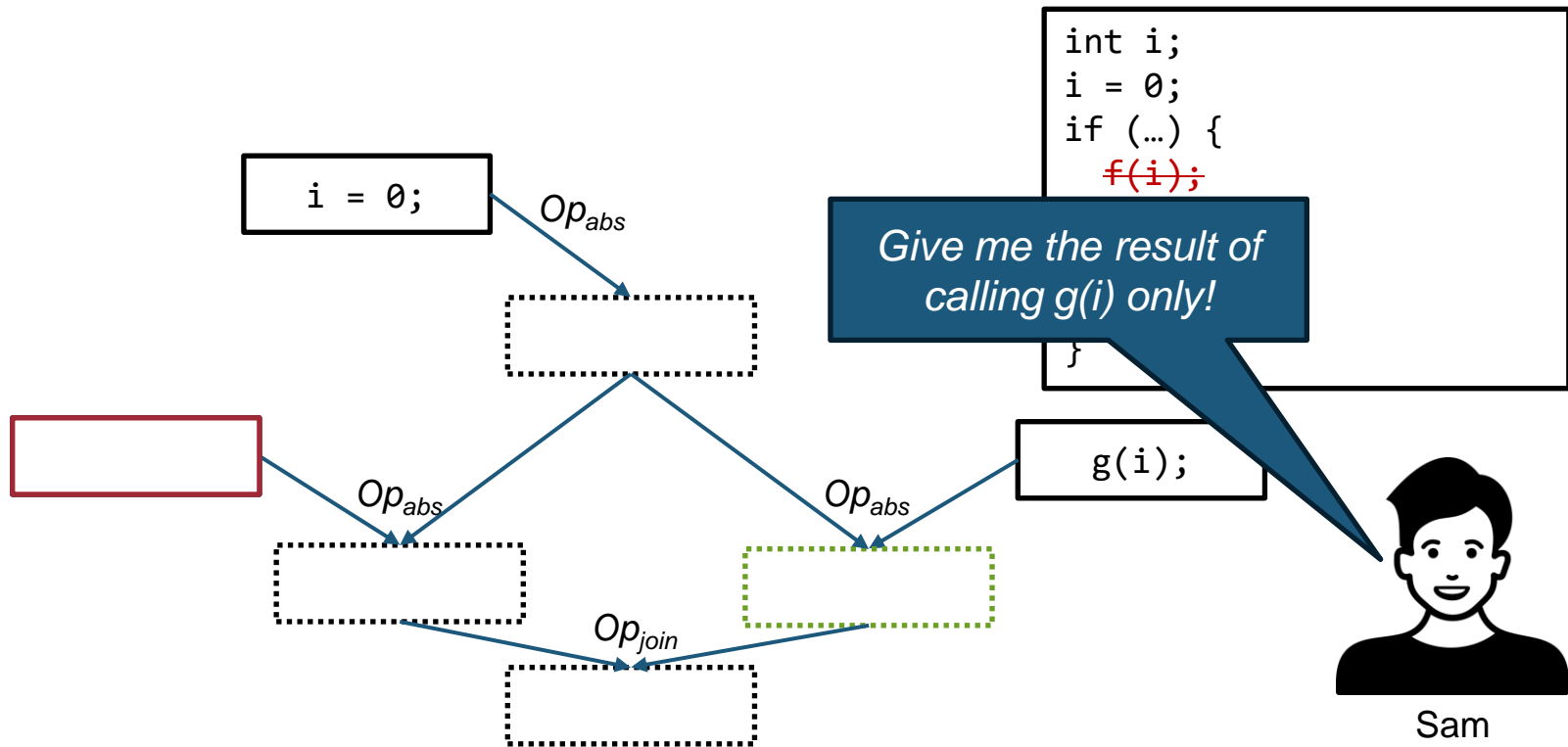


Sam

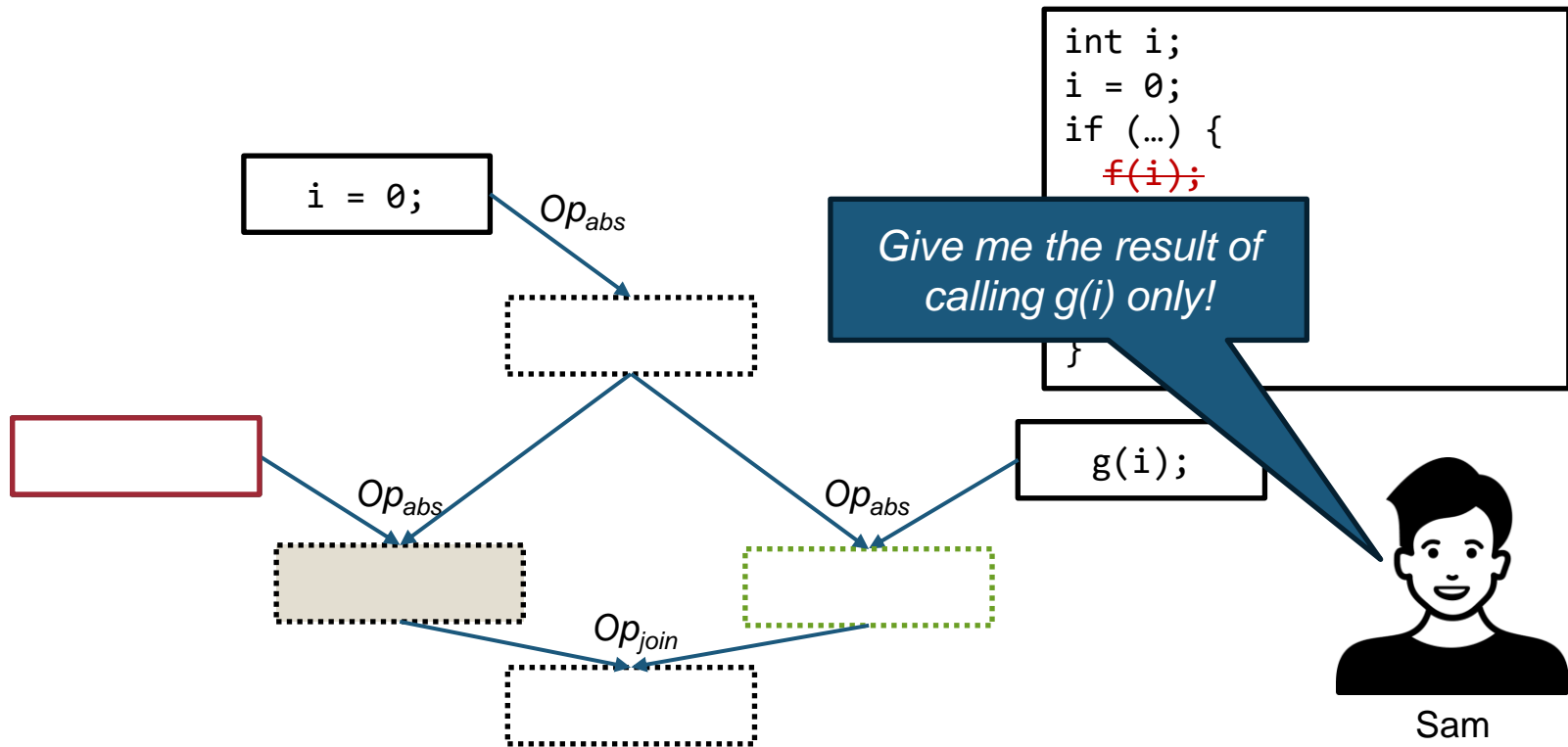
DAIG with Demand-Driven Evaluation



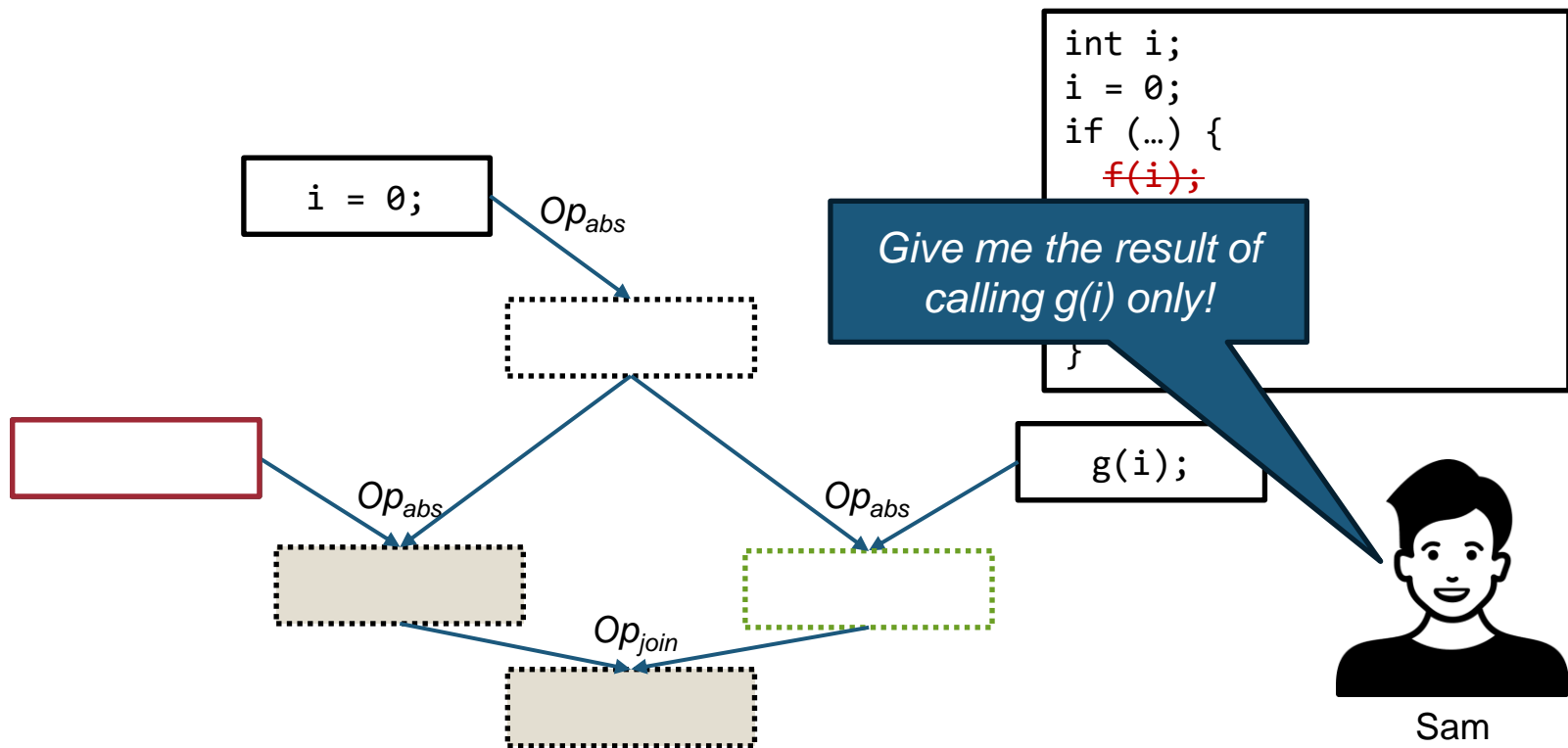
DAIG with Demand-Driven Evaluation



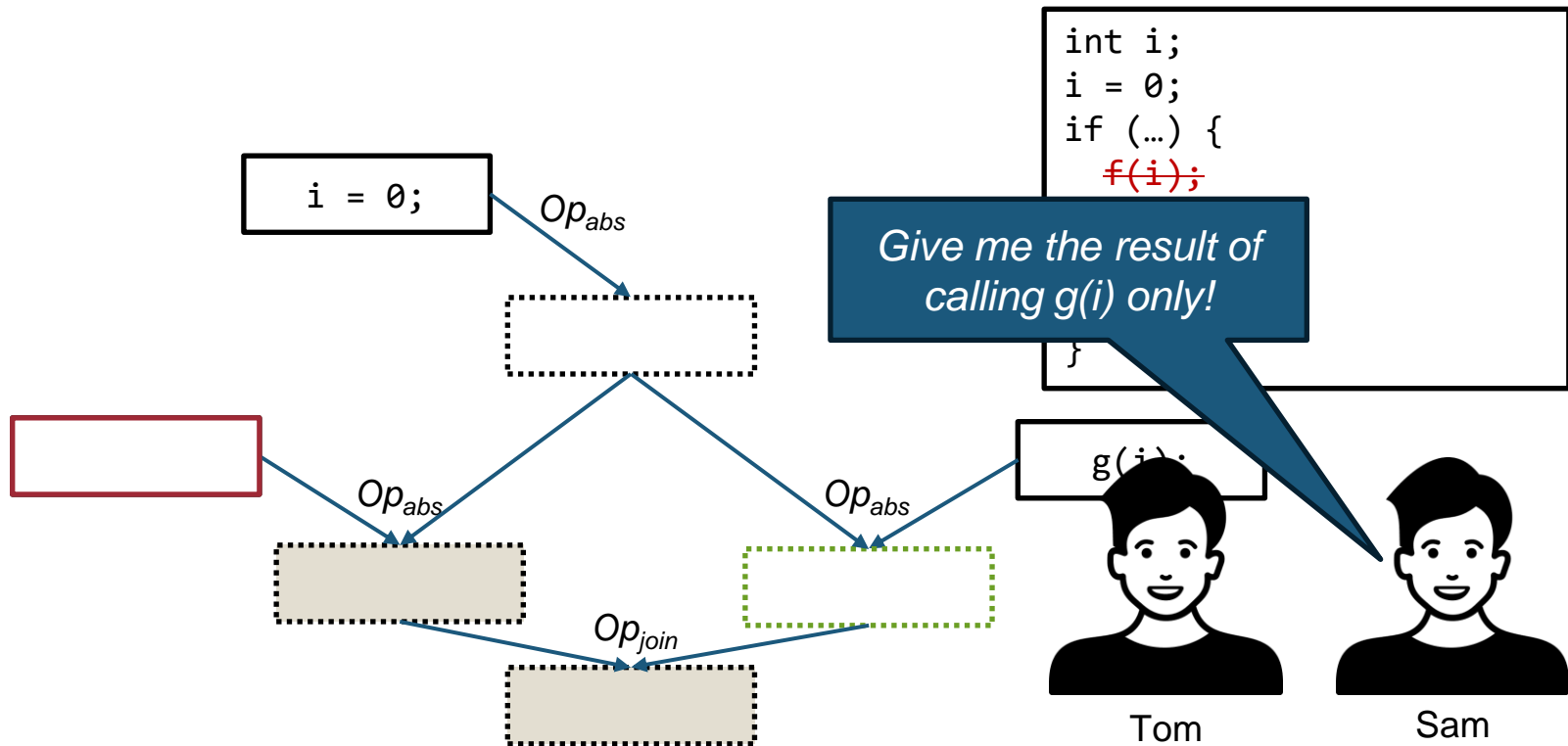
DAIG with Demand-Driven Evaluation



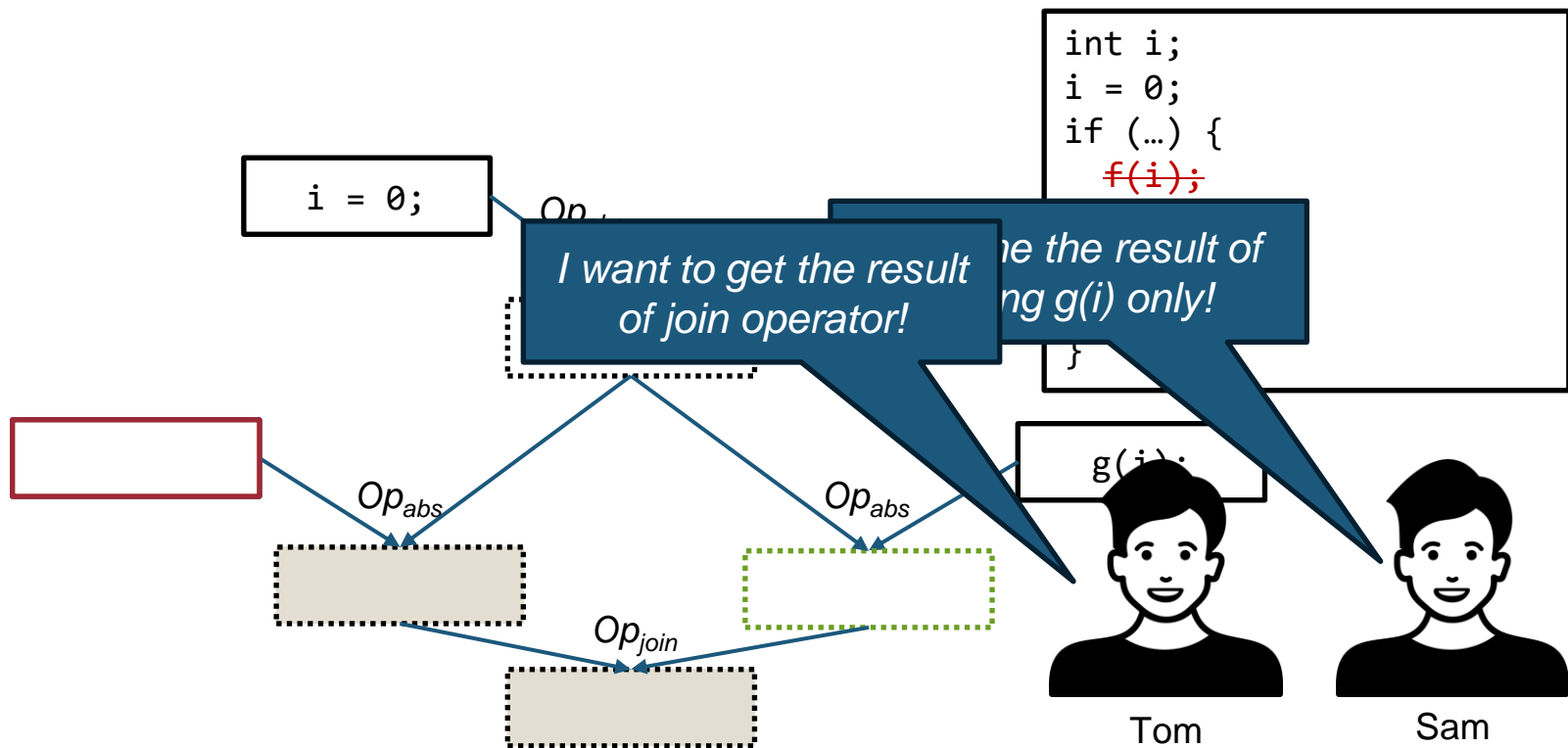
DAIG with Demand-Driven Evaluation



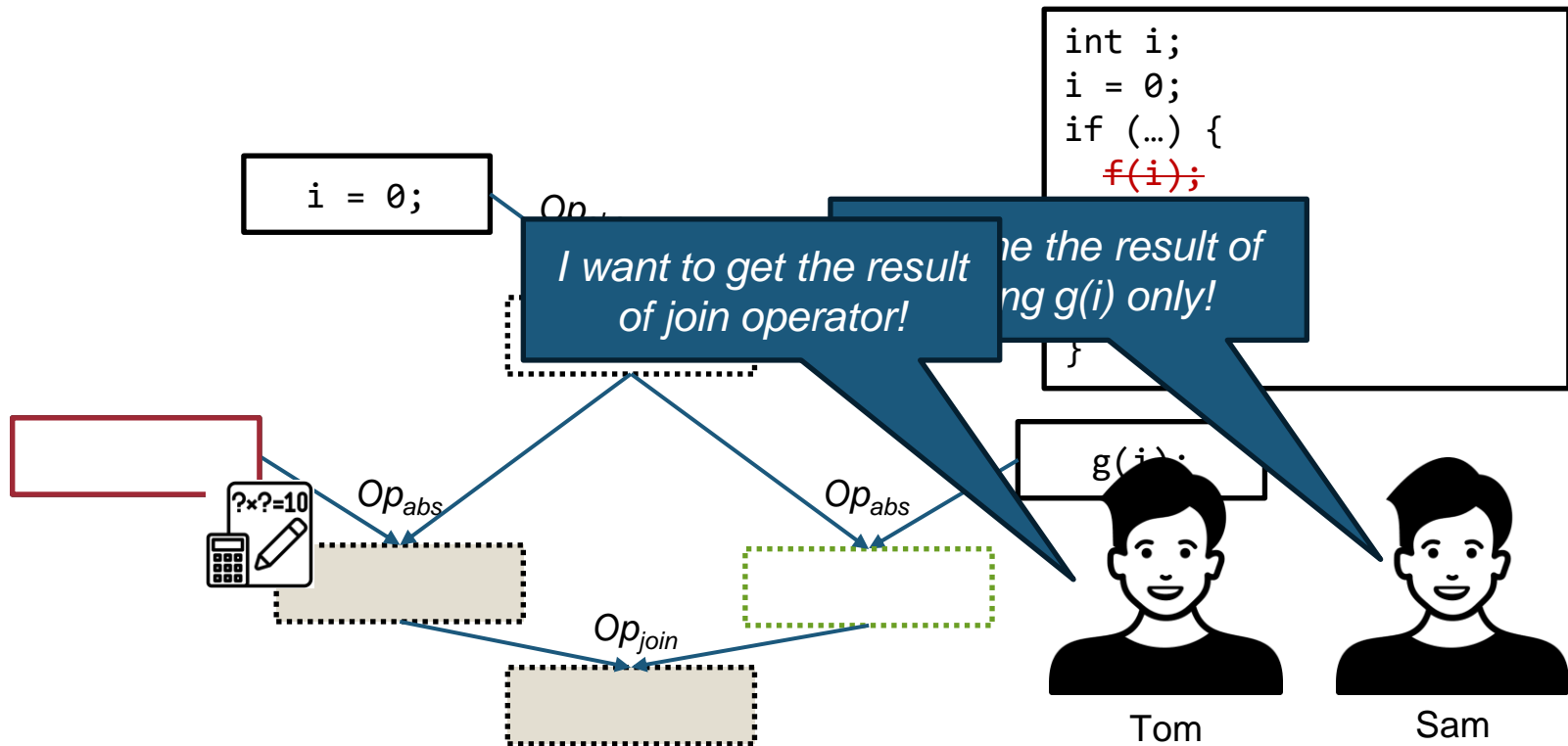
DAIG with Demand-Driven Evaluation



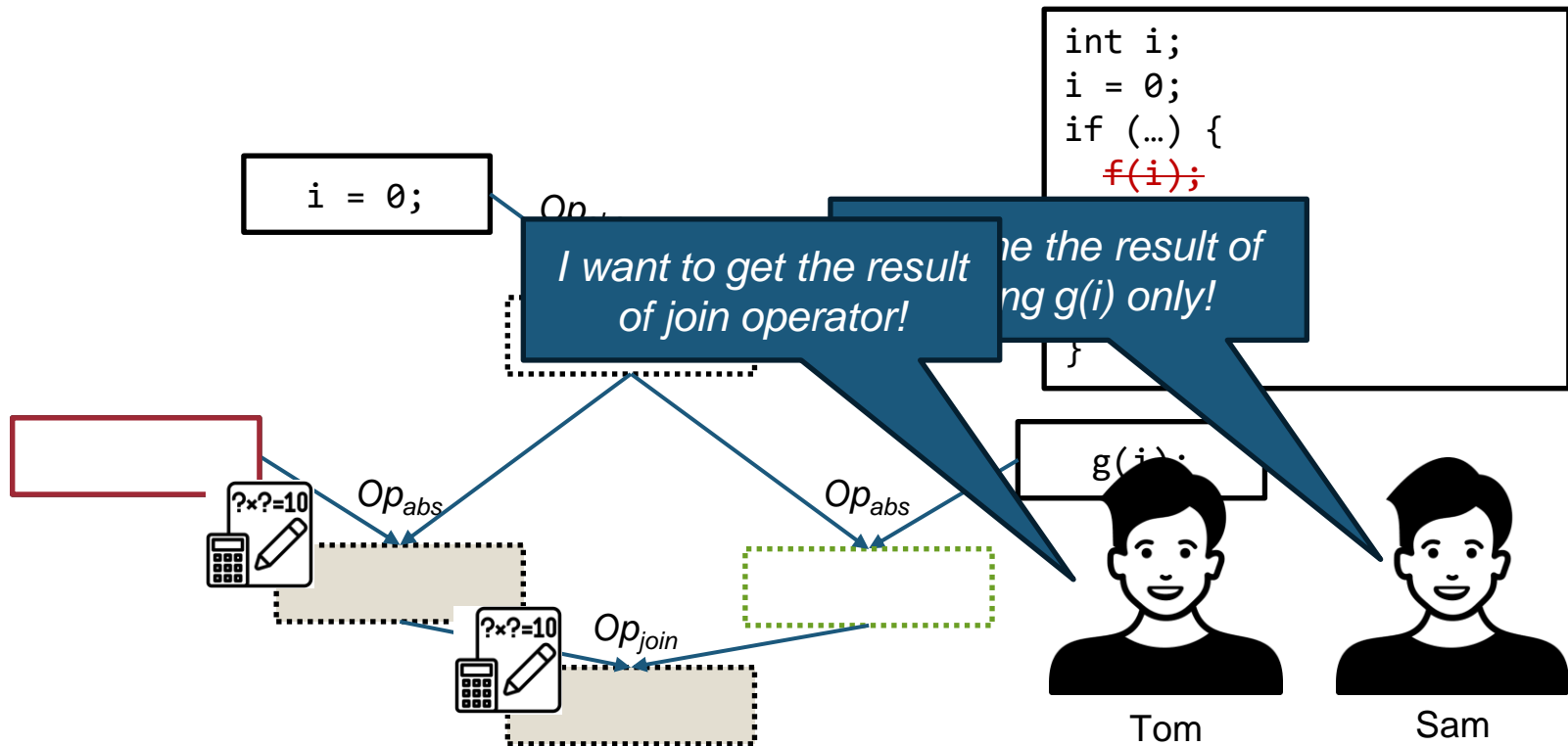
DAIG with Demand-Driven Evaluation



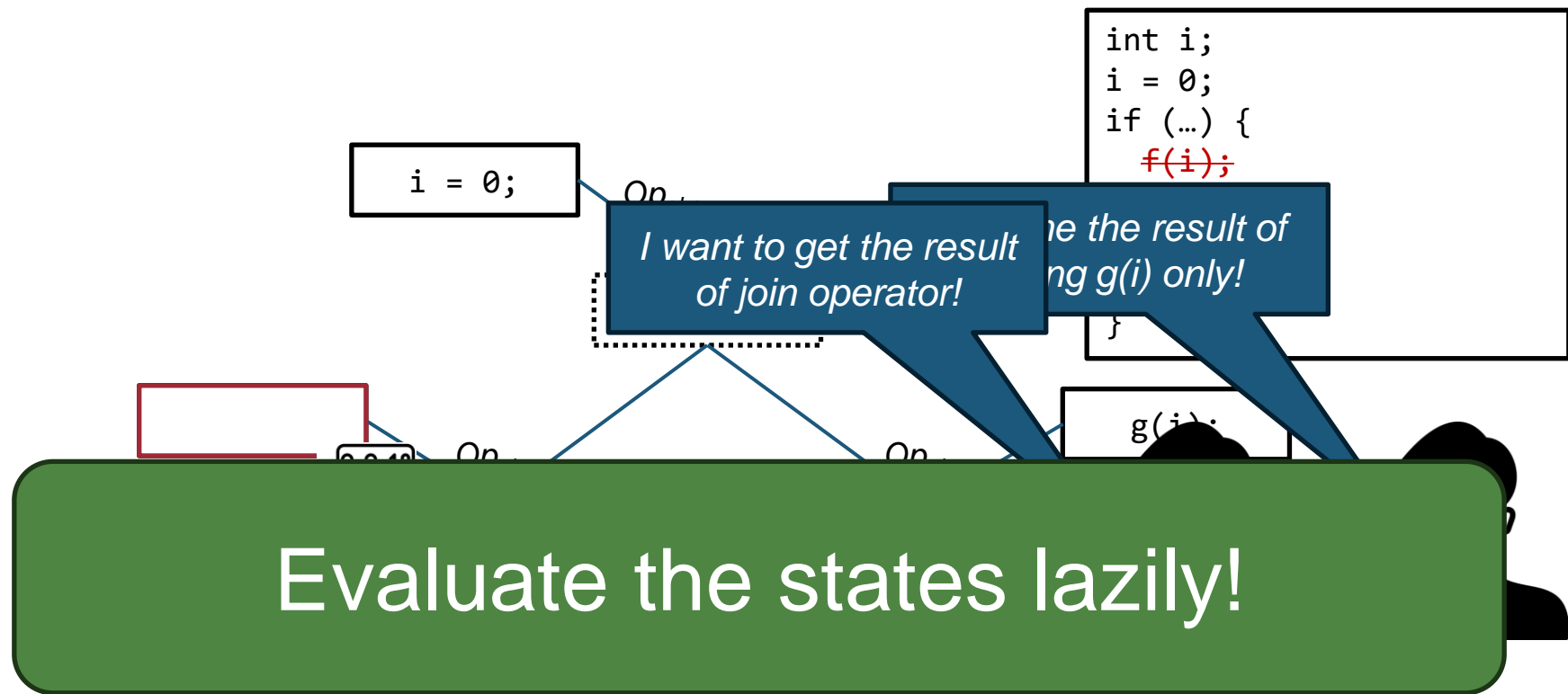
DAIG with Demand-Driven Evaluation



DAIG with Demand-Driven Evaluation



DAIG with Demand-Driven Evaluation



DAIG with Loop

DAIG with Loop

- A loop in a CFG breaks the *acyclicity* of DAIG.

DAIG with Loop

- A loop in a CFG breaks the *acyclicity* of DAIG.
 - It disables the demand-driven analysis.

DAIG with Loop

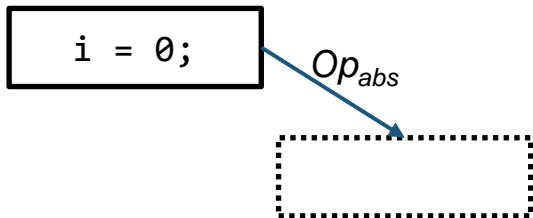
- A loop in a CFG breaks the *acyclicity* of DAIG.
 - It disables the demand-driven analysis.
- *Unroll* the recursion in AI.

DAIG with Loop

DAIG with Loop

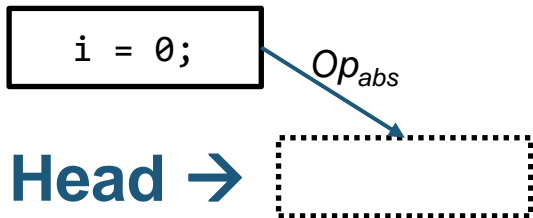
```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop



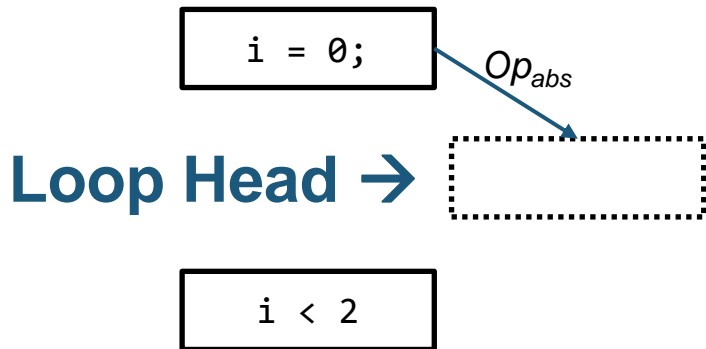
```
int i = 0;  
while (i < 2) {  
    i += 1;  
}
```


DAIG with Loop



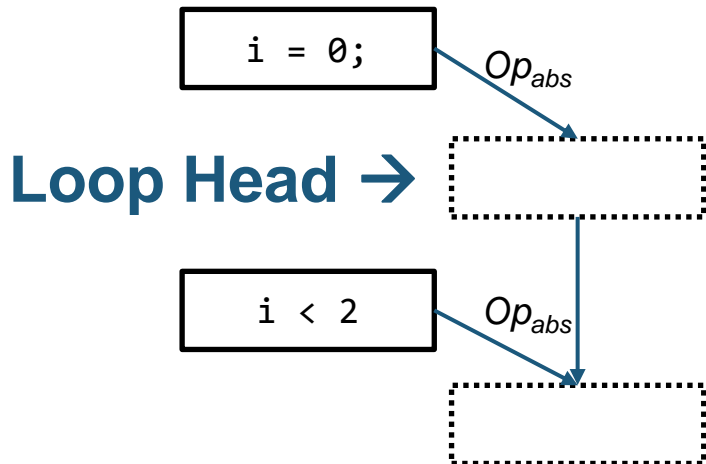
```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop



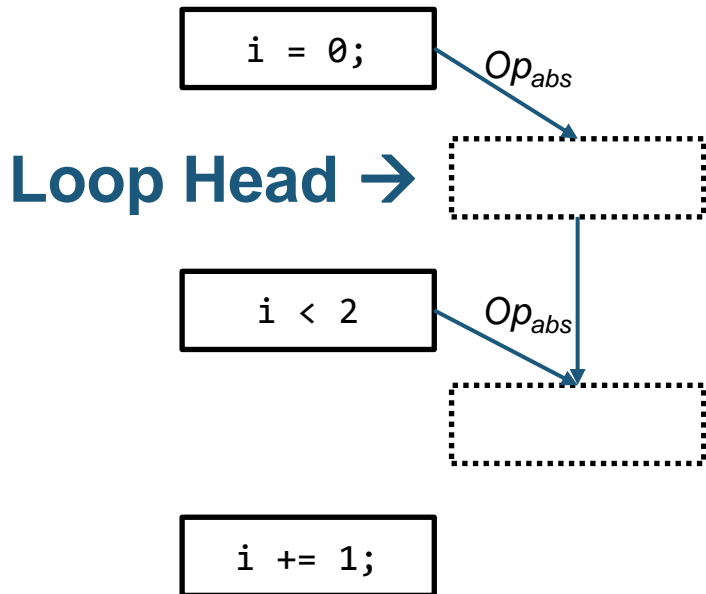
```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop



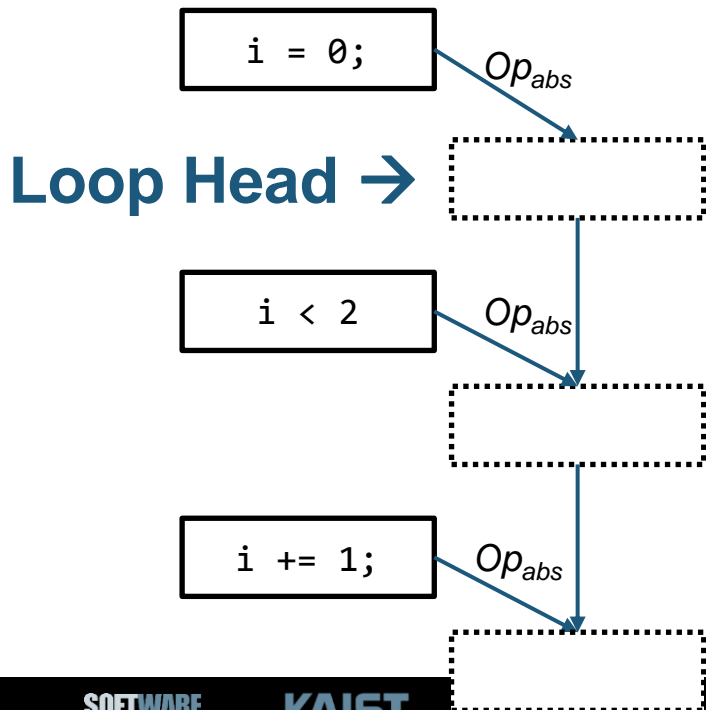
```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop



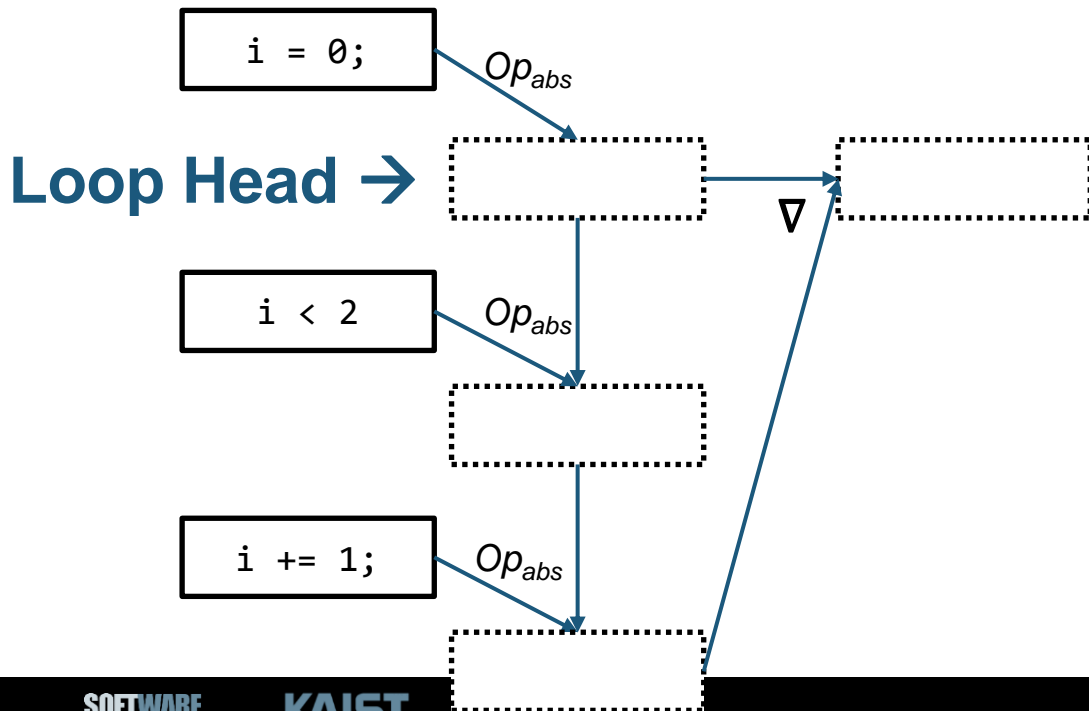
```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop



```
int i = 0;
while (i < 2) {
    i += 1;
}
```

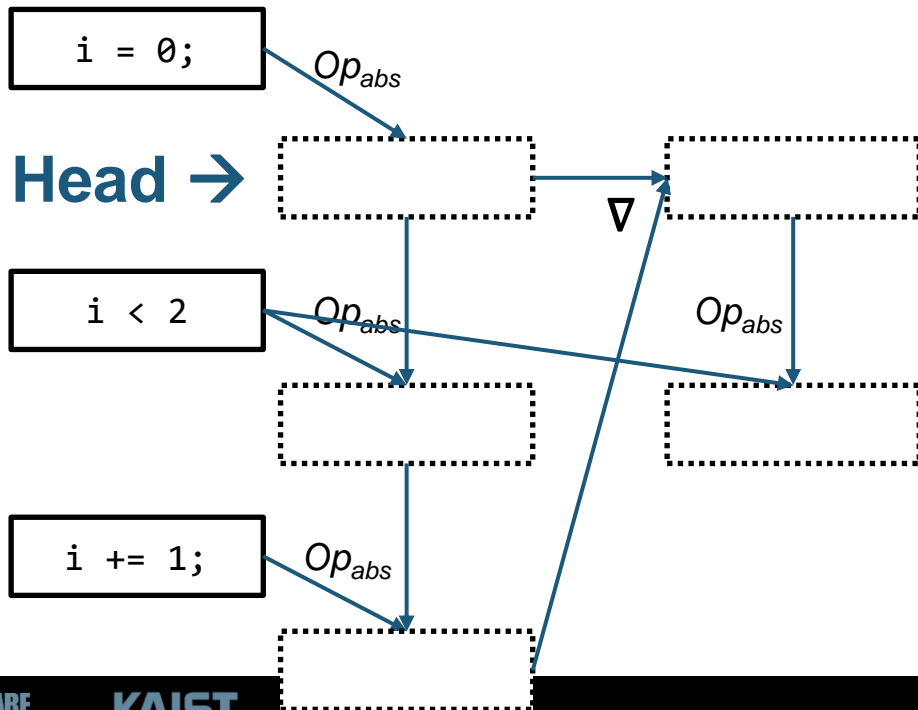
DAIG with Loop



```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop

Loop Head →

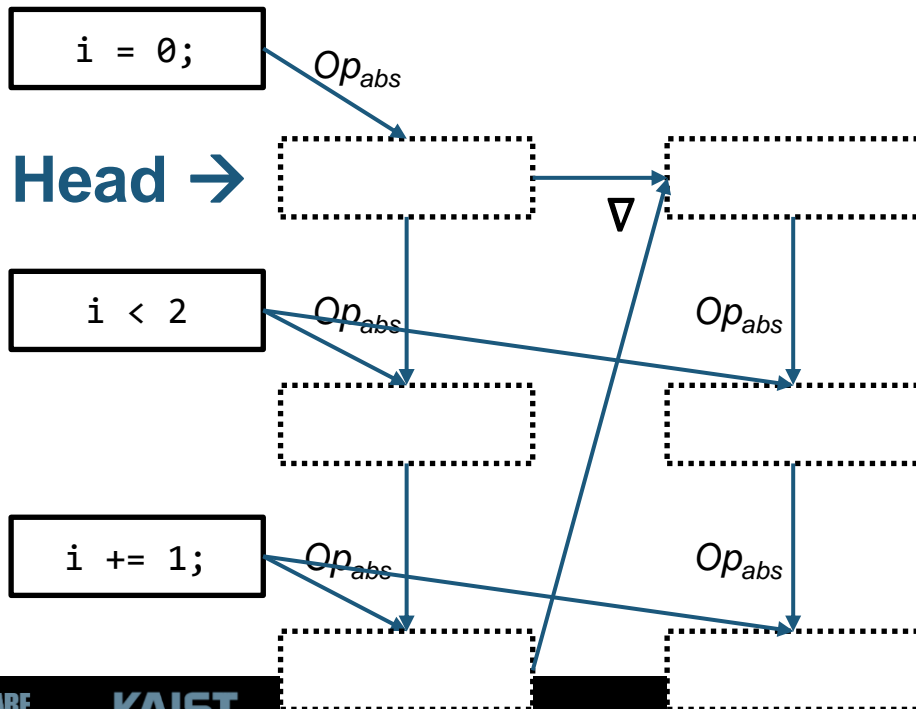


```
int i = 0;
while (i < 2) {
    i += 1;
}
```

DAIG with Loop

```
int i = 0;  
while (i < 2) {  
    i += 1;  
}
```

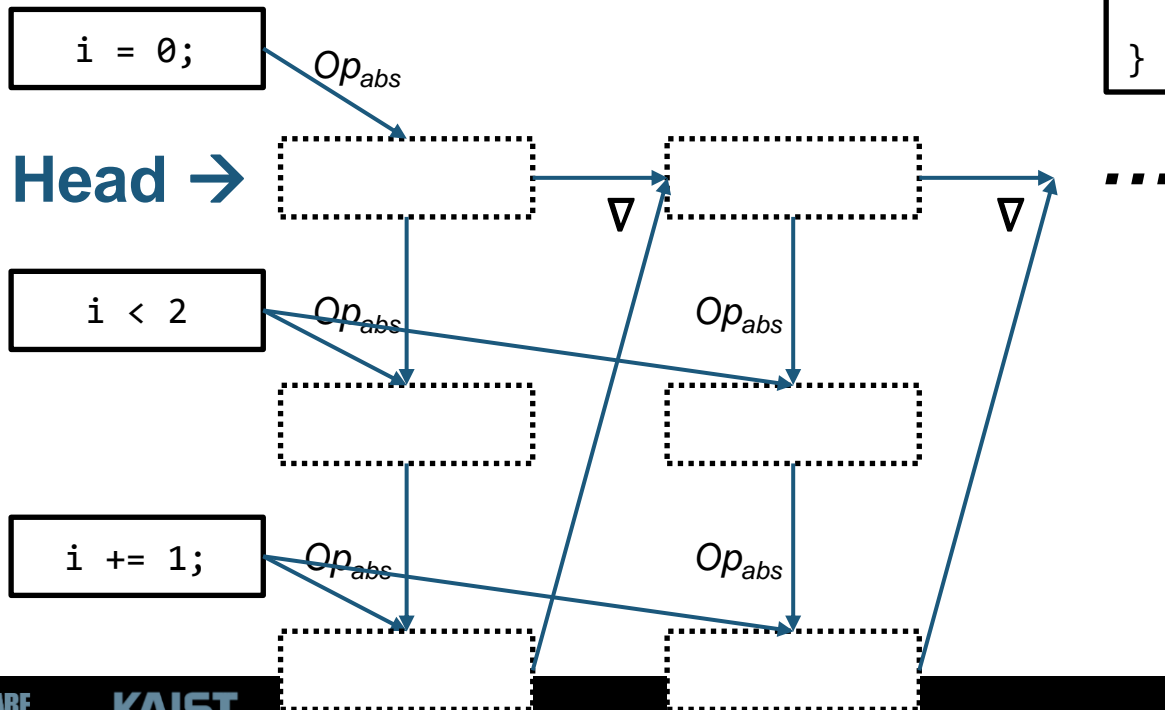
Loop Head →



DAIG with Loop

```
int i = 0;  
while (i < 2) {  
    i += 1;  
}
```

Loop Head →



Evaluation Results

	Mean	p50	p90	p95	p99
B	9.0	1.4	18.9	36.2	173.6
I	1.7	0.6	3.6	6.3	16.6
DD	1.5	0.1	3.7	7.9	16.7
I & DD	0.3	0.1	0.7	1.2	3.0

Batch (B): Apply offline algorithm on every single edit.

Incremental (I): Recalculate incrementally but eagerly.

Demand-Driven (DD): Recalculate lazily but from scratch.

I & DD: Recalculate incrementally and lazily.

* p50, p90, p95, p99: 50th, 90th, 95th, and 99th percentile.

Evaluation Results

	Mean	p50	p90	p95	p99
B	9.0	1.4	18.9	36.2	173.6
I	1.7	0.6	3.6	6.3	16.6
DD	1.5	0.1	3.7	7.9	16.7

DAIG on a dynamic CFG is *30 times* faster than an offline algorithm.

Questions

Questions

- Why propagate the states following the graph reachability?

Questions

- Why propagate the states following the graph reachability?
 - Can't we use the data-flow of variables?

Questions

- Why propagate the states following the graph reachability?
 - Can't we use the data-flow of variables?
 - The current design looks too much coarse-grained.

Questions

- Why propagate the states following the graph reachability?
 - Can't we use the data-flow of variables?
 - The current design looks too much coarse-grained.
 - ex) We may propagate the changes to nodes that would *not affected* at all.

Questions

- Why propagate the states following the graph reachability?
 - Can't we use the data-flow of variables?
 - The current design looks too much coarse-grained.
 - ex) We may propagate the changes to nodes that would *not affected* at all.
- Why explicitly generate DAIGs?

Questions

- Why propagate the states following the graph reachability?
 - Can't we use the data-flow of variables?
 - The current design looks too much coarse-grained.
 - ex) We may propagate the changes to nodes that would *not affected* at all.
- Why explicitly generate DAIGs?
 - Can't we implement the design seamlessly within an AI engine?

Questions

- Why propagate the states following the graph reachability?
 - Can't we use the data-flow of variables?
 - The current design looks too much coarse-grained.
 - ex) We may propagate the changes to nodes that would *not affected* at all.
- Why explicitly generate DAIGs?
 - Can't we implement the design seamlessly within an AI engine?
 - Why should we explicitly store the unrolled flow from an AI?

Summary

Summary

- DAIG is the first trial to make AI process both incremental and demand-driven.

Summary

- DAIG is the first trial to make AI process both incremental and demand-driven.
- To make it so, the authors recalculated information as little as possible by *minimum propagation* and *lazy evaluation*.

Summary

- DAIG is the first trial to make AI process both incremental and demand-driven.
- To make it so, the authors recalculated information as little as possible by *minimum propagation* and *lazy evaluation*.
 - *Incrementality* and *laziness* are always welcomed!

Summary

- DAIG is the first trial to make AI process both incremental and demand-driven.
- To make it so, the authors recalculated information as little as possible by *minimum propagation* and *lazy evaluation*.
 - *Incrementality* and *laziness* are always welcomed!
- DAIG can benefit the on-the-fly analysis.

Question?