Demanded Abstract Interpretation

Benno Stein
University of Colorado Boulder
benno.stein@colorado.edu

Bor-Yuh Evan Chang
University of Colorado Boulder & Amazon
evan.chang@colorado.edu

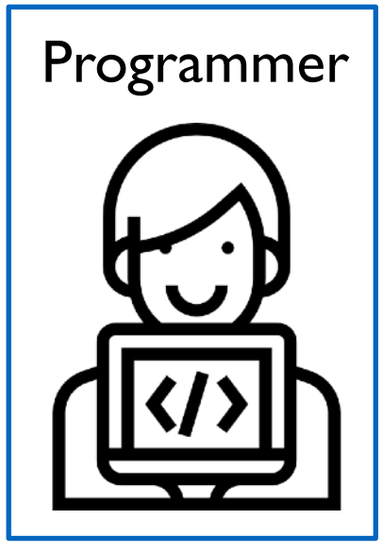
Manu Sridharan
University of California, Riverside
manu@cs.ucr.edu

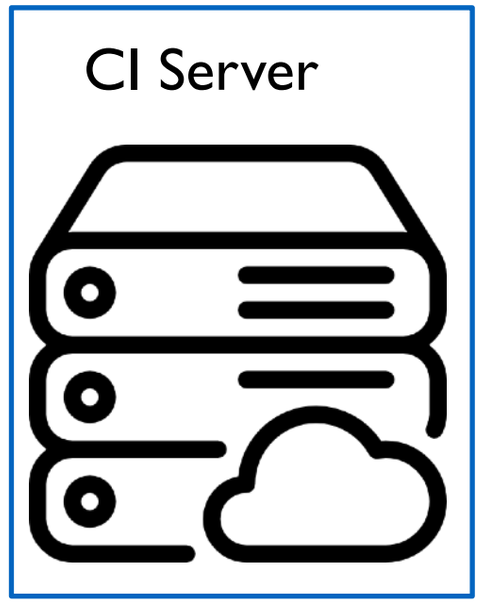
CUPLV

PLDI ‘21

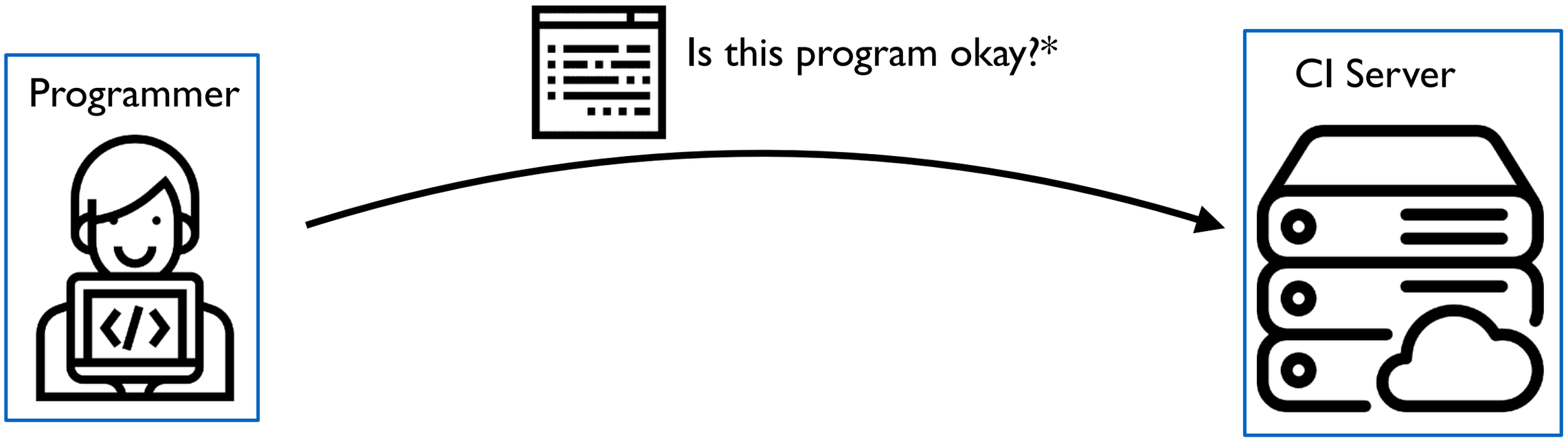
* Bor-Yuh Evan Chang holds concurrent appointments as an Associate Professor at the University of Colorado Boulder and as an Amazon Scholar. This talk describes work performed at the University of Colorado Boulder and is not associated with Amazon.

Batch Static Analysis Workflow



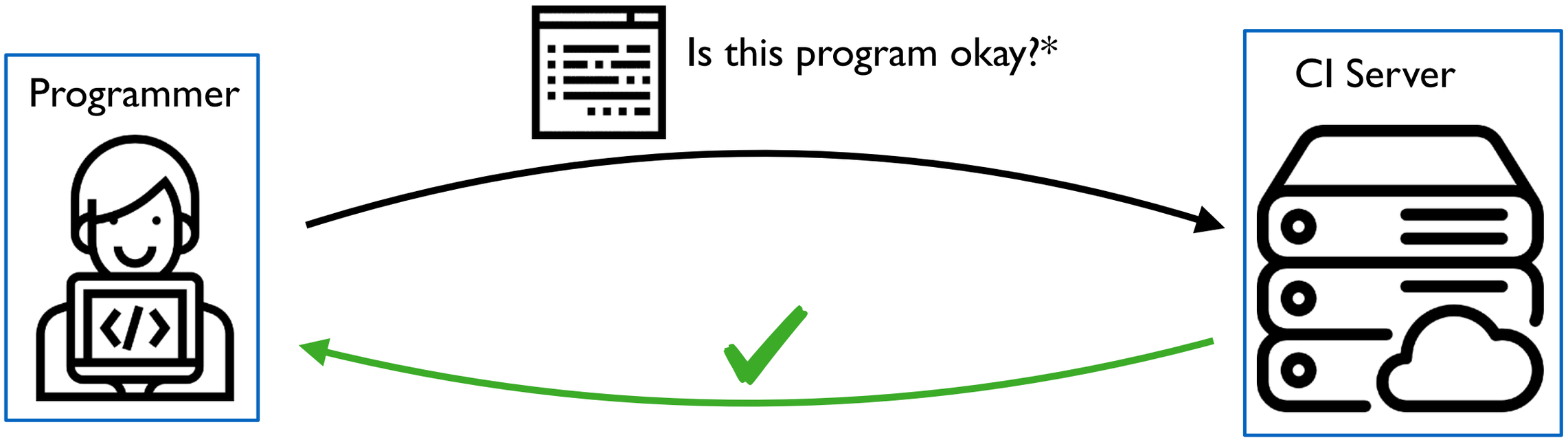


Batch Static Analysis Workflow



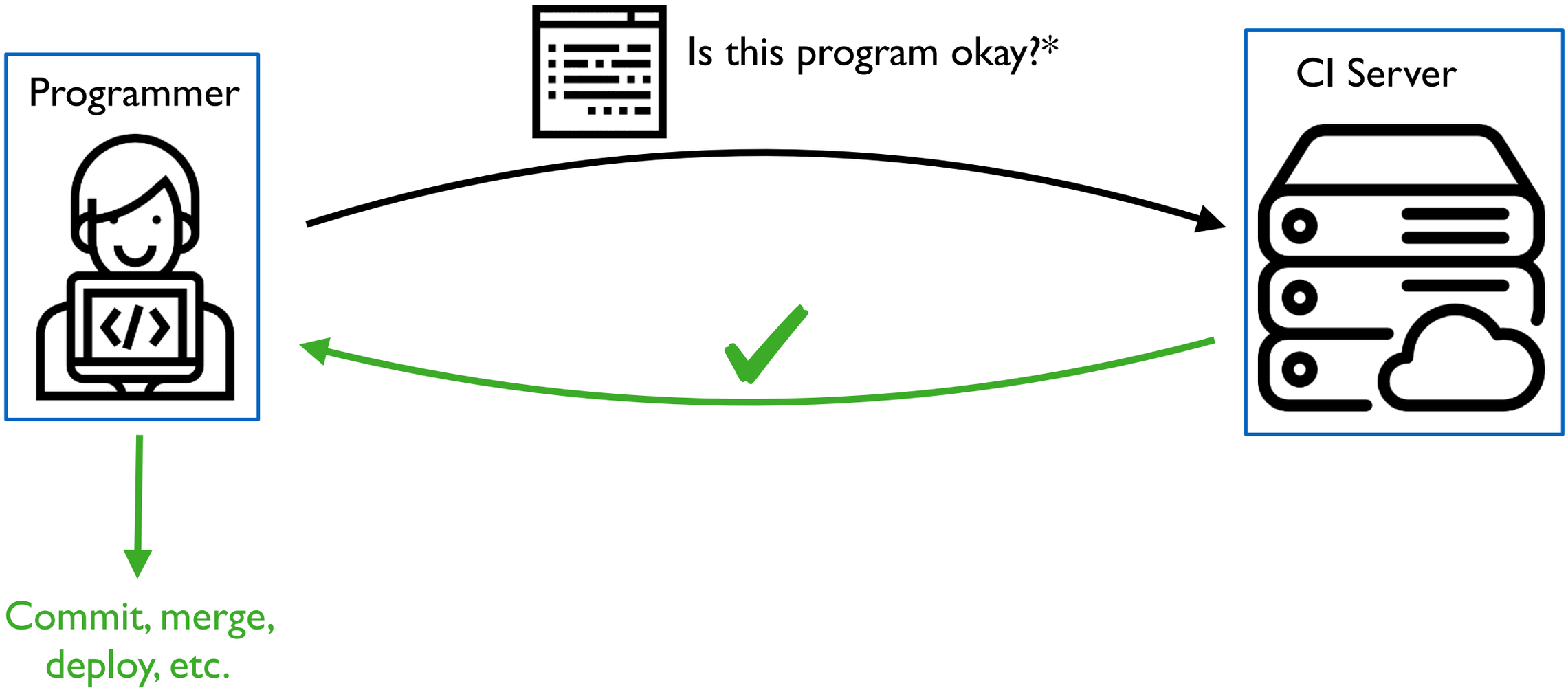
* i.e. safe with respect to some program analyzer

Batch Static Analysis Workflow



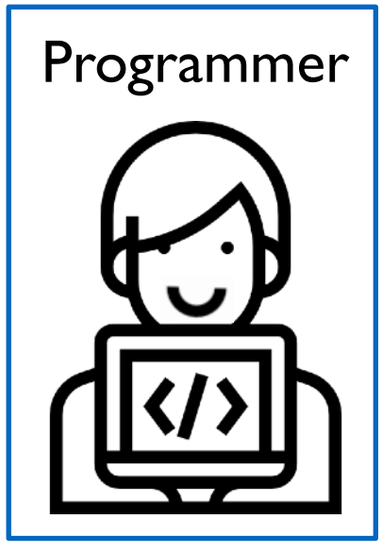
* i.e. safe with respect to some program analyzer

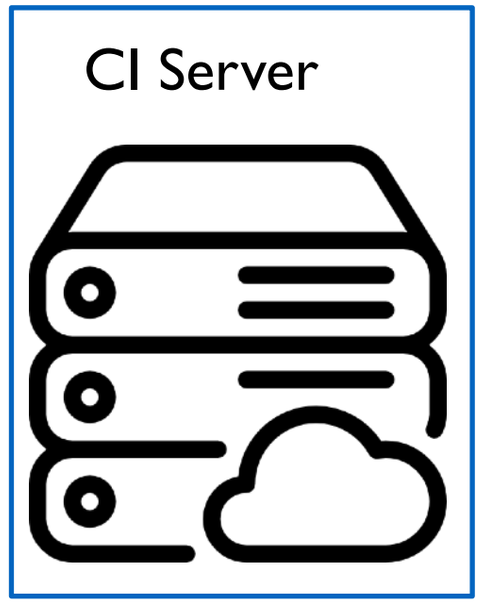
Batch Static Analysis Workflow



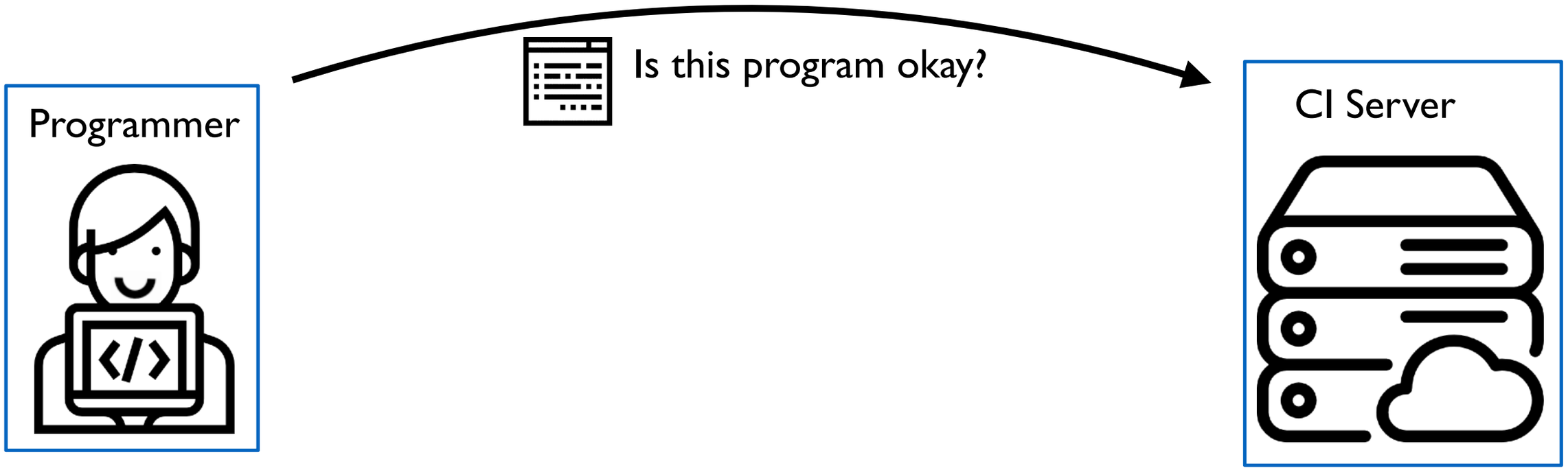
* i.e. safe with respect to some program analyzer

Batch Static Analysis Workflow

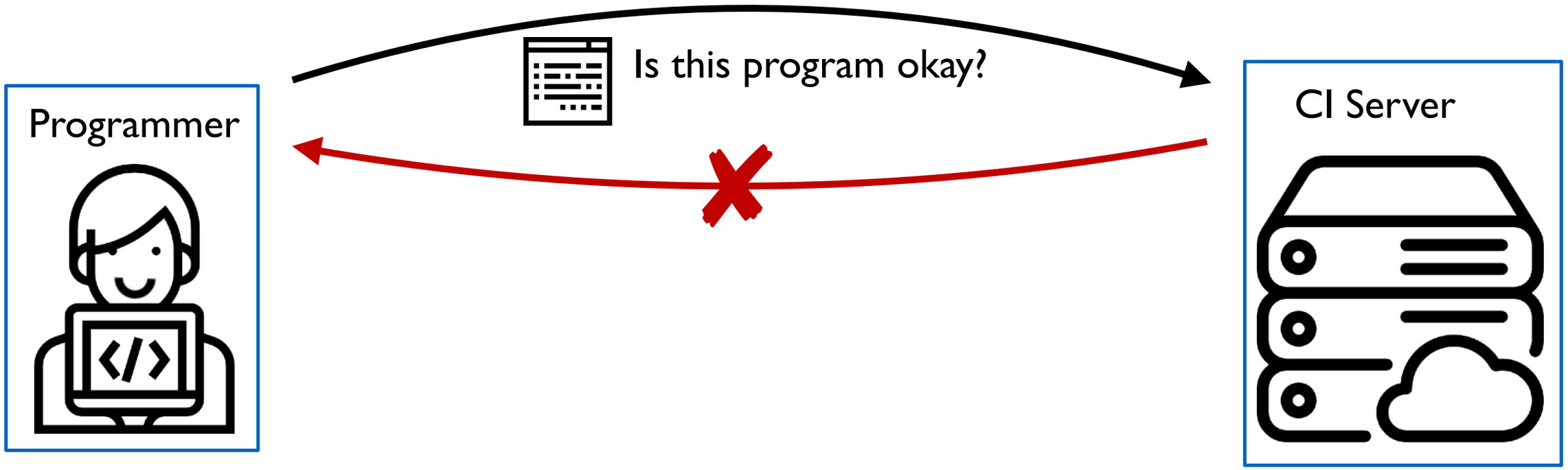




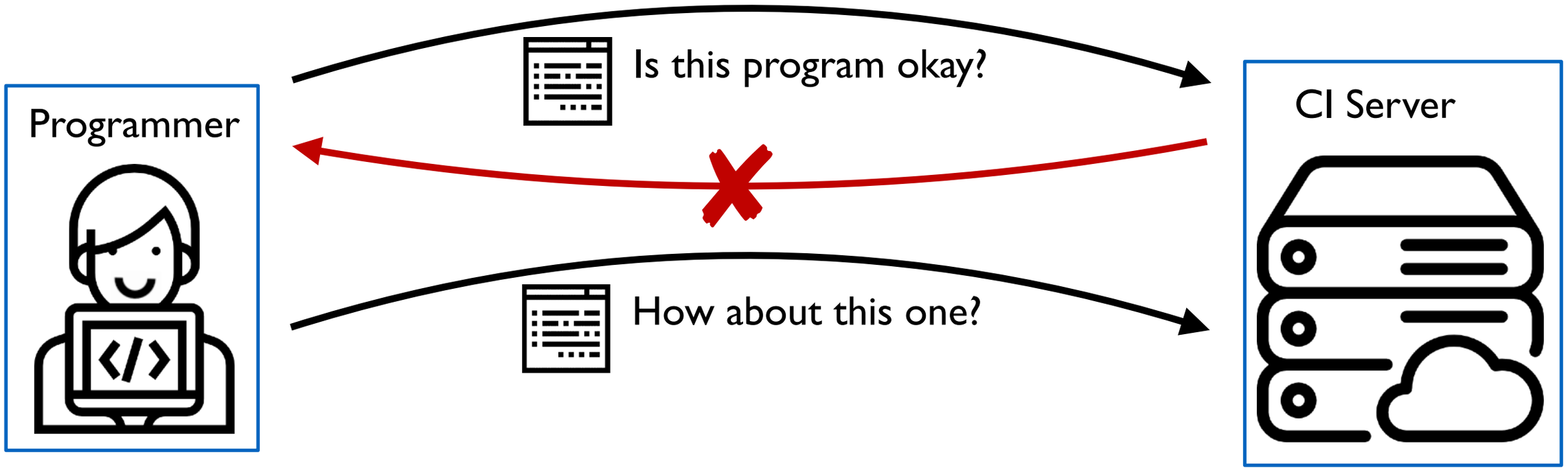
Batch Static Analysis Workflow



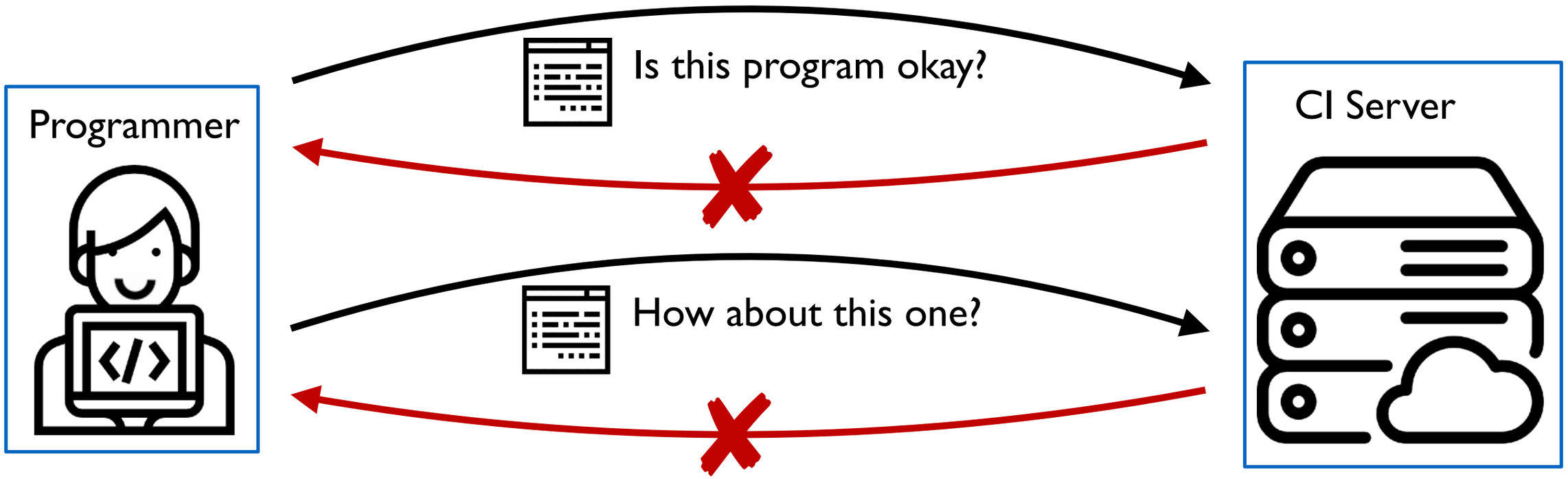
Batch Static Analysis Workflow



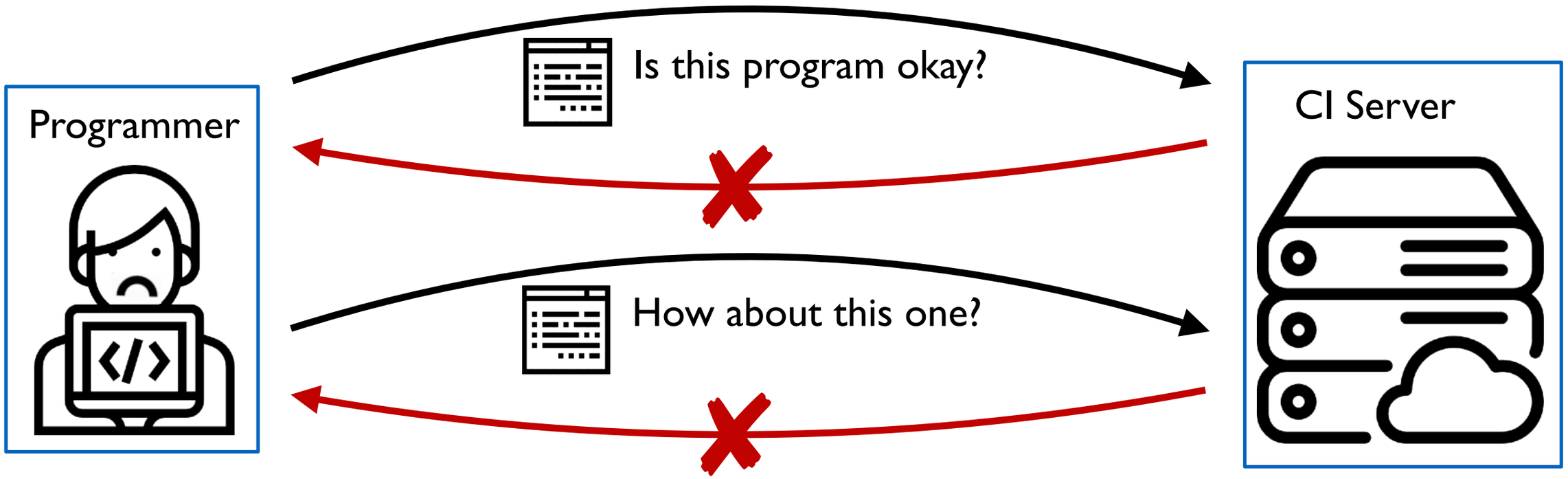
Batch Static Analysis Workflow



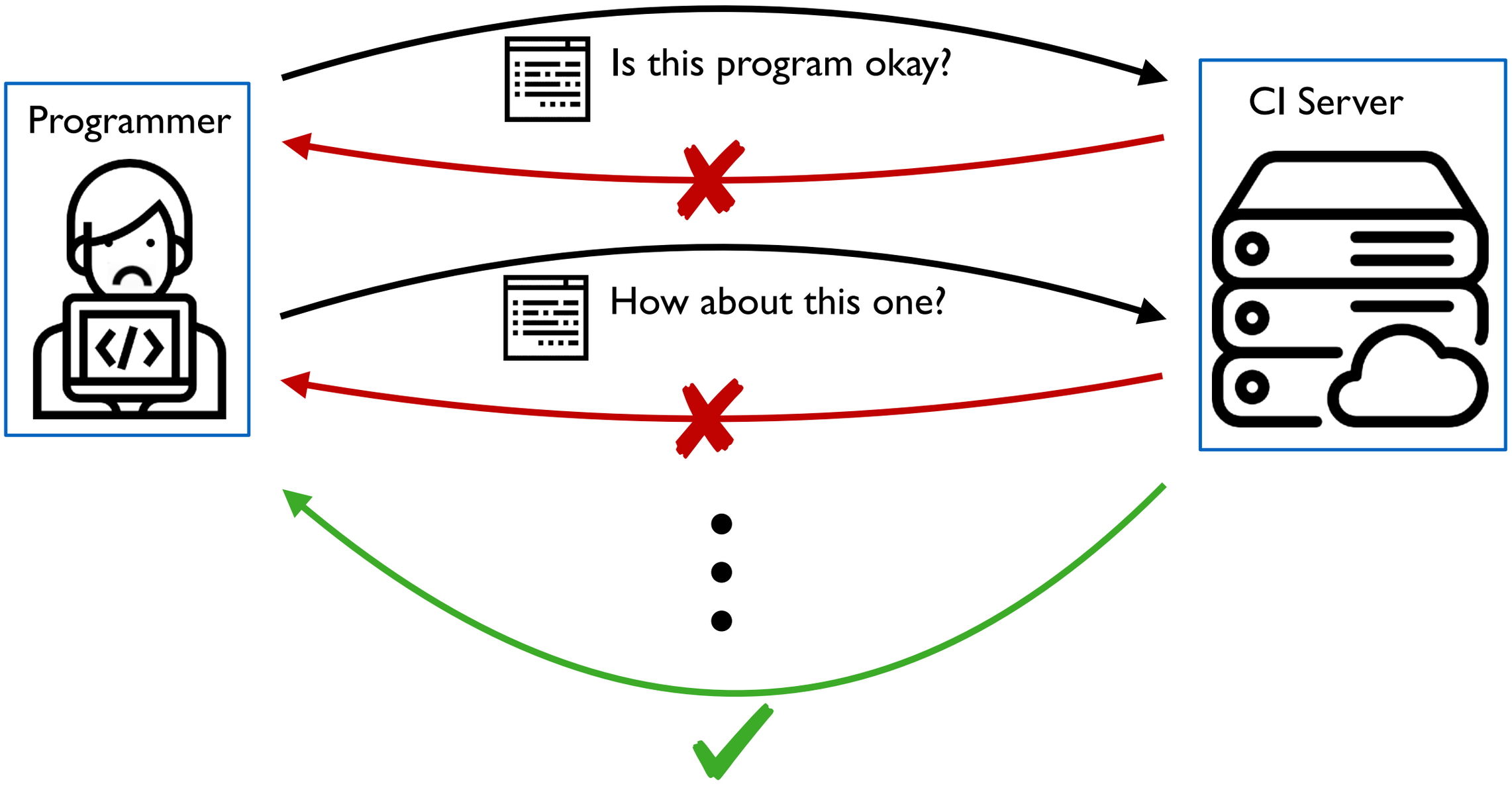
Batch Static Analysis Workflow



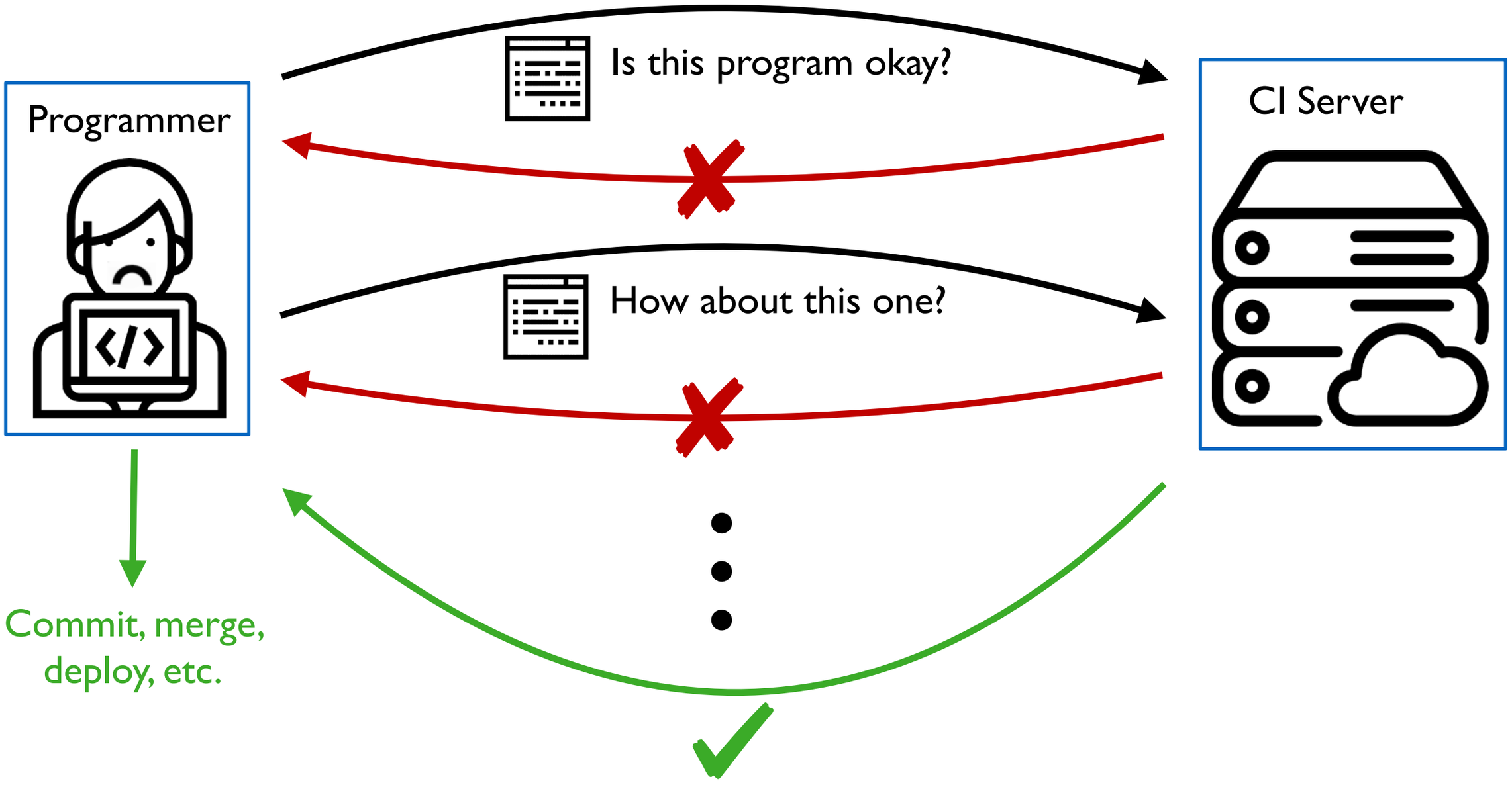
Batch Static Analysis Workflow



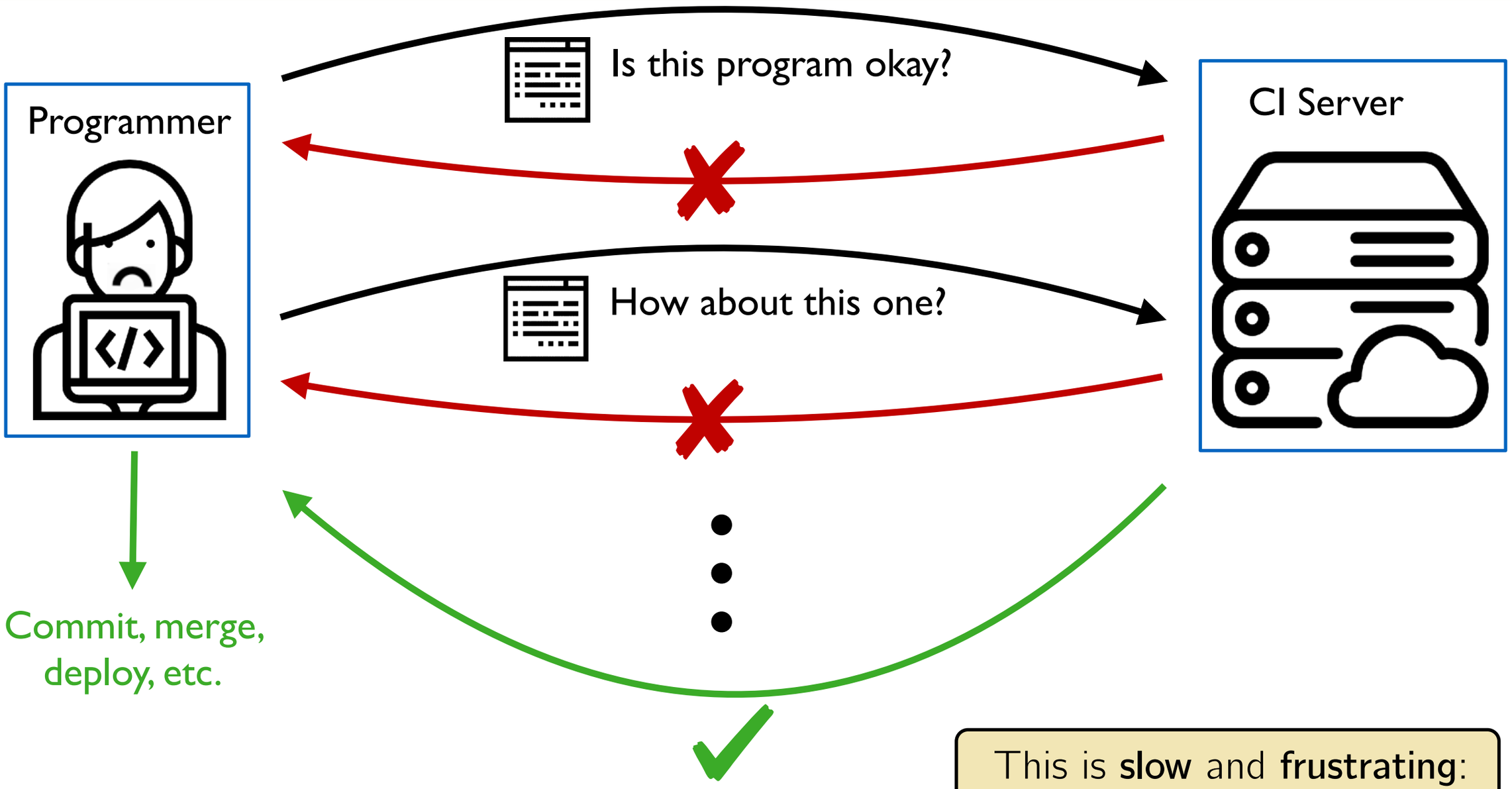
Batch Static Analysis Workflow



Batch Static Analysis Workflow

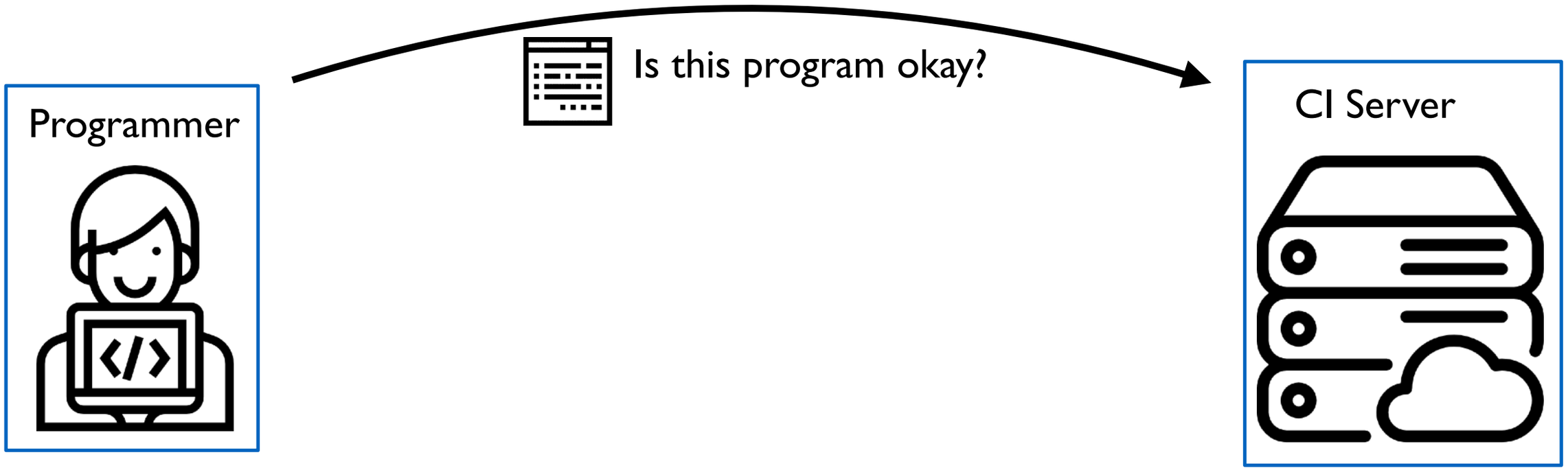


Batch Static Analysis Workflow

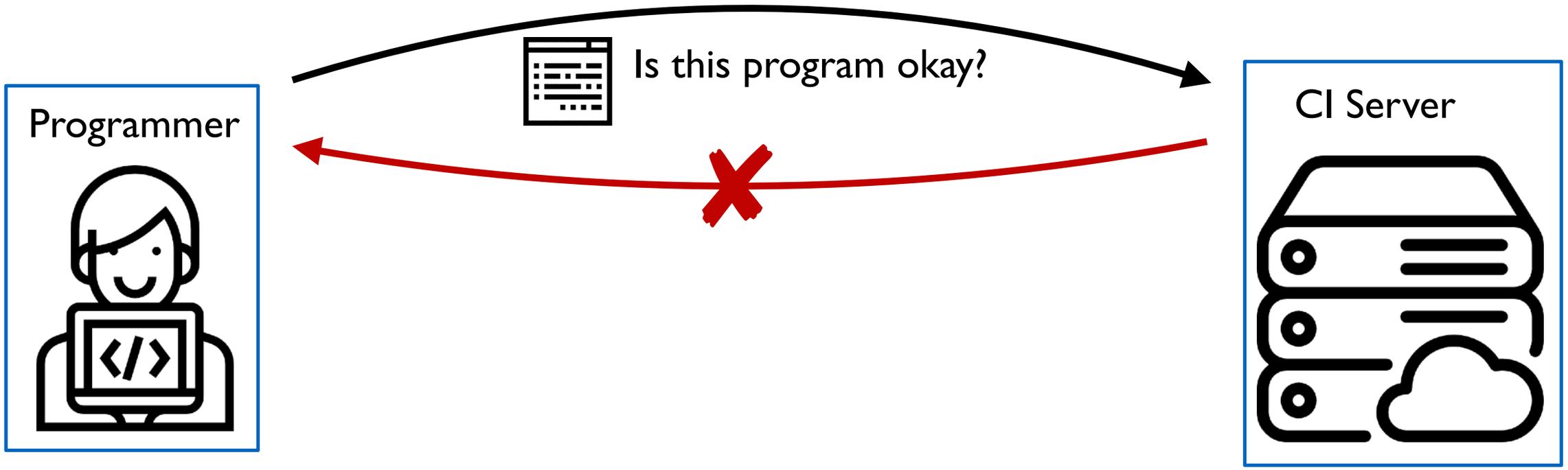


This is **slow** and **frustrating**: each iteration can be on the order of an hour or more.

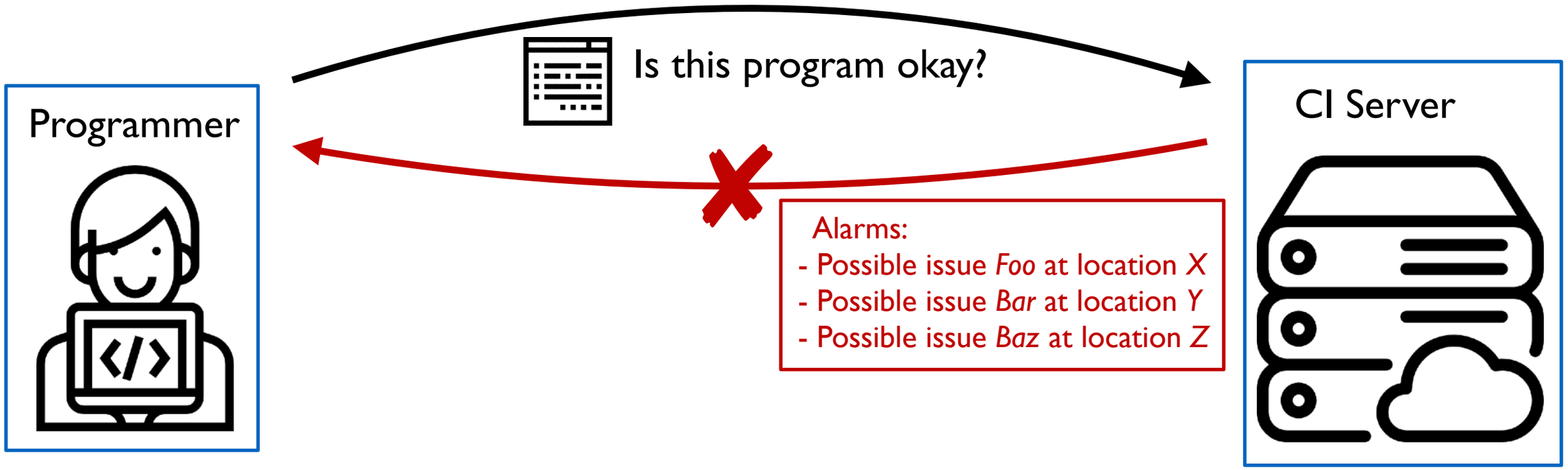
Interactive Static Analysis Workflow



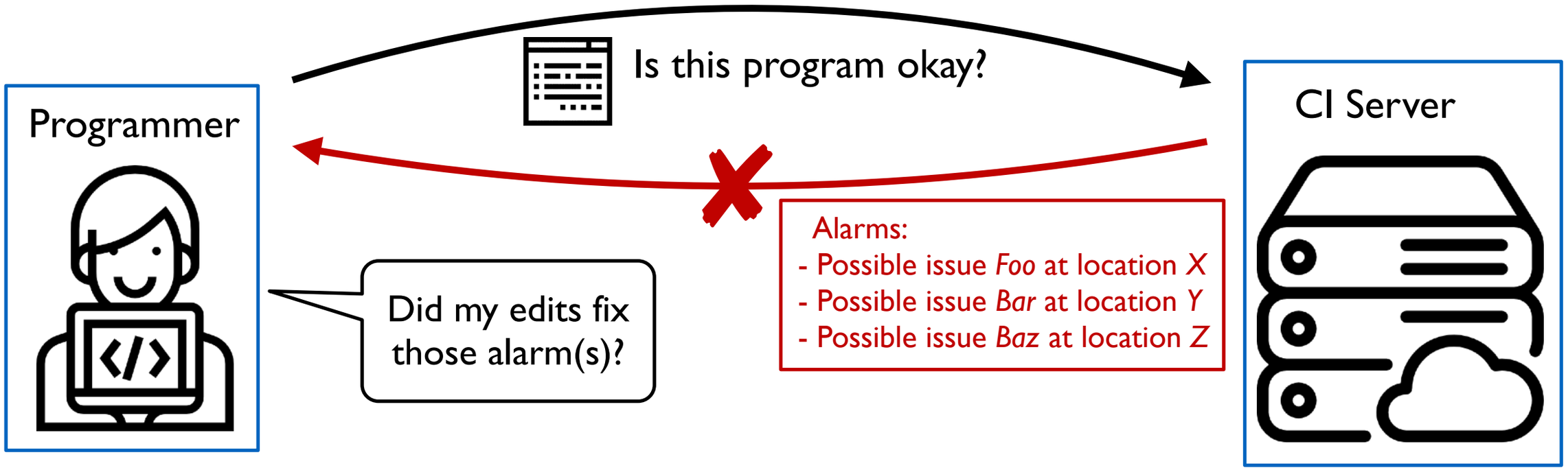
Interactive Static Analysis Workflow



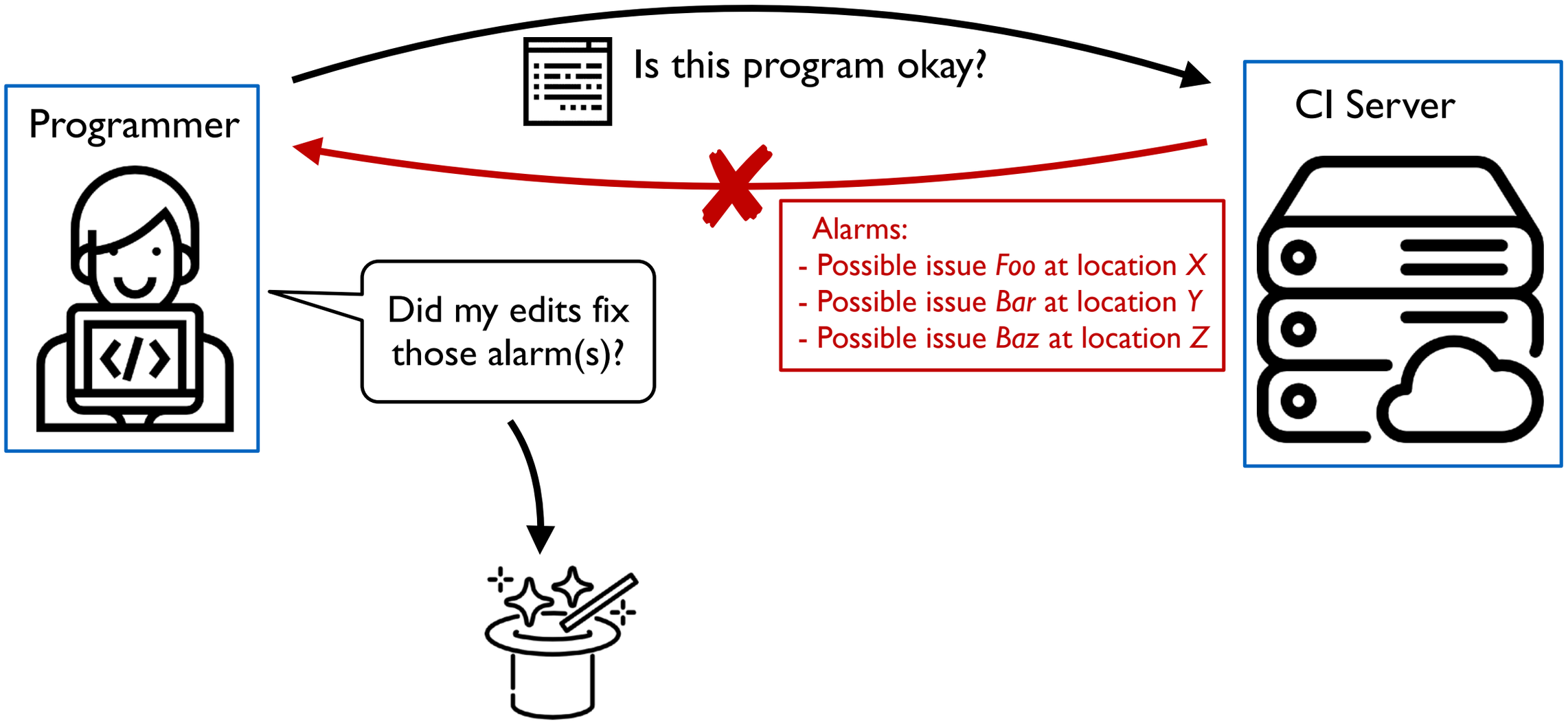
Interactive Static Analysis Workflow



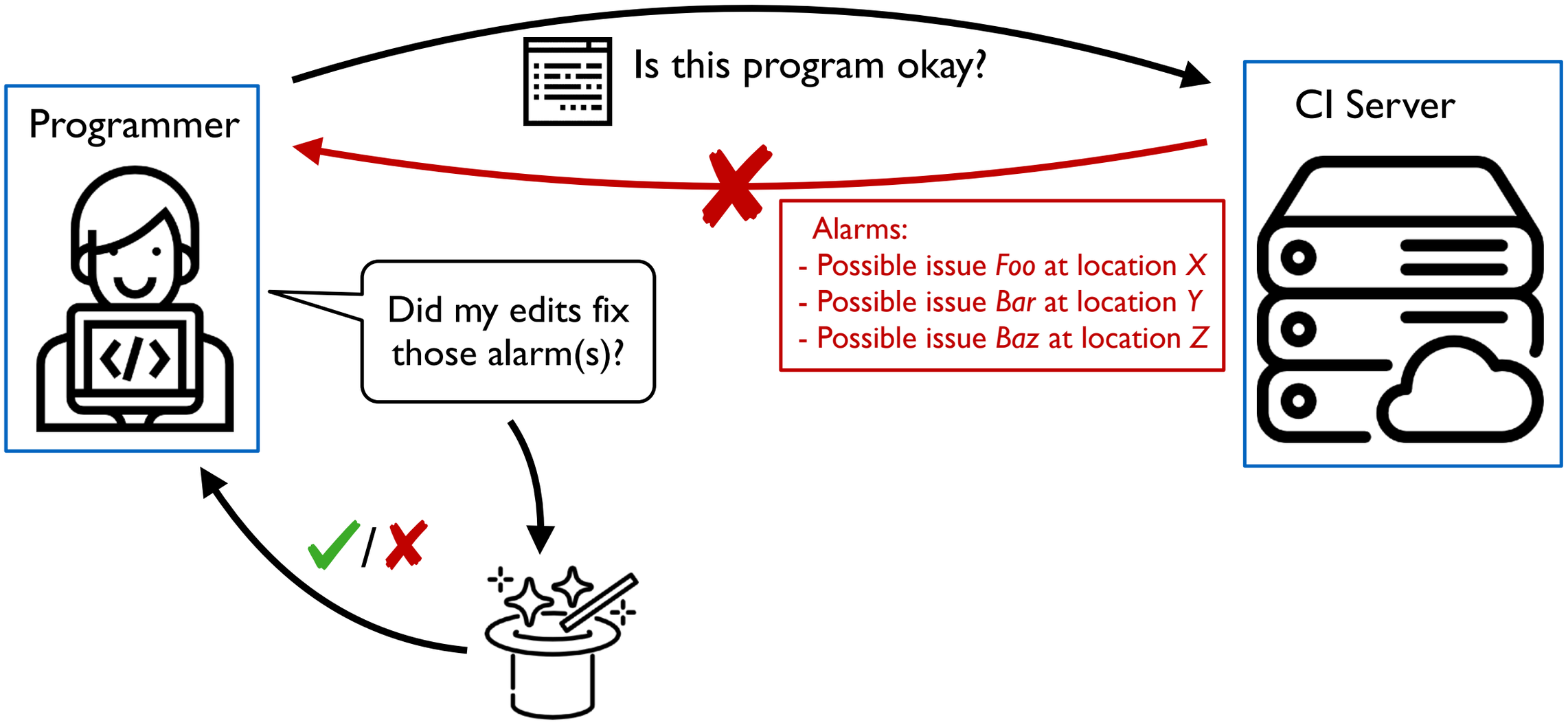
Interactive Static Analysis Workflow



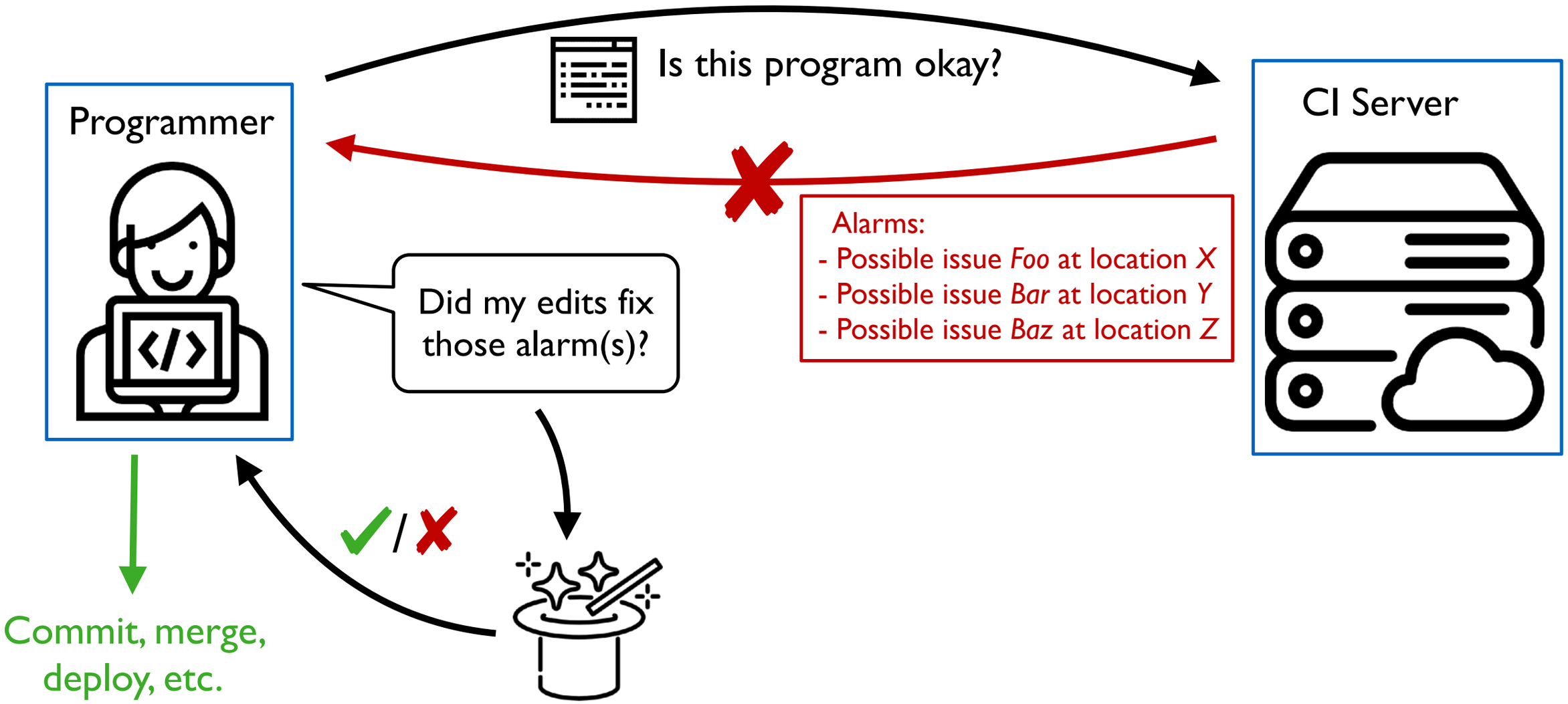
Interactive Static Analysis Workflow



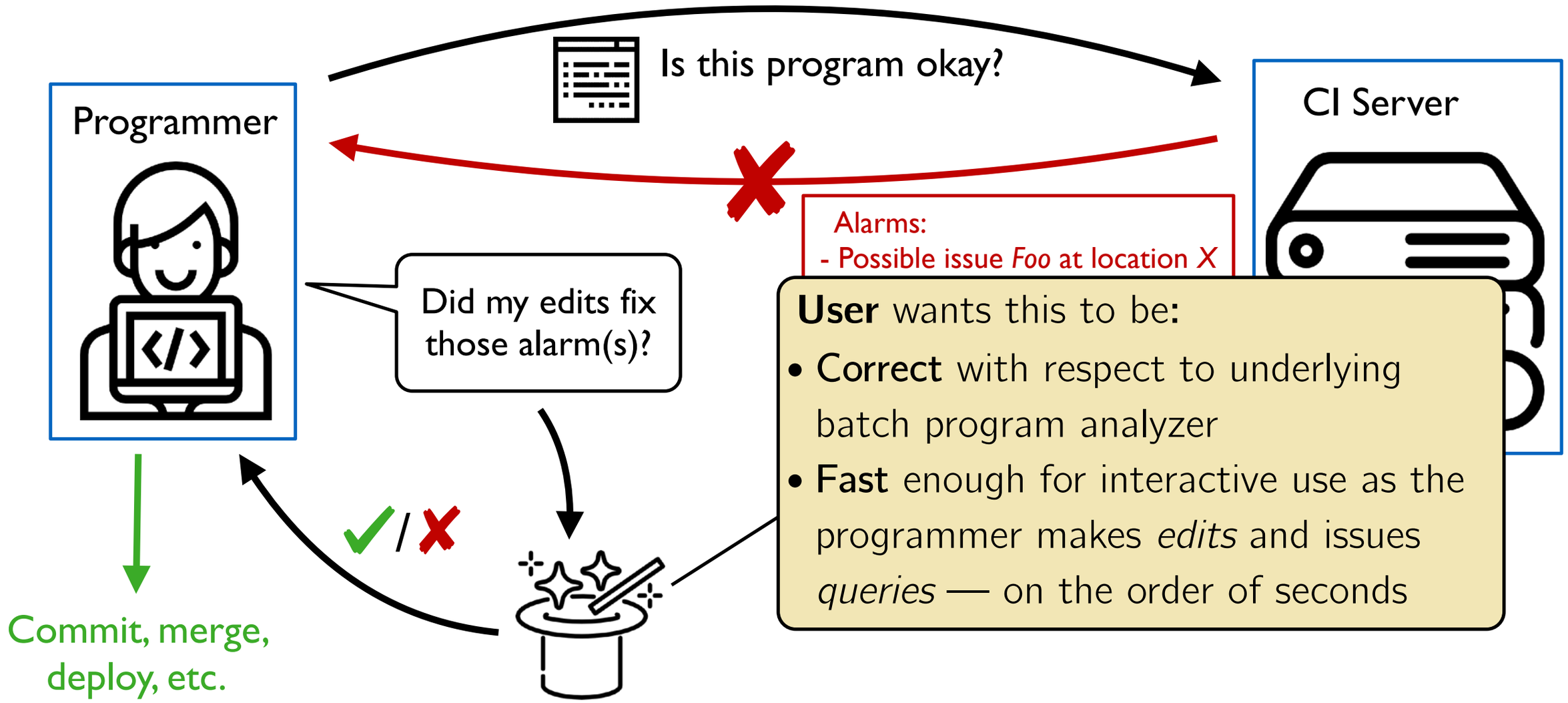
Interactive Static Analysis Workflow



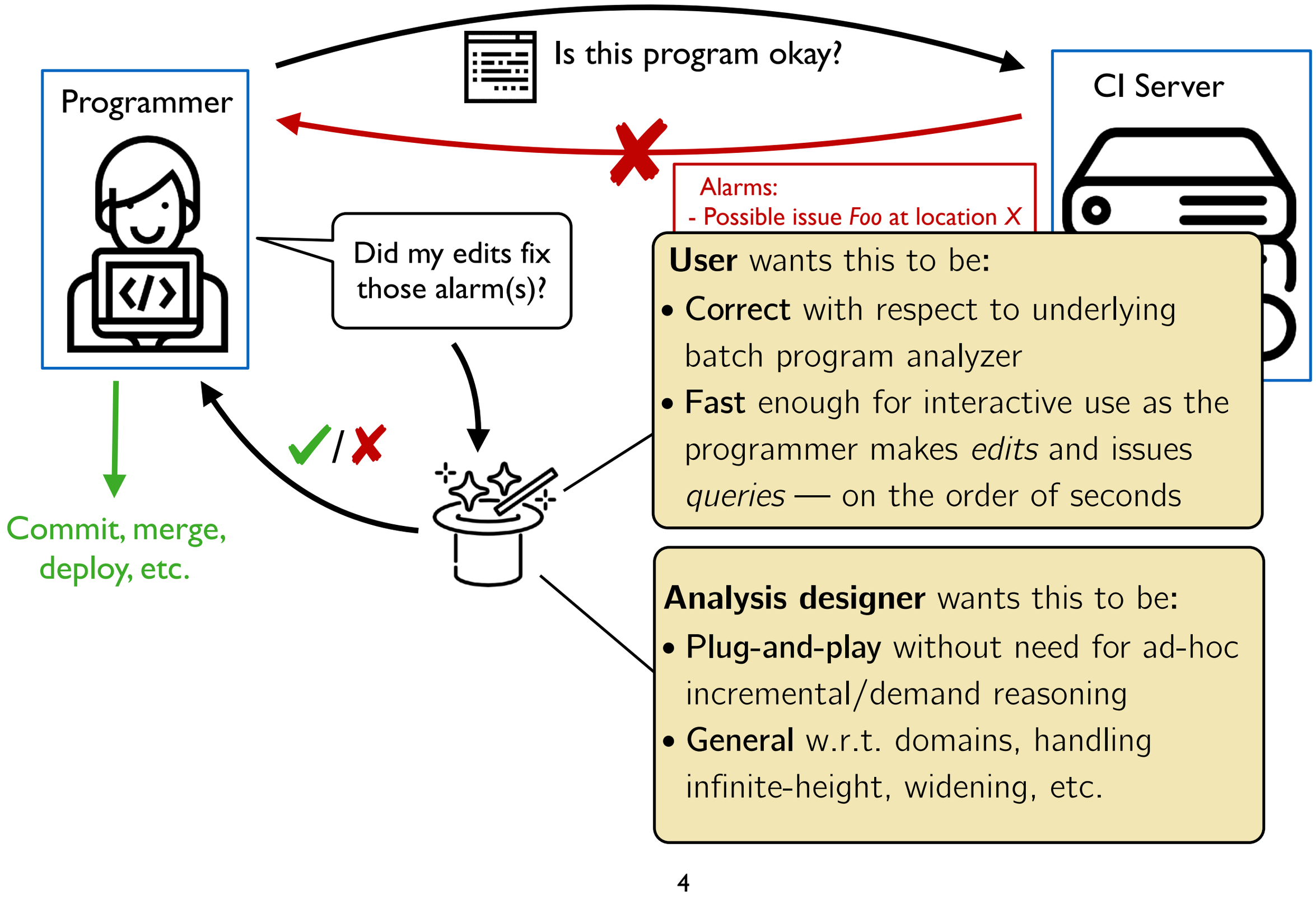
Interactive Static Analysis Workflow



Interactive Static Analysis Workflow



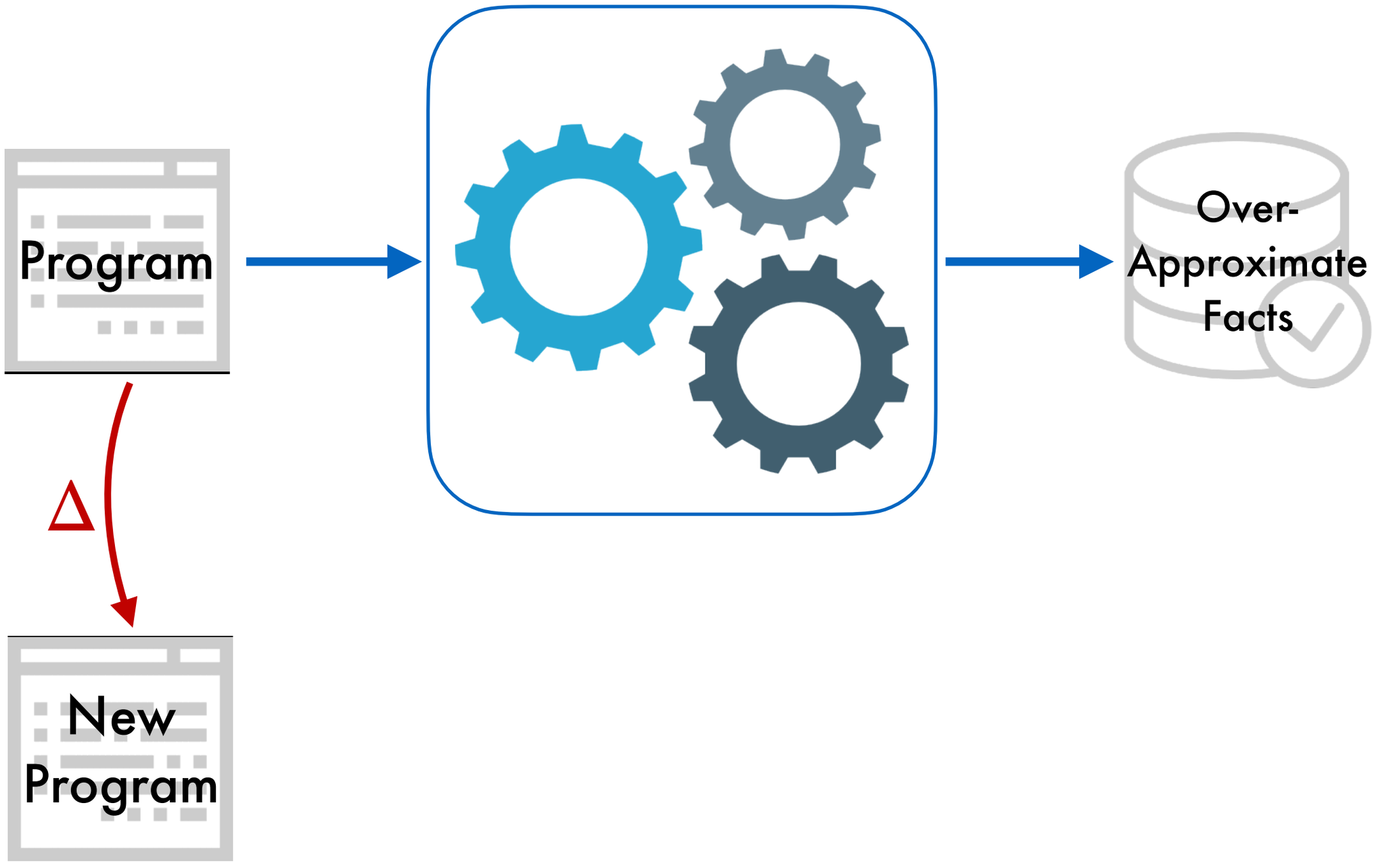
Interactive Static Analysis Workflow



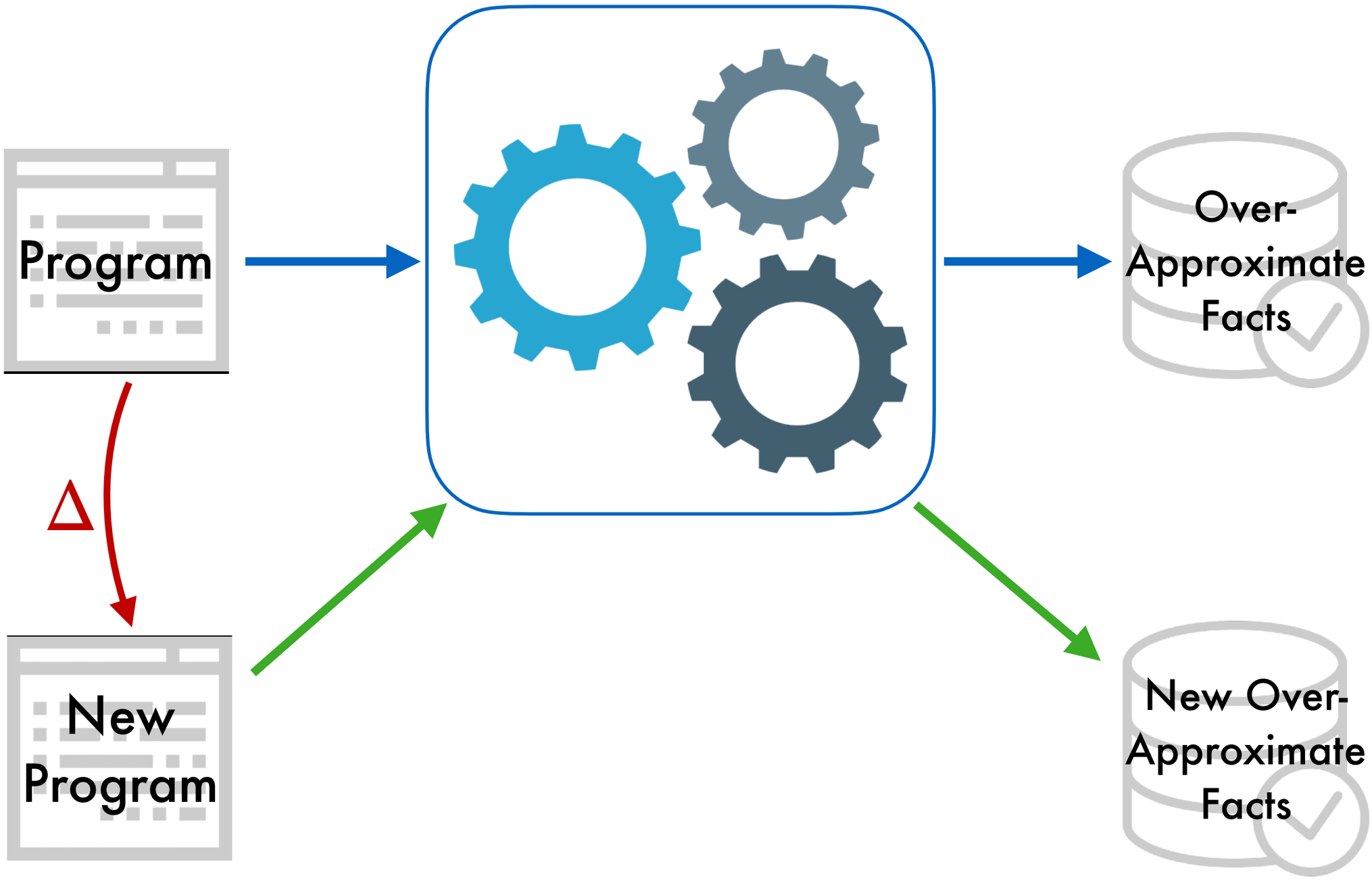
Batch Analysis

Incremental Analysis

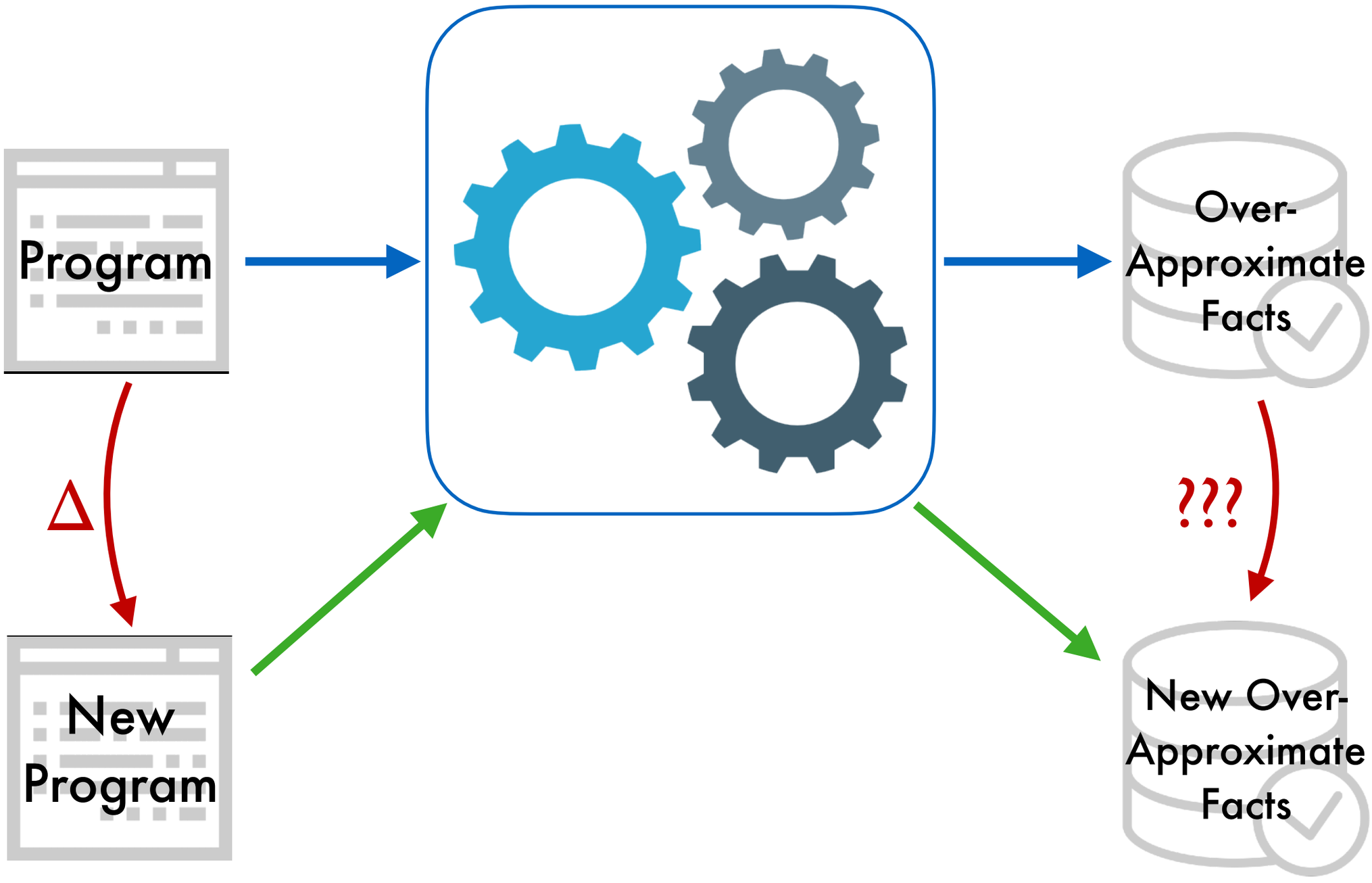
Incremental Analysis



Incremental Analysis

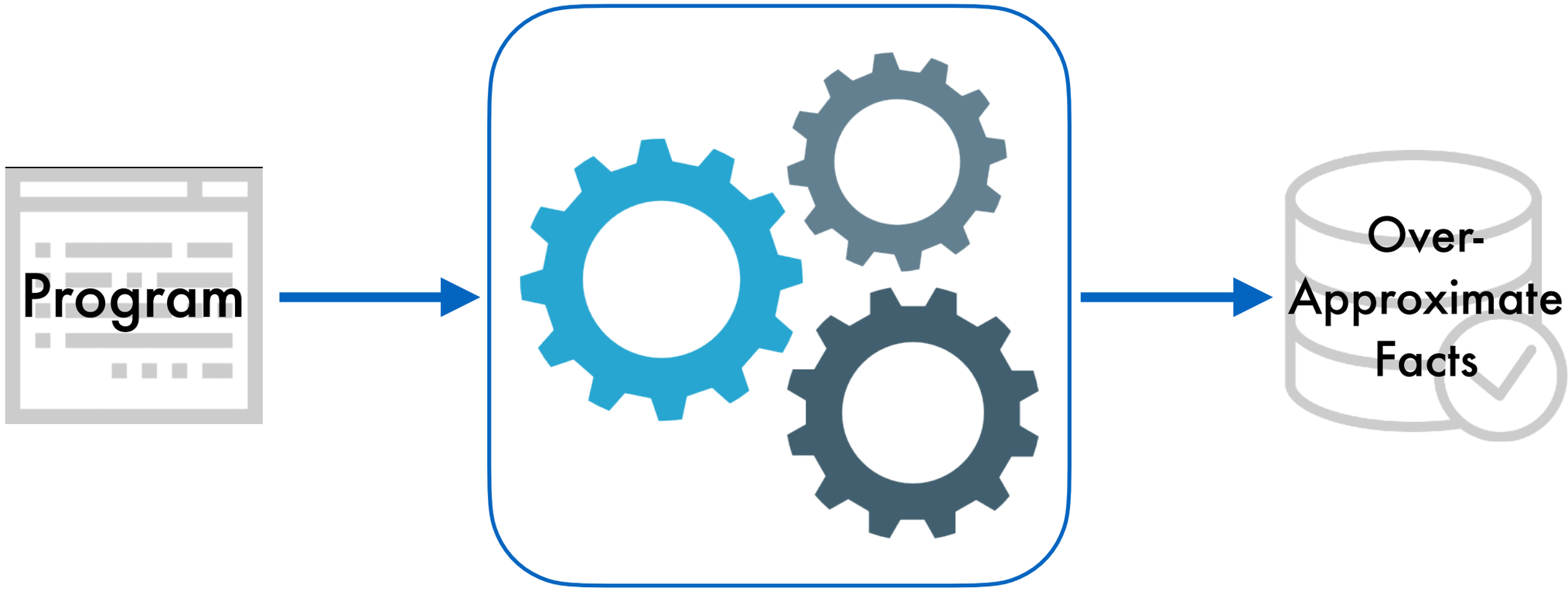


Incremental Analysis

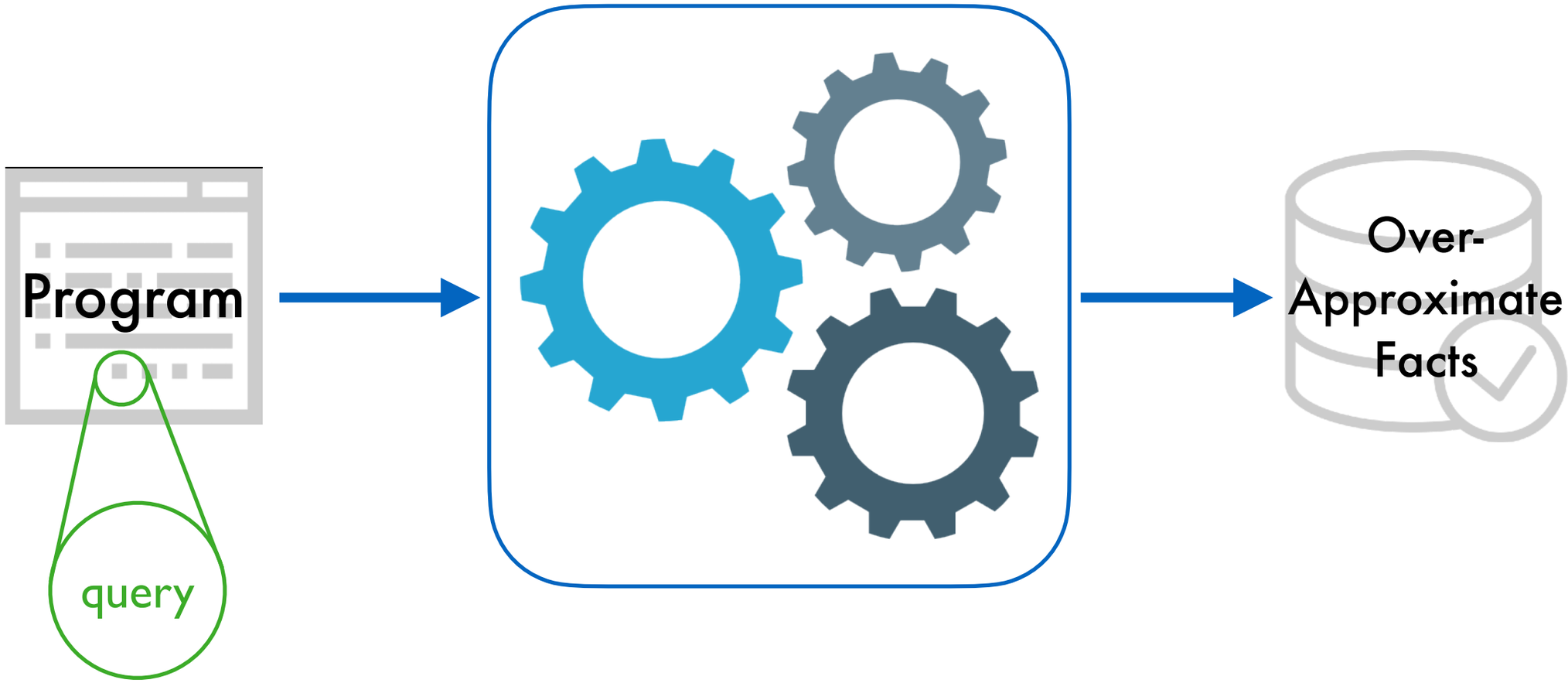


Incremental Analysis

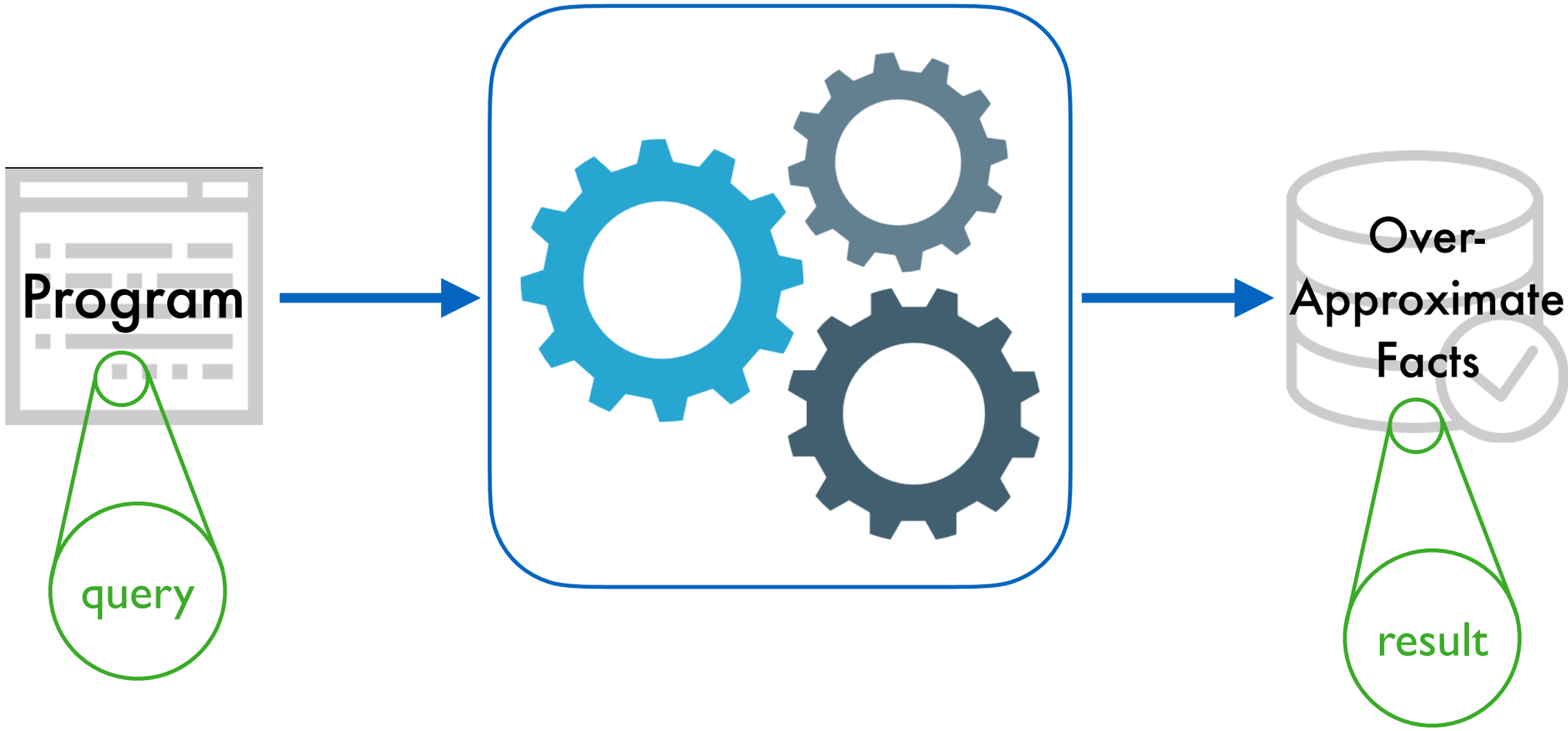
Demand-Driven Analysis



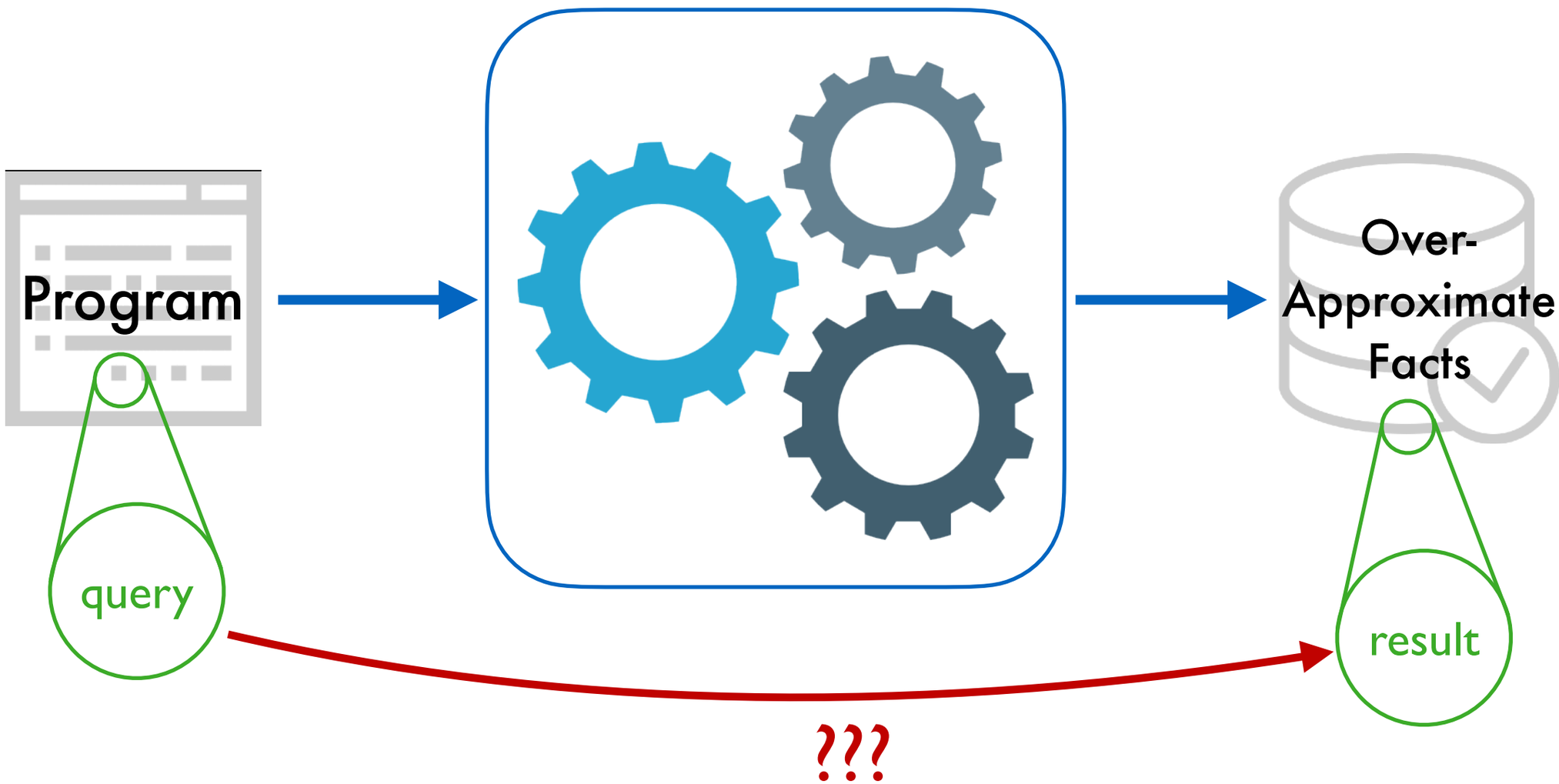
Demand-Driven Analysis



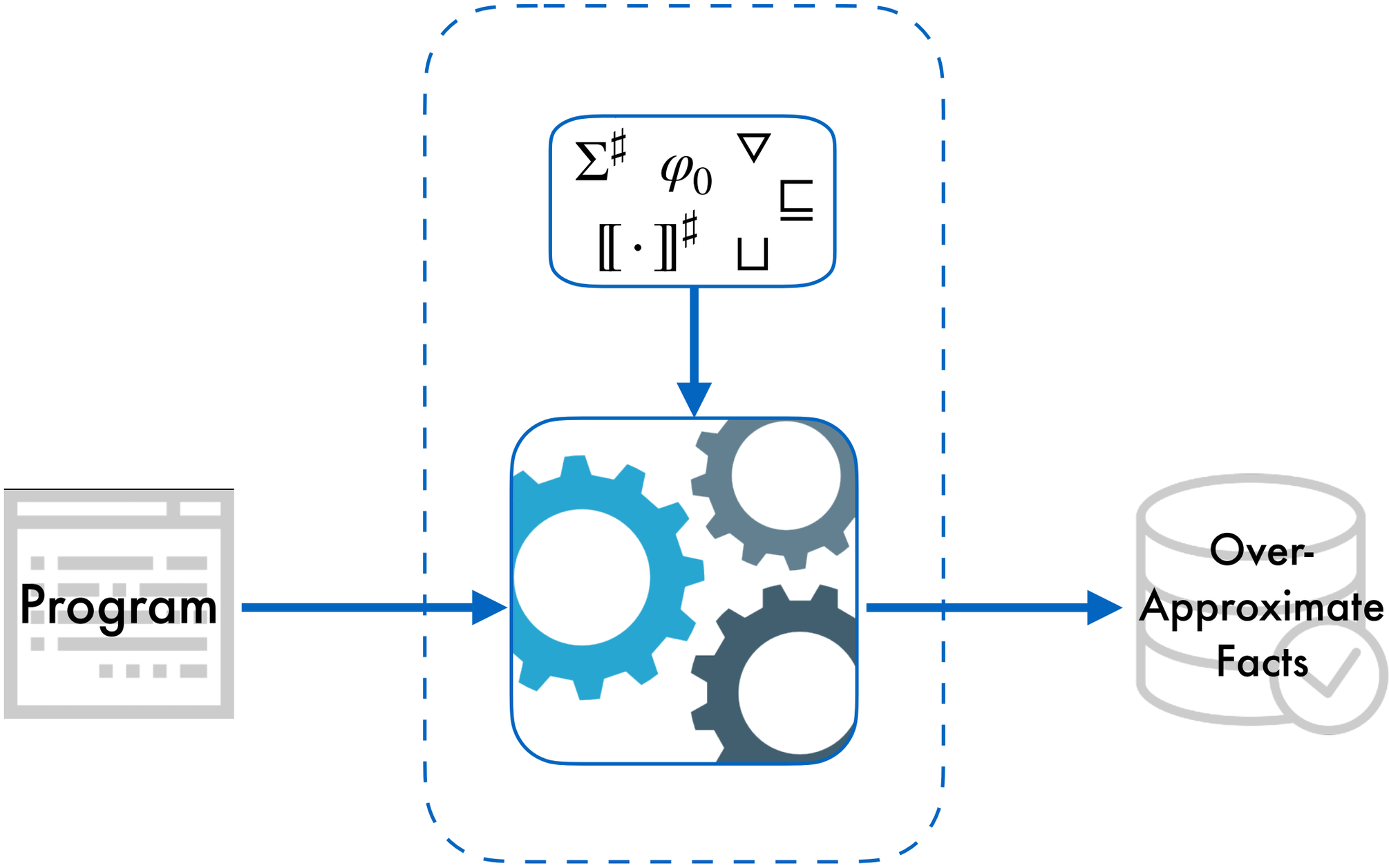
Demand-Driven Analysis



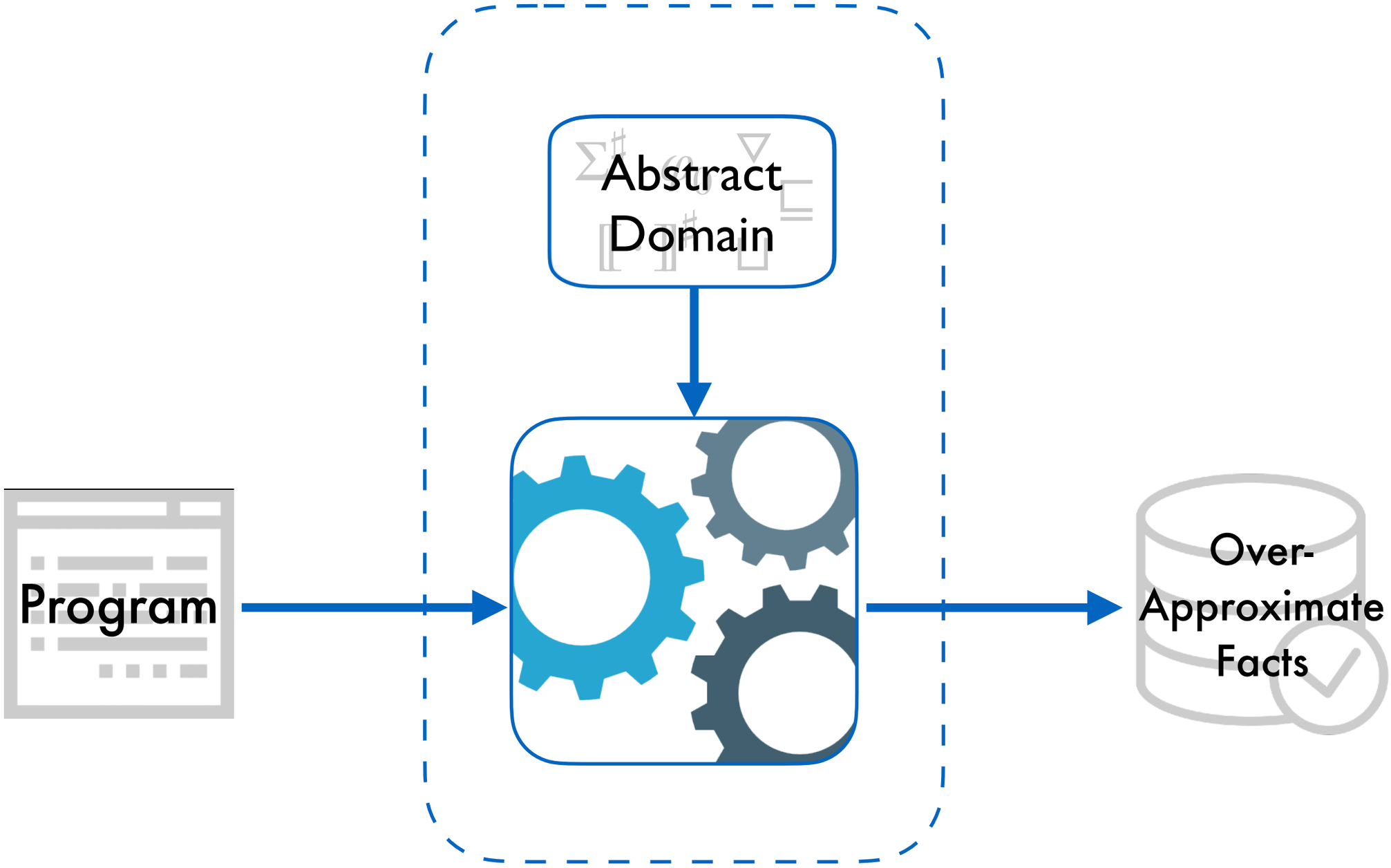
Demand-Driven Analysis



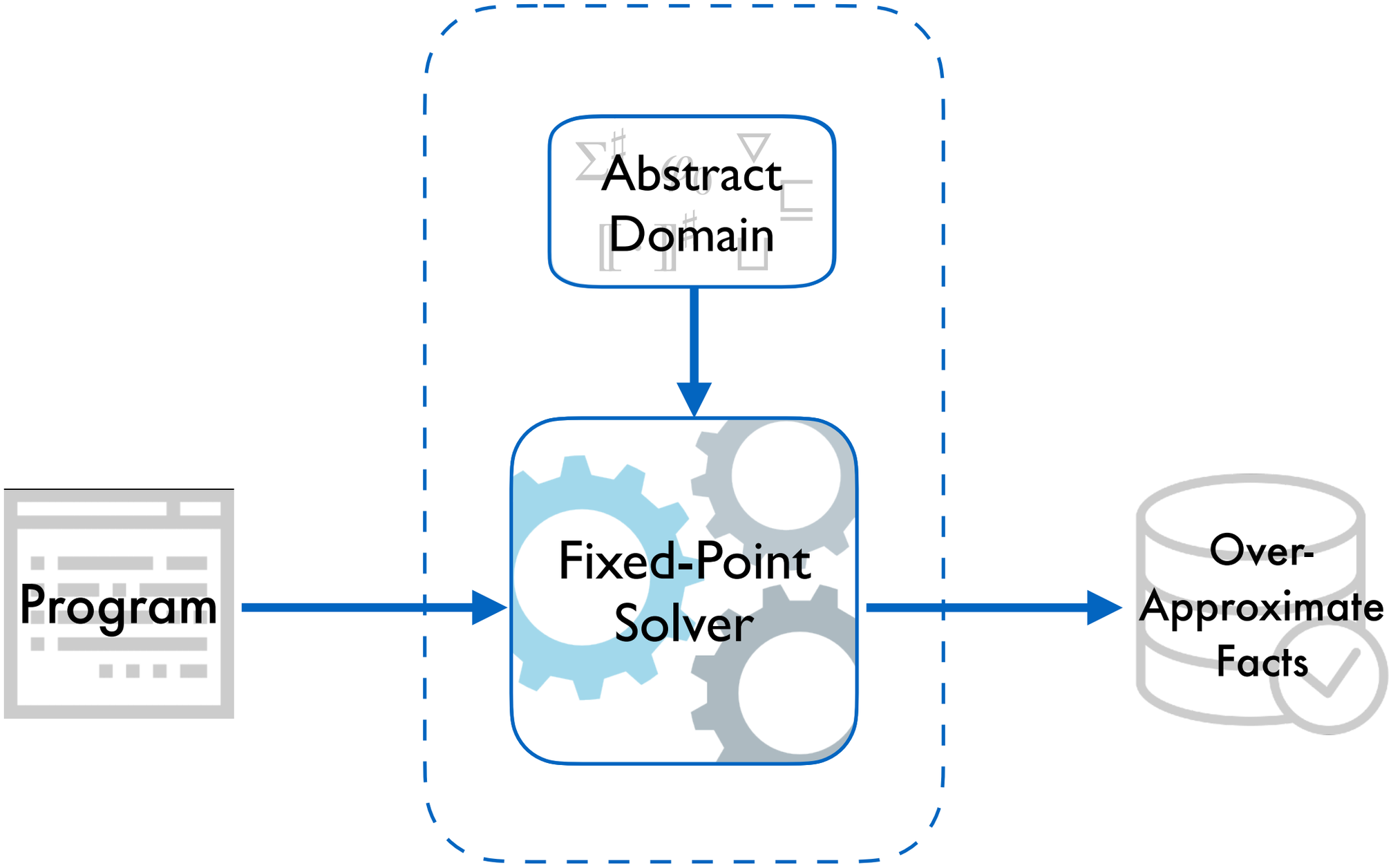
Abstract Interpretation



Abstract Interpretation



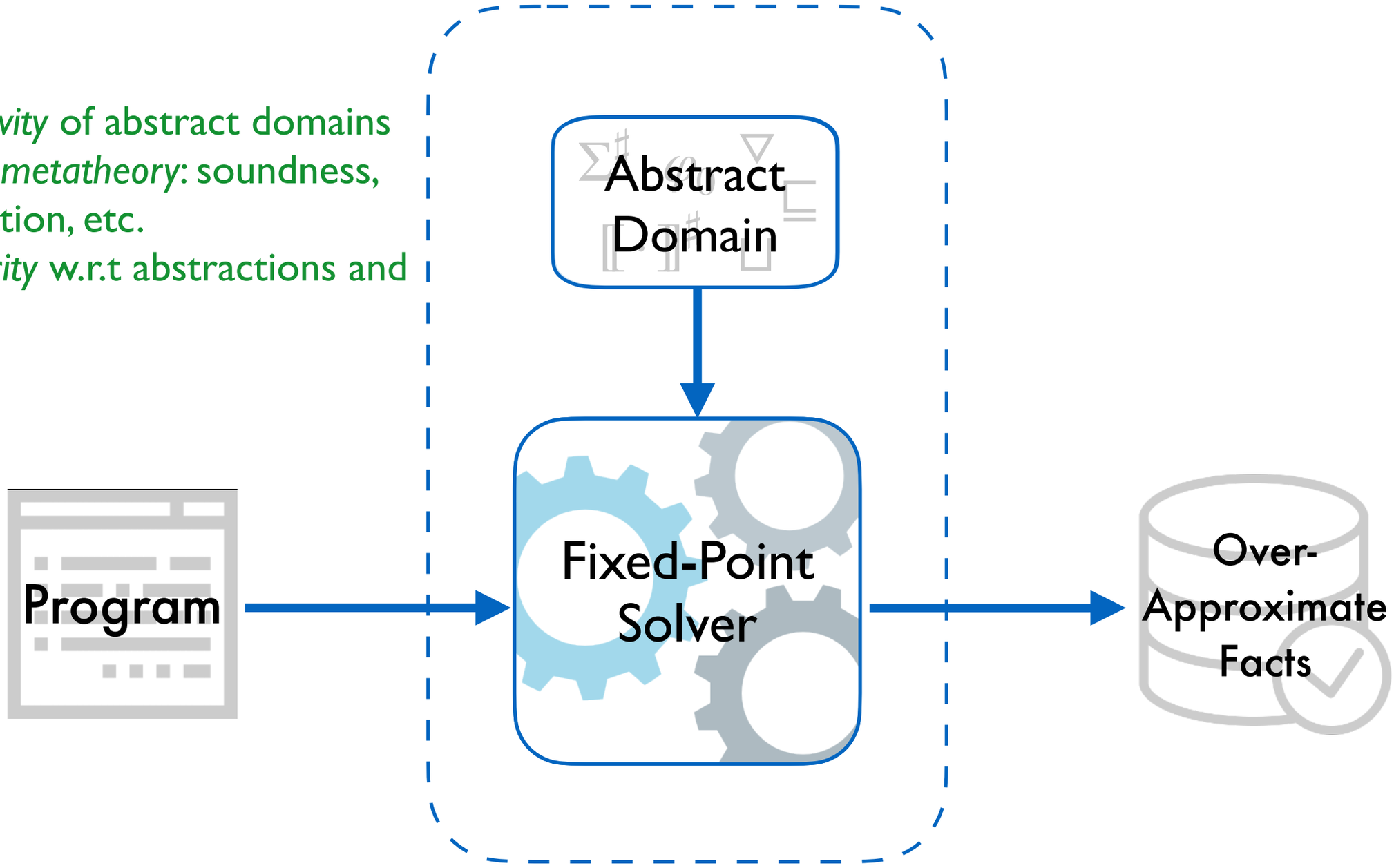
Abstract Interpretation



Abstract Interpretation

Pros:

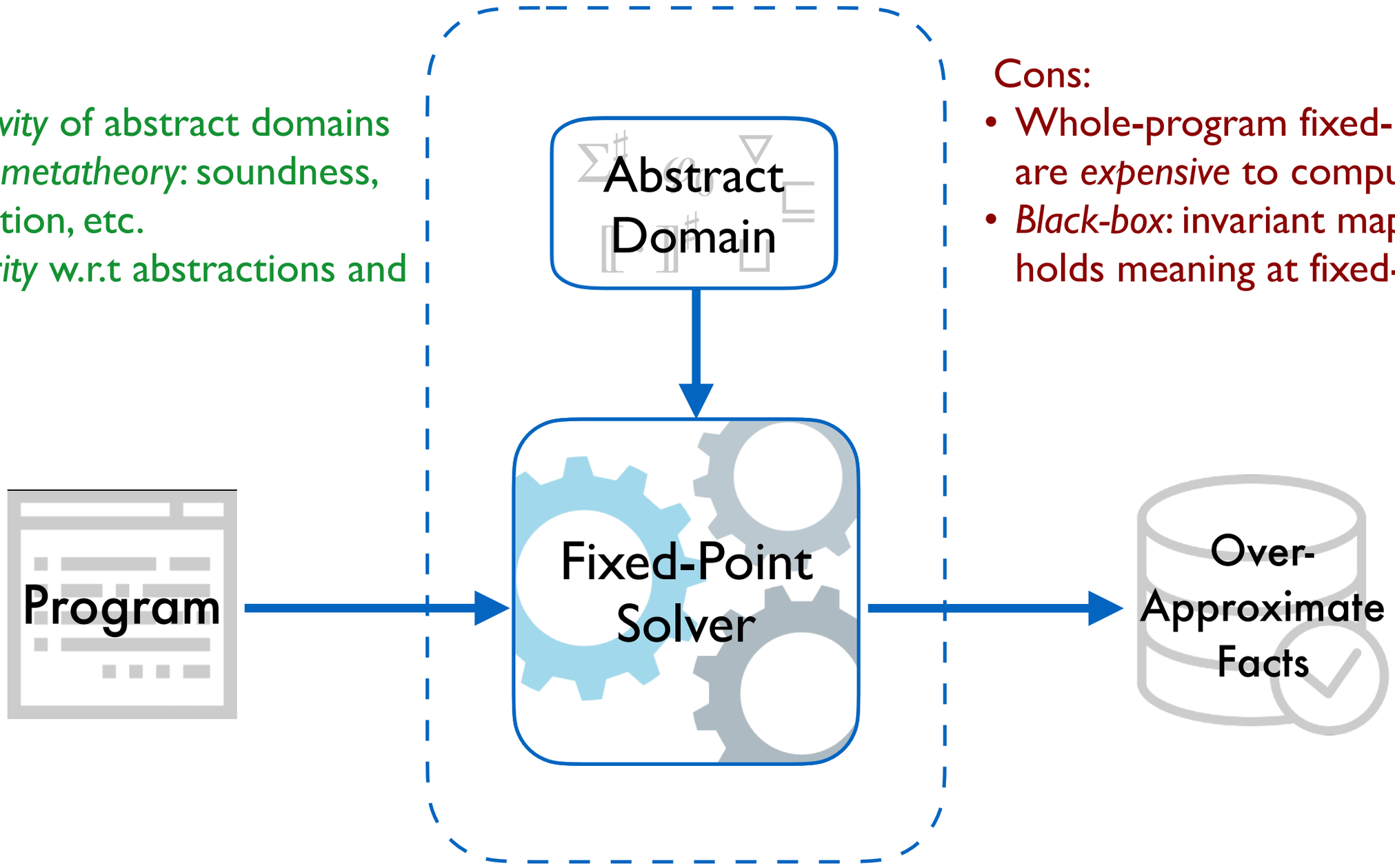
- *Expressivity* of abstract domains
- Robust *metatheory*: soundness, termination, etc.
- *Modularity* w.r.t abstractions and solvers



Abstract Interpretation

Pros:

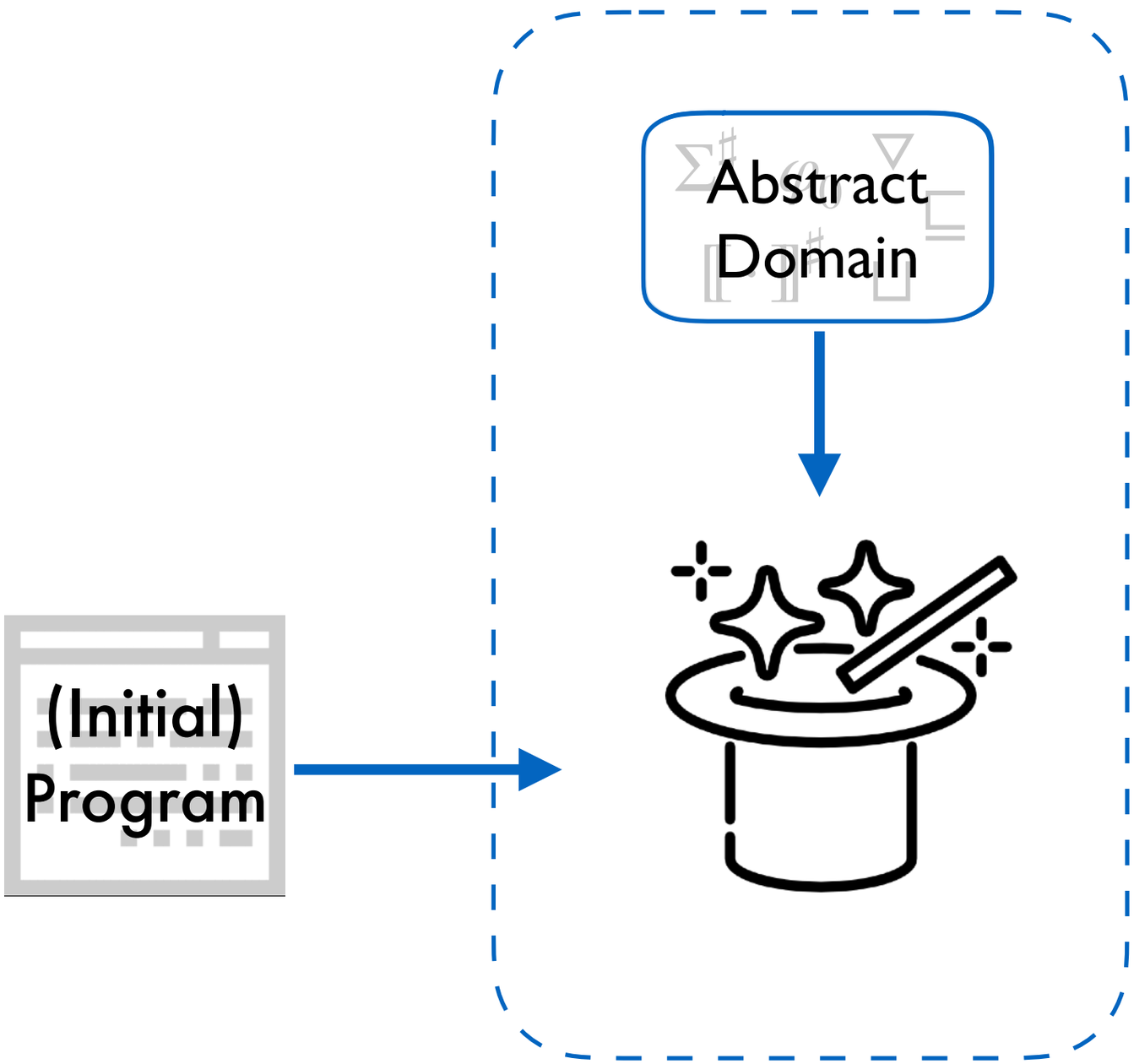
- *Expressivity* of abstract domains
- Robust *metatheory*: soundness, termination, etc.
- *Modularity* w.r.t abstractions and solvers



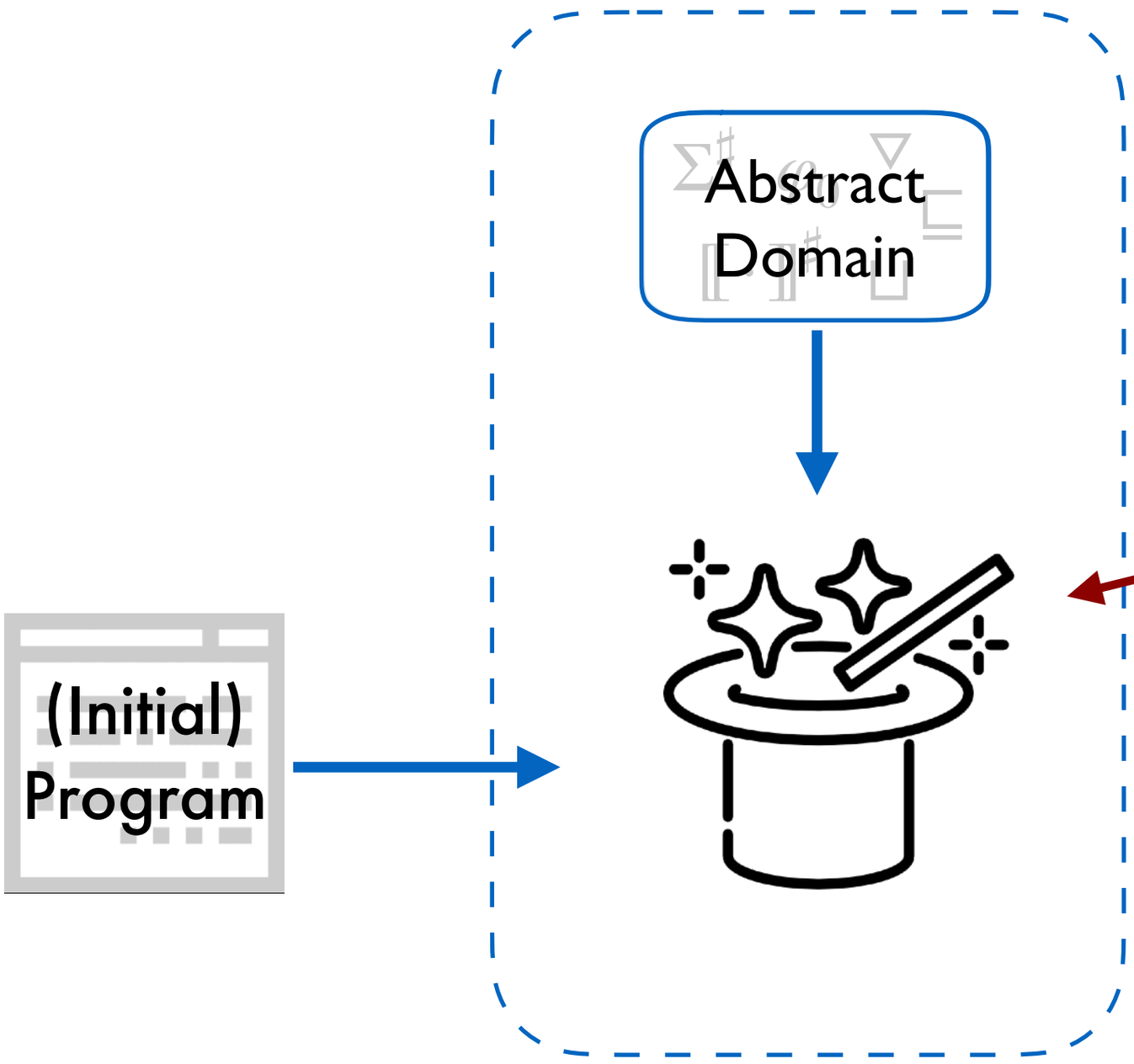
Cons:

- Whole-program fixed-points are *expensive* to compute
- *Black-box*: invariant map only holds meaning at fixed-point

Demanded Abstract Interpretation

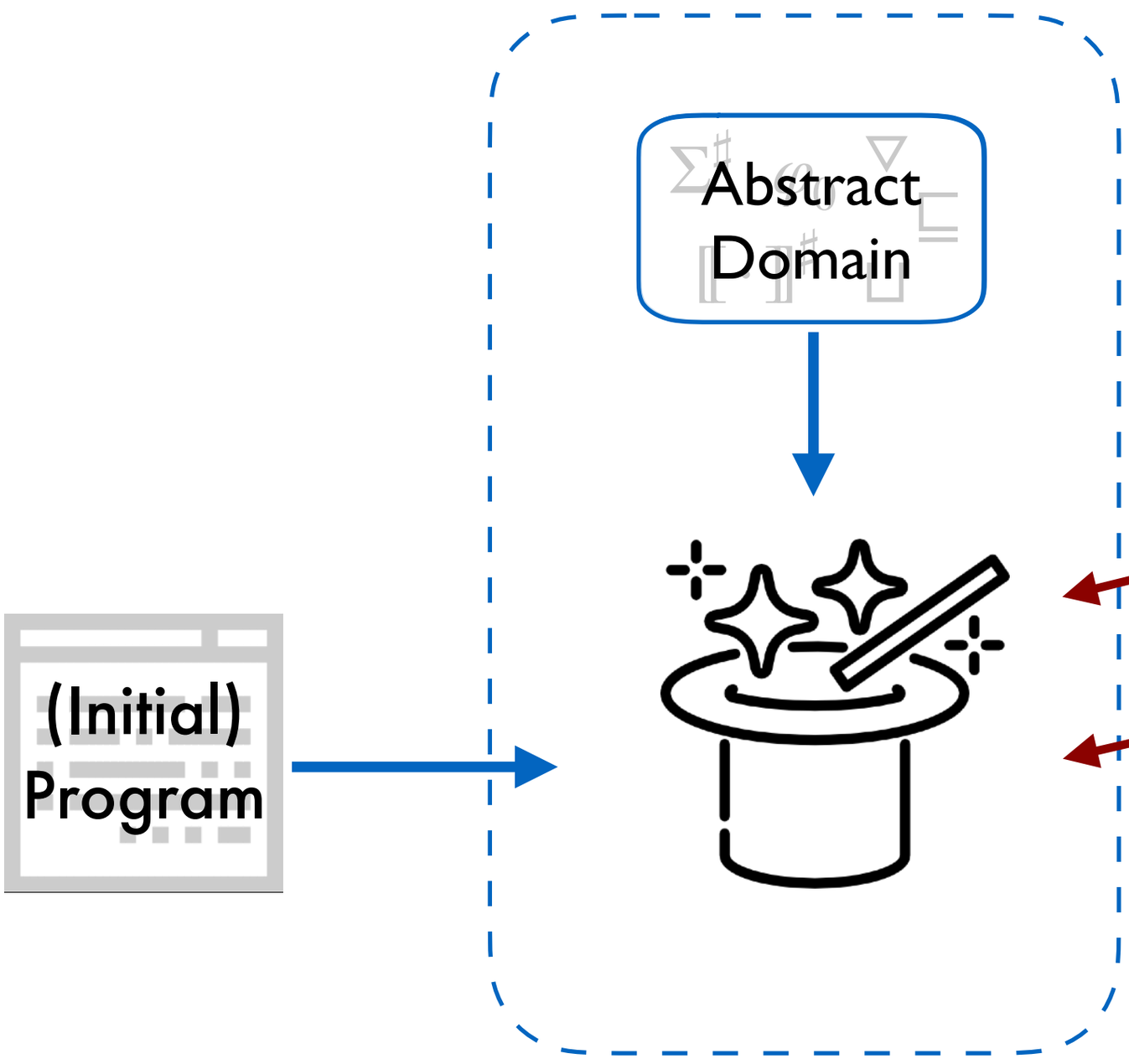


Demanded Abstract Interpretation



Demand-driven queries
"What is the abstract state at this program location?"

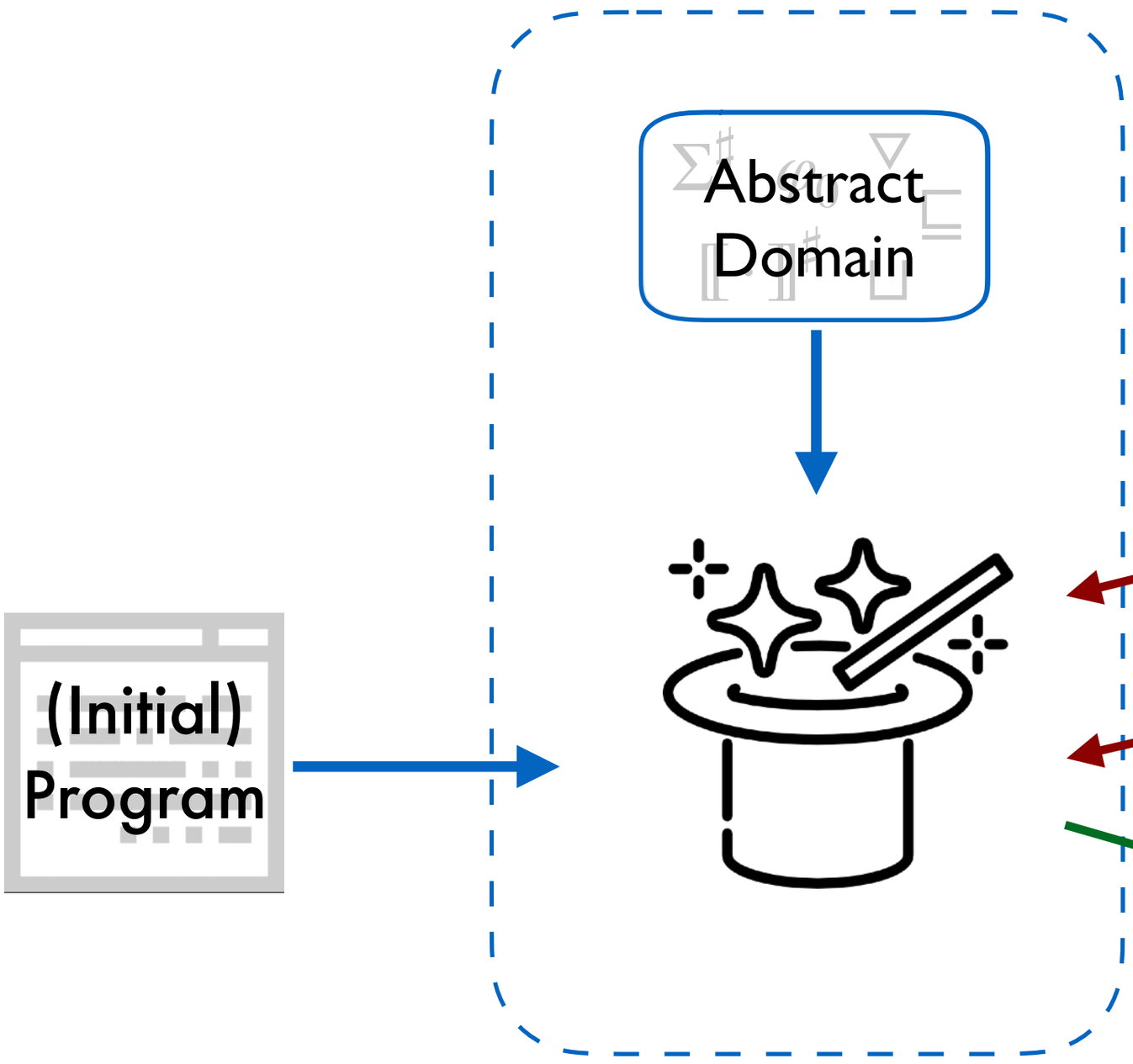
Demanded Abstract Interpretation



Demand-driven queries
"What is the abstract state at this program location?"

Incremental edits
Insert, modify, or delete statements

Demanded Abstract Interpretation

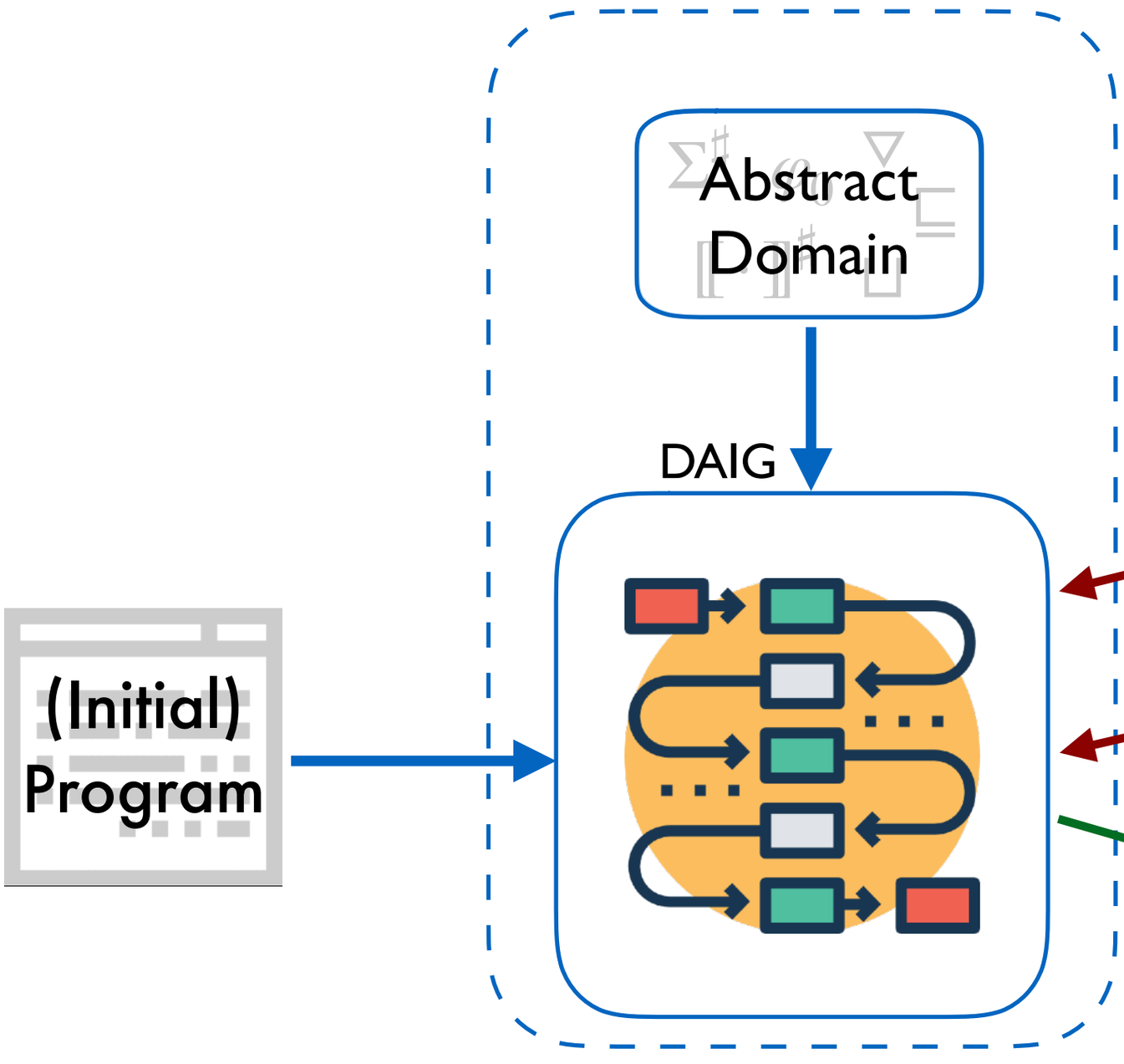


Demand-driven queries
"What is the abstract state at this program location?"

Incremental edits
Insert, modify, or delete statements

From-scratch consistent results
Query responses are guaranteed identical to batch analysis on current program version

Demanded Abstract Interpretation Graphs (DAIGs)



Demand-driven queries

"What is the abstract state at this program location?"

Incremental edits

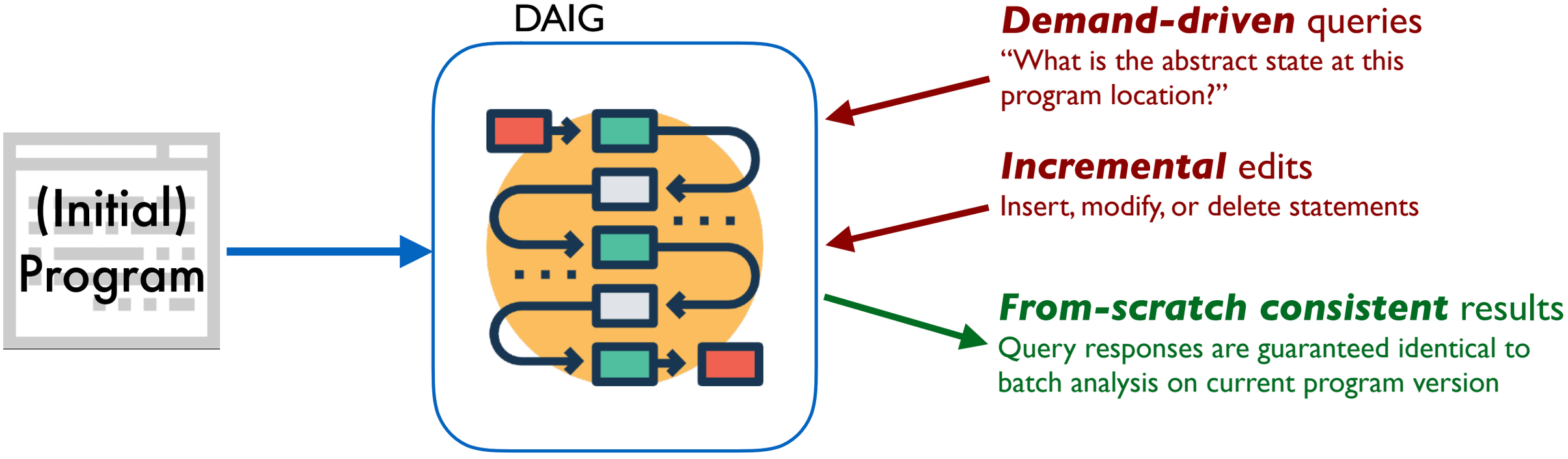
Insert, modify, or delete statements

From-scratch consistent results

Query responses are guaranteed identical to batch analysis on current program version

Demanded Abstract Interpretation Graphs (DAIGs)

A **DAIG** reifies the *dependency* structure of an abstract interpretation.



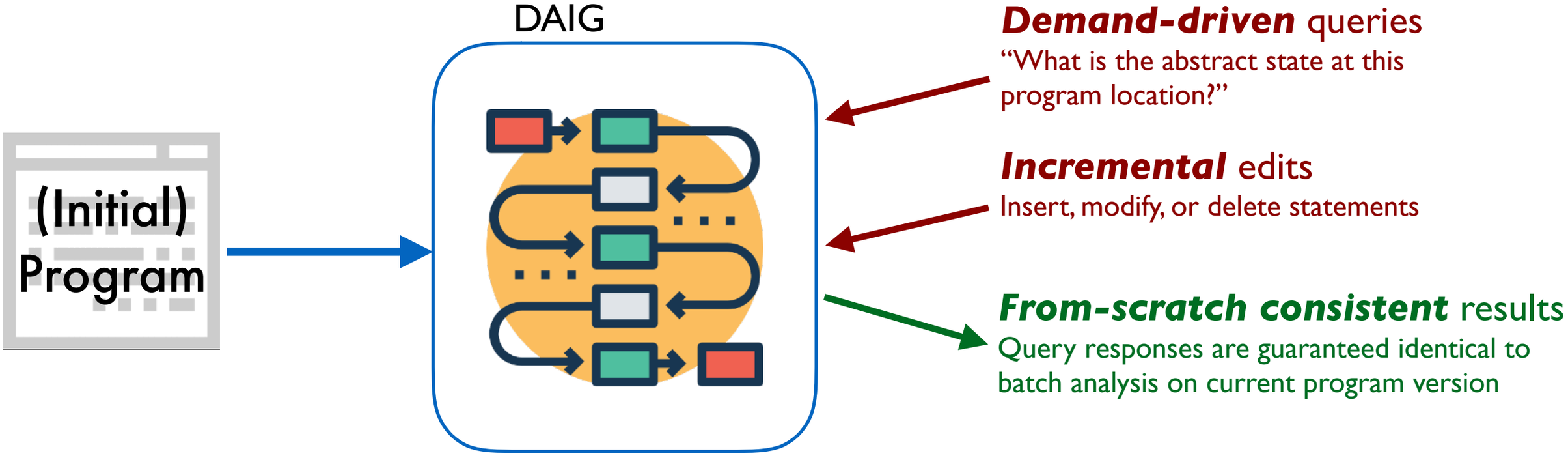
Demanded Abstract Interpretation Graphs (DAIGs)

A **DAIG** reifies the *dependency* structure of an abstract interpretation.

Reference cell vertices...

... contain intermediate analysis results & program syntax

... are uniquely named and potentially empty



Demanded Abstract Interpretation Graphs (DAIGs)

A **DAIG** reifies the *dependency* structure of an abstract interpretation.

Reference cell vertices...

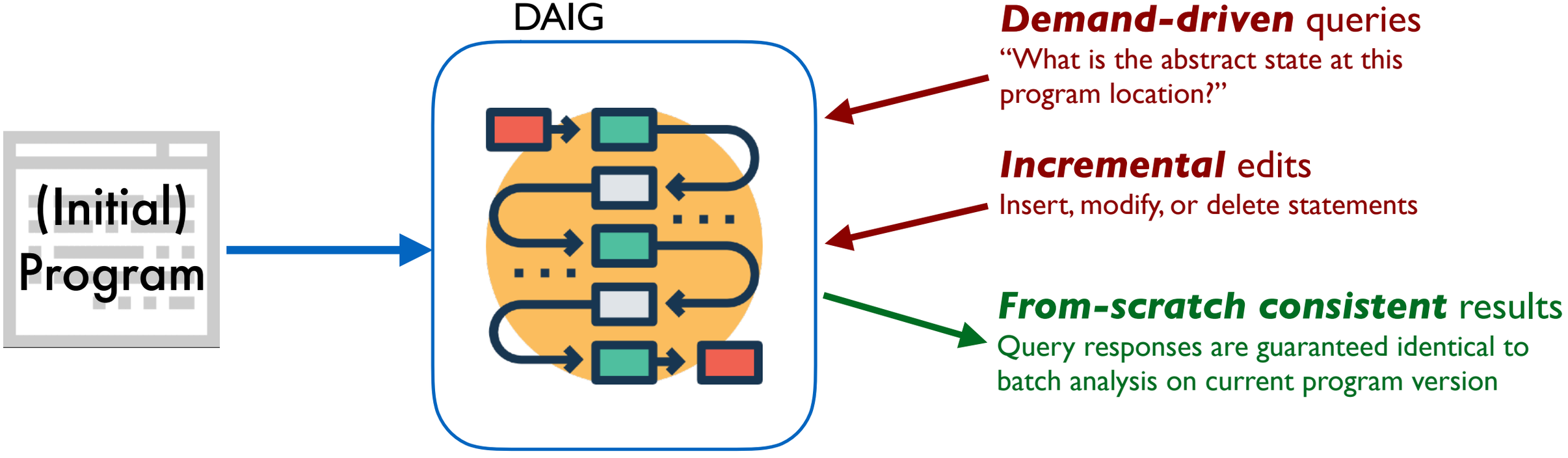
... contain intermediate analysis results & program syntax

... are uniquely named and potentially empty

Computation edges...

... acyclically connect reference cells

... denote analysis computations (e.g. $[\![\cdot]\!]^\sharp, \sqcup, \nabla$)



Contributions

A **DAIG** reifies the *dependency* structure of an abstract interpretation.

Reference cell vertices...

... contain program syntax & intermediate analysis results

... are uniquely *named* and potentially empty

Computation edges...

... acyclically connect reference cells

... are labelled by ar

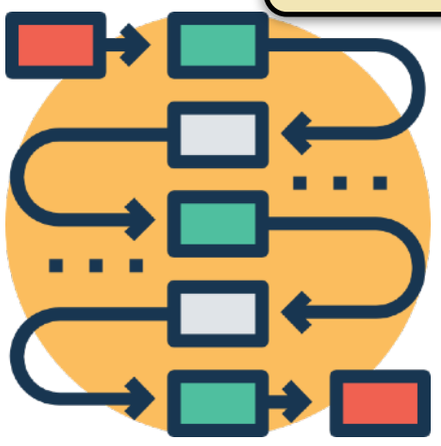
Finite & **acyclic** dependency graph, supporting incremental **and** demand-driven evaluation

ven queries

ct state at this

(Initial) Program

DAIG



Incremental edits

Insert, modify, or delete statements

From-scratch consistent results

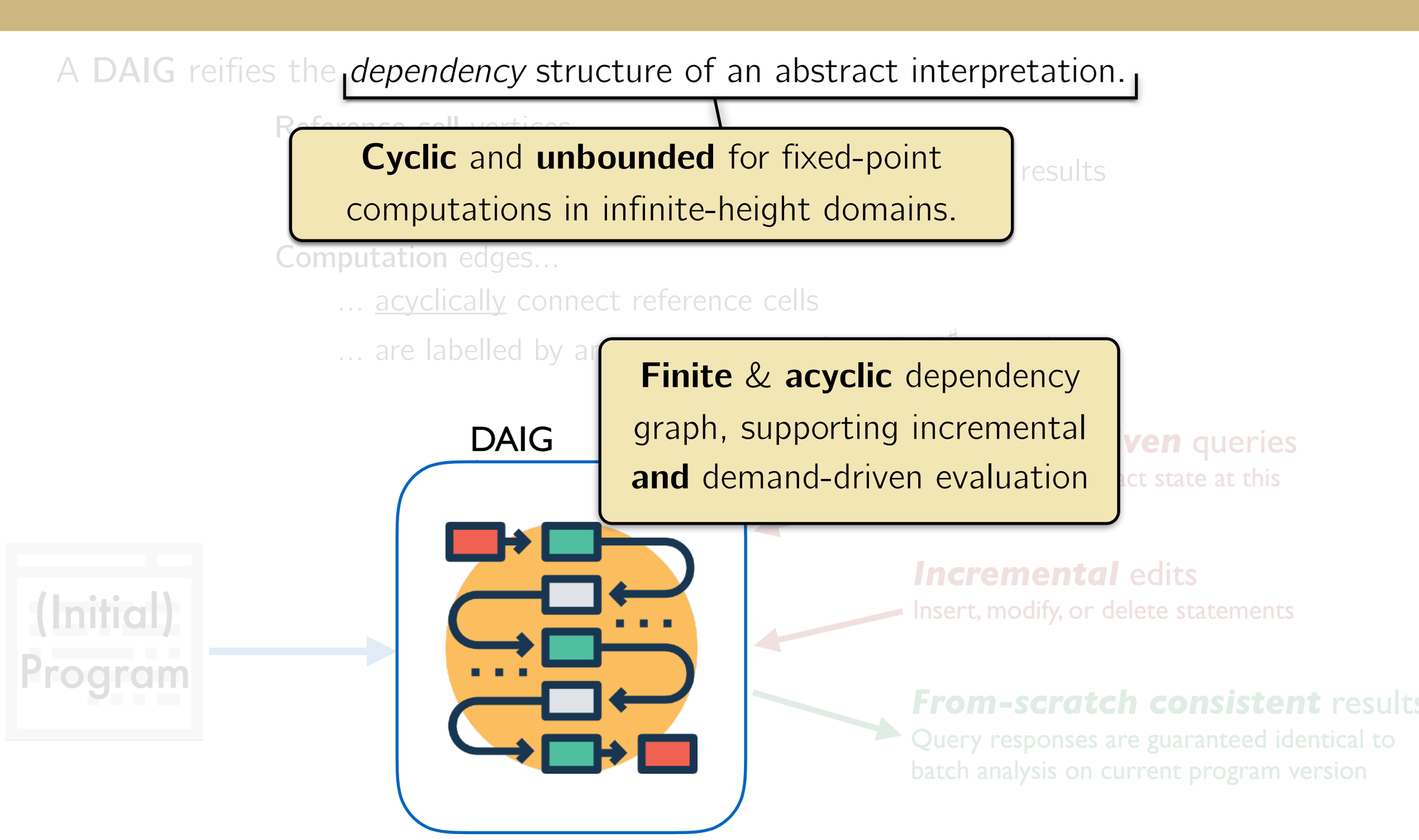
Query responses are guaranteed identical to batch analysis on current program version

Contributions

A **DAIG** reifies the *dependency* structure of an abstract interpretation.

Cyclic and **unbounded** for fixed-point computations in infinite-height domains.

Finite & **acyclic** dependency graph, supporting incremental **and** demand-driven evaluation



Contributions

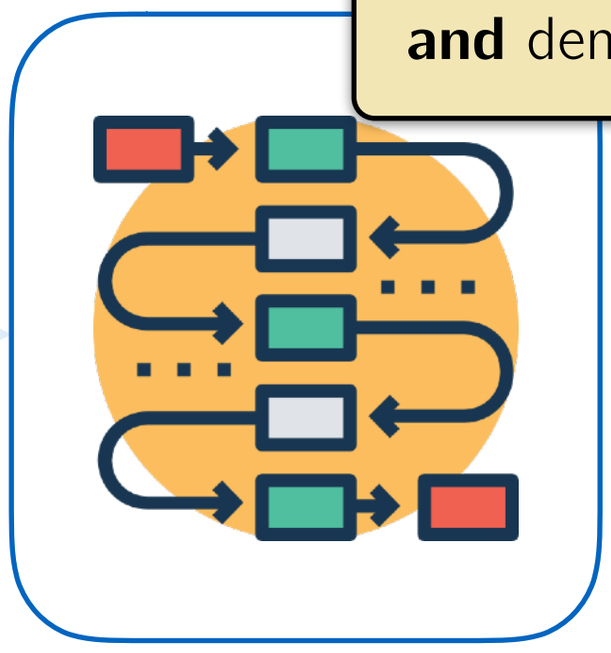
A **DAIG** reifies the *dependency* structure of an abstract interpretation.

Cyclic and **unbounded** for fixed-point computations in infinite-height domains.

via *demanded unrolling* of fixed-point computations

Finite & **acyclic** dependency graph, supporting incremental **and** demand-driven evaluation

DAIG



(Initial) Program

Incremental edits
Insert, modify, or delete statements

From-scratch consistent results
Query responses are guaranteed identical to batch analysis on current program version

Contributions

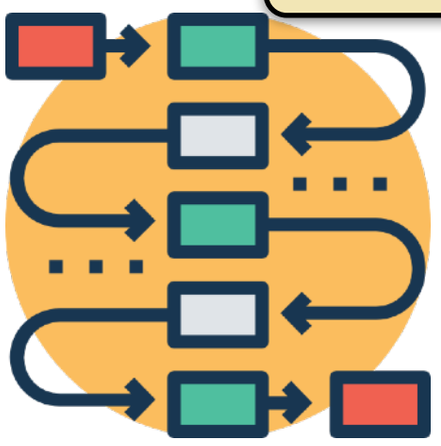
dependency structure of an abstract interpretation.

Cyclic and **unbounded** for fixed-point computations in infinite-height domains.

via *demanded unrolling* of fixed-point computations

Finite & **acyclic** dependency graph, supporting incremental **and** demand-driven evaluation

DAIG

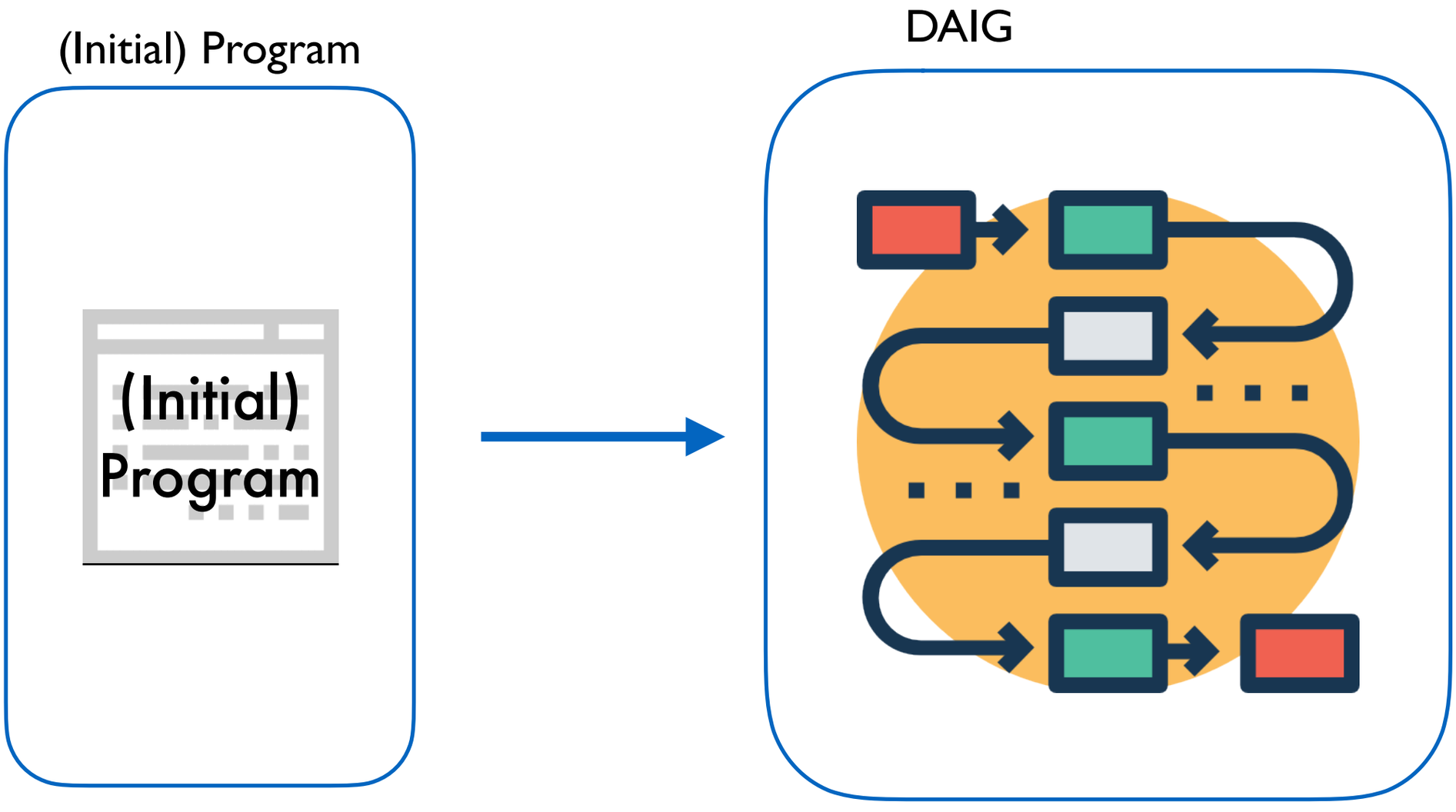


(Initial) Program

Incremental edits
Insert, modify, or delete statements

From-scratch consistent results
Query responses are guaranteed identical to batch analysis on current program version

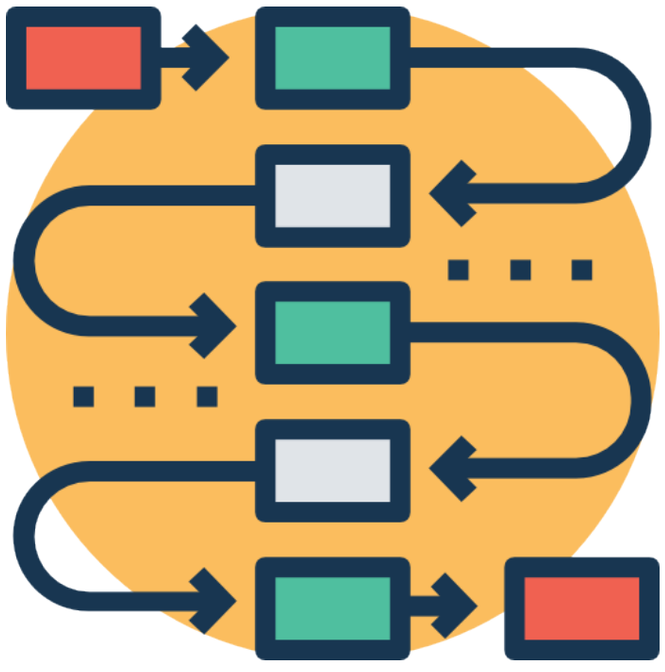
Demanded Abstract Interpretation Graphs (DAIGs)



Demanded Abstract Interpretation Graphs (DAIGs)

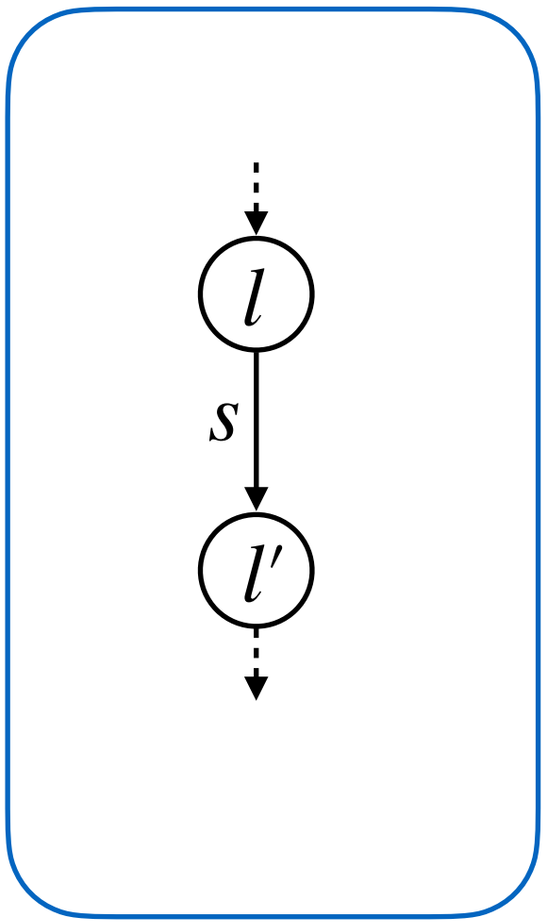
(Initial) Program

DAIG



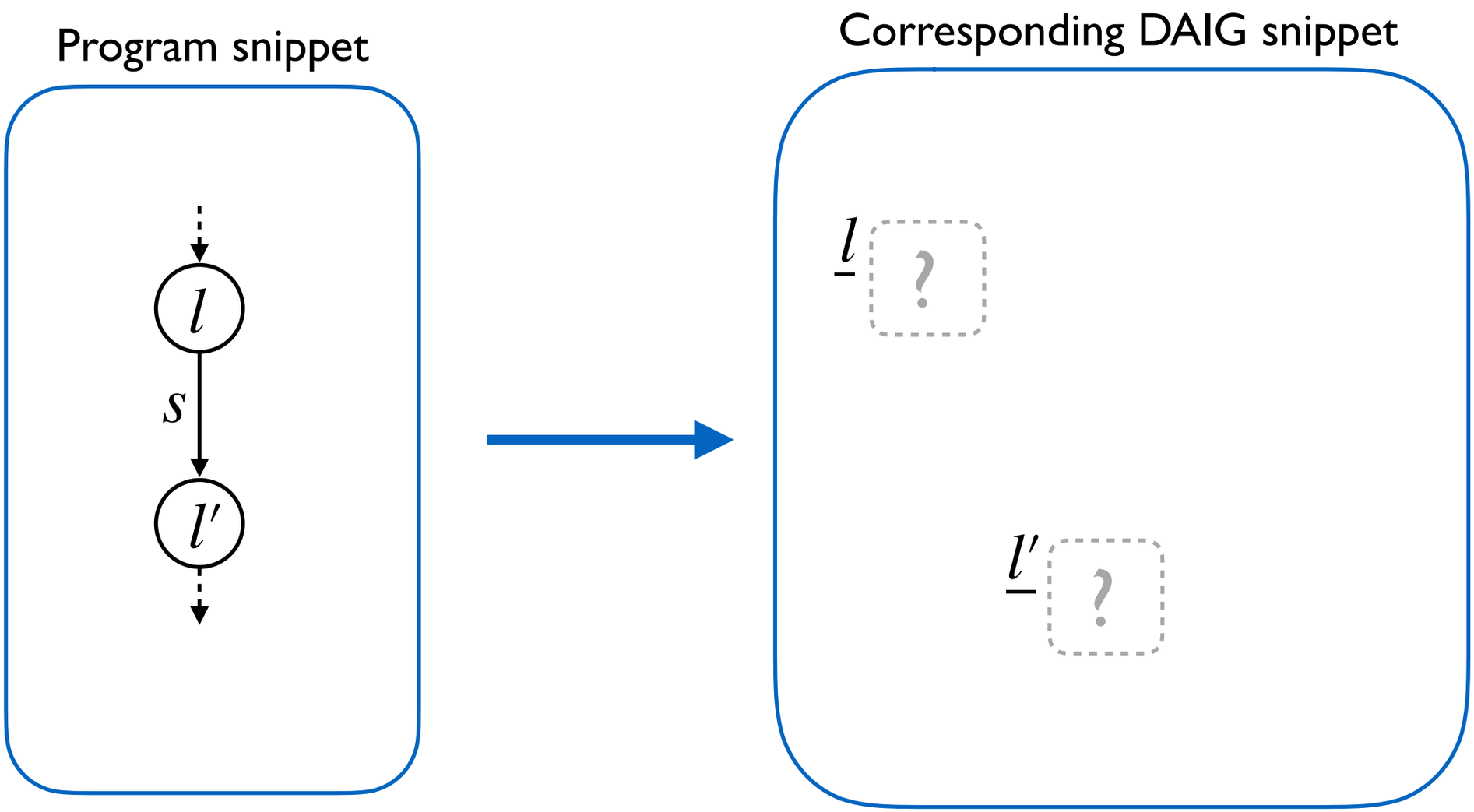
CFG → DAIG Encoding (by example)

Program snippet

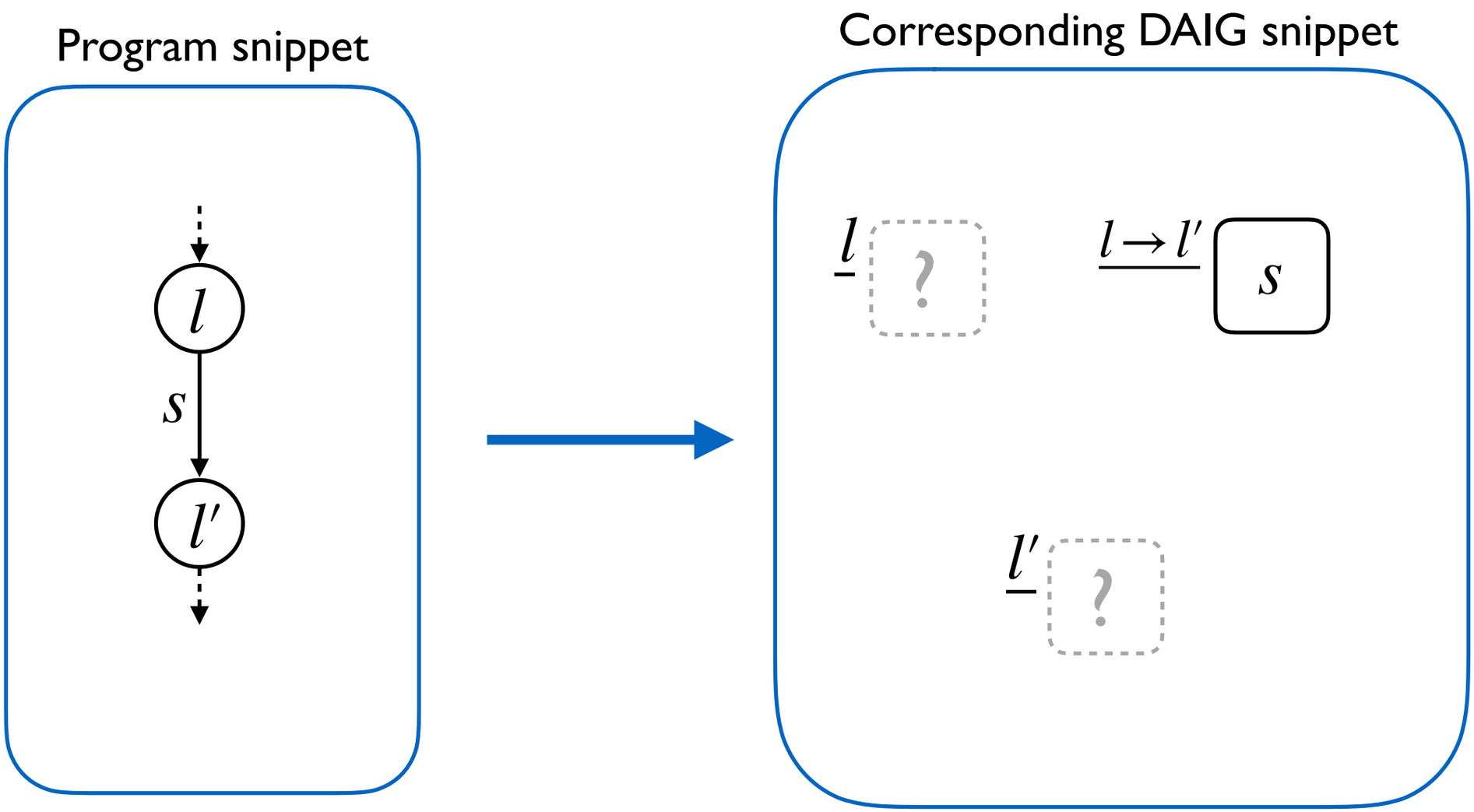


Corresponding DAIG snippet

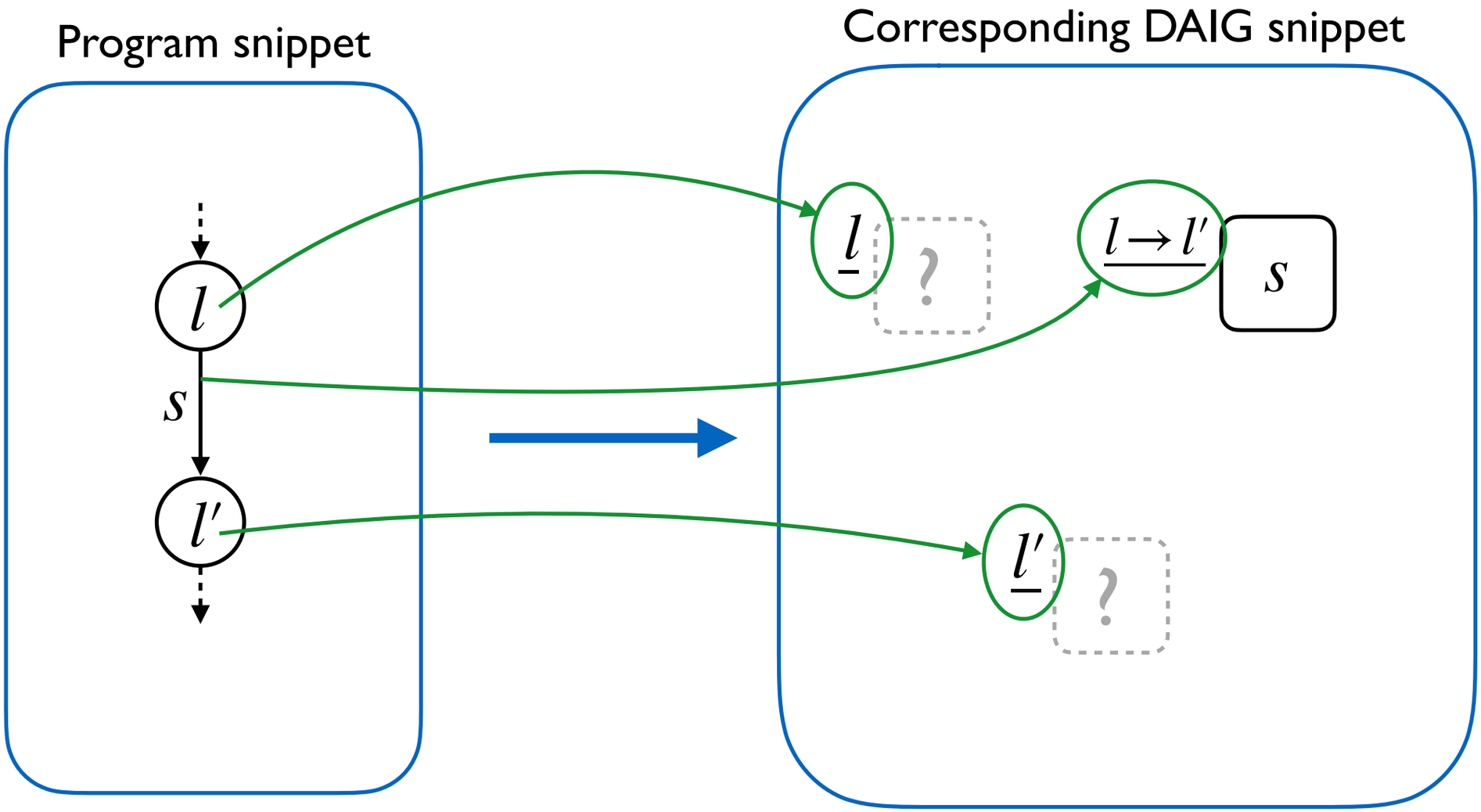
CFG → DAIG Encoding (by example)



CFG → DAIG Encoding (by example)

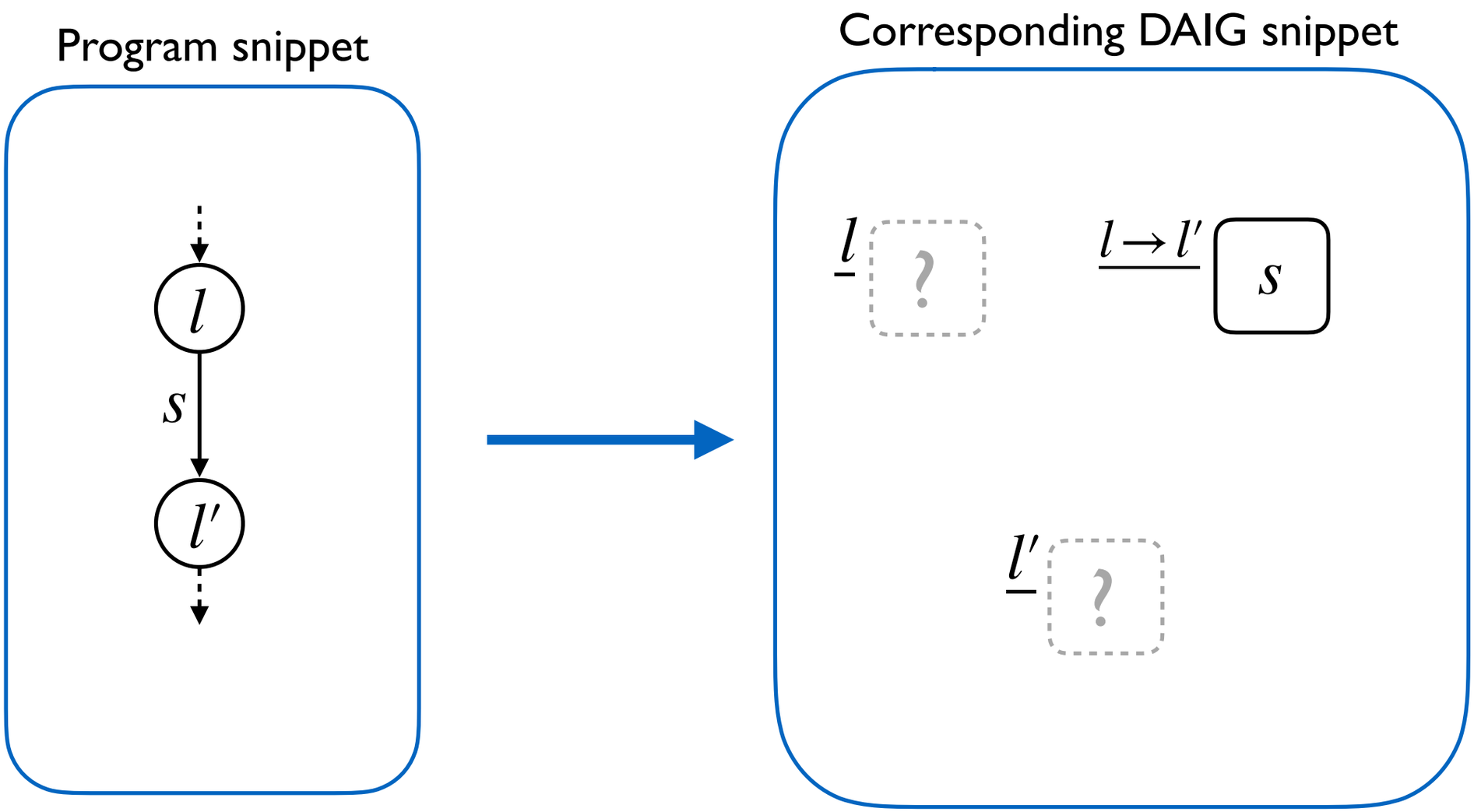


CFG → DAIG Encoding (by example)

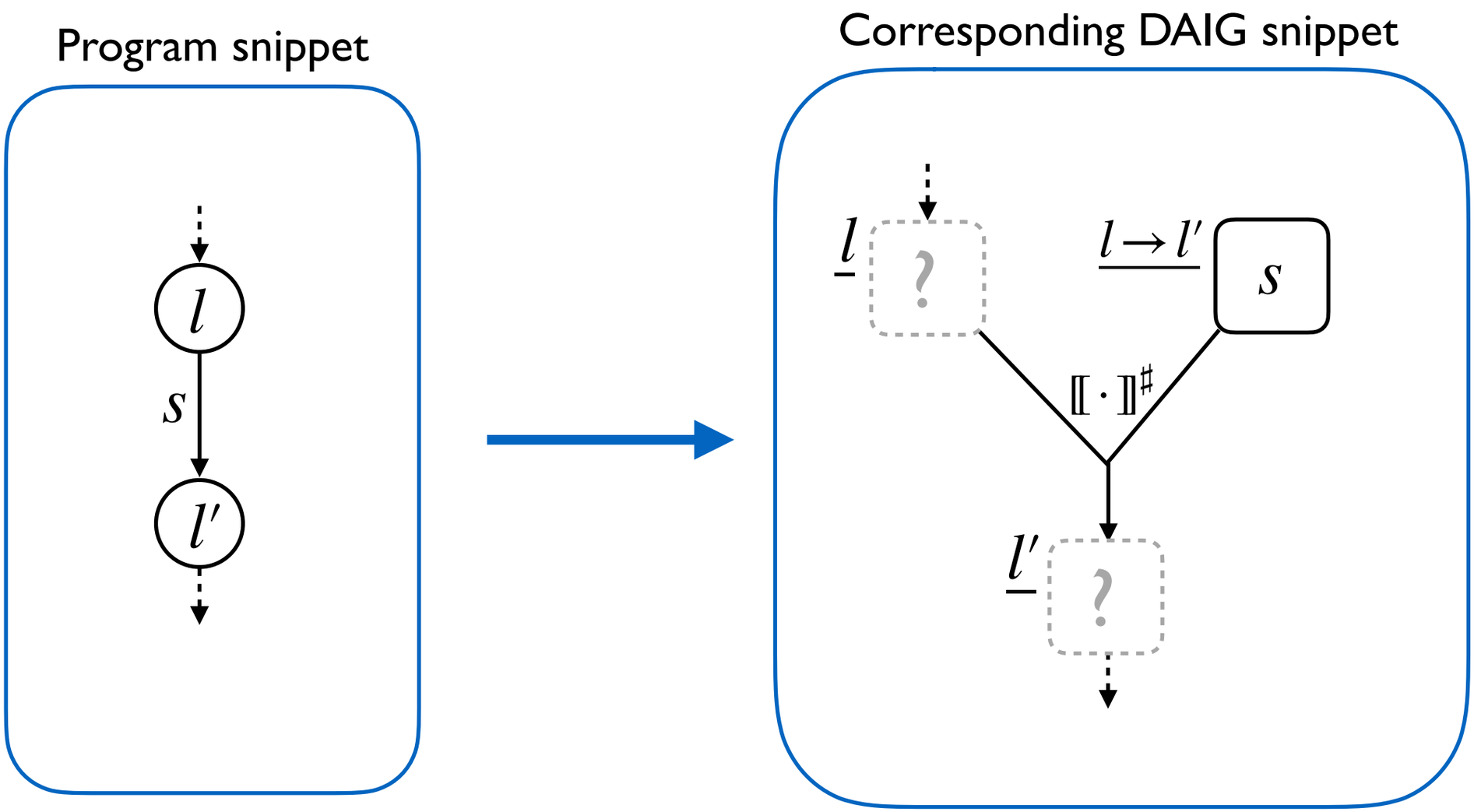


Underlined terms are *names* derived e.g. by hashing

CFG → DAIG Encoding (by example)

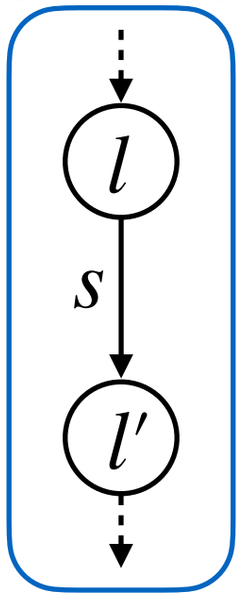


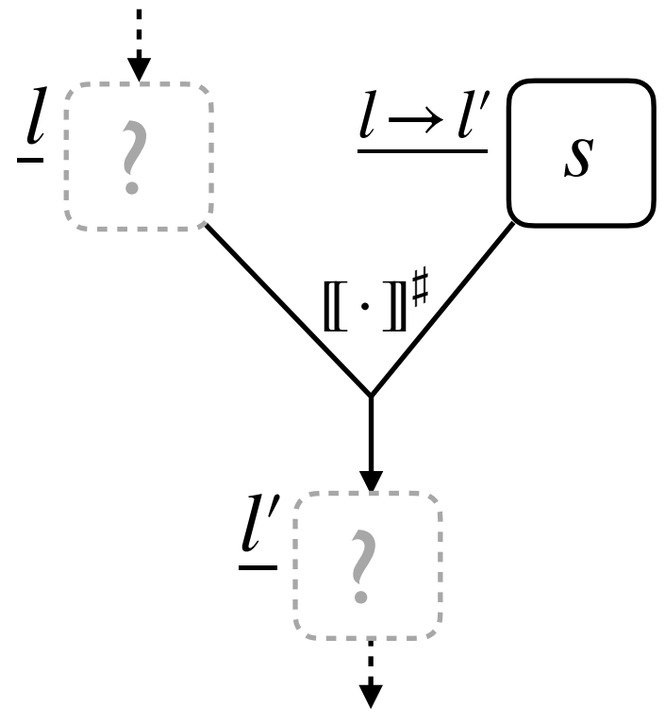
CFG → DAIG Encoding (by example)



Demand-Driven Query Evaluation

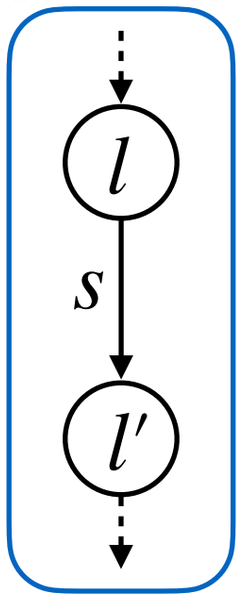
Program snippet

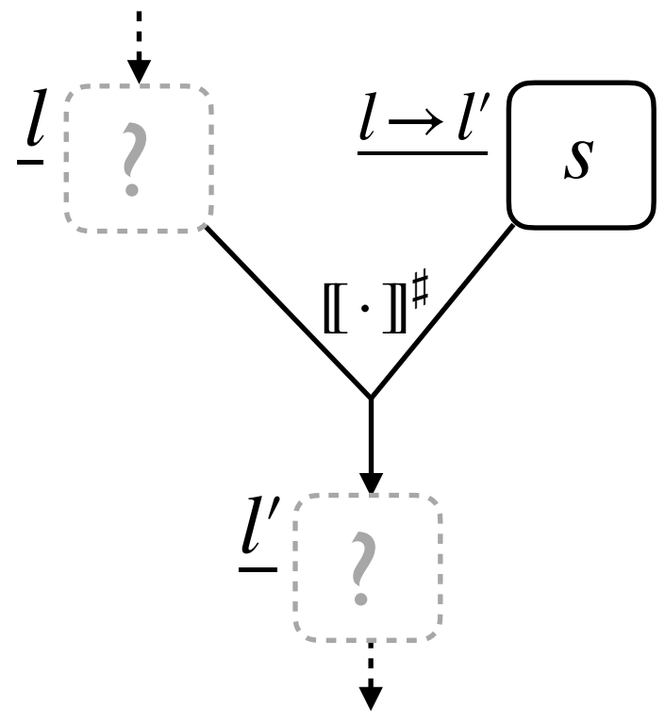




Demand-Driven Query Evaluation

Program snippet





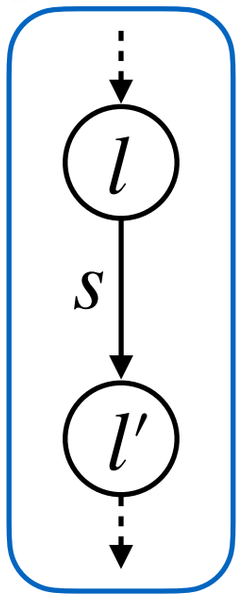
Demand-driven query:

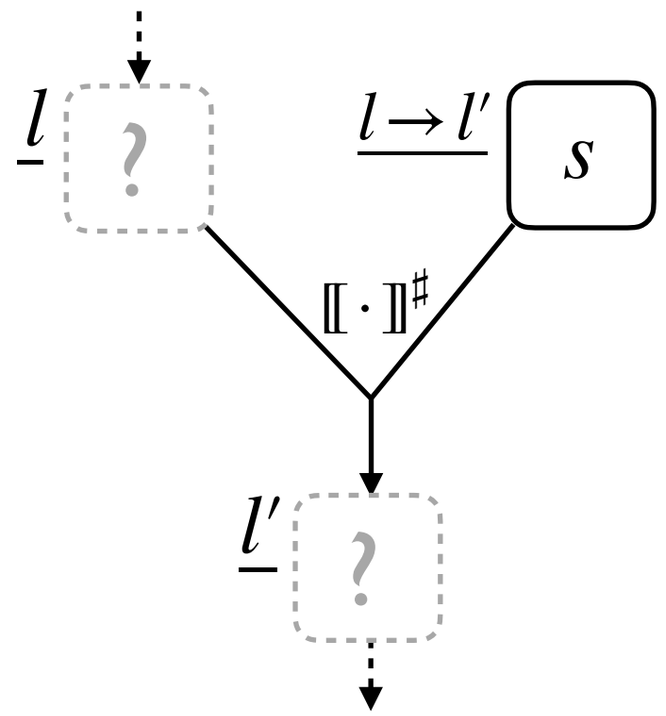
"What is the abstract state at l' ?"

i.e. "What is the value of cell $\underline{l'}$?"

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

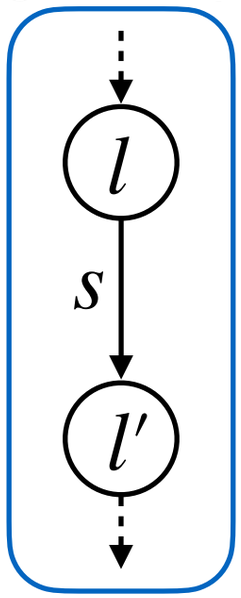
"What is the abstract state at l' ?"

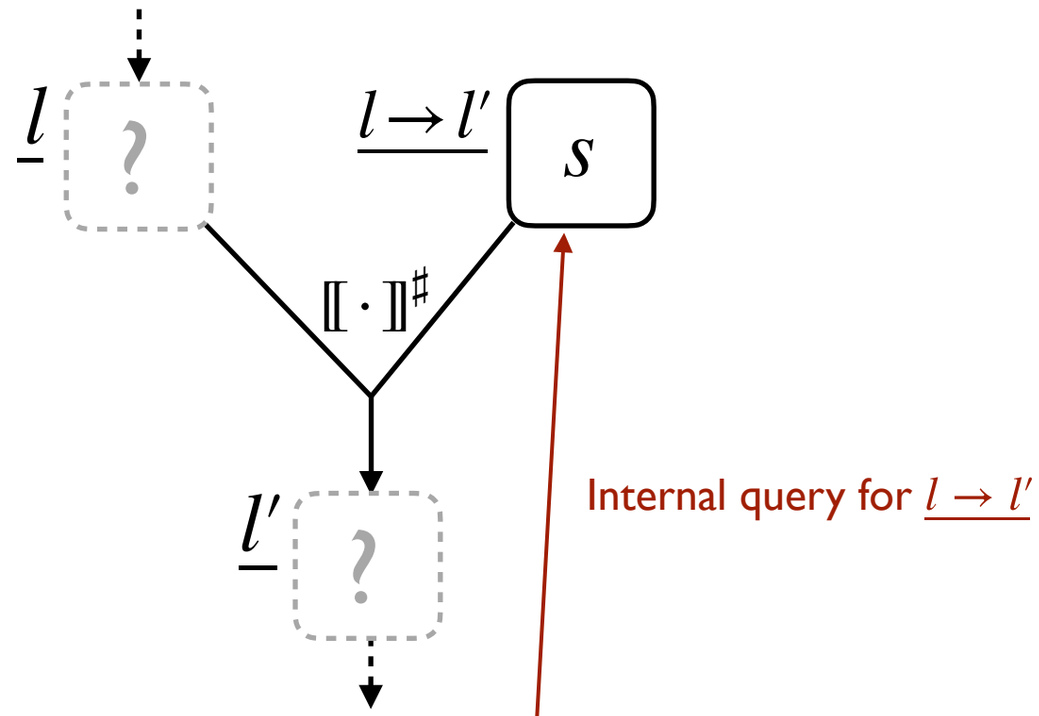
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![\underline{l \to l'}]\!]^\sharp \underline{l}$$

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

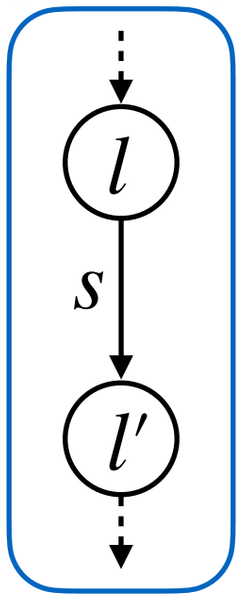
"What is the abstract state at l' ?"

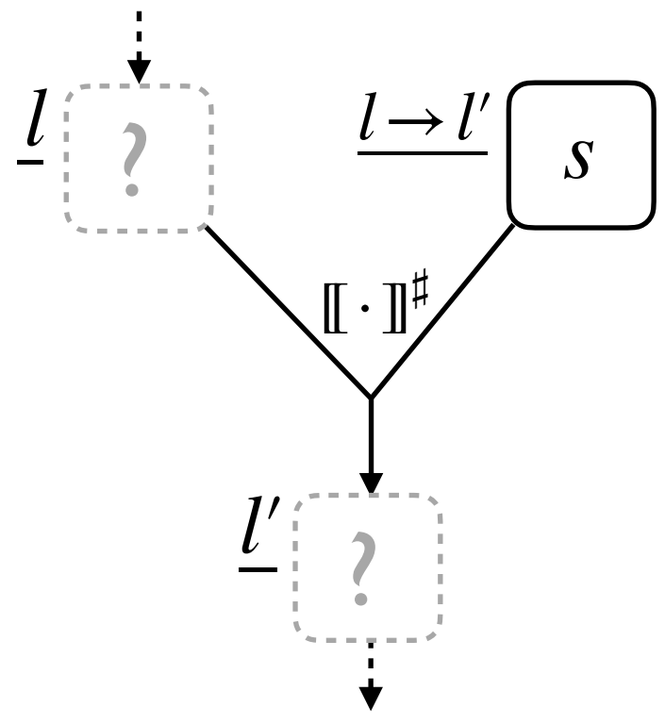
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![\underline{l \to l'}]\!]^\sharp \underline{l}$$

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

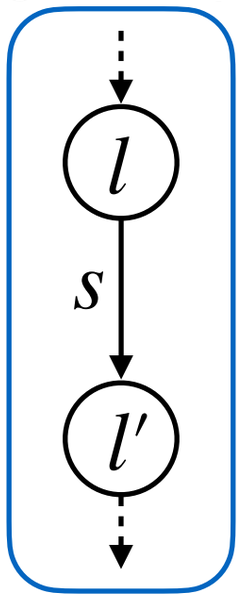
"What is the abstract state at l' ?"

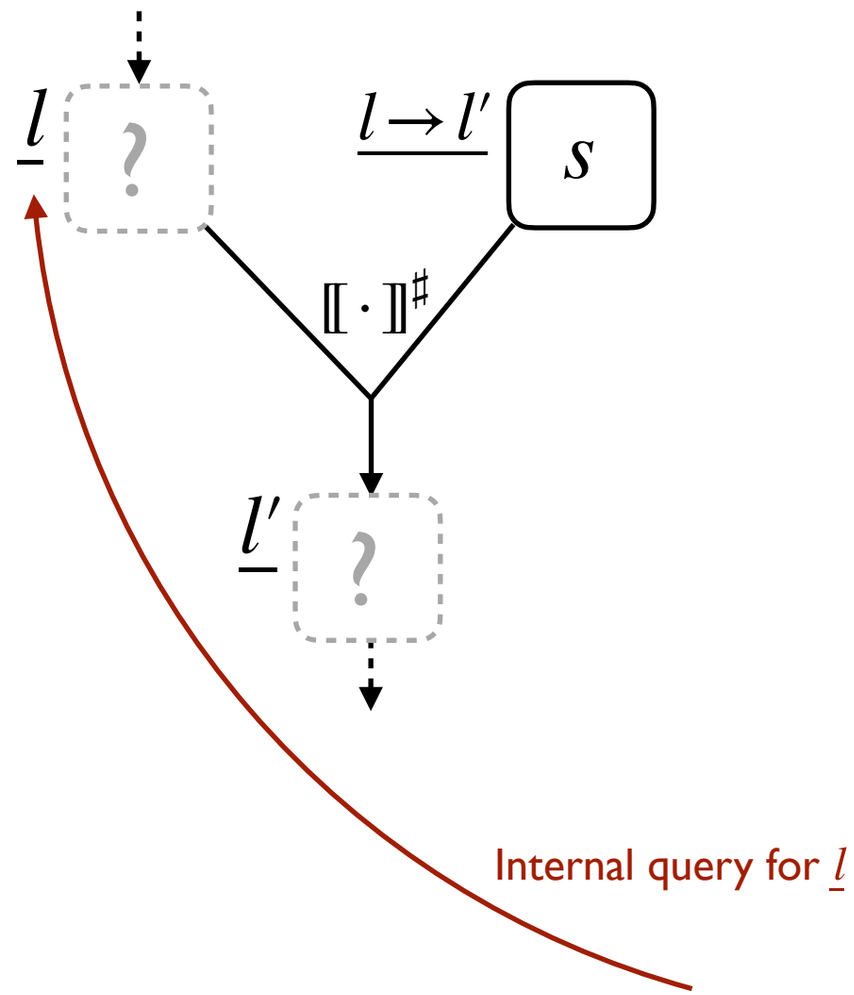
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![s]\!]^\sharp \underline{l}$$

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

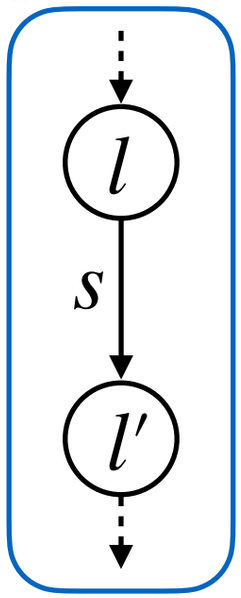
"What is the abstract state at l' ?"

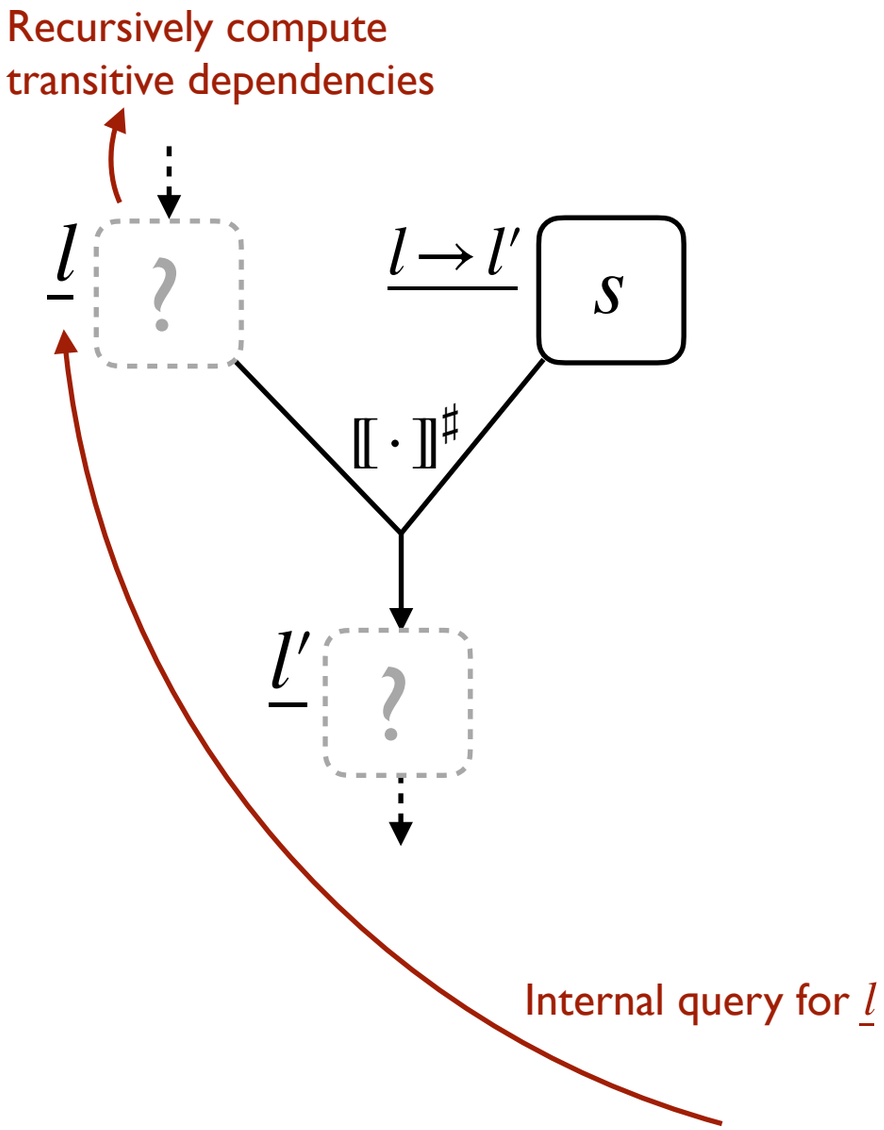
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![s]\!]^\sharp \, \underline{l}$$

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

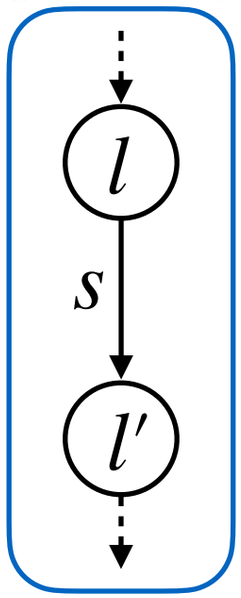
"What is the abstract state at l' ?"

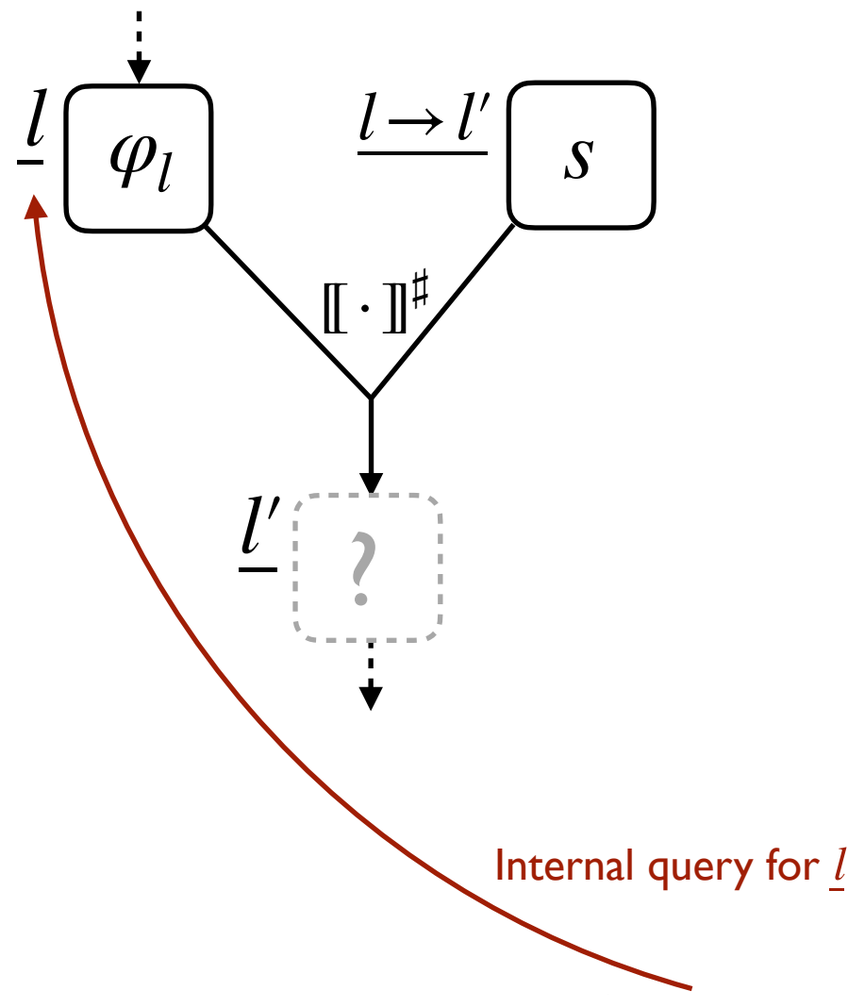
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![s]\!]^\sharp \, \underline{l}$$

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

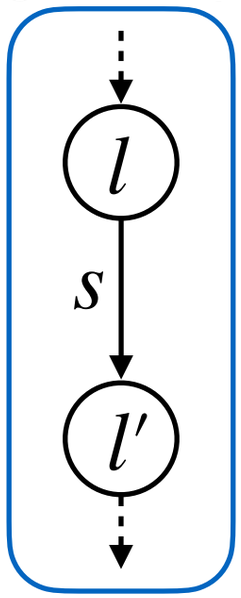
"What is the abstract state at l' ?"

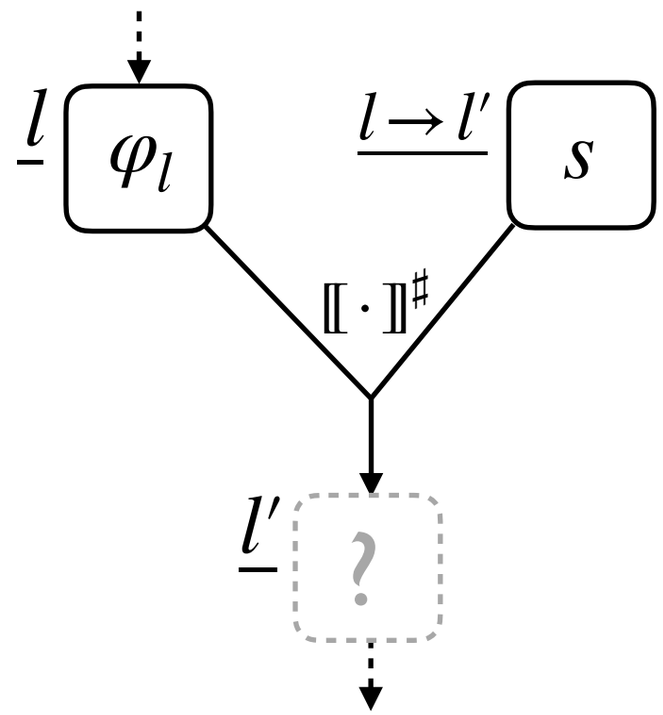
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![s]\!]^\sharp \underline{l}$$

Demand-Driven Query Evaluation

Program snippet





Demand-driven query:

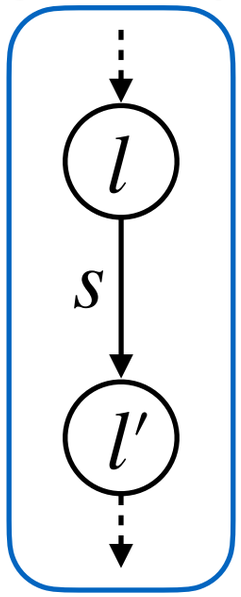
"What is the abstract state at l' ?"

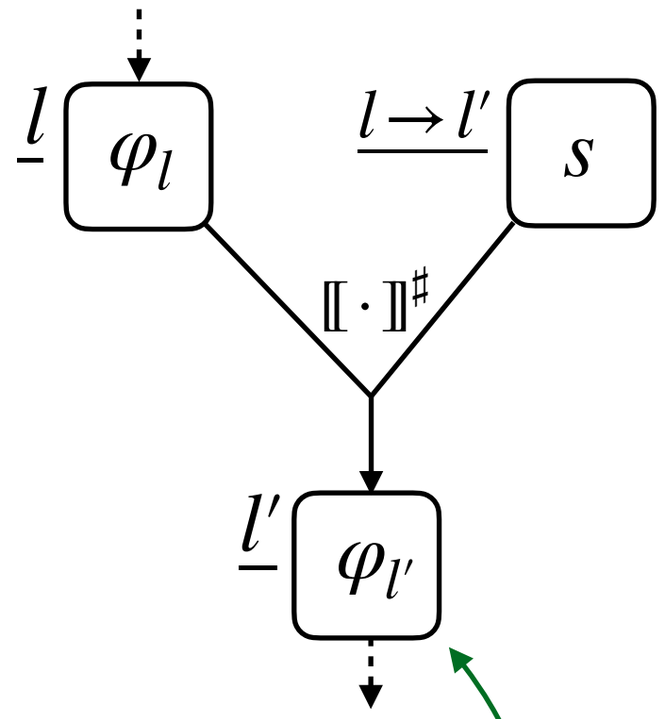
i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![s]\!]^\sharp \varphi_l$$

Demand-Driven Query Evaluation

Program snippet





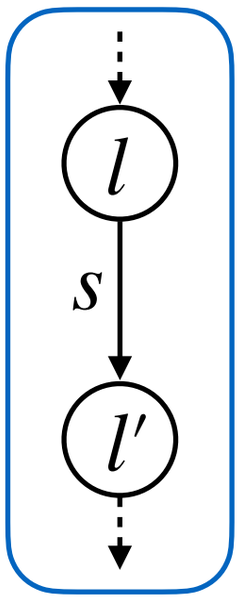
Demand-driven query:

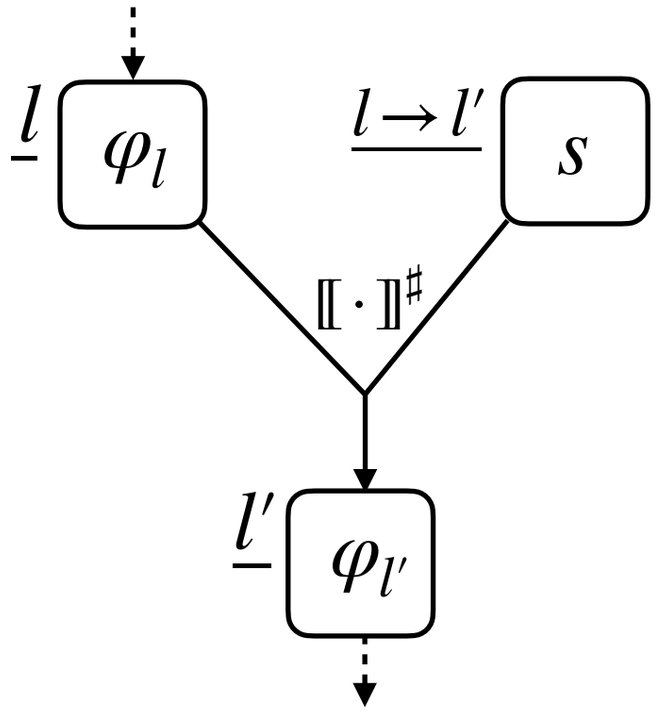
"What is the abstract state at l' ?"

i.e. "What is the value of cell $\underline{l'}$?"

$$\underline{l'} = [\![s]\!]^\sharp \varphi_l$$

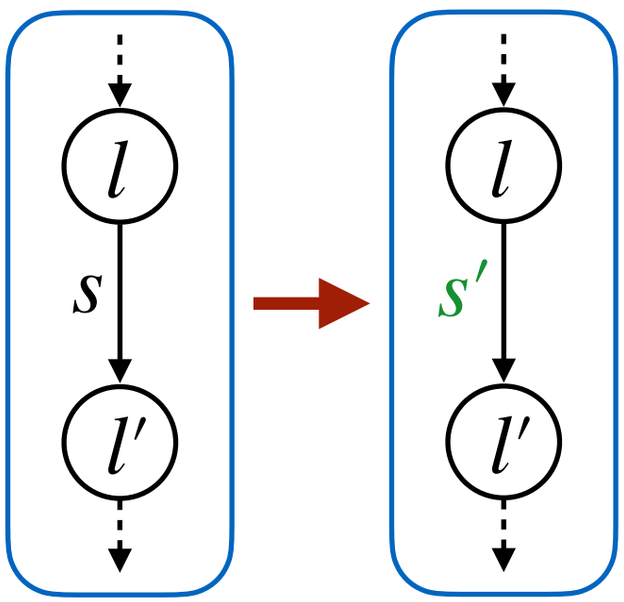
Incremental Change Propagation

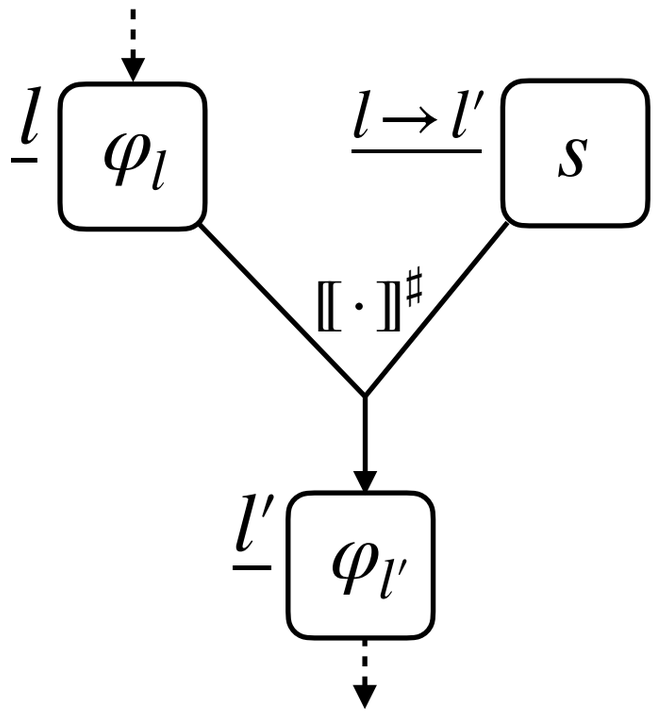




Incremental Change Propagation

Program edit:
Modify statement s to s'

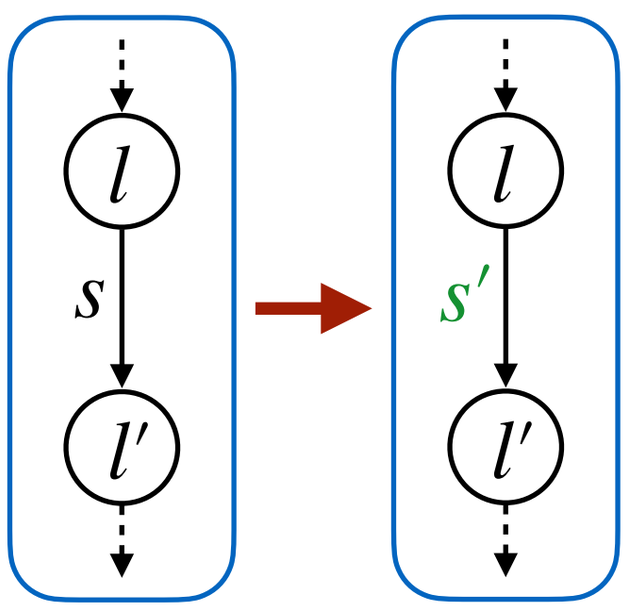


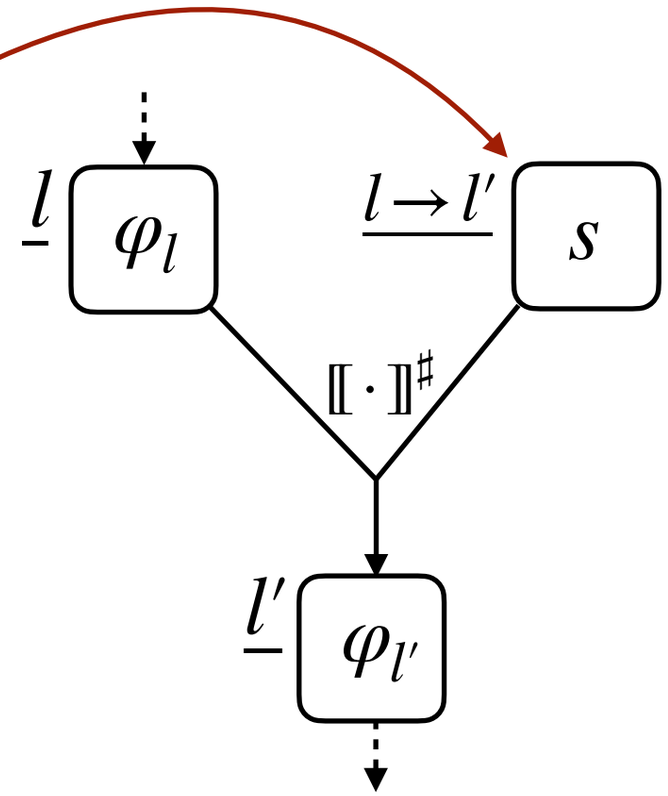


Incremental Change Propagation

Program edit:
Modify statement s to s'

i.e. "Write s' to the cell named $\underline{l \to l'}$"

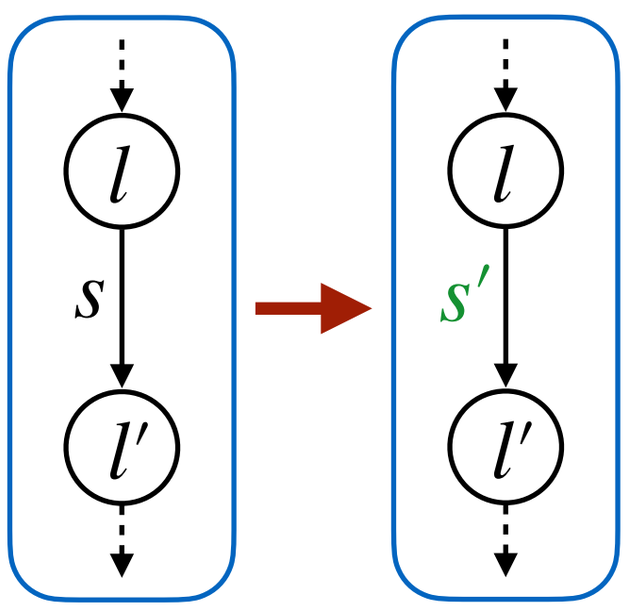


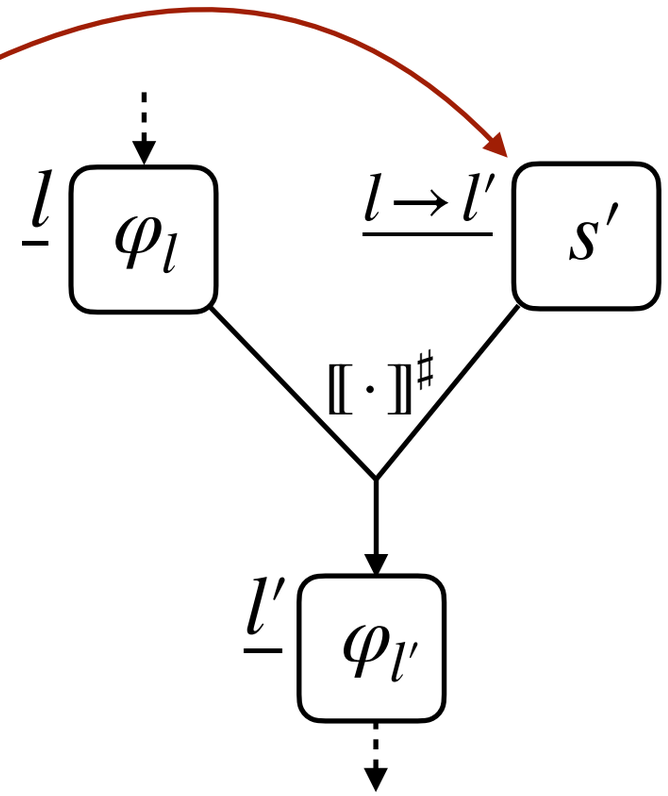


Incremental Change Propagation

Program edit:
Modify statement s to s'

i.e. "Write s' to the cell named $\underline{l \to l'}$"

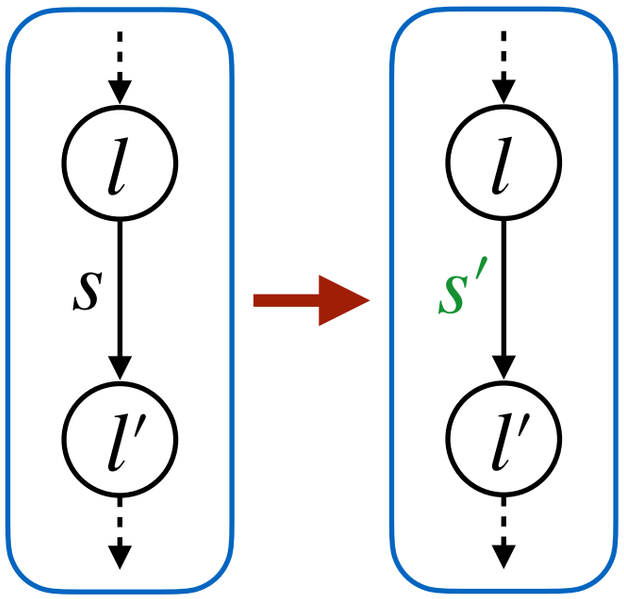


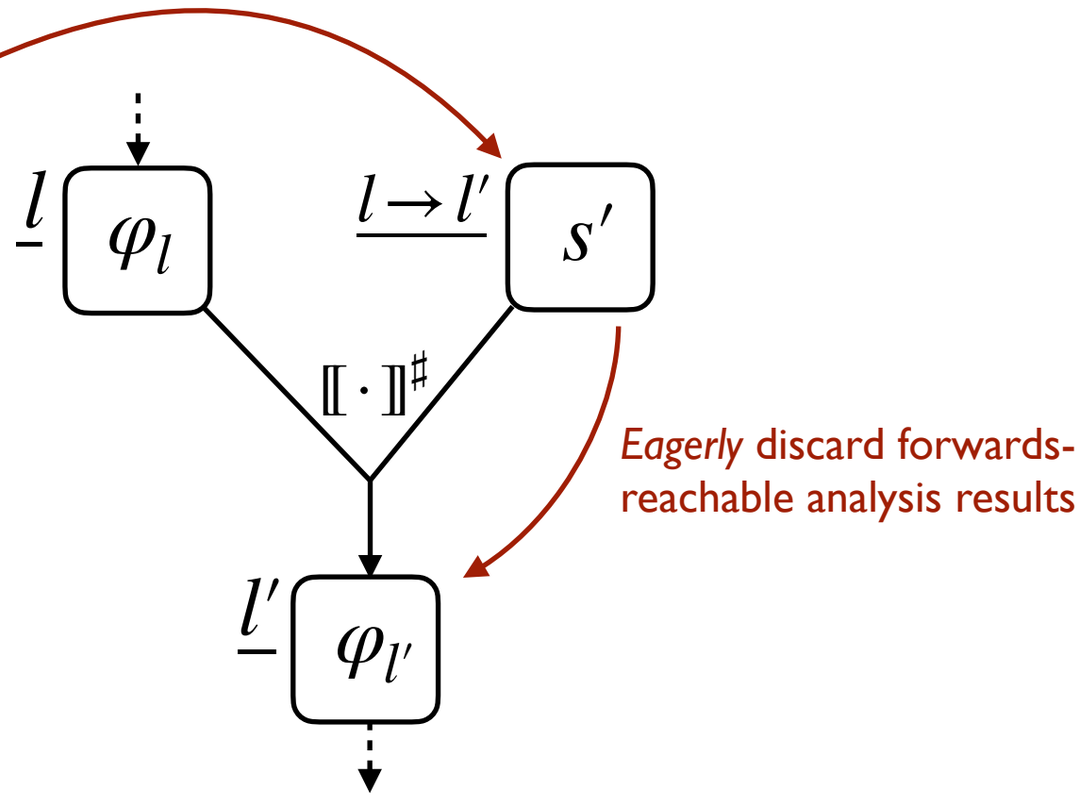


Incremental Change Propagation

Program edit:
Modify statement s to s'

i.e. "Write s' to the cell named $\underline{l \to l'}$"



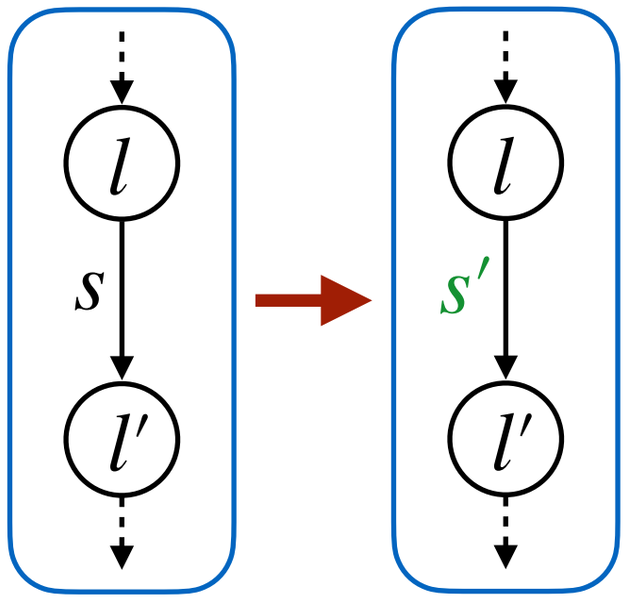


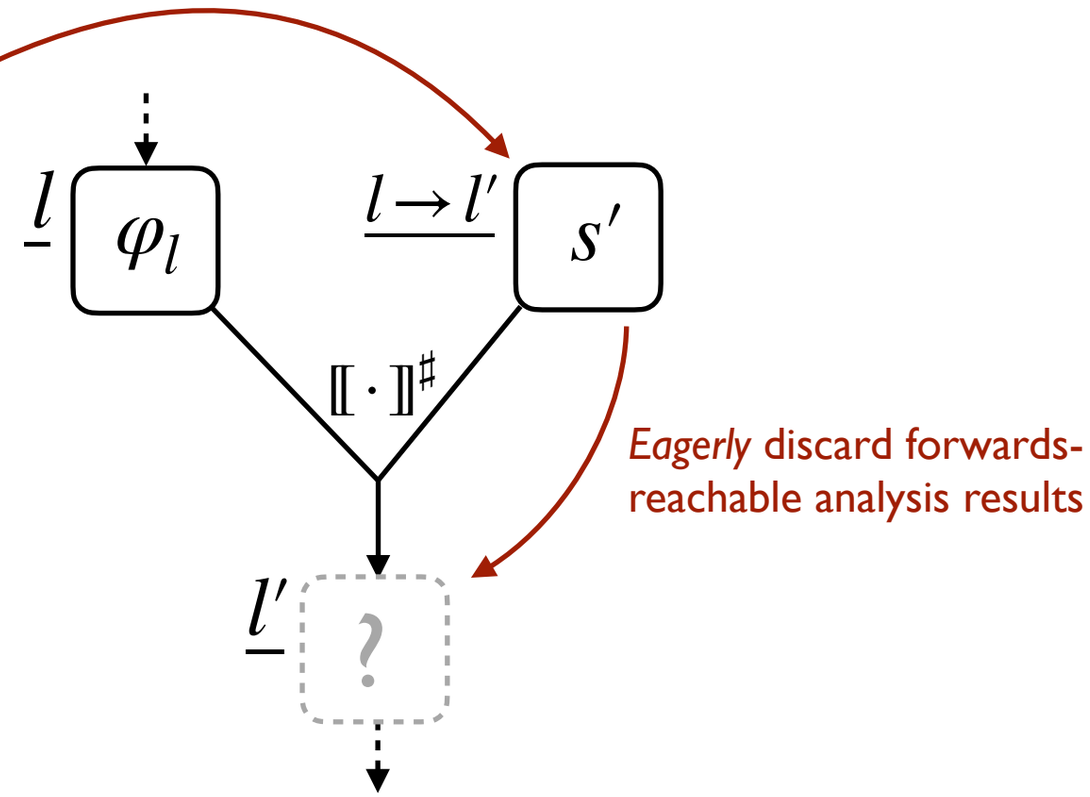
Eagerly discard forwards-reachable analysis results

Incremental Change Propagation

Program edit:
Modify statement s to s'

i.e. "Write s' to the cell named $\underline{l \to l'}$"



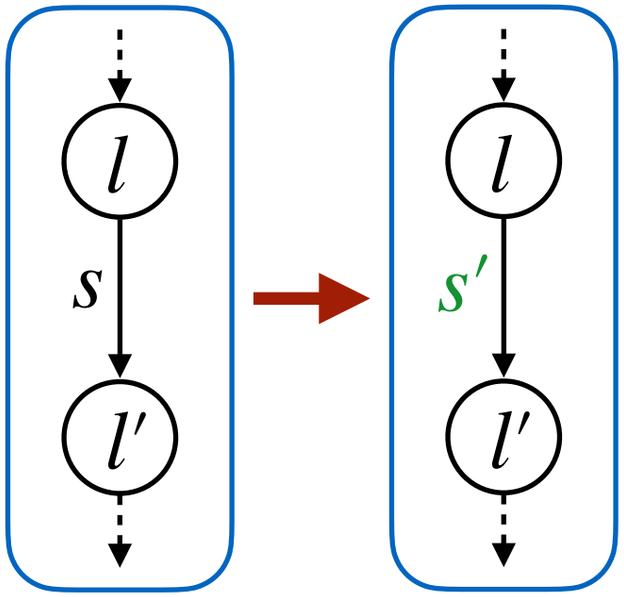


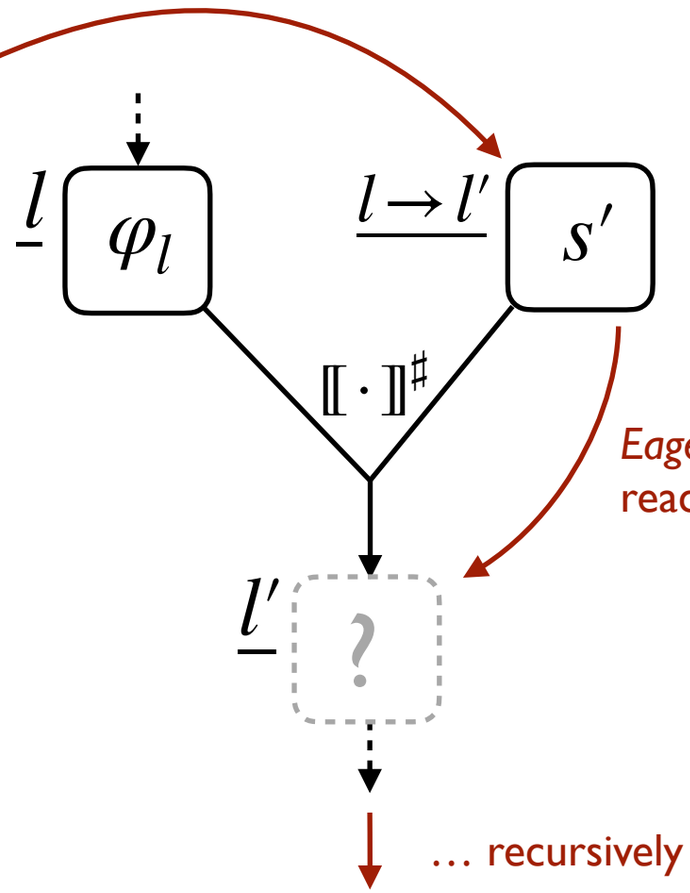
Eagerly discard forwards-reachable analysis results

Incremental Change Propagation

Program edit:
Modify statement s to s'

i.e. "Write s' to the cell named $\underline{l \to l'}$"



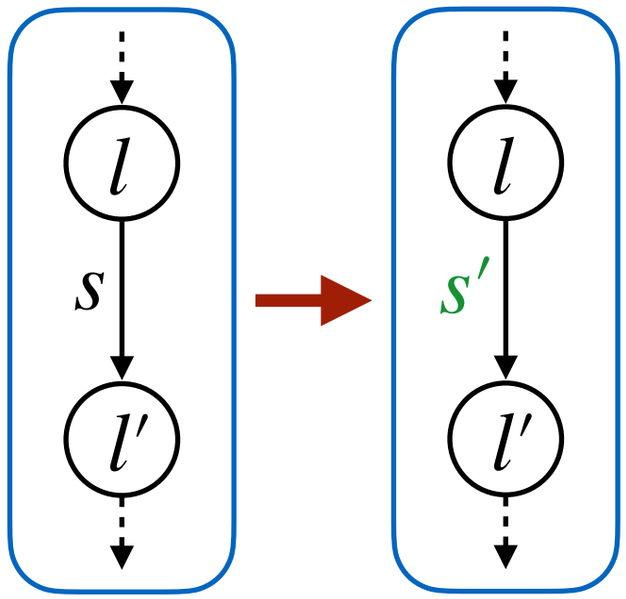


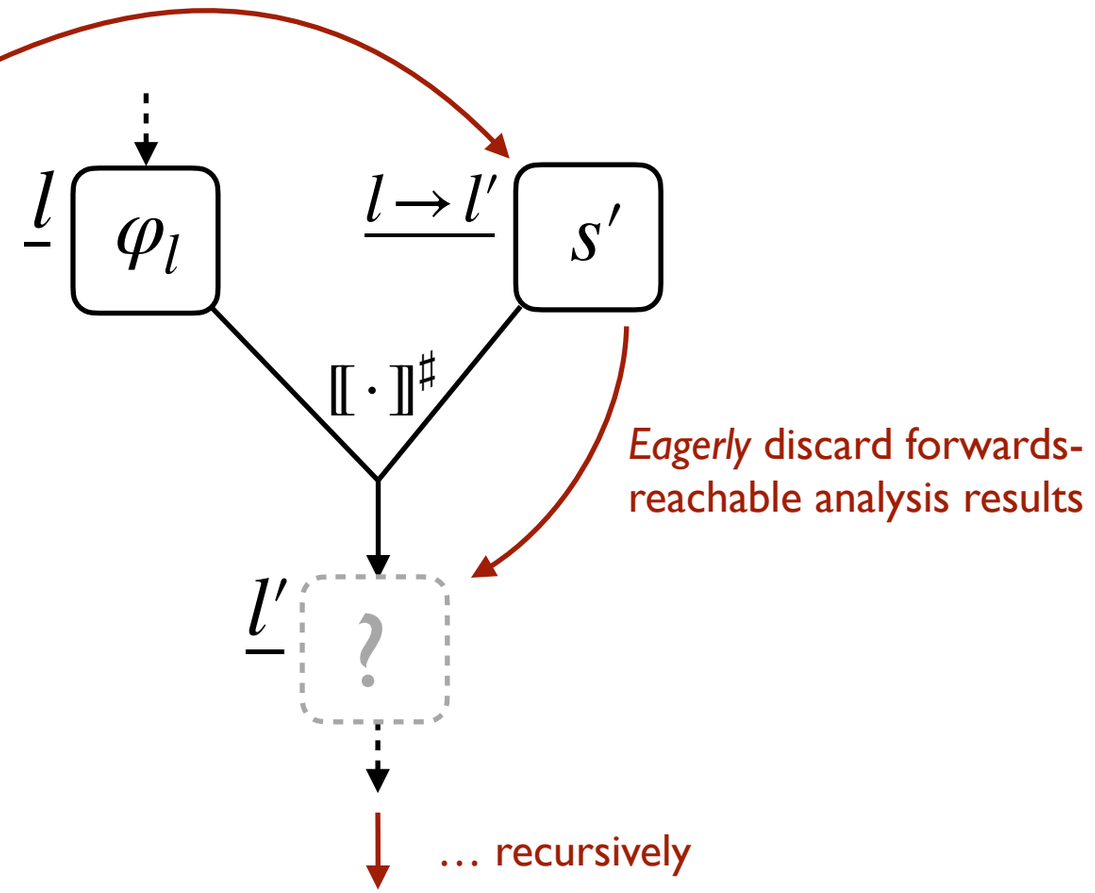
Eagerly discard forwards-reachable analysis results

Incremental Change Propagation

Program edit:
Modify statement s to s'

i.e. "Write s' to the cell named $\underline{l \to l'}$"



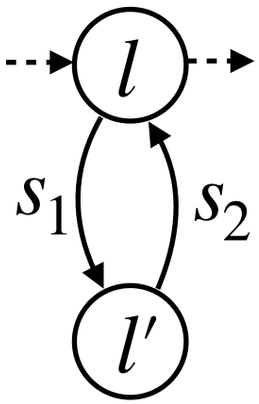


Eagerly discard forwards-reachable analysis results

Subsequent queries only recompute those analysis results potentially affected by the edit!

Cyclic Control Flow

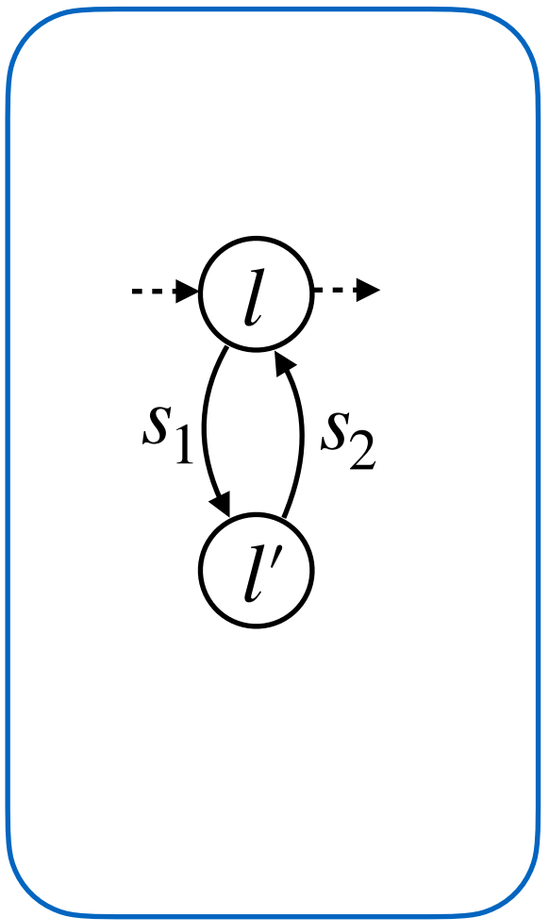
Program snippet



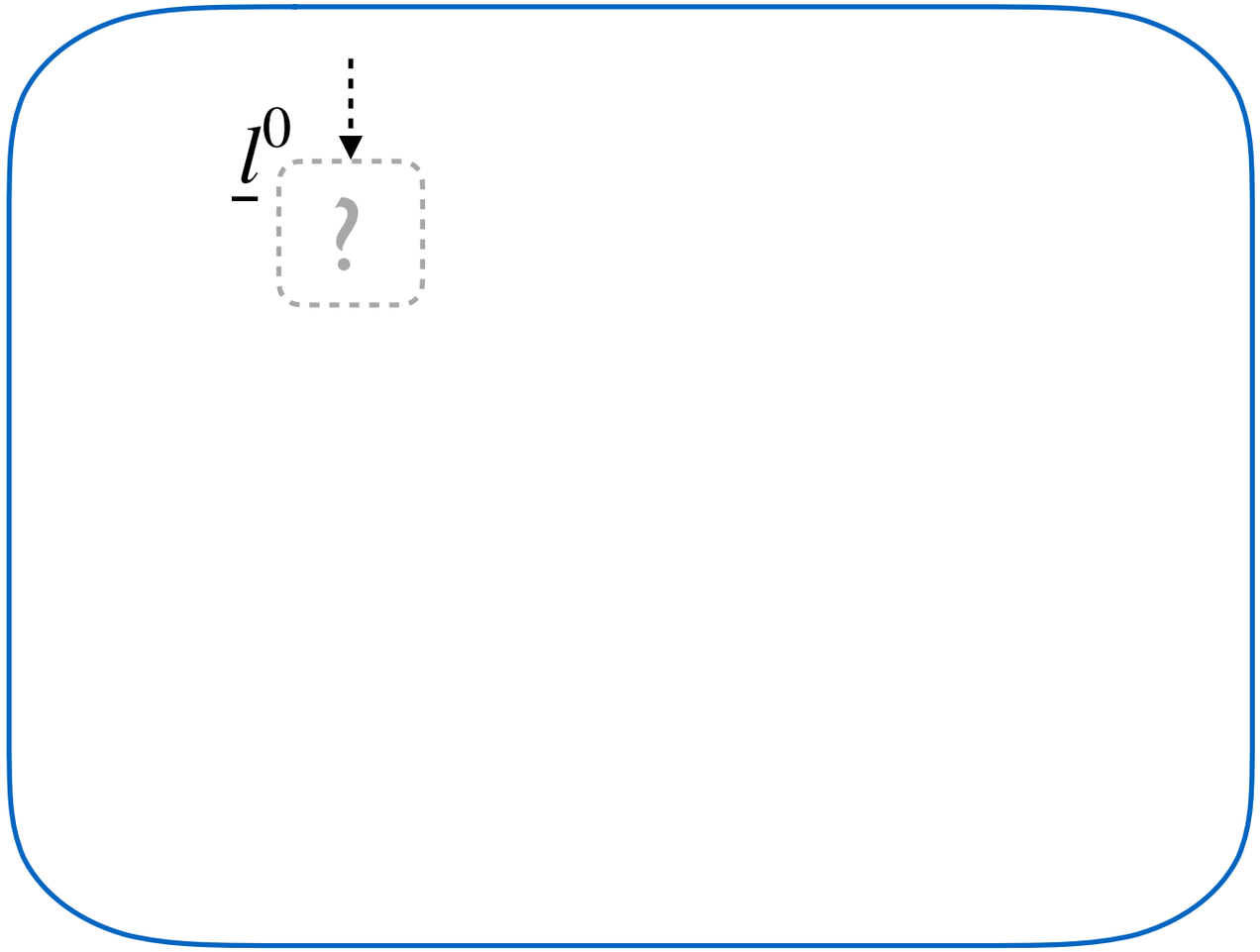
Corresponding DAIG snippet

Cyclic Control Flow

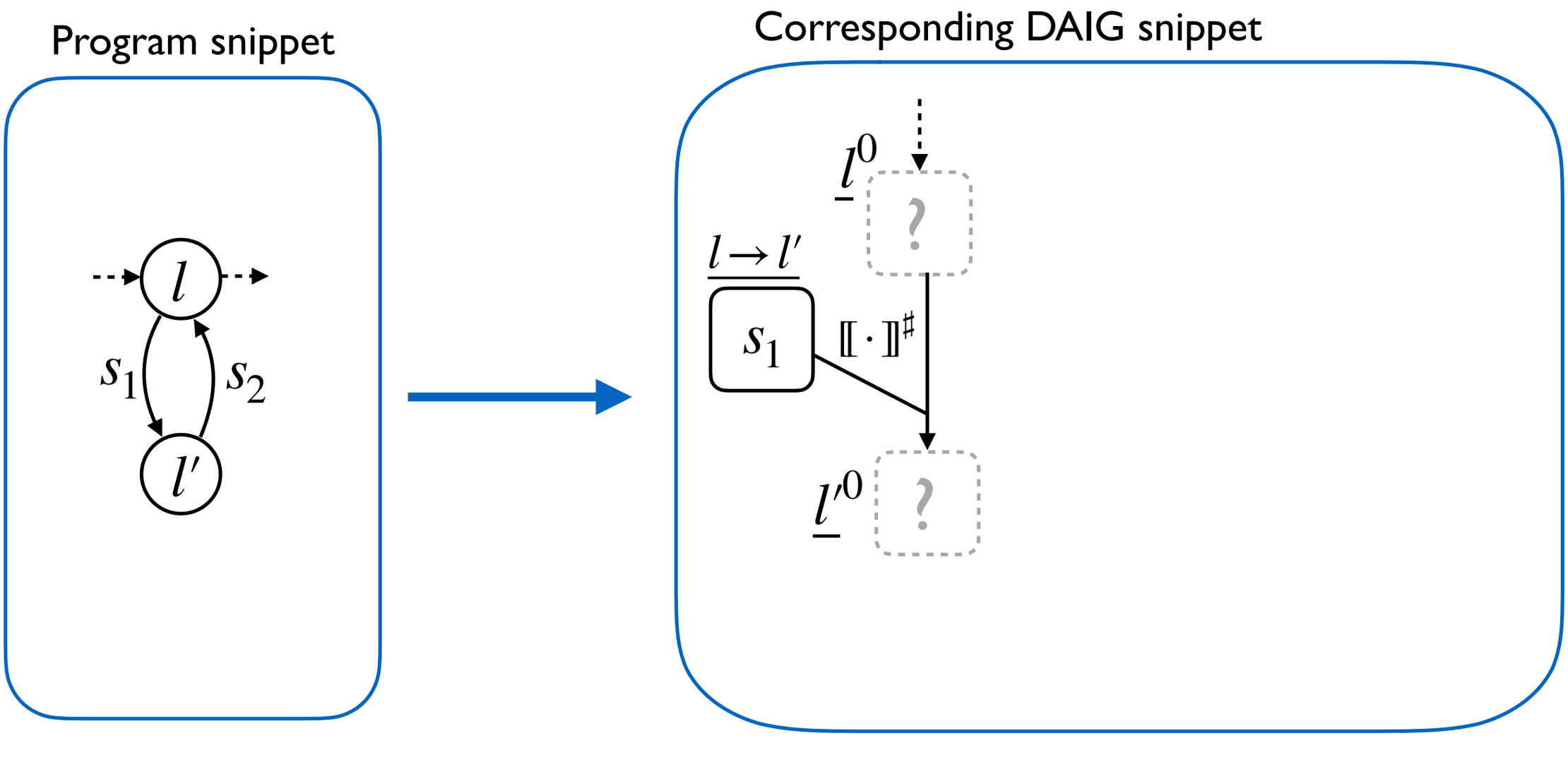
Program snippet



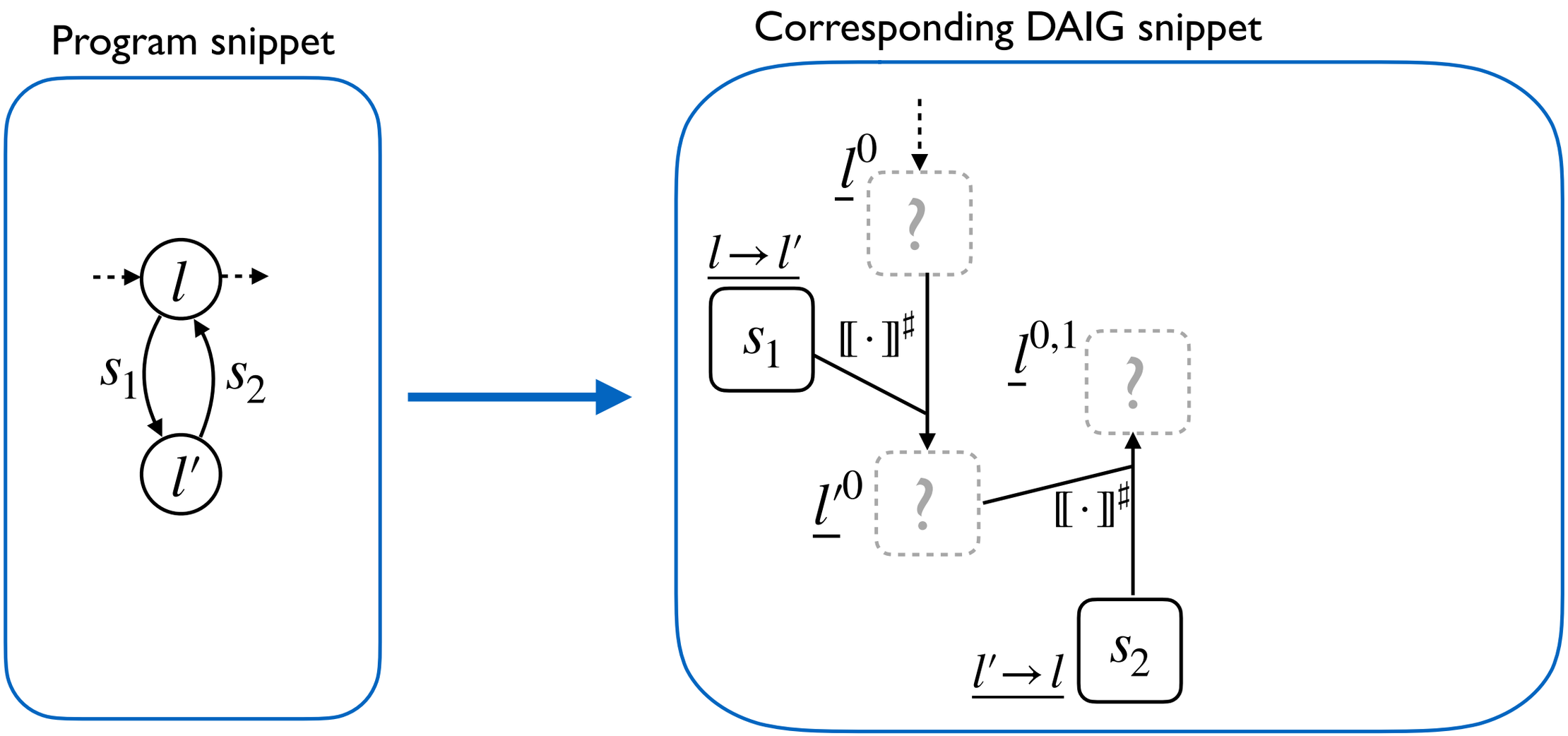
Corresponding DAIG snippet



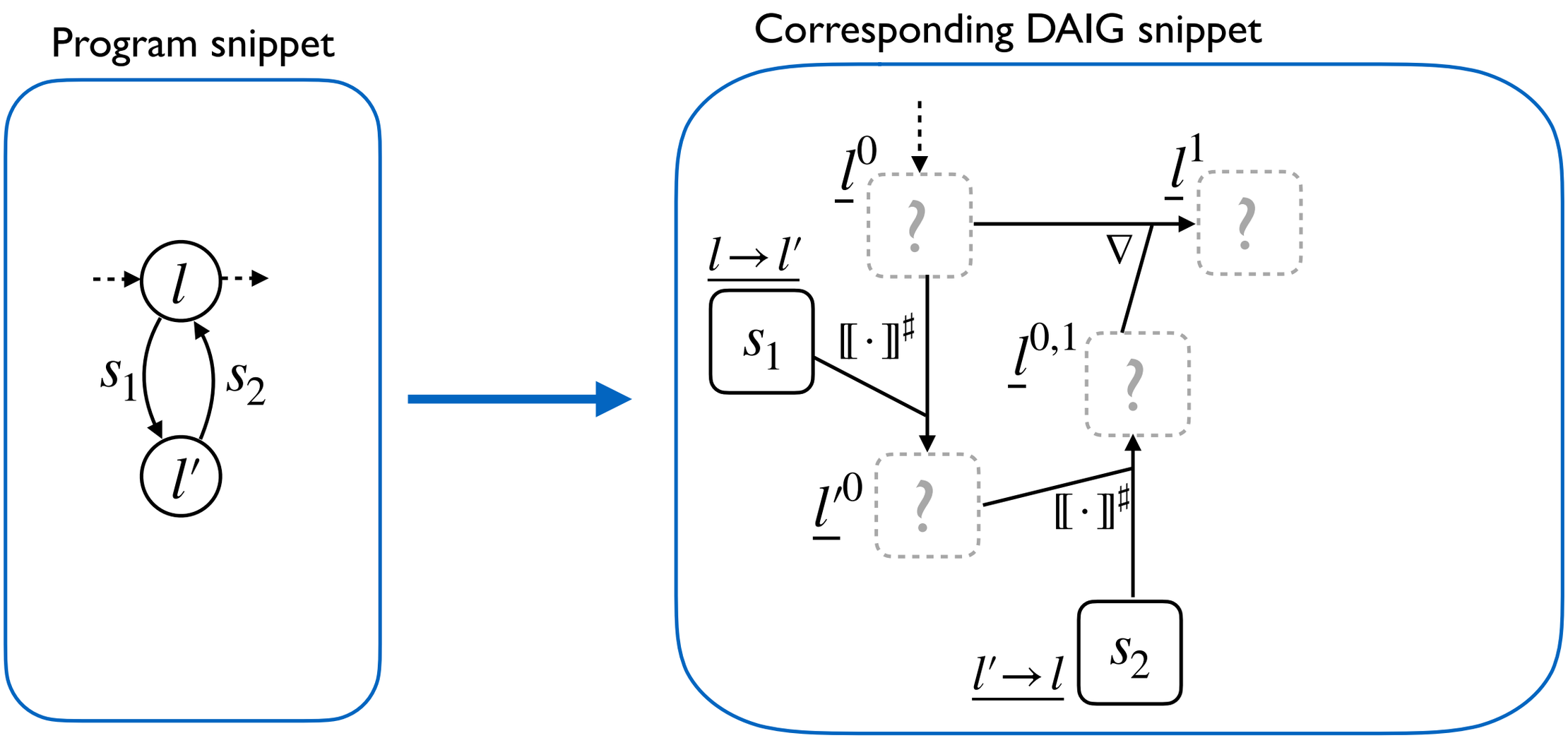
Cyclic Control Flow



Cyclic Control Flow

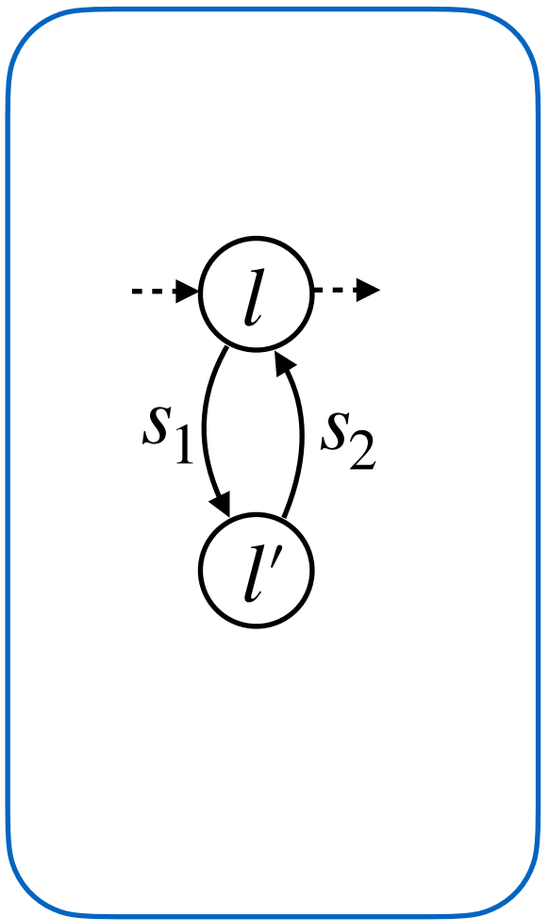


Cyclic Control Flow

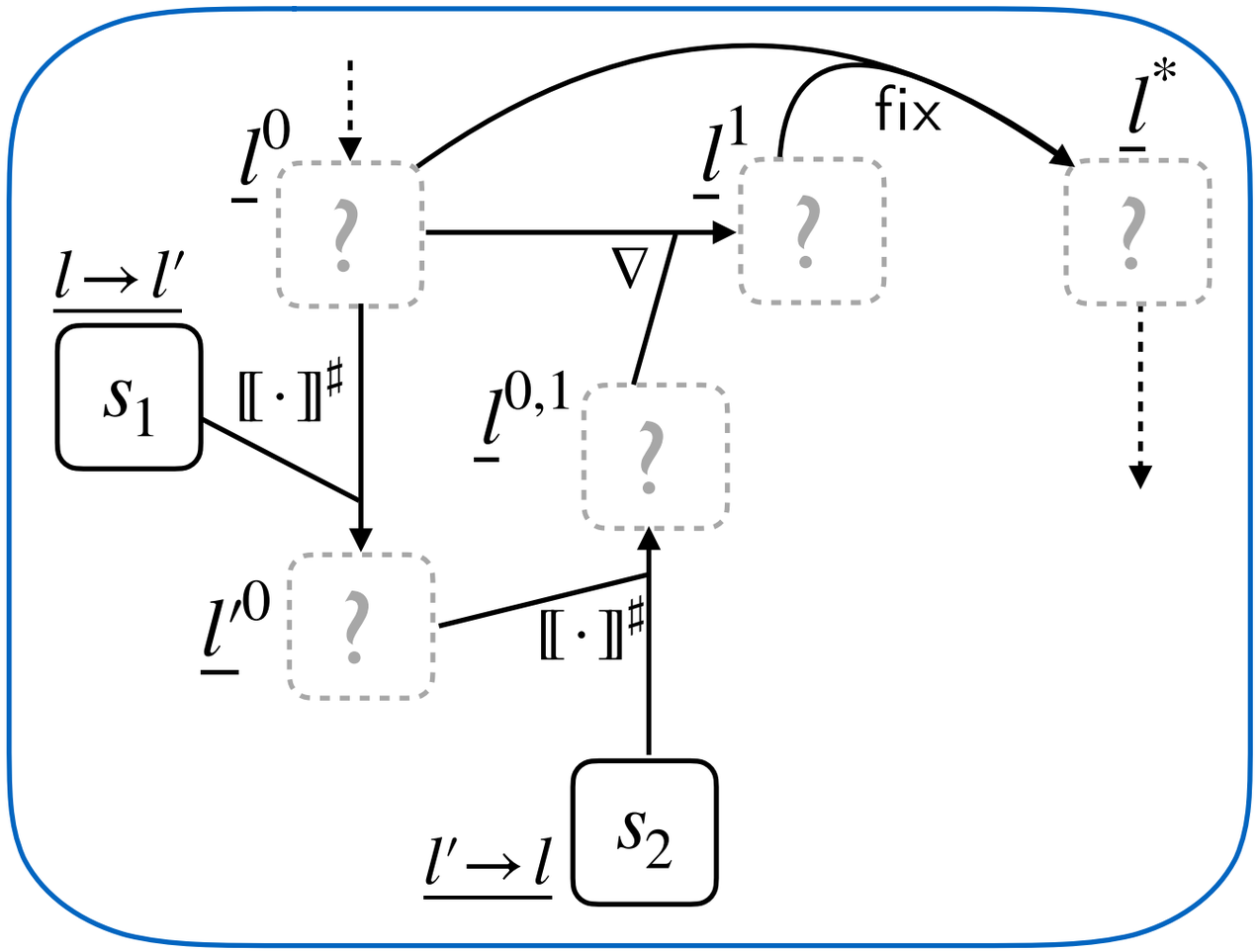


Cyclic Control Flow

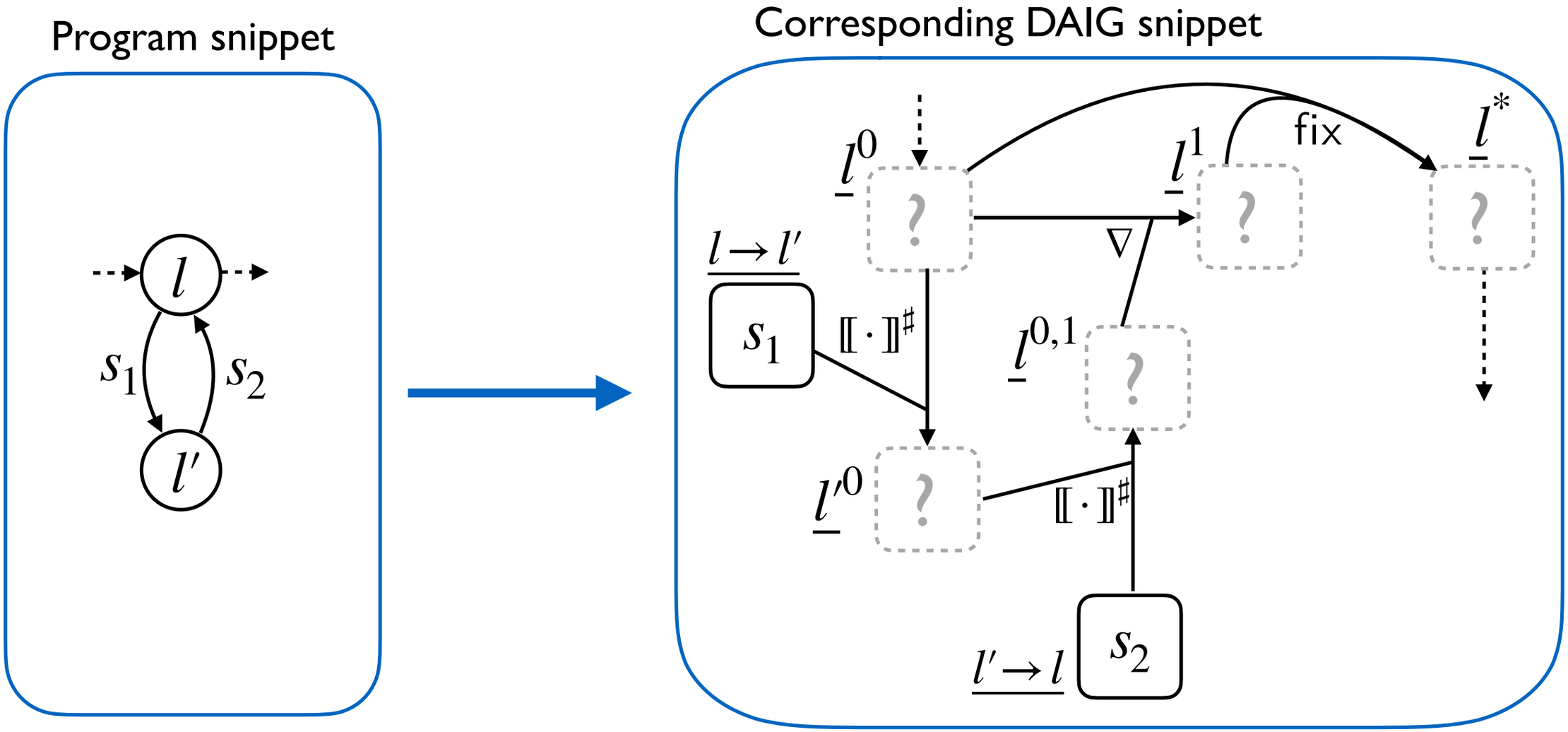
Program snippet



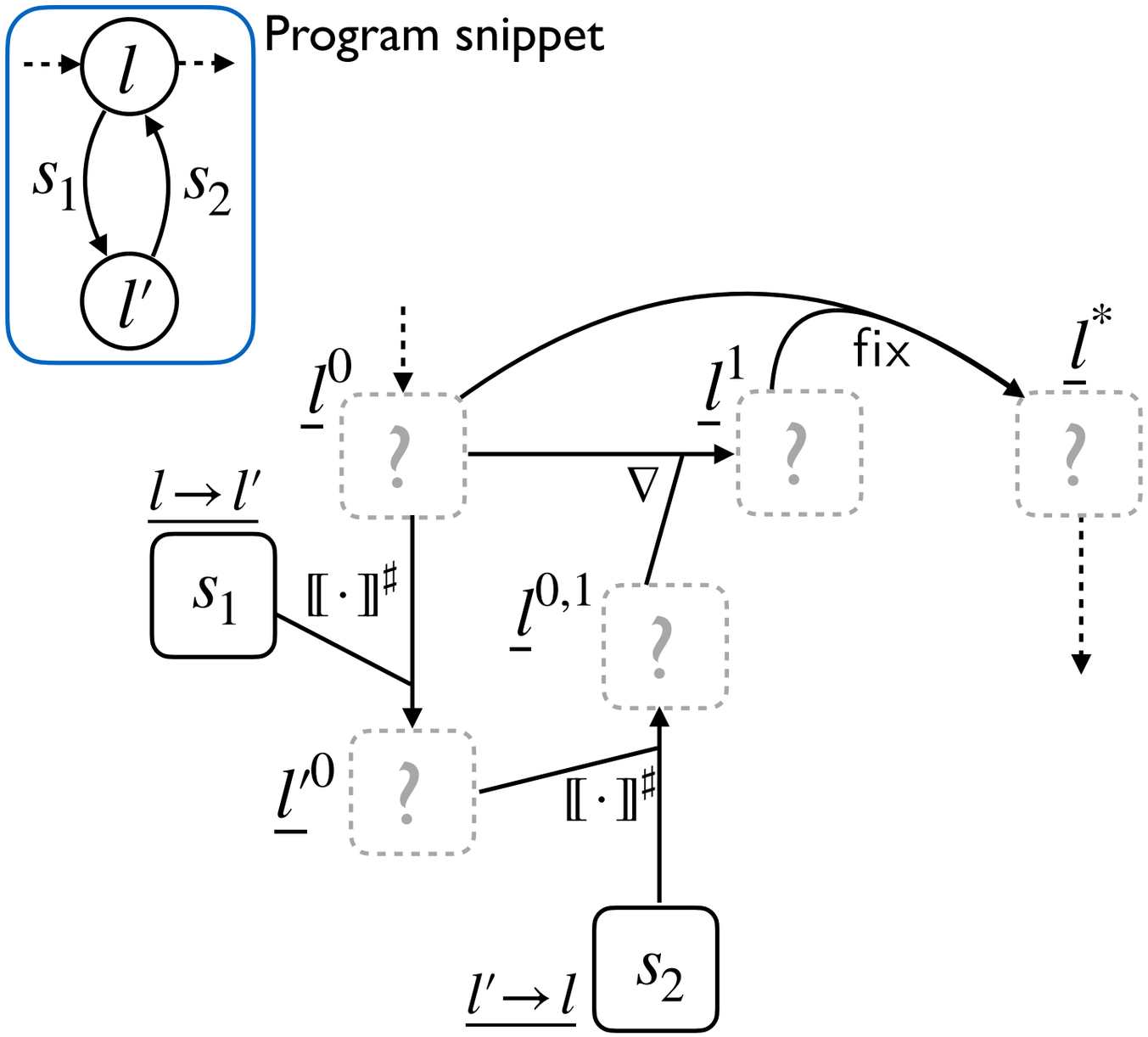
Corresponding DAIG snippet



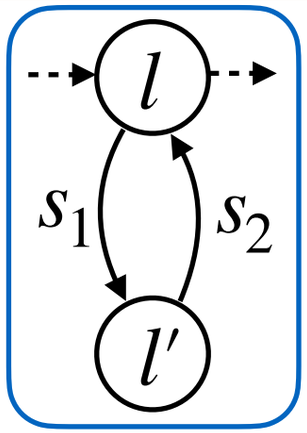
Cyclic Control Flow



DAIG Fixed-point Computation

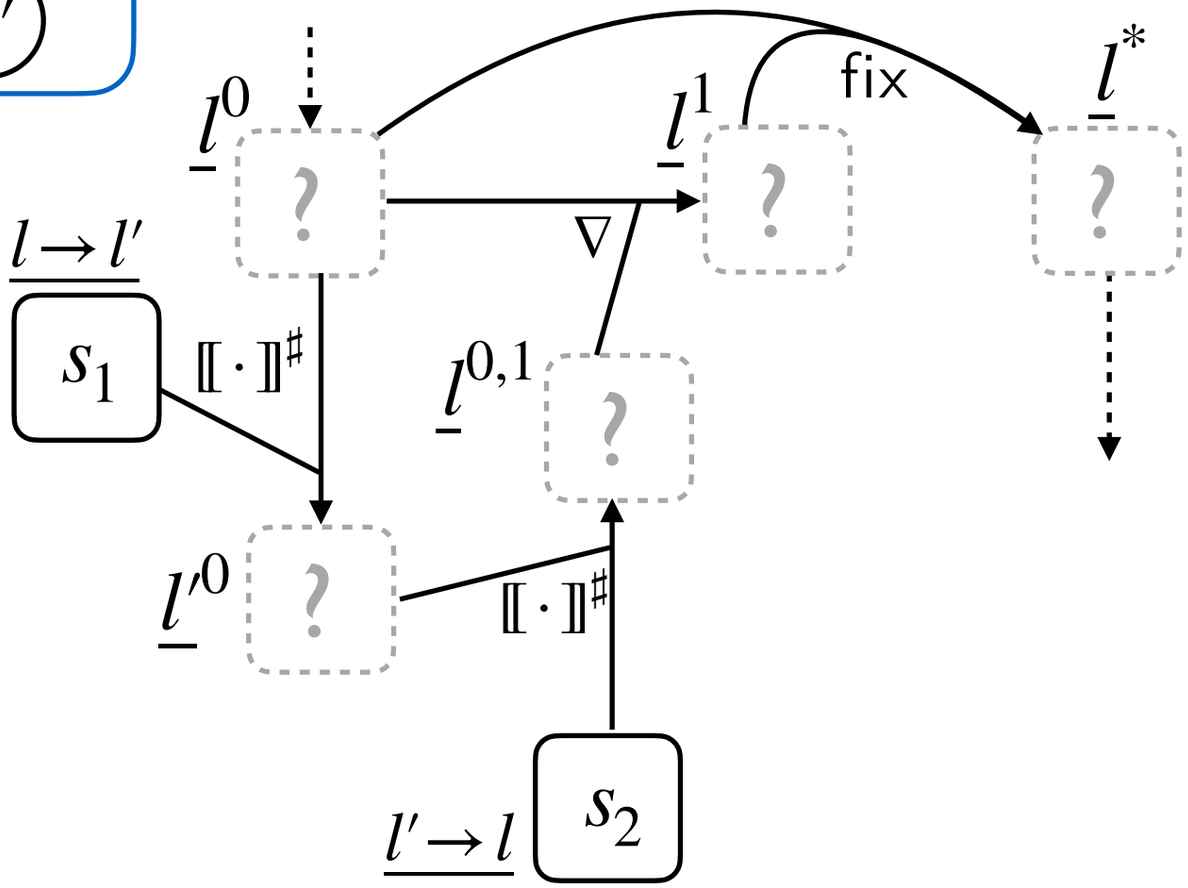


DAIG Fixed-point Computation

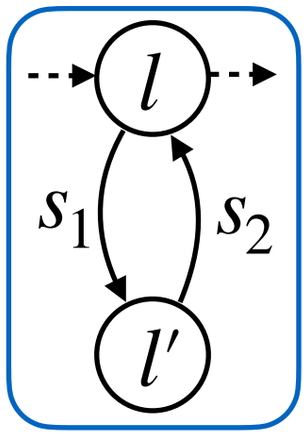


Program snippet

Demand-driven query:
"What is the *fixed-point* abstract state at l ?"



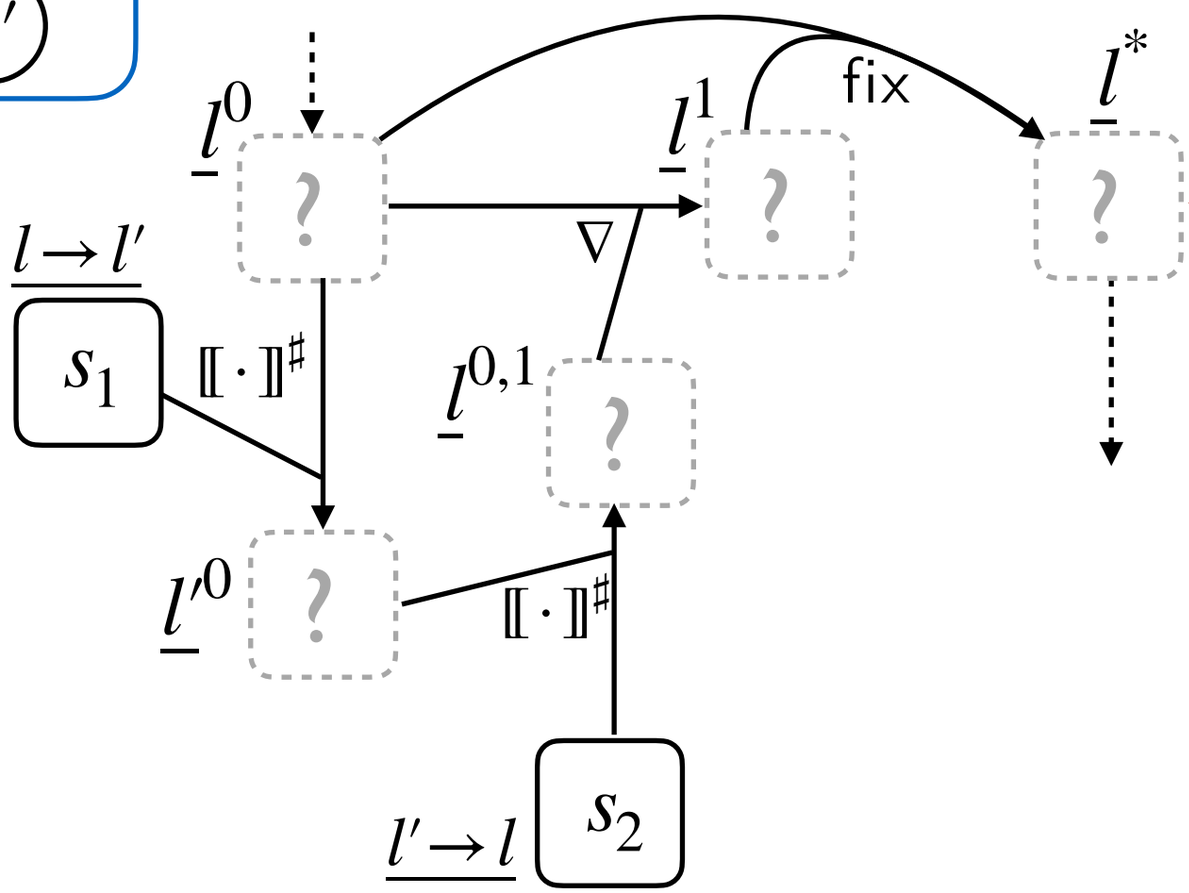
DAIG Fixed-point Computation



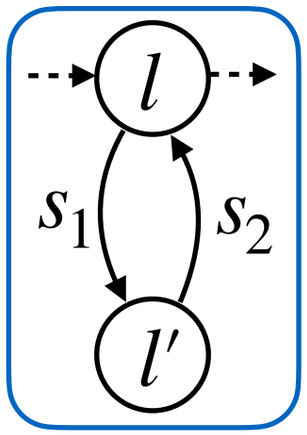
Program snippet

Demand-driven query:
"What is the *fixed-point* abstract state at l ?"

i.e. "What is the value of cell $\underline{l}^*$?"



DAIG Fixed-point Computation

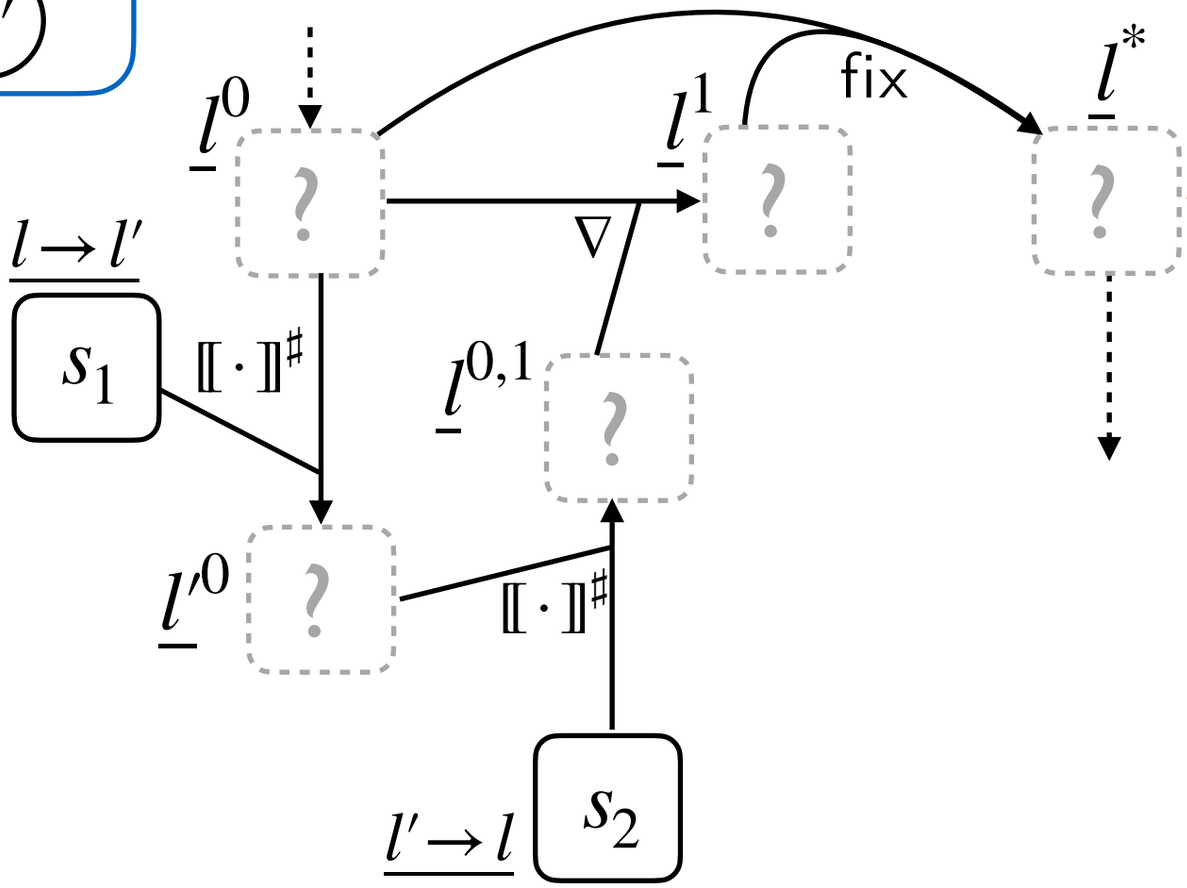


Program snippet

Demand-driven query:

"What is the *fixed-point* abstract state at l ?"

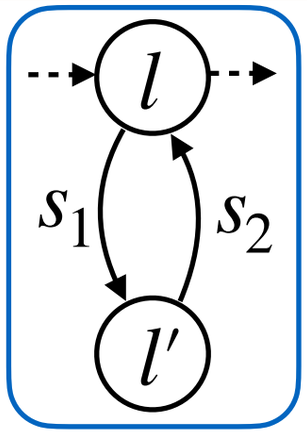
i.e. "What is the value of cell $\underline{l}^*$?"



Compute dependencies as normal…

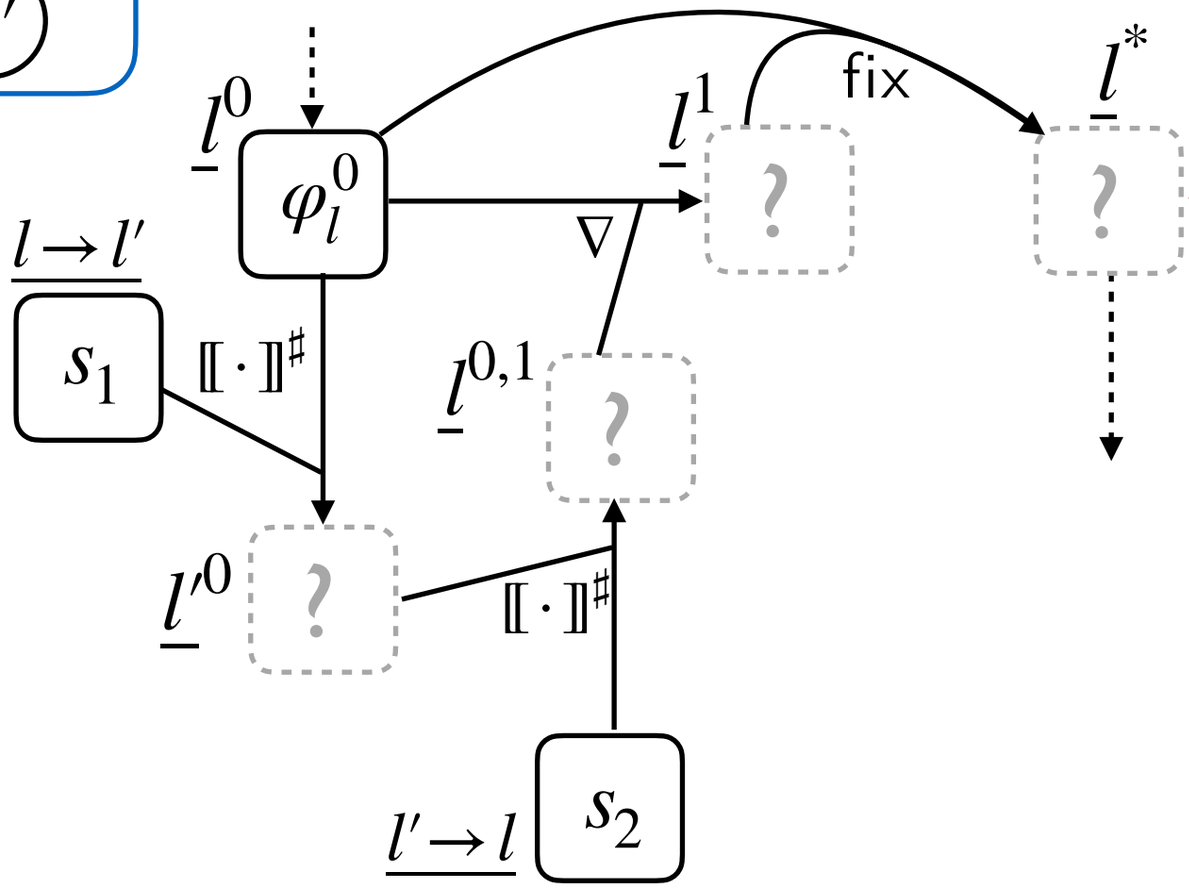
DAIG Fixed-point Computation

Program snippet



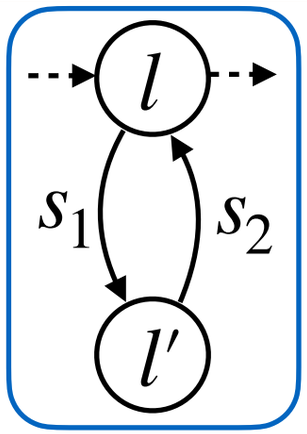
Demand-driven query:
"What is the *fixed-point* abstract state at l ?"

i.e. "What is the value of cell $\underline{l}^*$?"



Compute dependencies as normal…

DAIG Fixed-point Computation

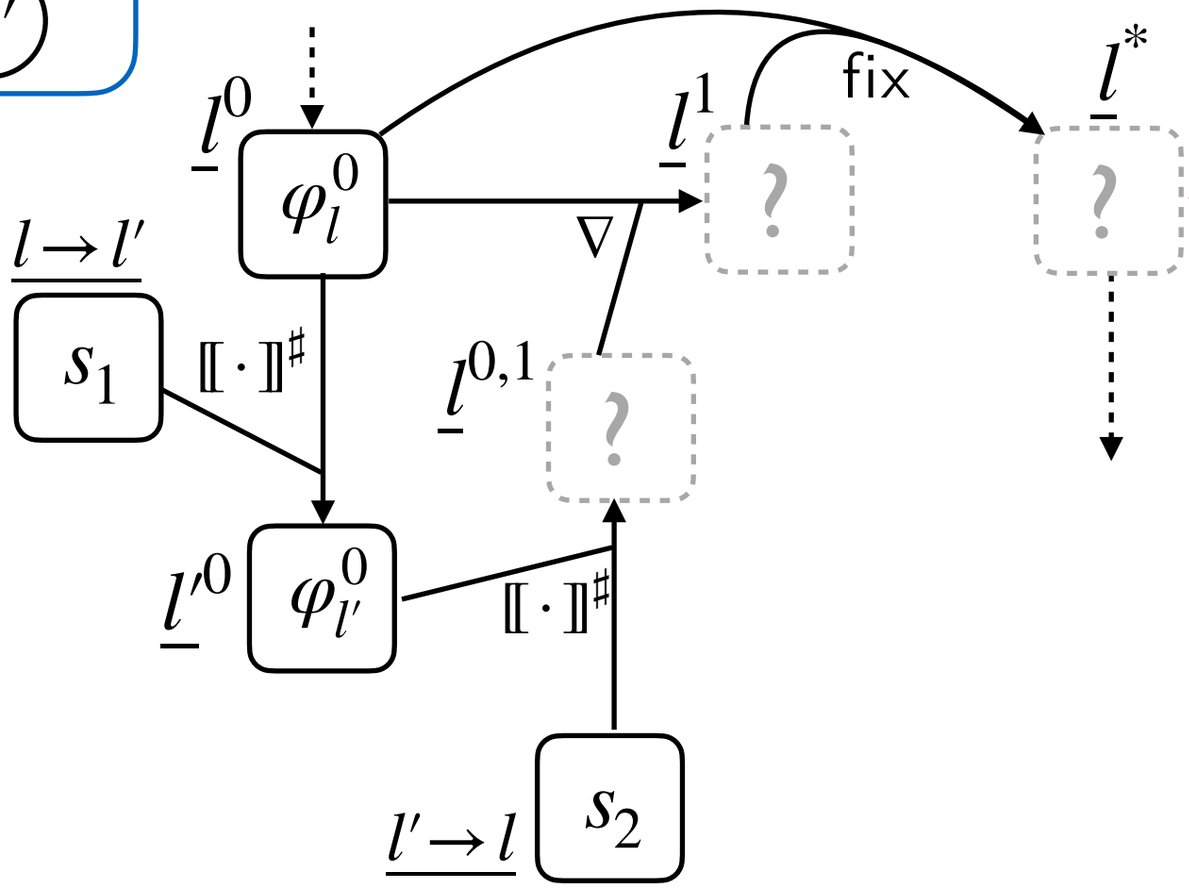


Program snippet

Demand-driven query:

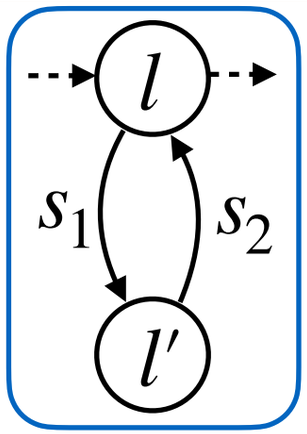
"What is the *fixed-point* abstract state at l ?"

i.e. "What is the value of cell $\underline{l}^*$?"



Compute dependencies as normal…

DAIG Fixed-point Computation

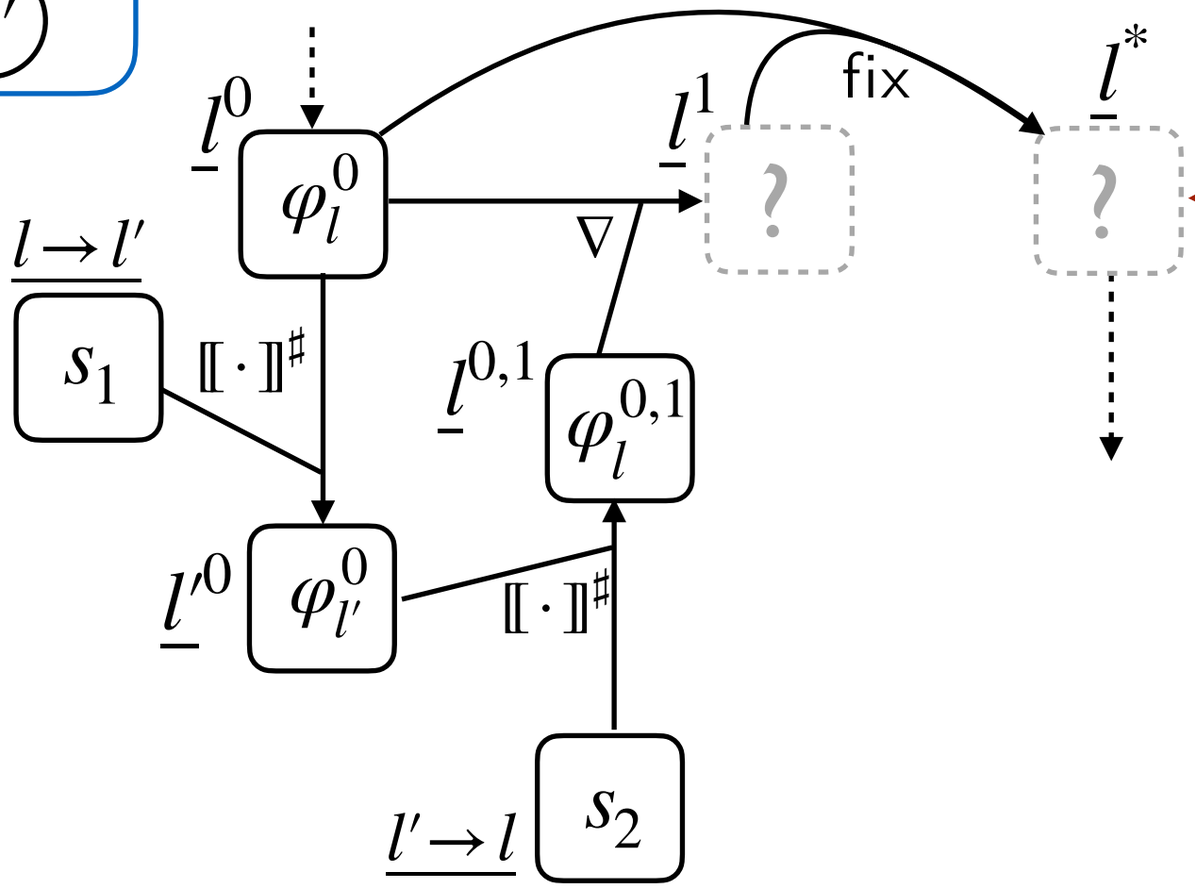


Program snippet

Demand-driven query:

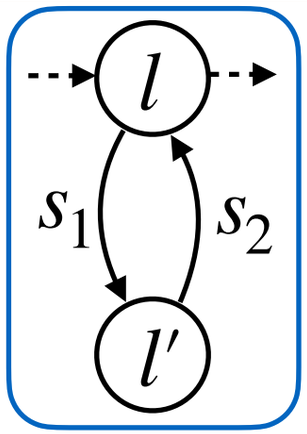
"What is the *fixed-point* abstract state at l ?"

i.e. "What is the value of cell $\underline{l}^*$?"



Compute dependencies as normal…

DAIG Fixed-point Computation

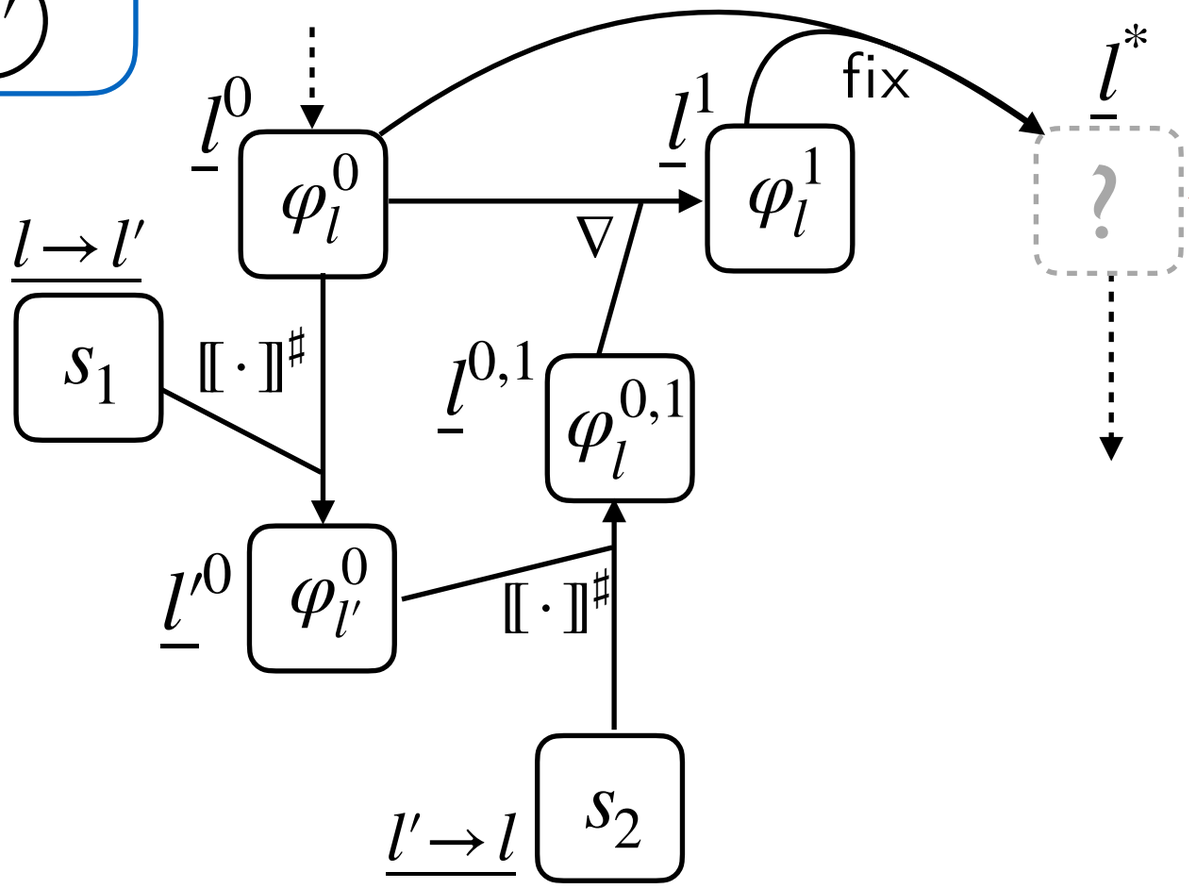


Program snippet

Demand-driven query:

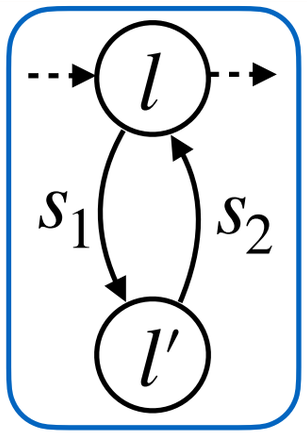
"What is the *fixed-point* abstract state at l ?"

i.e. "What is the value of cell $\underline{l}^*$?"



Compute dependencies as normal…

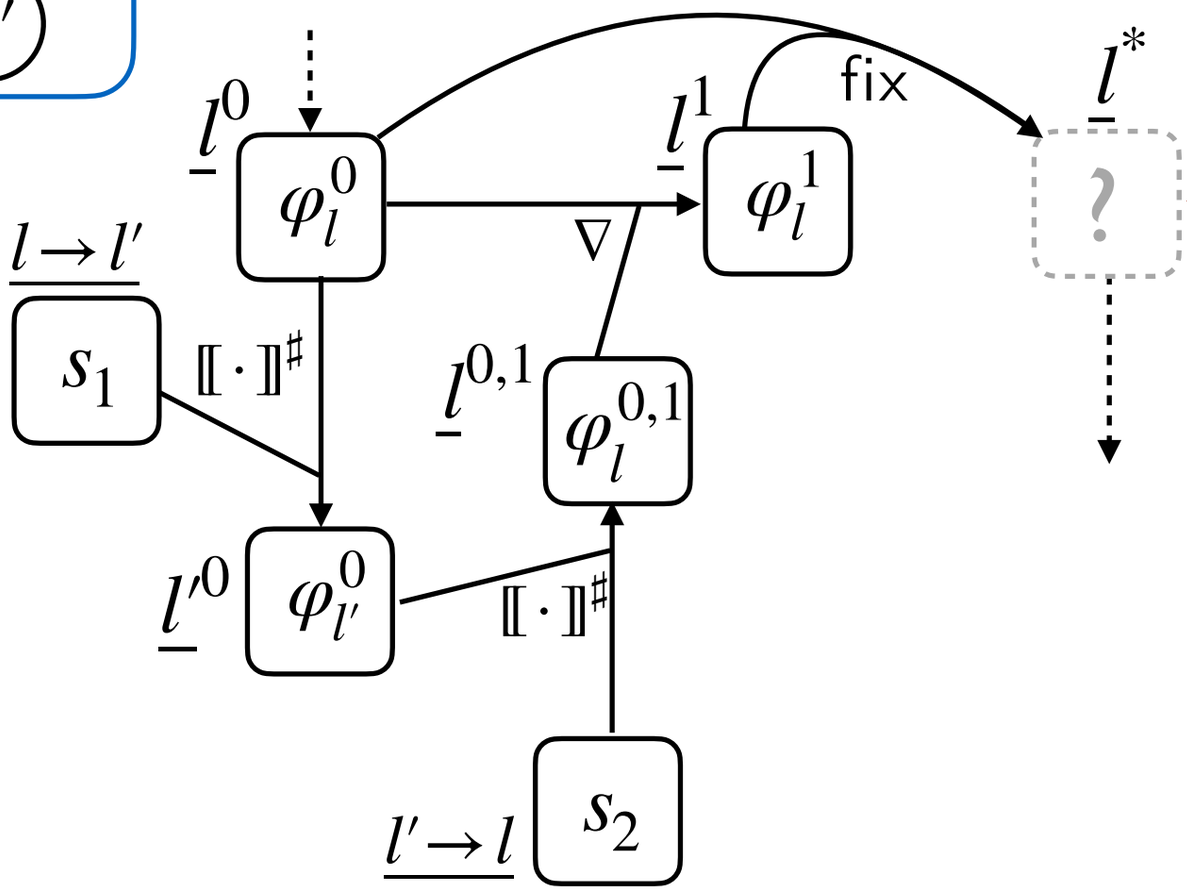
DAIG Fixed-point Computation



Program snippet

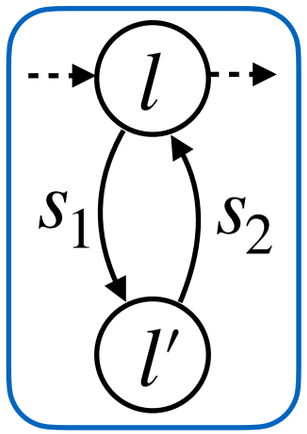
Demand-driven query:
"What is the *fixed-point* abstract state at l ?"

i.e. "What is the value of cell $\underline{l}^*$?"



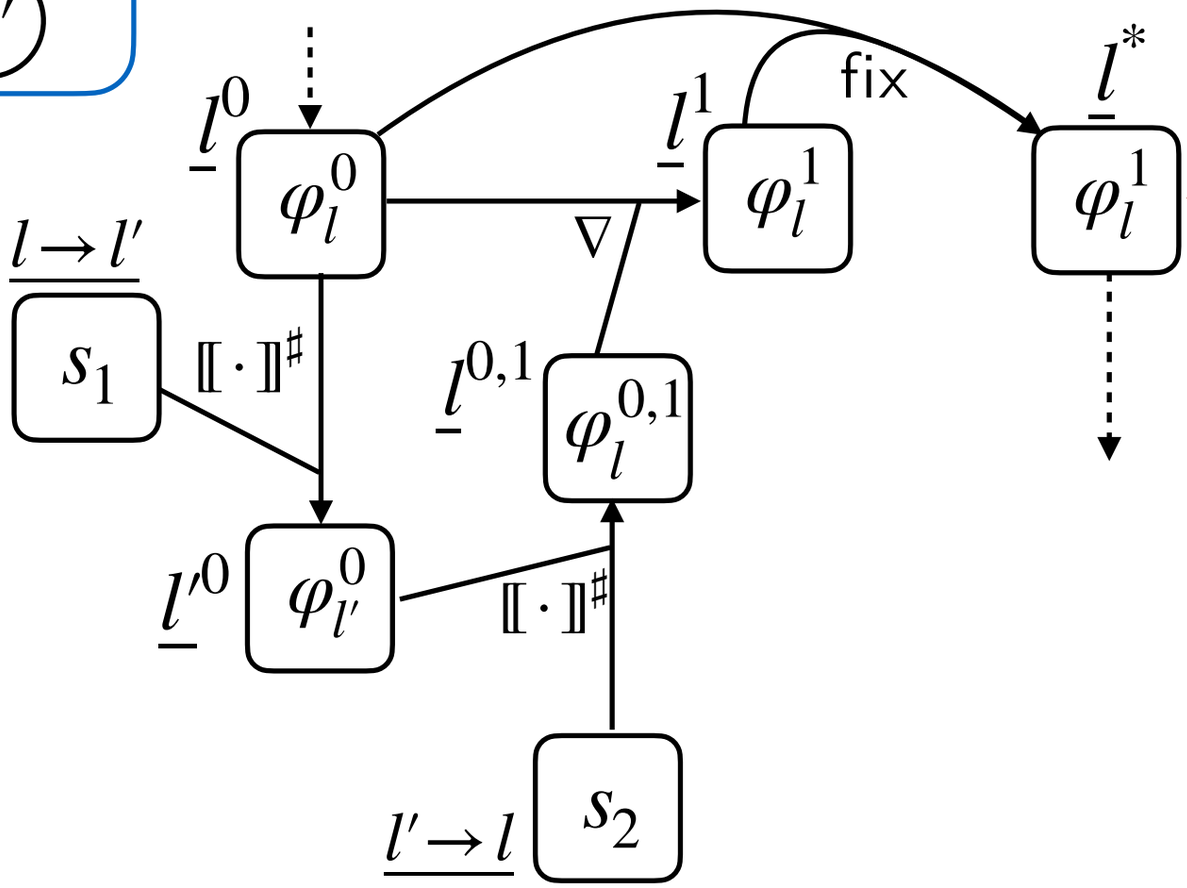
If the zeroth and first abstract iterates are equal (i.e. $\varphi_l^0 = \varphi_l^1$) then that's the fixed point; write to $\underline{l}^*$ and return.

DAIG Fixed-point Computation



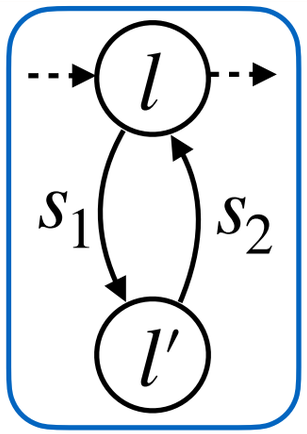
Program snippet

Demand-driven query:
"What is the *fixed-point* abstract state at l ?"
i.e. "What is the value of cell $\underline{l}^*$?"

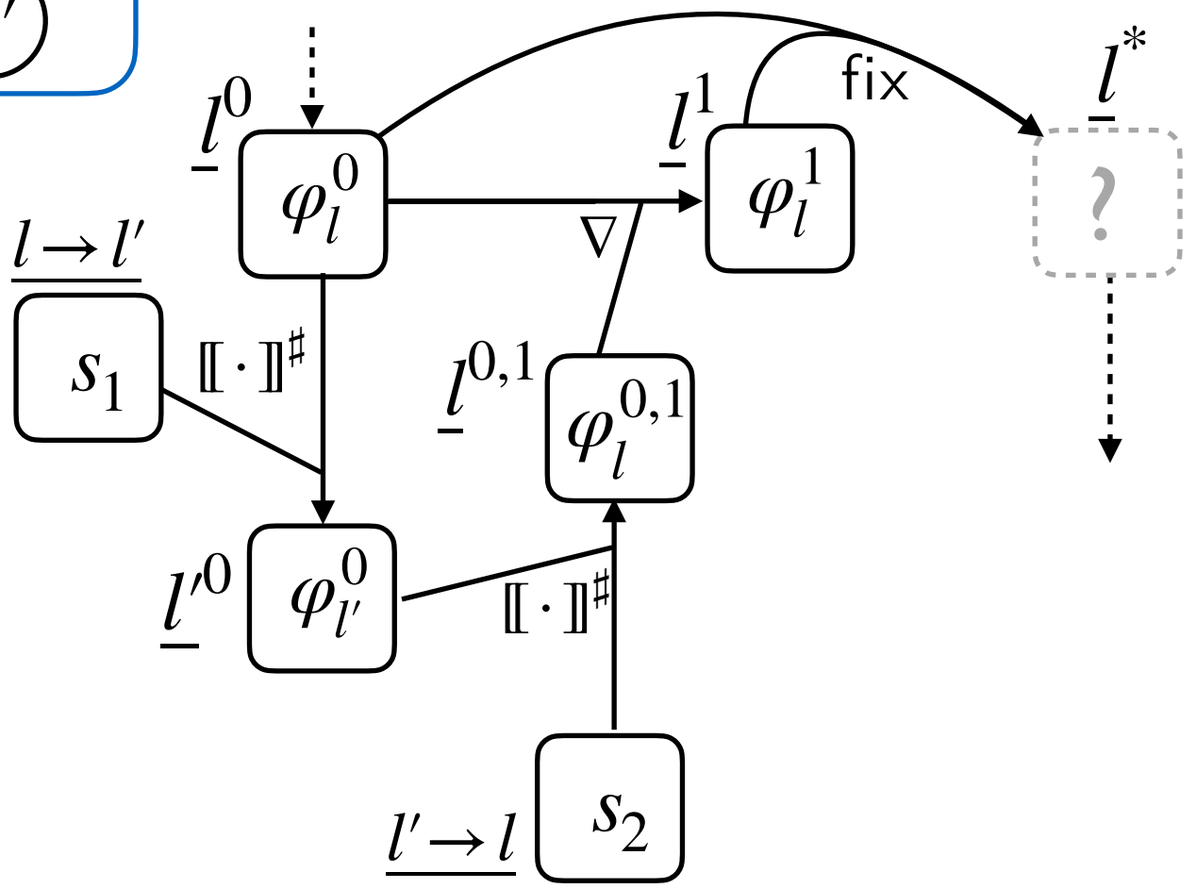


If the zeroth and first abstract iterates are equal (i.e. $\varphi_l^0 = \varphi_l^1$) then that's the fixed point; write to $\underline{l}^*$ and return.

DAIG Fixed-point Computation



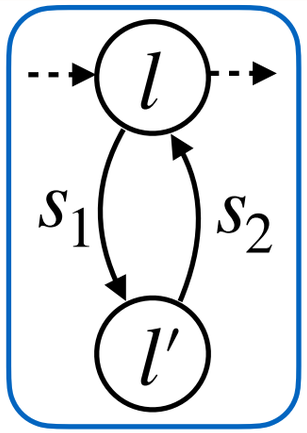
Program snippet



Demand-driven query:
"What is the *fixed-point* abstract state at l ?"

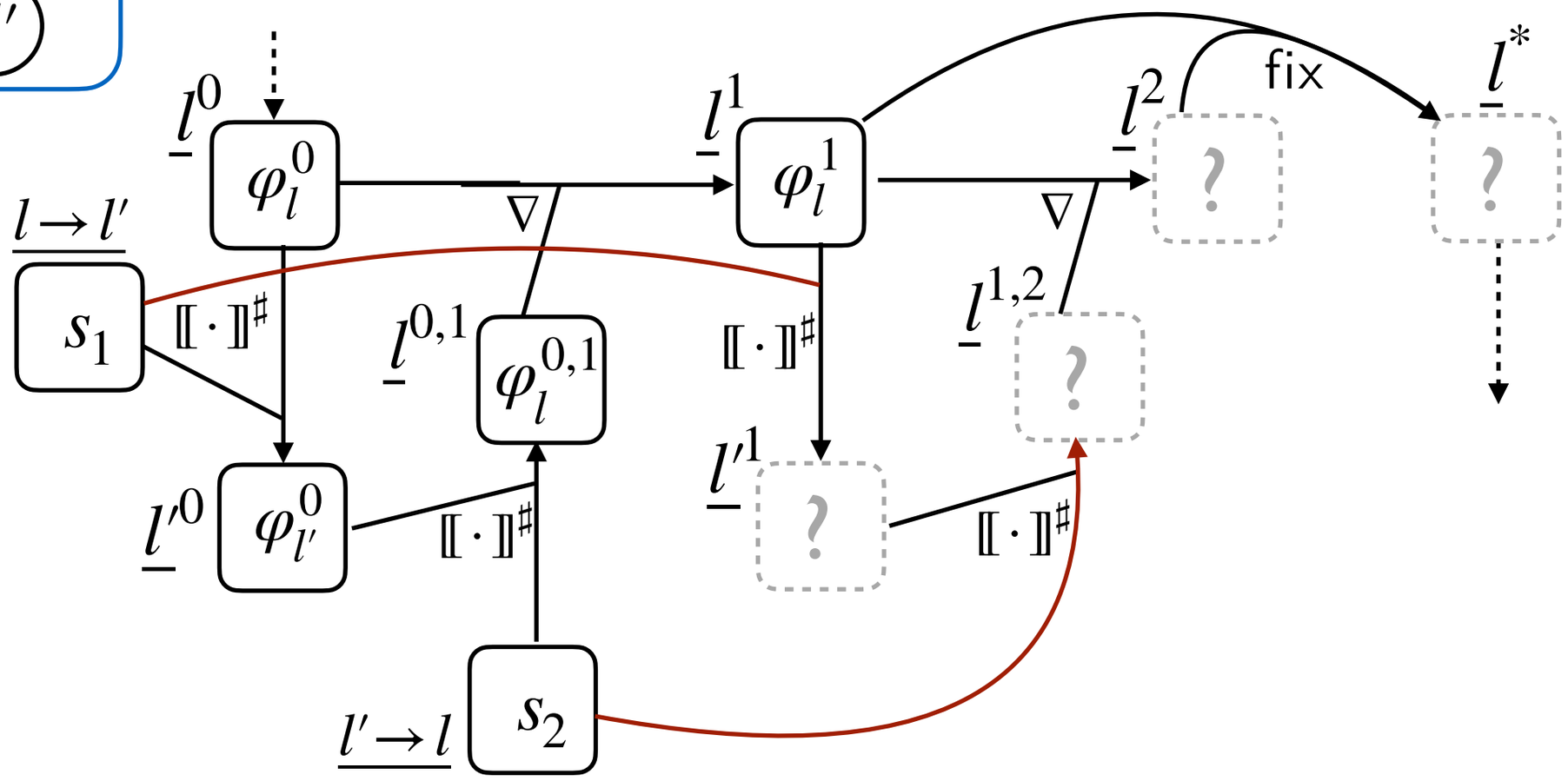
Otherwise, unroll the DAIG's loop representation, re-query, and continue.

DAIG Fixed-point Computation



Program snippet

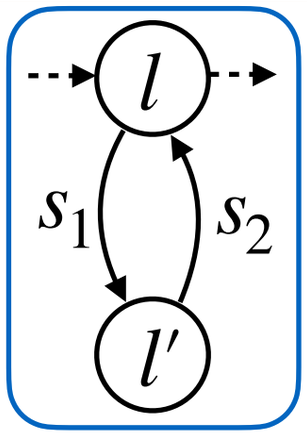
Demand-driven query:
"What is the *fixed-point* abstract state at l ?"



Statement cells not unrolled!

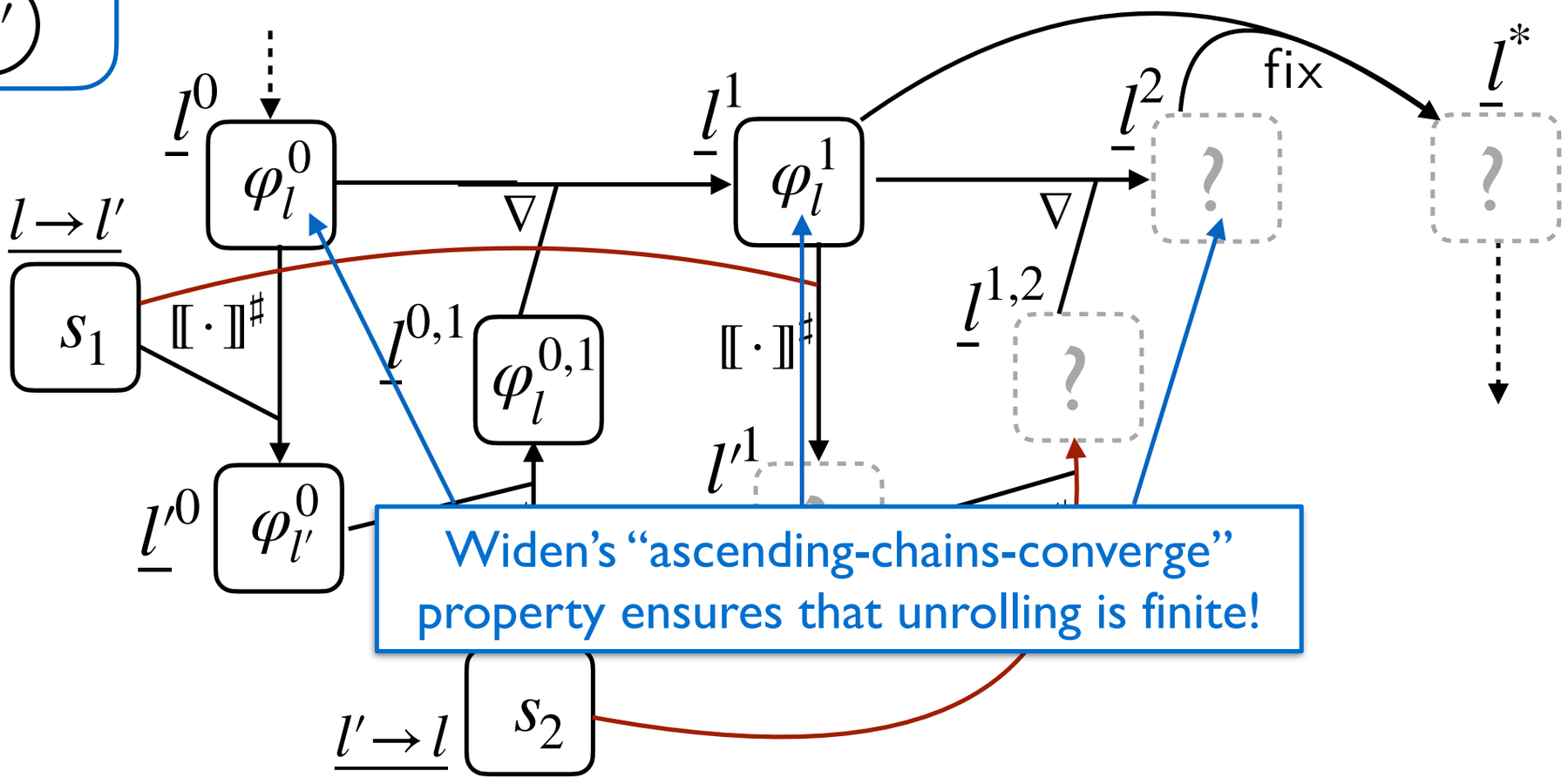
Otherwise, unroll the DAIG's loop representation, re-query, and continue.

DAIG Fixed-point Computation



Program snippet

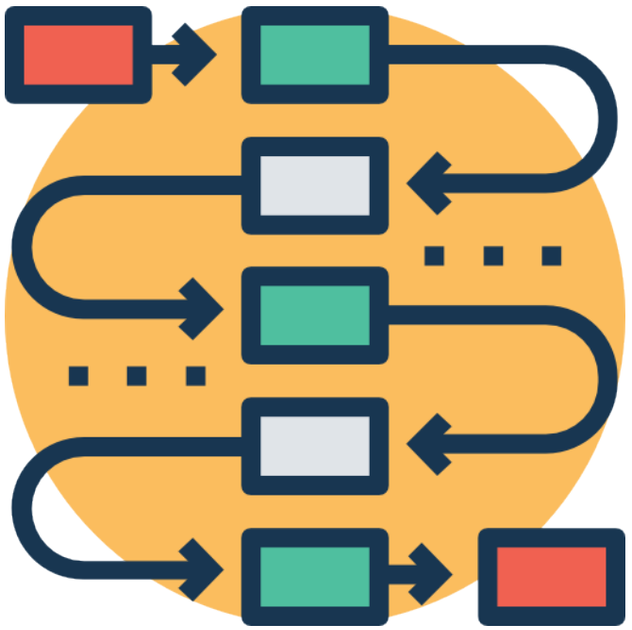
Demand-driven query:
"What is the *fixed-point* abstract state at l ?"



Statement cells not unrolled!

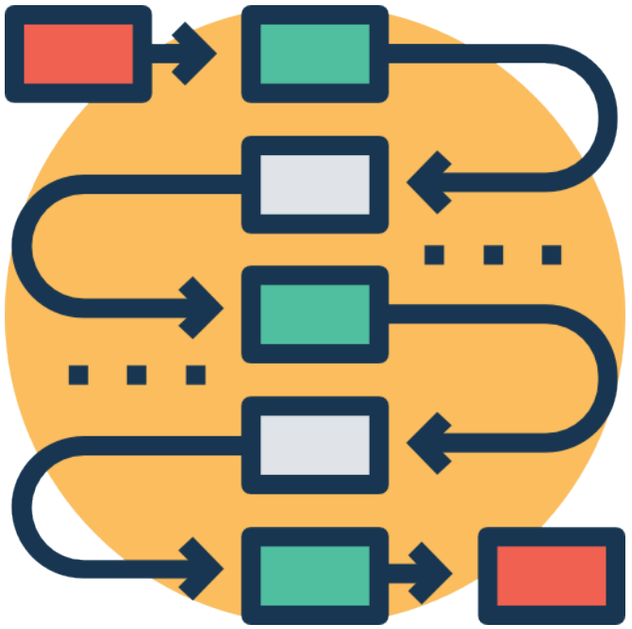
Otherwise, unroll the DAIG's loop representation, re-query, and continue.

DAIG Formalism



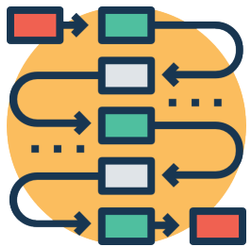
DAIG Formalism

A DAIG is an *explicit* representation of a partially-evaluated abstract interpretation



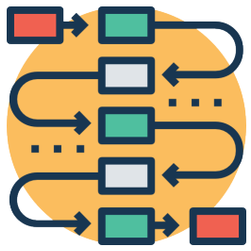
DAIG Formalism

Query evaluation == (small-step) operational semantics



DAIG Formalism

Query evaluation == (small-step) operational semantics

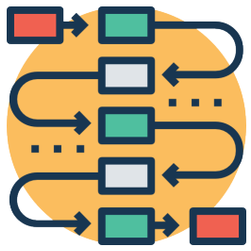


Given an initial DAIG*...

* this elides some details of the actual semantics

DAIG Formalism

Query evaluation == (small-step) operational semantics



$$\vdash n \Rightarrow v$$

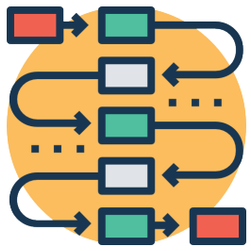
Given an initial DAIG*...

... a query for the value named n yields v...

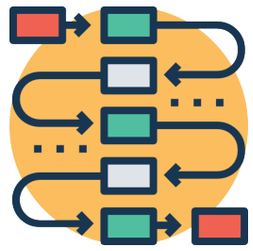
* this elides some details of the actual semantics

DAIG Formalism

Query evaluation == (small-step) operational semantics



$$\vdash n \Rightarrow v \;;$$



Given an initial DAIG*...

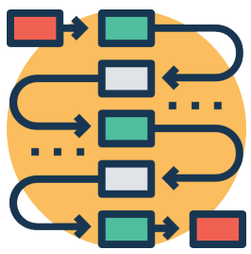
... a query for the value named n yields v...

... and an updated DAIG*.

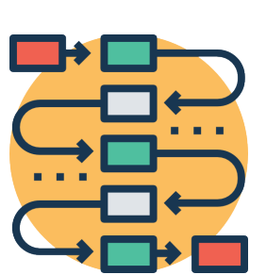
* this elides some details of the actual semantics

DAIG Formalism

Query evaluation == (small-step) operational semantics



$$\vdash n \Rightarrow v \;;$$



Given an initial DAIG*...

... a query for the value named n yields v...

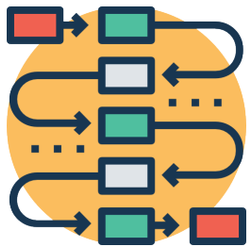
... and an updated DAIG*.

Edit handling/change propagation is *also* a small-step operational semantics

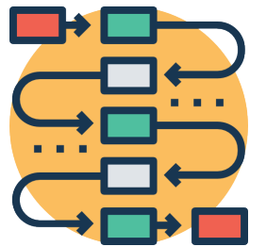
* this elides some details of the actual semantics

DAIG Formalism

Query evaluation == (small-step) operational semantics



$$\vdash n \Rightarrow v \;;$$

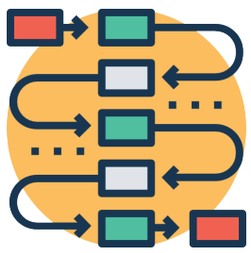


Given an initial DAIG*...

... a query for the value named n yields v...

... and an updated DAIG*.

Edit handling/change propagation is *also* a small-step operational semantics

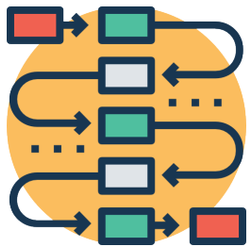


Given an initial DAIG...

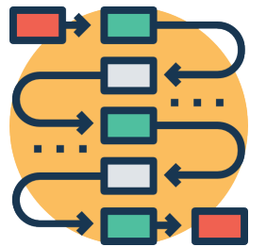
* this elides some details of the actual semantics

DAIG Formalism

Query evaluation == (small-step) operational semantics



$$\vdash n \Rightarrow v \;;$$

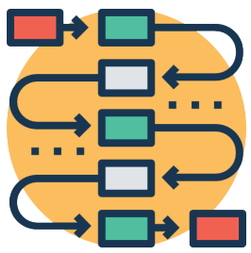


Given an initial DAIG*...

... a query for the value named n yields v...

... and an updated DAIG*.

Edit handling/change propagation is *also* a small-step operational semantics



$$\vdash n \Leftarrow v$$

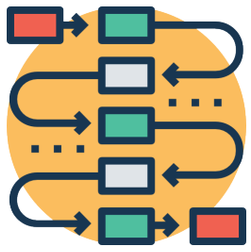
Given an initial DAIG...

... an edit that writes value v to cell n ...

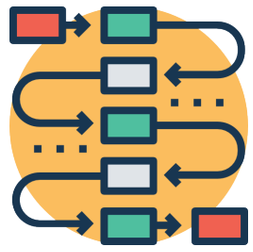
* this elides some details of the actual semantics

DAIG Formalism

Query evaluation == (small-step) operational semantics



$$\vdash n \Rightarrow v \;;$$

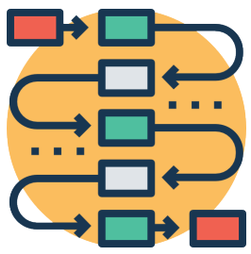


Given an initial DAIG*...

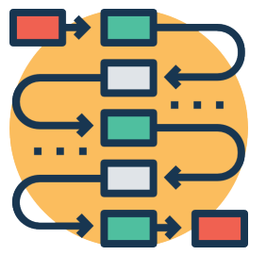
... a query for the value named n yields v...

... and an updated DAIG*.

Edit handling/change propagation is *also* a small-step operational semantics



$$\vdash n \Leftarrow v \;;$$



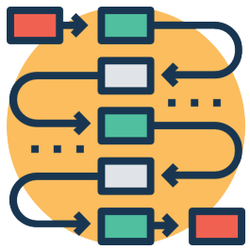
Given an initial DAIG...

... an edit that writes value v to cell n ...

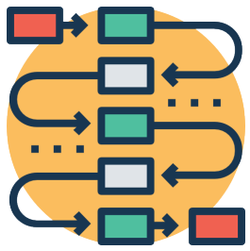
... yields an updated DAIG.

* this elides some details of the actual semantics

Termination

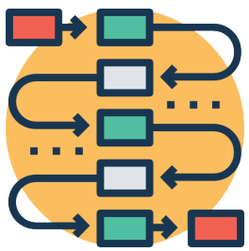


Termination



If this initial DAIG is a valid abstract interpretation state for some program...

Termination

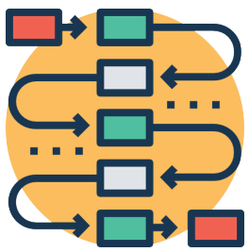


$\vdash n$

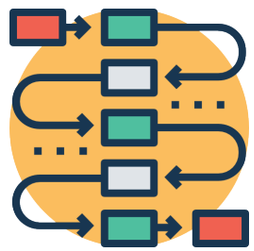
If this initial DAIG is a valid abstract interpretation state for some program...

... then a query for any n therein ...

Termination



$$\vdash n \Rightarrow v \; ;$$



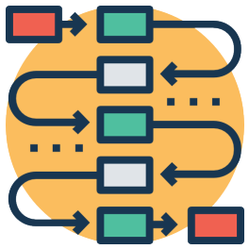
$'$

If this initial DAIG is a valid abstract interpretation state for some program...

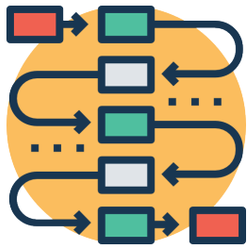
... then a query for any n therein ...

... will terminate with some value v and an updated DAIG.

From-Scratch Consistency

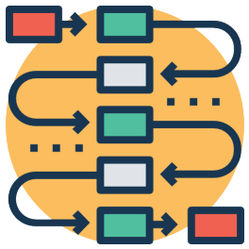


$$\vdash n \Rightarrow v \; ;$$

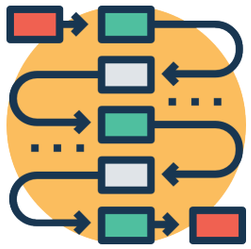


$'$

From-Scratch Consistency



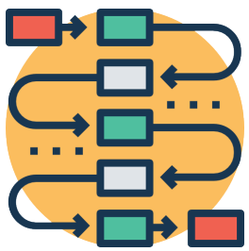
$$\vdash n \Rightarrow v \;;$$



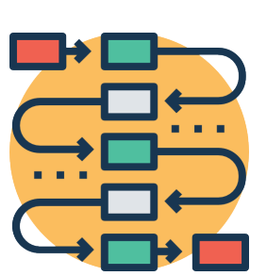
$'$

If the current DAIG is a valid abstract interpretation state for some program…

From-Scratch Consistency



$$\vdash n \Rightarrow v \; ;$$

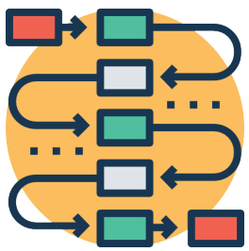


$'$

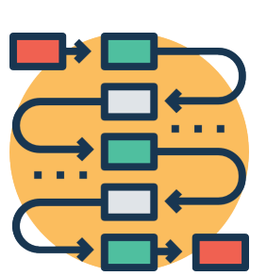
If the current DAIG is a valid abstract interpretation state for some program...

... and n is the name of the abstract state at a program location l ...

From-Scratch Consistency



$$\vdash n \Rightarrow v \; ;$$

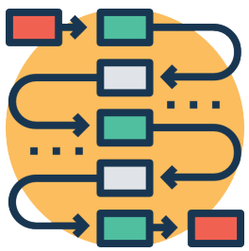


If the current DAIG is a valid abstract interpretation state for some program...

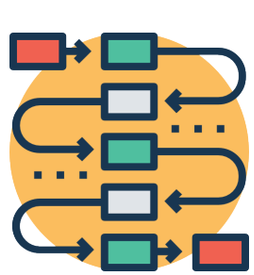
... and n is the name of the abstract state at a program location l ...

... then v is *precisely* the same value that a batch abstract interpreter would compute at l.

From-Scratch Consistency



$$\vdash n \Rightarrow v \; ;$$



$'$

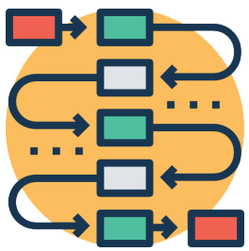
If the current DAIG is a valid abstract interpretation state for some program...

... and n is the name of the abstract state at a program location l ...

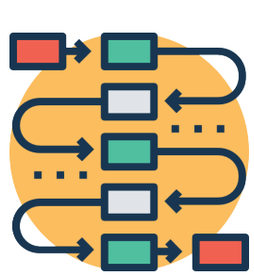
... then v is *precisely* the same value that a batch abstract interpreter would compute at l.

No loss of precision due to incrementality/demand!

From-Scratch Consistency



$$\vdash n \Rightarrow v \; ;$$



$'$

If the current DAIG is a valid abstract interpretation state for some program...

... and n is the name of the abstract state at a program location l ...

... then v is *precisely* the same value that a batch abstract interpreter would compute at l.

No loss of precision due to incrementality/demand!

Corollary: DAIG query results are **sound**.

Expressivity

Does a DAIG-based analysis framework support rich analysis domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Expressivity

Does a DAIG-based analysis framework support rich analysis domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Prototype Implementation: github.com/cuplv/dai

- ~2500 lines of OCaml code
- Parametric in a statement language and abstract domain

Expressivity

Does a DAIG-based analysis framework support rich analysis domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Prototype Implementation: github.com/cuplv/dai

- ~2500 lines of OCaml code
- Parametric in a statement language and abstract domain

```
module type Domain = sig

  type t

  val init : t
  val interpret : stmt -> t -> t
  val implies : t -> t -> bool
  val join : t -> t -> t
  val widen : t -> t -> t

  (* elided: equal, hash, etc. *)
end
```

Expressivity

Does a DAIG-based analysis framework support rich analysis domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Prototype Implementation: github.com/cuplv/dai

- ~2500 lines of OCaml code
- Parametric in a statement language and abstract domain

abstract states $\Sigma^\sharp$

initial abstract state φ_0

transfer function $[\![\cdot]\!]^\sharp$

partial order $\sqsubseteq$

join $\sqcup$

widening ∇

```
module type Domain = sig

  type t

  val init : t
  val interpret : stmt -> t -> t
  val implies : t -> t -> bool
  val join : t -> t -> t
  val widen : t -> t -> t

  (* elided: equal, hash, etc. *)
end
```

Expressivity

Does a DAIG-based analysis framework support rich analysis domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Prototype Implementation: github.com/cuplv/dai

- ~2500 lines of OCaml code
- Parametric in a statement language and abstract domain

abstract states $\Sigma^\sharp$

initial abstract state φ_0

transfer function $[\![\cdot]\!]^\sharp$

partial order $\sqsubseteq$

join $\sqcup$

widening ∇

```
module type Domain = sig

  type t

  val init : t
  val interpret : stmt -> t -> t
  val implies : t -> t -> bool
  val join : t -> t -> t
  val widen : t -> t -> t

  (* elided: equal, hash, etc. *)
end
```

Domain implementer doesn't need to reason about incrementality or demand!

Expressivity

Do DAIGs support rich abstract domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Expressivity

Do DAIGs support rich abstract domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Interval Analysis

- Abstract values $[x, y]$ model integers $\{i \,|\, x \leq i \leq y\}$
- Used to verify array accesses in-bounds in a JS data structure library.

Expressivity

Do DAIGs support rich abstract domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Interval Analysis

- Abstract values $[x, y]$ model integers $\{i \mid x \leq i \leq y\}$
- Used to verify array accesses in-bounds in a JS data structure library.

Shape Analysis

- Separation logic formulae over linked-list-segment primitive **lseg**$(\hat{x}, \hat{y})$
- Used to verify memory safety of linked-list **append**, **reverse**, etc.

Expressivity

Do DAIGs support rich abstract domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Interval Analysis

- Abstract values $[x, y]$ model integers $\{i \,|\, x \leq i \leq y\}$
- Used to verify array accesses in-bounds in a JS data structure library.

Shape Analysis

- Separation logic formulae over linked-list-segment primitive **lseg**$(\hat{x}, \hat{y})$
- Used to verify memory safety of linked-list **append**, **reverse**, etc.

Octagon Analysis

- Invariants of the form $\pm x \pm y \leq c$
- Used in scalability experiments

Expressivity

Do DAIGs support rich abstract domains that cannot be handled by existing incremental and/or demand-driven frameworks?

Interval Analysis

- Abstract values $[x, y]$ model integers $\{i \mid x \leq i \leq y\}$
- Used to verify array accesses in-bounds in a JS data structure library.

Shape Analysis

- Separation logic formulae over linked-list-segment primitive **lseg**$(\hat{x}, \hat{y})$
- Used to verify memory safety of linked-list **append**, **reverse**, etc.

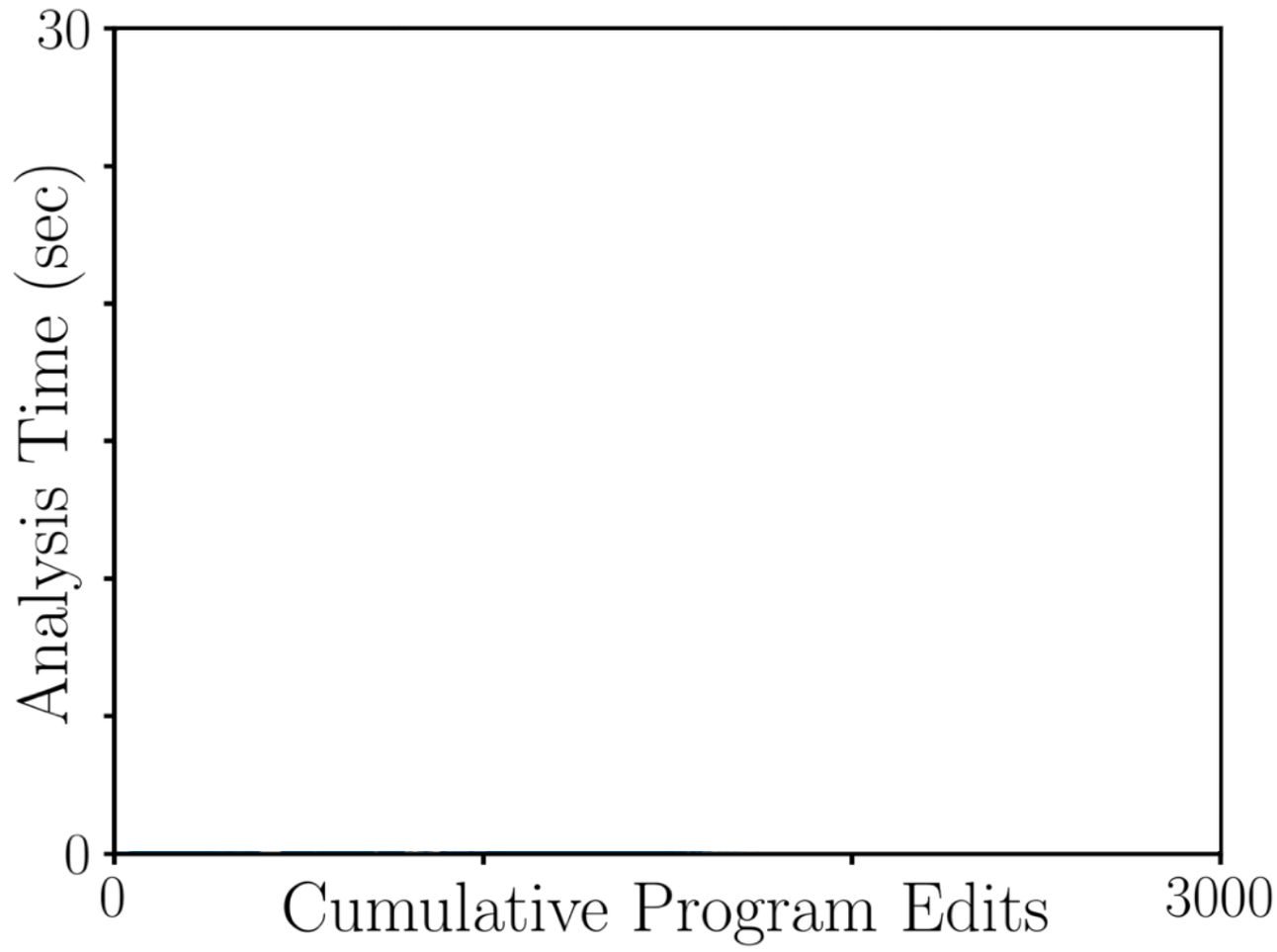
Octagon Analysis

- Invariants of the form $\pm x \pm y \leq c$
- Used in scalability experiments

Intervals & Octagons built with APRON — optimized open-source numerical domains in C

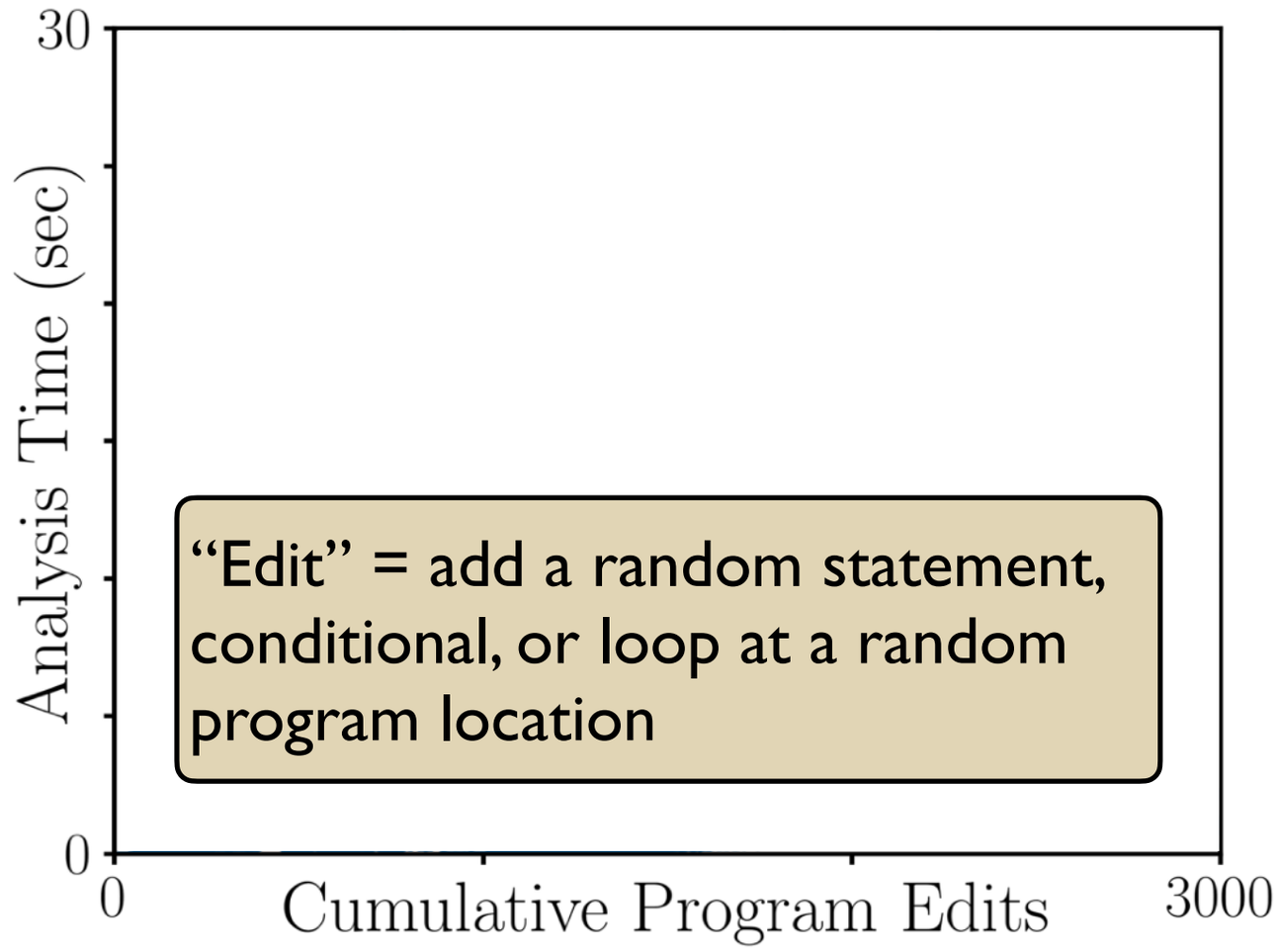
Scalability

For these rich analysis domains, what degree of performance improvement can be obtained by performing incremental and/or demand-driven analysis?



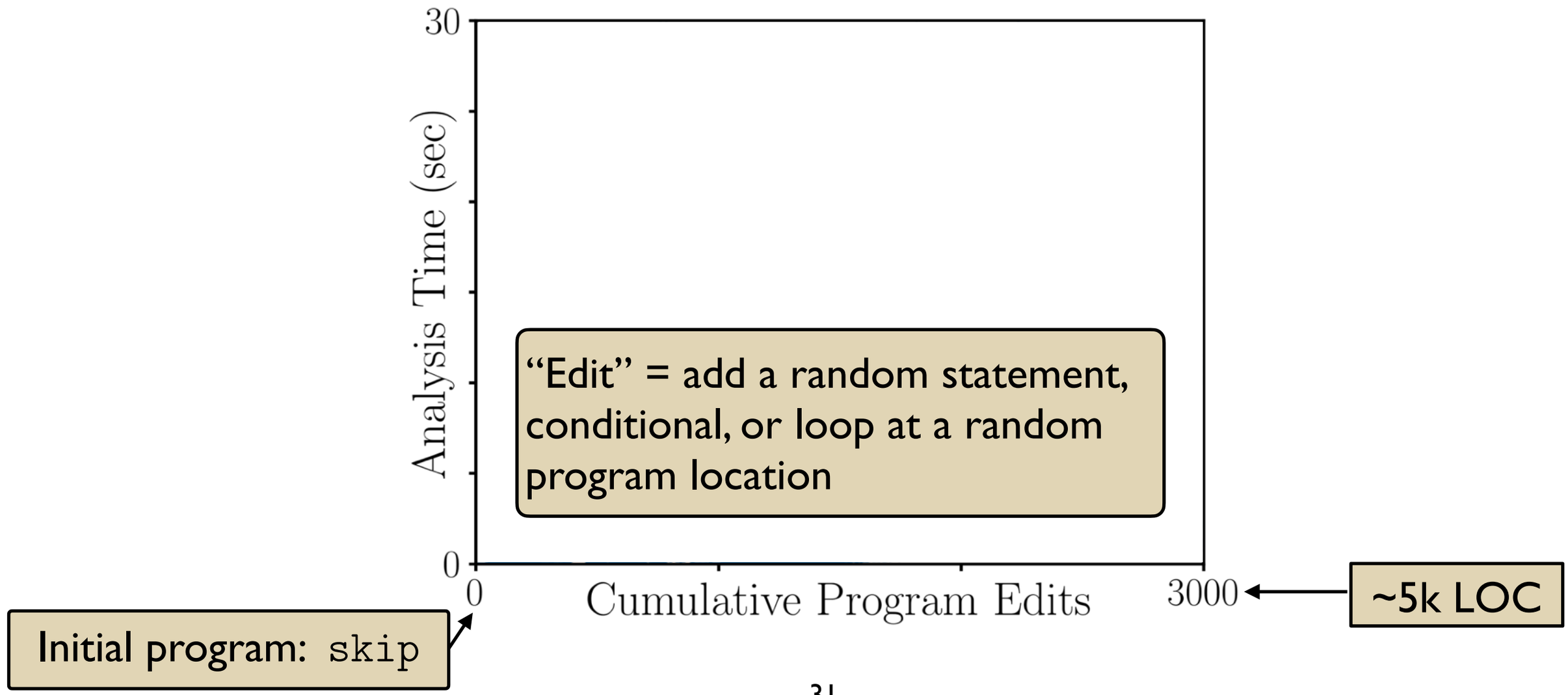
Scalability

For these rich analysis domains, what degree of performance improvement can be obtained by performing incremental and/or demand-driven analysis?



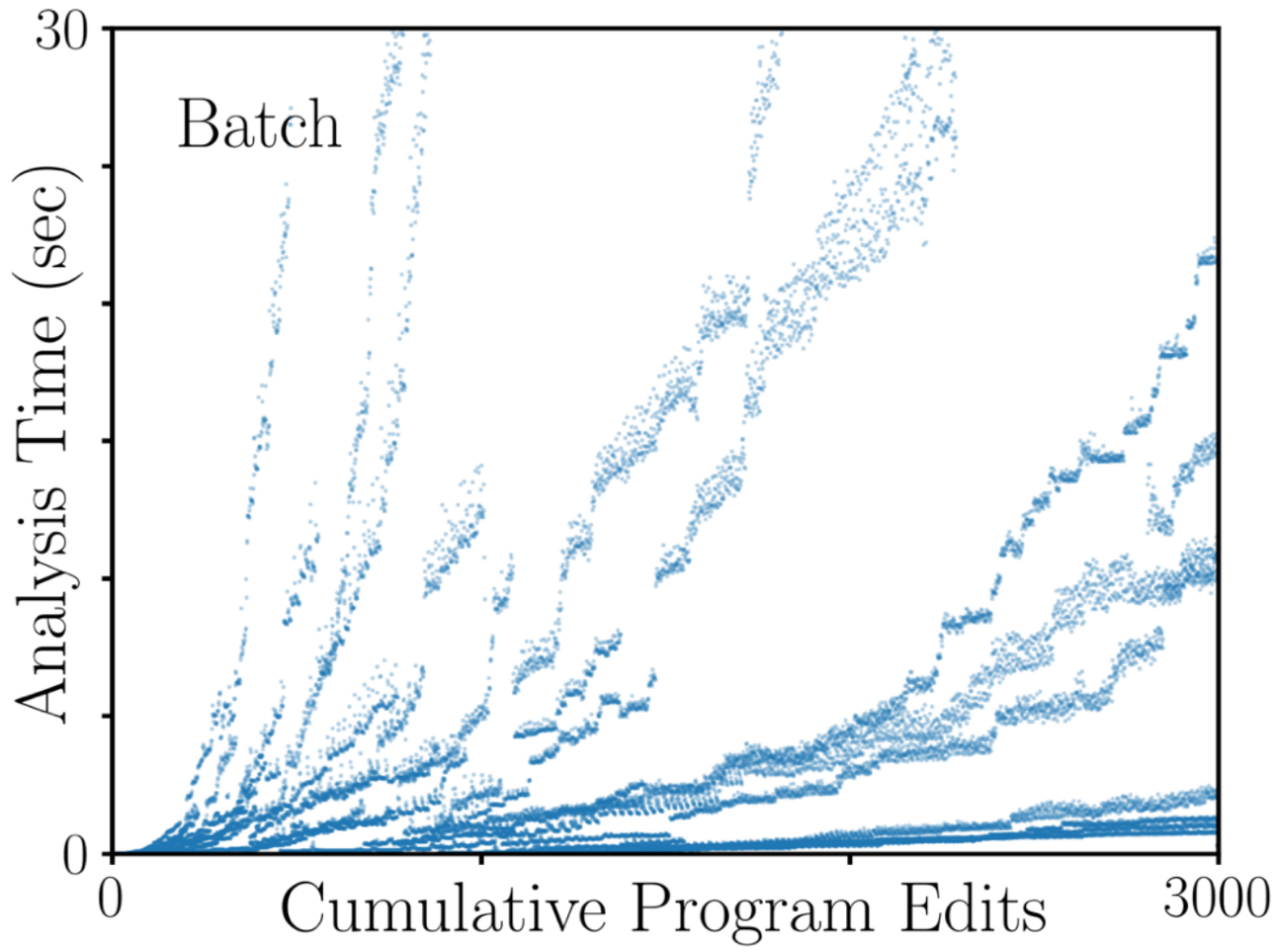
Scalability

For these rich analysis domains, what degree of performance improvement can be obtained by performing incremental and/or demand-driven analysis?

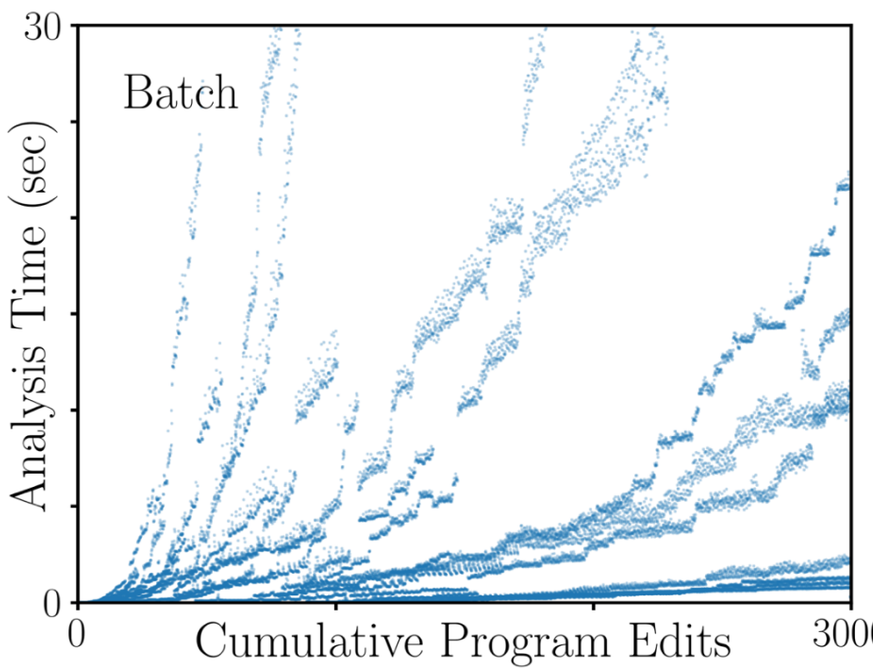


Scalability

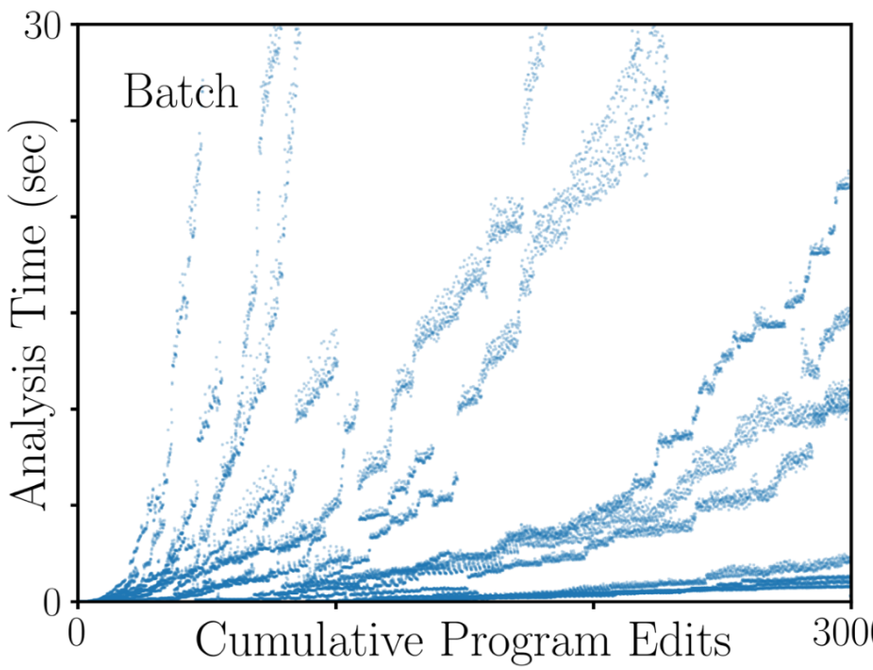
For these rich analysis domains, what degree of performance improvement can be obtained by performing incremental and/or demand-driven analysis?

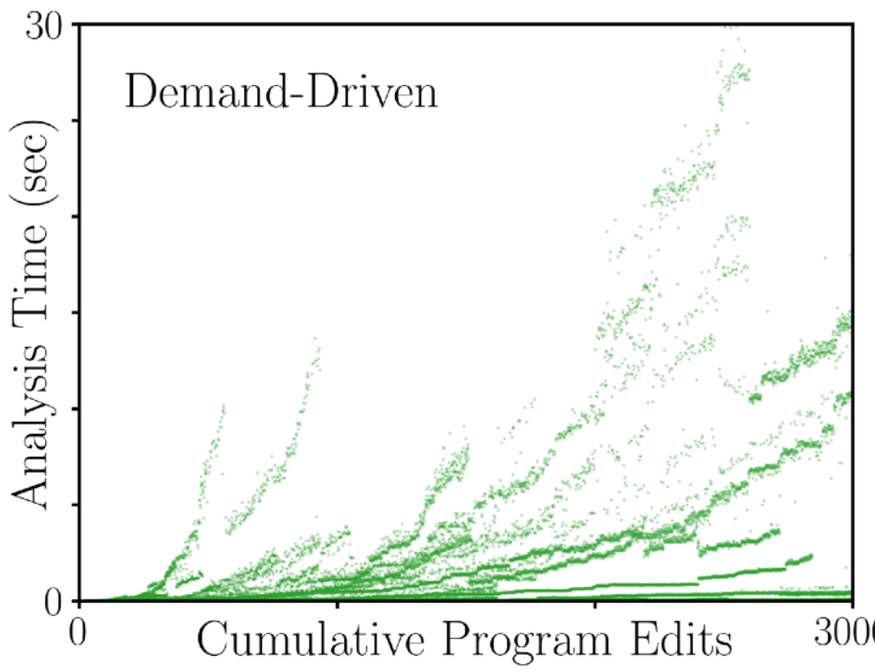


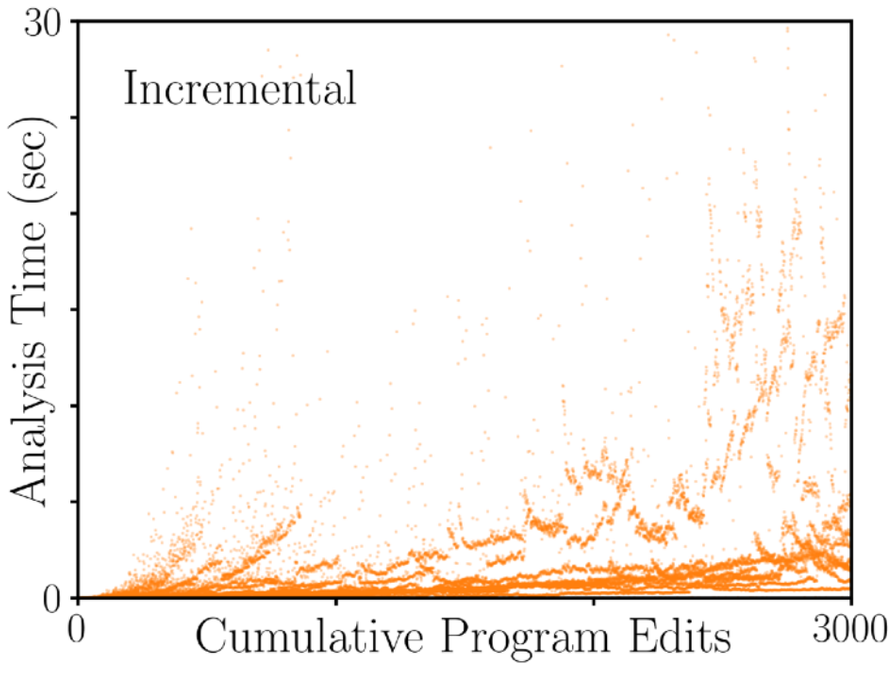
Scalability



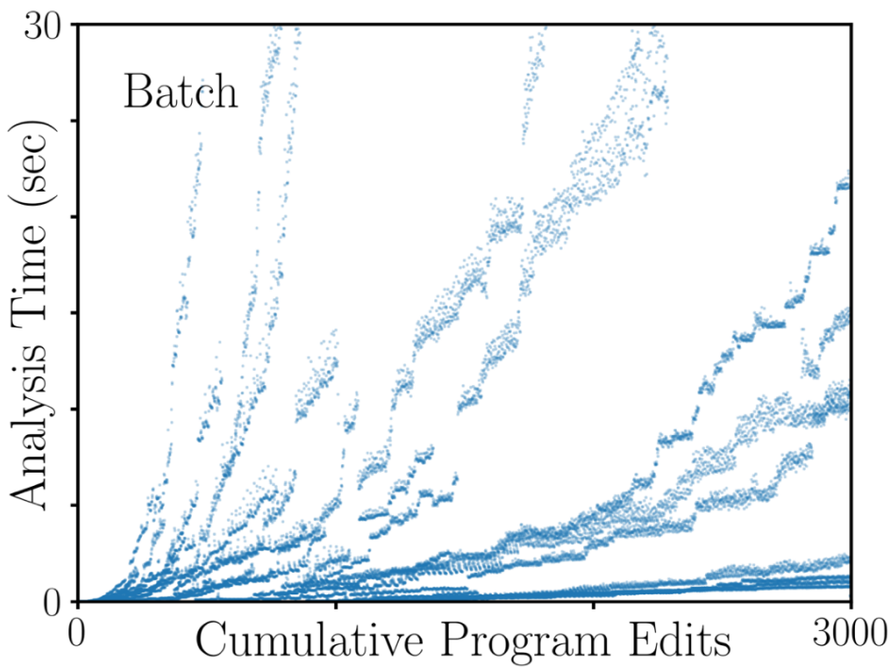
Scalability

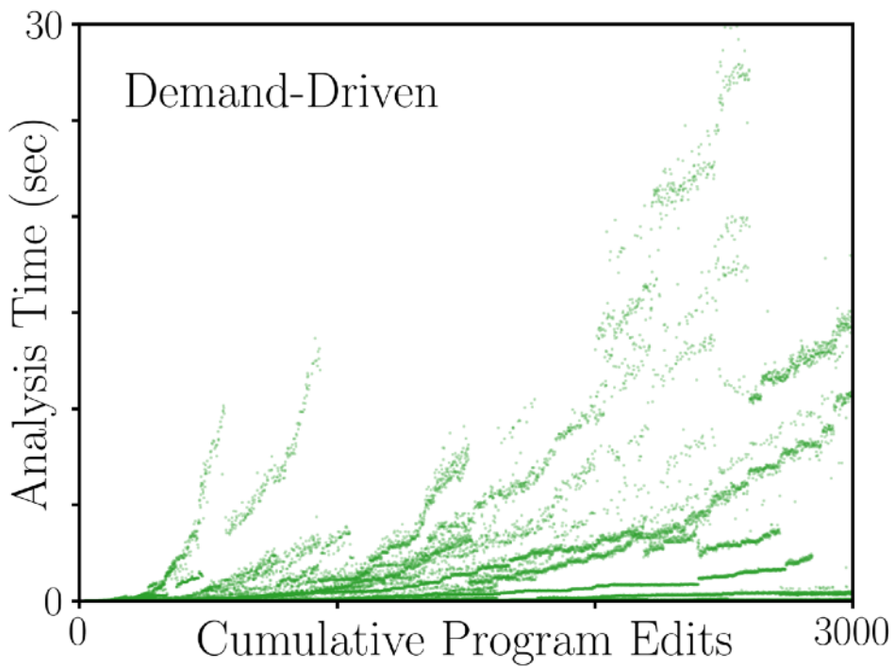


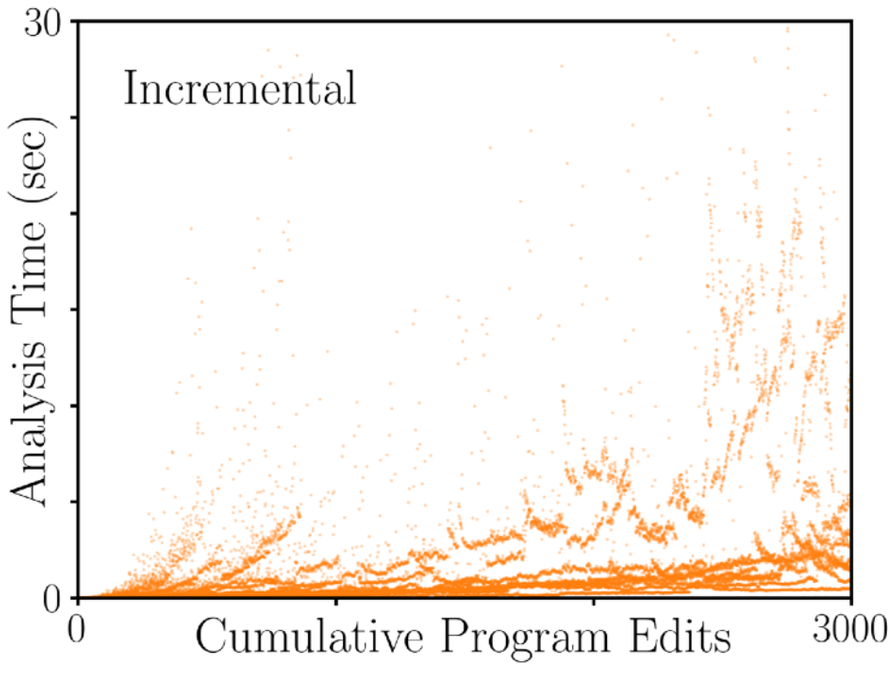


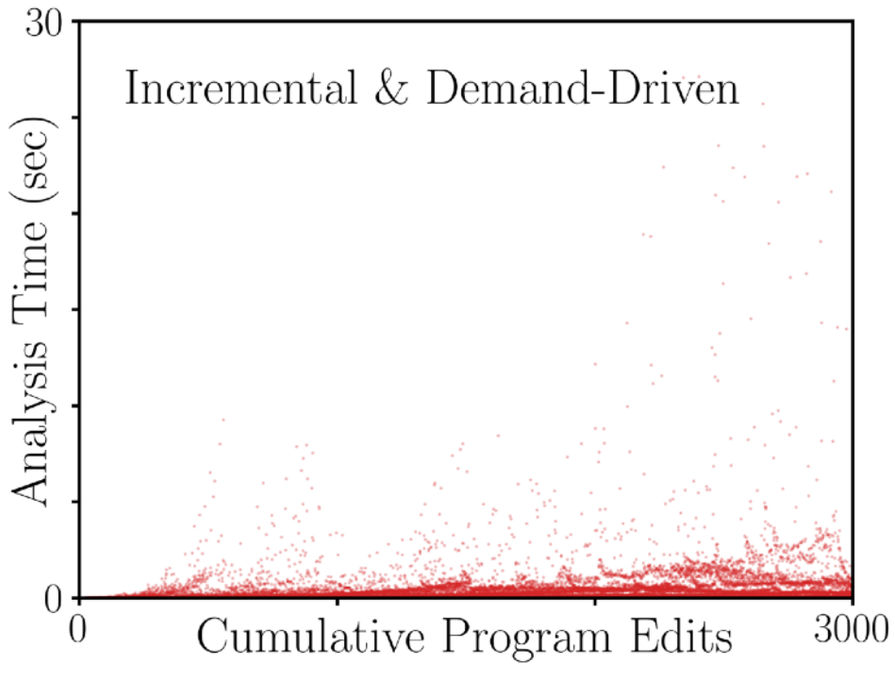


Scalability



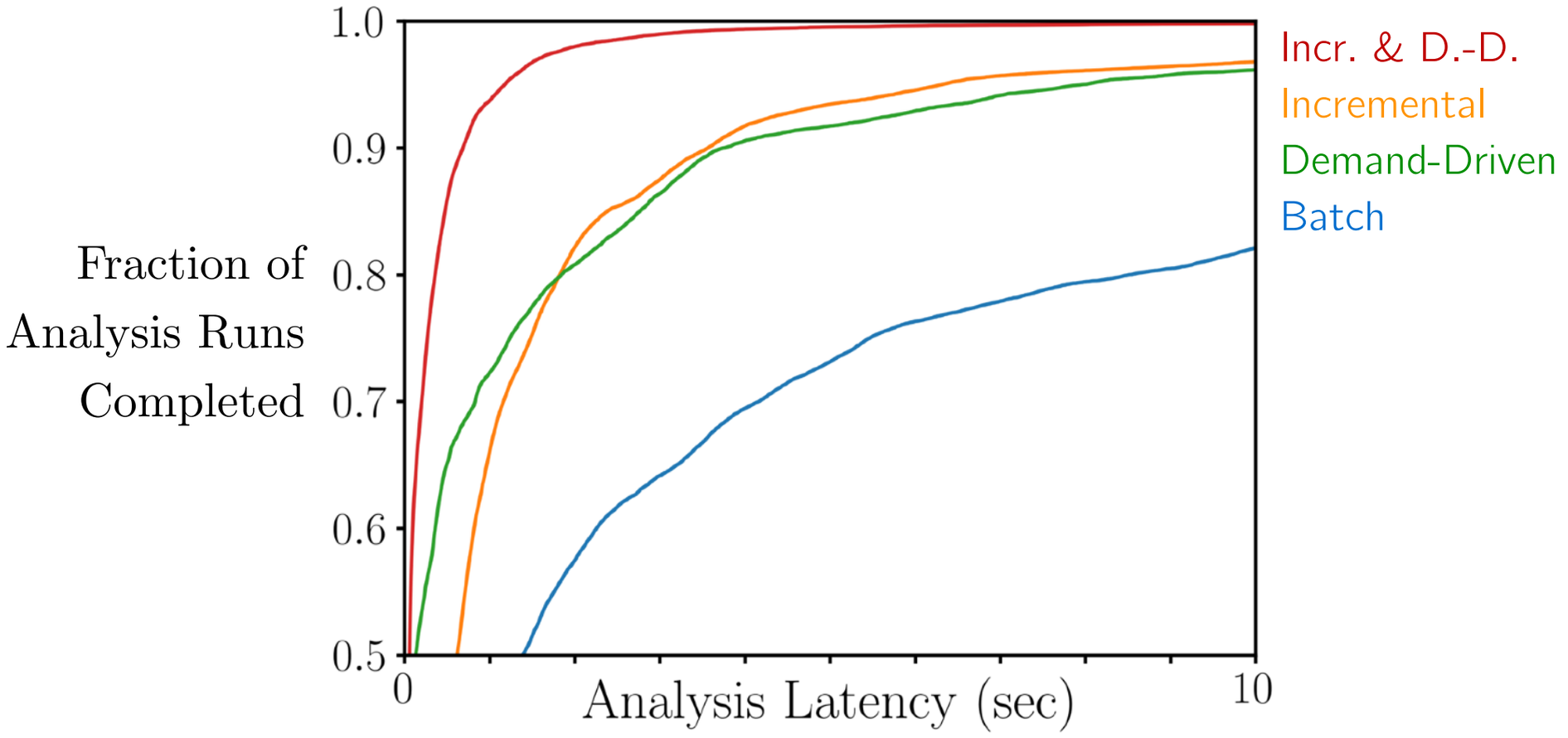






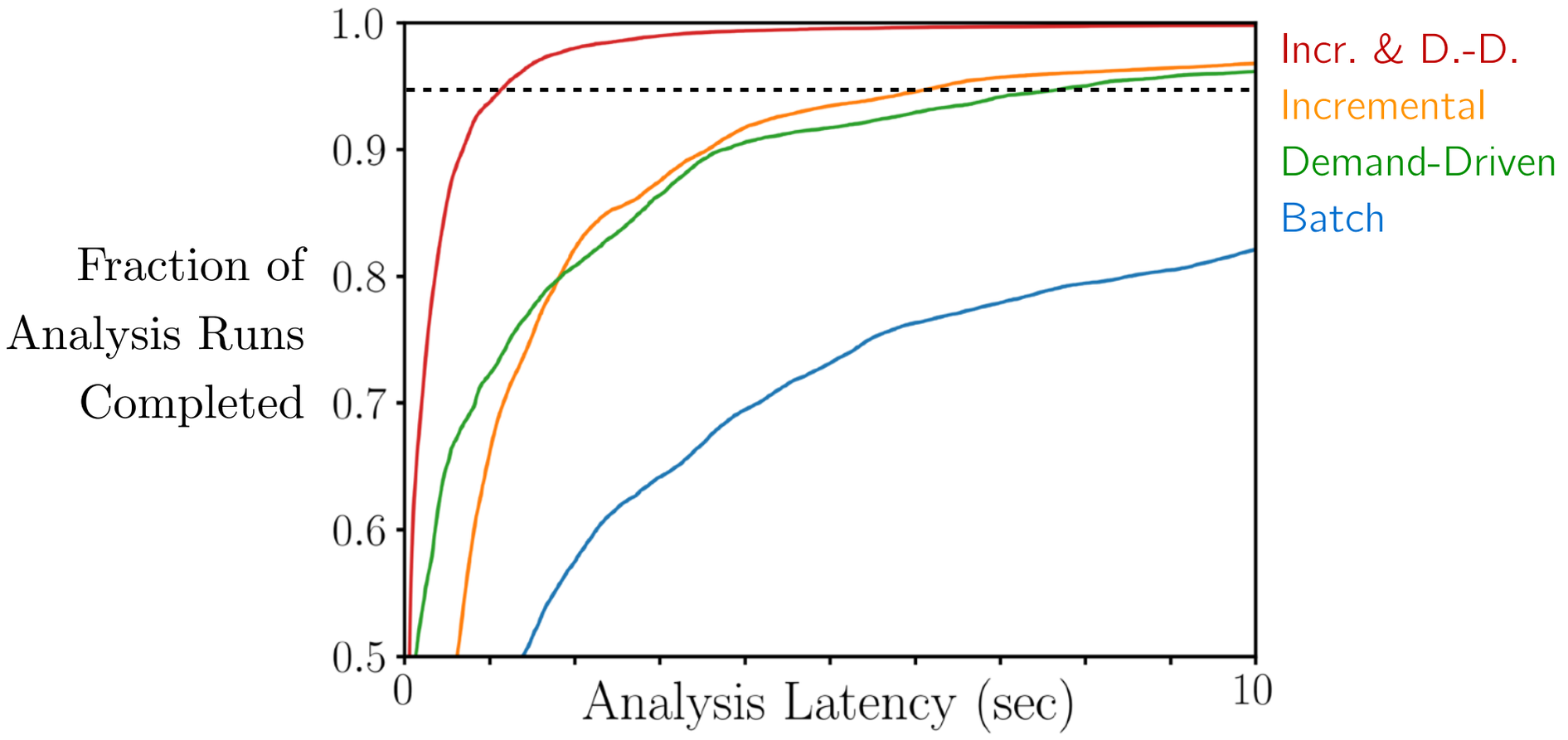
Scalability

The *combination* of incrementality and demand consistently obtains lower latencies than either incrementality or demand alone.



Scalability

The *combination* of incrementality and demand consistently obtains lower latencies than either incrementality or demand alone.



p95: 1.2s ; 6.3s ; 7.9s ; 36.2s

Thanks for watching!

Check out our paper or come chat at the Q&A for more details.

Conclusion:

- Batch whole-program analysis is too costly to support real-time developer interaction, but existing incremental and demand-driven analyses are often limited in expressivity or granularity
- By leveraging graph-based incremental computation techniques, we define an engine for incremental and demand-driven evaluation of *arbitrary* abstract interpreters (and prove it sound & from-scratch consistent)

CUPLV

Benno Stein
University of Colorado Boulder
benno.stein@colorado.edu

PLDI '21