Thesis Proposal: Hybrid Resource-Bound Analysis of Programs

Long Pham[1]
Committee: Jan Hoffmann[1], Feras Saad[1], Matt Fredrikson[1], François Pottier[2]
[1]Carnegie Mellon University
[2]Inria Paris
January 17th, 2025

Resource Analysis

Goal of resource analysis

Infer a worst-case bound of the cost of a program as a function of inputs

Program

```
fun quicksort x
= <body>
```

resource analysis

Worst-case cost bound

$|x|(|x|-1)/2$

Resource metric

Time Memory Energy

Applications of Resource Analysis

1. Prevent algorithmic complexity attacks by inferring worst-case resource usage / inputs

2. Estimate job size for job scheduling in cloud computing

3. **Infer** tool used at Meta/Facebook

https://github.com/facebook/infer

4. Worst-case execution time (WCET) for safety-critical embedded systems

Existing Approaches to Resource Analysis

(Automatic) static analysis of the source code

+ Sound: any result is a valid bound

- Incomplete: cannot handle all programs

Data-driven analysis of runtime cost data

+ Always returns a result

- Only sound with respect to runtime cost data

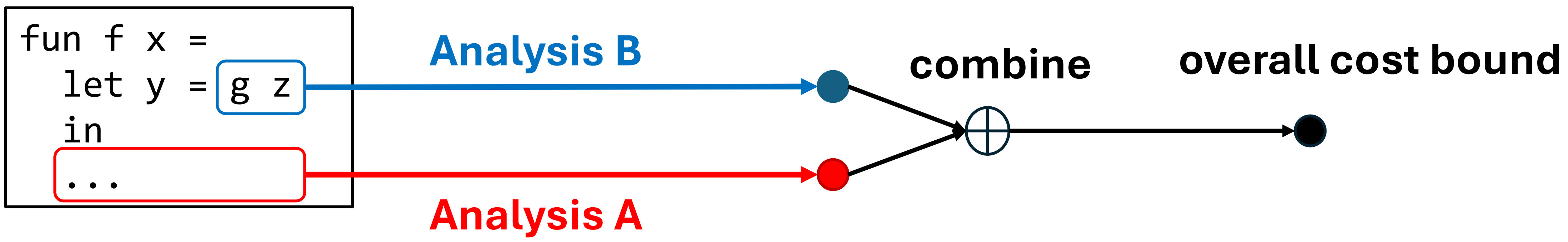
Interactive analysis with proof assistants (e.g., Coq and Agda)

+ Sound and expressive cost bounds

- Not fully automatic

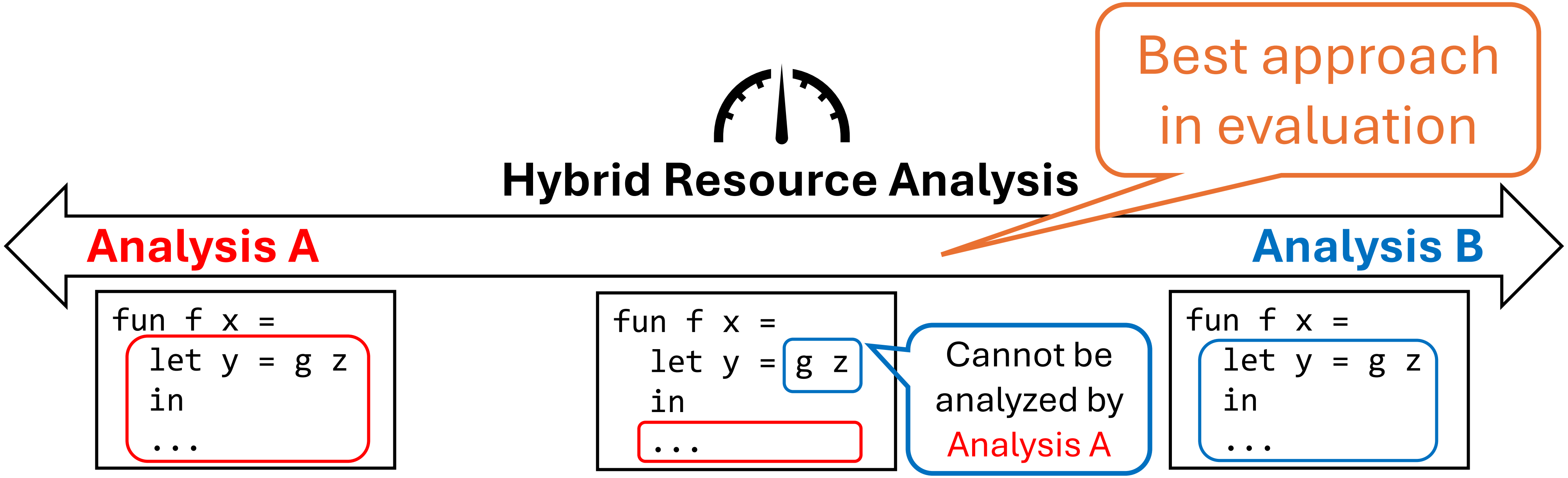
Proposal: Hybrid Resource Analysis

Idea Integrate complementary analysis techniques to overcome their respective limitations



Proposal: Hybrid Resource Analysis

Hybrid resource analysis covers a spectrum ranging from Analysis A to Analysis B



Proposal: Hybrid Resource Analysis

Benefit of hybrid analysis

Retains the strengths of analysis techniques while mitigating their respective weaknesses

- Analyze a larger class of programs
- Infer a larger class of symbolic cost bounds

Thesis statement

Hybrid resource analysis enables the analysis of programs and inference of cost bounds that are beyond the reach of individual analysis techniques

Structure of the Thesis

Static analysis

- Background on AARA (§2.1)
- Polynomial-time completeness of AARA (§2.2) [1]

Data-driven analysis

- Bayesian data-driven analysis (§2.3) [2]
- Program-input generator inference (§2.7)

Hybrid analysis

- Hybrid AARA (§2.4) [2]
- Resource decomposition (§2.5, 2.6) [3]

[1] **Long Pham**, Jan Hoffmann. *Typable Fragments of Polynomial Automatic Amortized Resource Analysis*. Published at CSL 2021

[2] **Long Pham**, Feras Saad, Jan Hoffmann. *Robust Resource Bounds with Static Analysis and Bayesian Inference*. Published at PLDI 2024

[3] **Long Pham**, Yue Niu, Nathan Glover, Feras Saad, Jan Hoffmann. *Integrating Resource Analyses via Resource Decomposition*. Under submission

Outline

- ☑ Motivation and overview
- ❑ Background on AARA
- ❑ Contribution: Bayesian data-driven analysis
- ❑ Contribution: Hybrid AARA
- ❑ Contribution: resource decomposition
- ❑ Proposed work: inference of program-input generator
- ❑ Timeline and conclusion

State of the Art in Static Analysis

Static analysis examines the source code, constructs constraints defining the worst-case behavior, and solves them

- Type systems (e.g., AARA by Hoffmann, Hofmann, Jost et al.)
- Recurrence relations (e.g., COSTA by Albert et al.)
- Ranking functions (e.g., AProVE and KoAT by Giesl et al.)

Advantage

+ Soundness guarantee

Disadvantages

- Incomplete due to the undecidability of resource analysis
- Rewriting a program is difficult for non-expert users

Static Analysis: AARA

<u>Automatic Amortized Resource Analysis (AARA)</u>

Type-based resource analysis that automates the potential method of amortized analysis

<u>Example</u> partition function

<u>Cost</u> one unit per comparison

<u>Input</u> [1, 2, 3, 4] ⇒ **<u>Output</u>** ([,], [,])

Cost is indicated by tick q for $q \in \mathbb{Q}$

Static Analysis: AARA

Automatic Amortized Resource Analysis (AARA)

Type-based resource analysis that automates the potential method of amortized analysis

Example partition function

Cost one unit per comparison

Input [1, 2, 3, 4]

Output ([,], [,])

One unit of potential has been consumed

Cost is indicated by tick q for $q \in \mathbb{Q}$

Static Analysis: AARA

Automatic Amortized Resource Analysis (AARA)

Type-based resource analysis that automates the potential method of amortized analysis

Example partition function

Cost one unit per comparison

Cost is indicated by $\mathrm{tick}\ q$ for $q \in \mathbb{Q}$

Input [1, 2, 3, 4]

Output ([1,], [,])

The input list element is added to the output

Static Analysis: AARA

Automatic Amortized Resource Analysis (AARA)

Type-based resource analysis that automates the potential method of amortized analysis

Example partition function

Cost one unit per comparison

Cost is indicated by tick q for $q \in \mathbb{Q}$

Input [1, 2, 3, 4] ⇒ **Output** ([1, 2], [3, 4])

Resource-annotated typing judgment

$$\text{partition}: \underbrace{(\text{int} \times L^{2}(\text{int}))}_{\text{Input potential: } 2 \cdot n} \rightarrow \underbrace{(L^{1}(\text{int}) \times L^{1}(\text{int}))}_{\text{Output potential: } 1 \cdot n_1 + 1 \cdot n_2}$$

Static Analysis: AARA

1. Assign variables

$$\text{partition}: (\text{int} \times L^{p}(\text{int})) \to (L^{q_1}(\text{int}) \times L^{q_2}(\text{int}))$$

AARA can express polynomial cost bounds in general

2. Collect linear constraints

$$p \geq 1 + q_1$$
$$p \geq 1 + q_2$$

Input potential (p) ≥ Cost (1) + Output potential (q_1, q_2)

Why AARA for static analysis

+ Compositionality offered by types
+ Automatic bound inference by LP solving
+ Precise cost bounds by amortized analysis
+ Soundness guarantee

Sound: any cost bound inferred by AARA is a valid worst-case cost bound

Incomplete: there exists a polynomial-cost program that AARA cannot analyze because resource analysis is undecidable in general

Outline

- ☑ Motivation and overview
- ☑ Background on AARA
- ❑ Contribution: Bayesian data-driven analysis
- ❑ Contribution: Hybrid AARA
- ❑ Contribution: resource decomposition
- ❑ Proposed work: inference of program-input generator
- ❑ Timeline and conclusion

State of the Art in Data-Driven Analysis (Opt.)

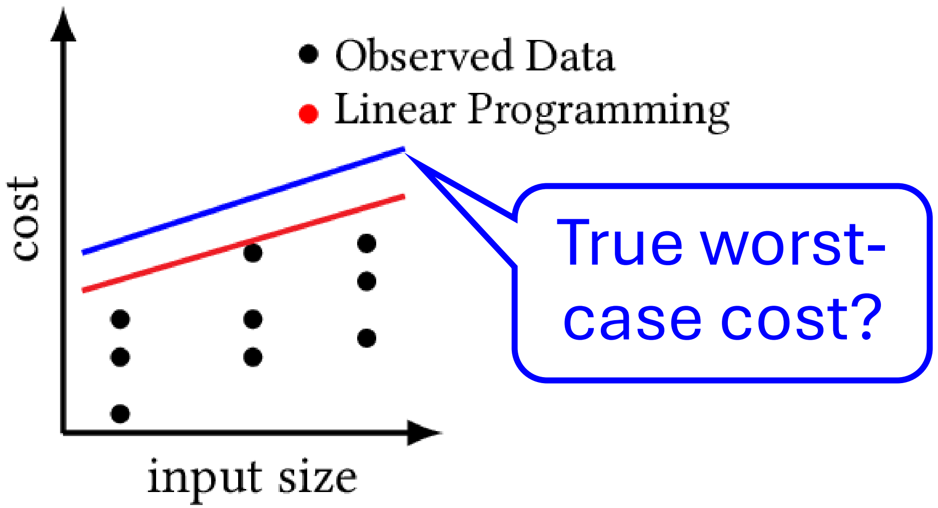
<u>Examples</u> Input-sensitive profiling (Coppa et al.), Algorithmic profiling (Zaparanuks et al.), Dynaplex (Ishimwe et al.)

1. Collect cost measurements of inputs $v_1, \dots, v_N$

$$\left.\begin{array}{c} v_1 \\ \vdots \\ v_N \end{array}\right\} \longrightarrow \boxed{P} \longrightarrow \left\{\begin{array}{c} (|v_1|, c_1) \\ \vdots \\ (|v_N|, c_N) \end{array}\right.$$

2. Optimize cost bound (red line)

Minimize	red line − black dots
Subject to	red line ≥ black dots



<u>Disadvantages of optimization</u>

- Does not incorporate the user's domain knowledge
- No quantitative measure of statistical uncertainty

Contribution: Bayesian Data-Driven Analysis

Bayesian data-driven resource analysis

1. Define a probabilistic model $\pi(\theta, D)$

   θ : latent parameter (cost bound)

   D: observed data (cost measurements)

2. Collect observed data D_{obs}

3. Compute the posterior distribution

   Bayes' rule: $\pi(\, \theta \mid D = D_{\text{obs}}\,) = \frac{\pi(\theta, D{=}D_{\text{obs}})}{\int \pi(\theta, D{=}D_{\text{obs}})\, \mathrm{d}\theta}$

   Draw posterior samples: $\theta_1, \ldots, \theta_M \sim \pi(\theta \mid D = D_{\text{obs}})$

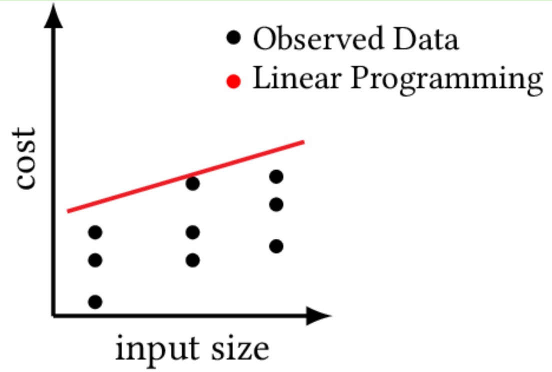
Advantages over optimization

+ Can incorporate the domain knowledge in the probabilistic model

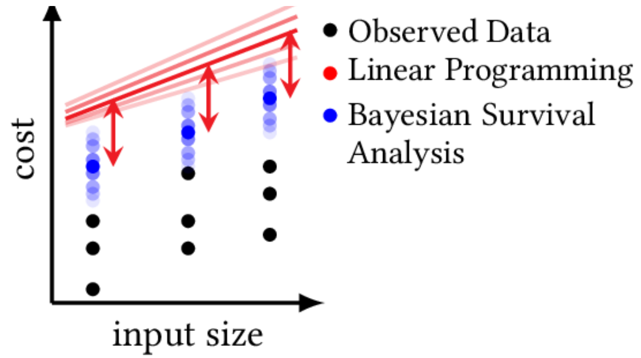
+ Posterior distribution captures statistical uncertainty

Bayesian Data-Driven Analysis: Overview

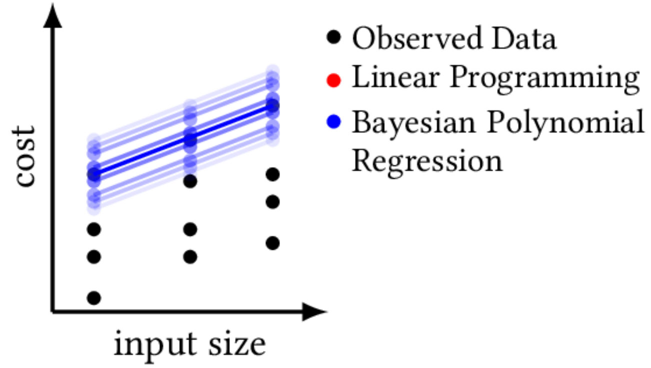
Previous: Optimization (Opt)



New: Bayesian inference of worst-case costs (BayesWC)



New: Bayesian inference of polynomial coefficients (BayesPC)



Bayesian Data-Driven Analysis: BayesWC

Bayesian inference of worst-case costs (BayesWC)

1. Define a probabilistic model $\pi(\theta, \boldsymbol{c}^{\max}, \boldsymbol{c})$

   $\boldsymbol{c}^{\max}$: hidden worst-case cost (blue dots)

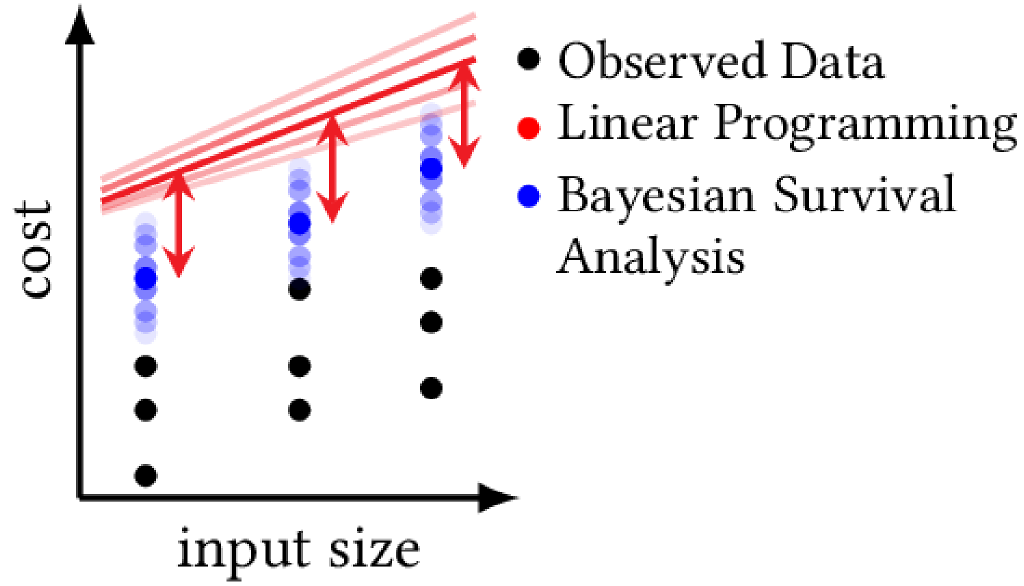
   $\boldsymbol{c}$: observed cost (black dots)

2. Draw posterior samples of $\boldsymbol{c}^{\max}$

3. Optimize cost bound (red line)

   Minimize red line − blue dots

   Subject to red line ≥ blue dots



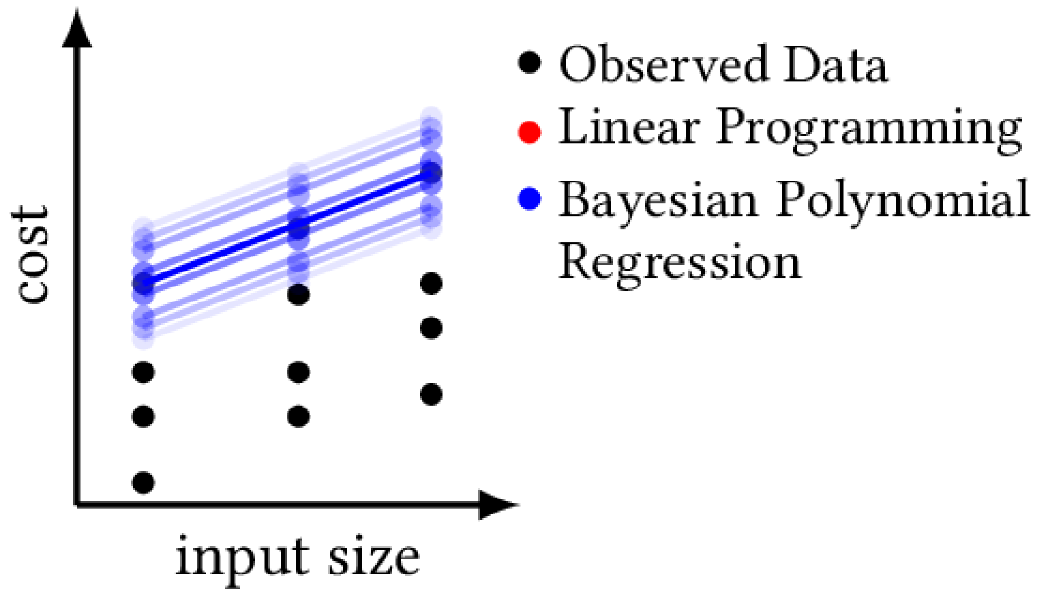
Bayesian Data-Driven Analysis: BayesPC

Bayesian inference of polynomial coefficients (BayesPC)

1. Define a probabilistic model $\pi(p, \boldsymbol{c})$

    p: cost bond (blue line)

    $\boldsymbol{c}$: observed cost (black dots)



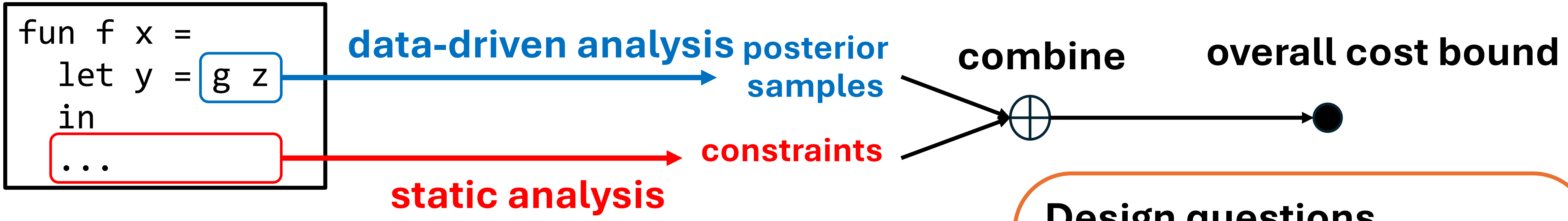
2. Draw posterior samples of cost bound p (blue line)

Outline

- ☑ Motivation and overview
- ☑ Background on AARA
- ☑ Contribution: Bayesian data-driven analysis
- ❑ Contribution: Hybrid AARA
- ❑ Contribution: resource decomposition
- ❑ Proposed work: inference of program-input generator
- ❑ Timeline and conclusion

Hybrid Analysis: Goal and Challenge

Goal



Design questions

1. What is the format of inference results?
2. What information needs to be exchanged during inference?

Key challenge

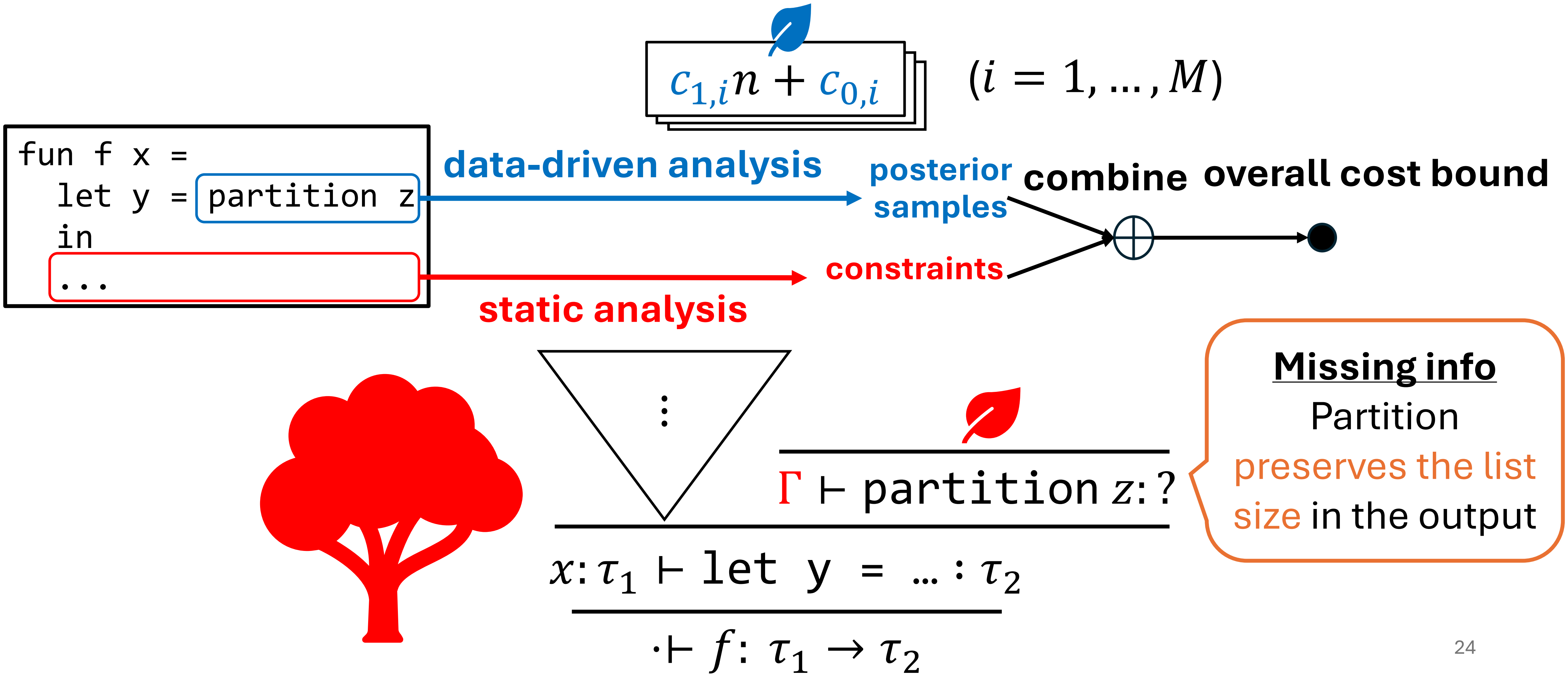
Hybrid analysis needs an interface between:

1. Bayesian data-driven analysis draws samples
2. Static analysis solves constraints

How do we coherently combine constraints and posterior samples?

Hybrid Analysis: Interface Design Attempt 1

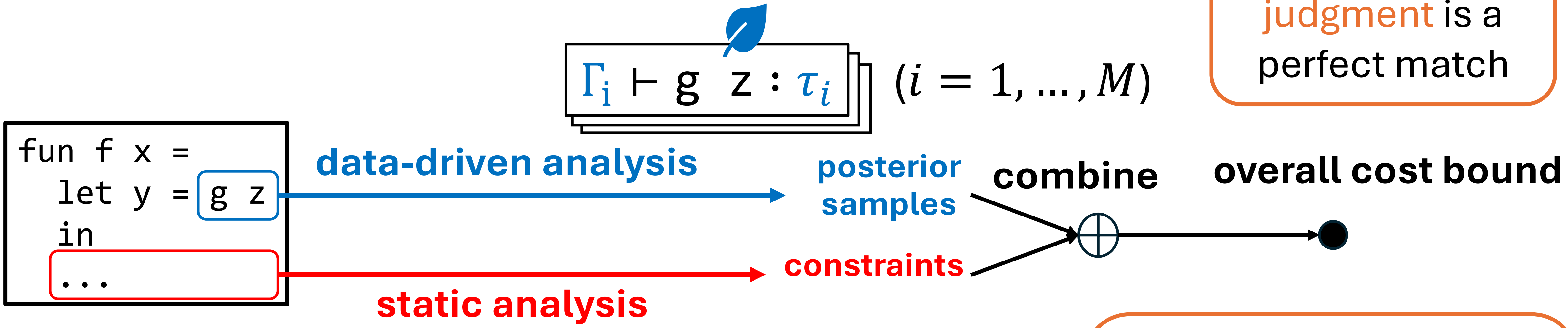
<u>Unsuccessful interface</u> Symbolic cost bound (e.g., $1.1n + 1.2$)



Hybrid Analysis: Interface Design Attempt 2

Interface Potential functions in the input and output

AARA's typing judgment is a perfect match



$$\dfrac{\dfrac{\vdots \qquad \Gamma \vdash \texttt{g z} : \tau}{x : \tau_1 \vdash \texttt{let y = } \ldots : \tau_2}}{\cdot \vdash f : \tau_1 \to \tau_2}$$

Challenge

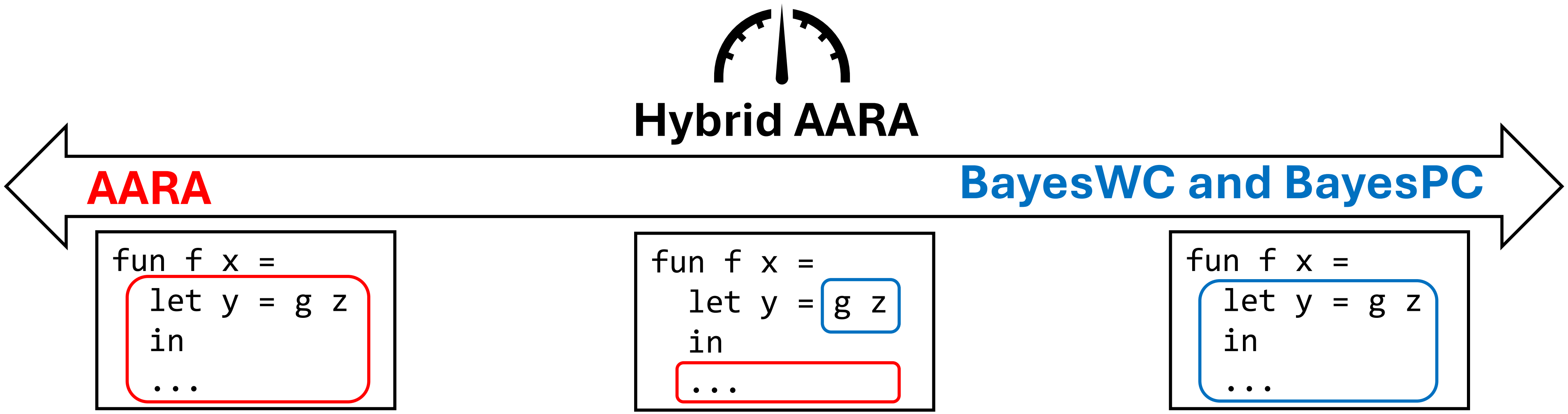
(Γ_i, τ_i) cannot be arbitrary samples, which may violate linear constraints

Hybrid AARA: Contribution

<u>Hybrid AARA</u> It integrates

- Bayesian data-driven analysis (BayesWC and BayesPC) and
- Automatic Amortized Resource Analysis (AARA)

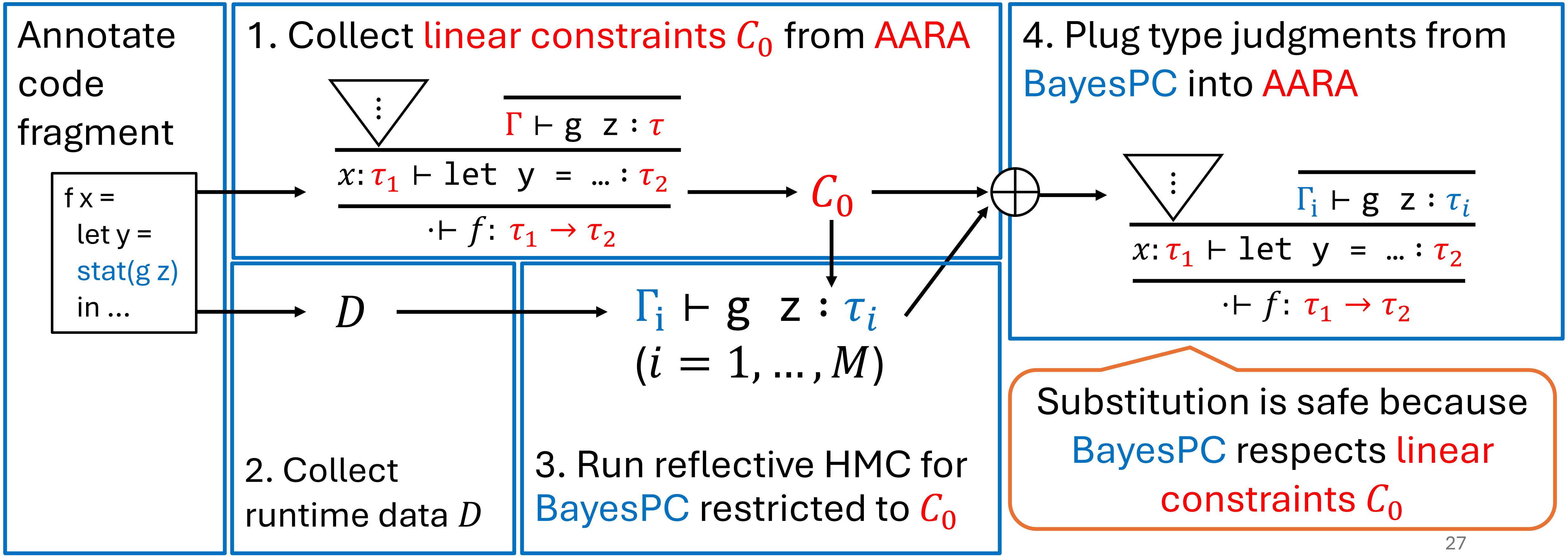
by a novel interface between sampling algorithms and linear programming



Hybrid AARA: AARA + BayesPC

Key idea Draw samples from a probability distribution restricted to a convex polytope defined by linear constraints

Reflective Hamiltonian Monte Carlo (Chalkis et al., 2023)



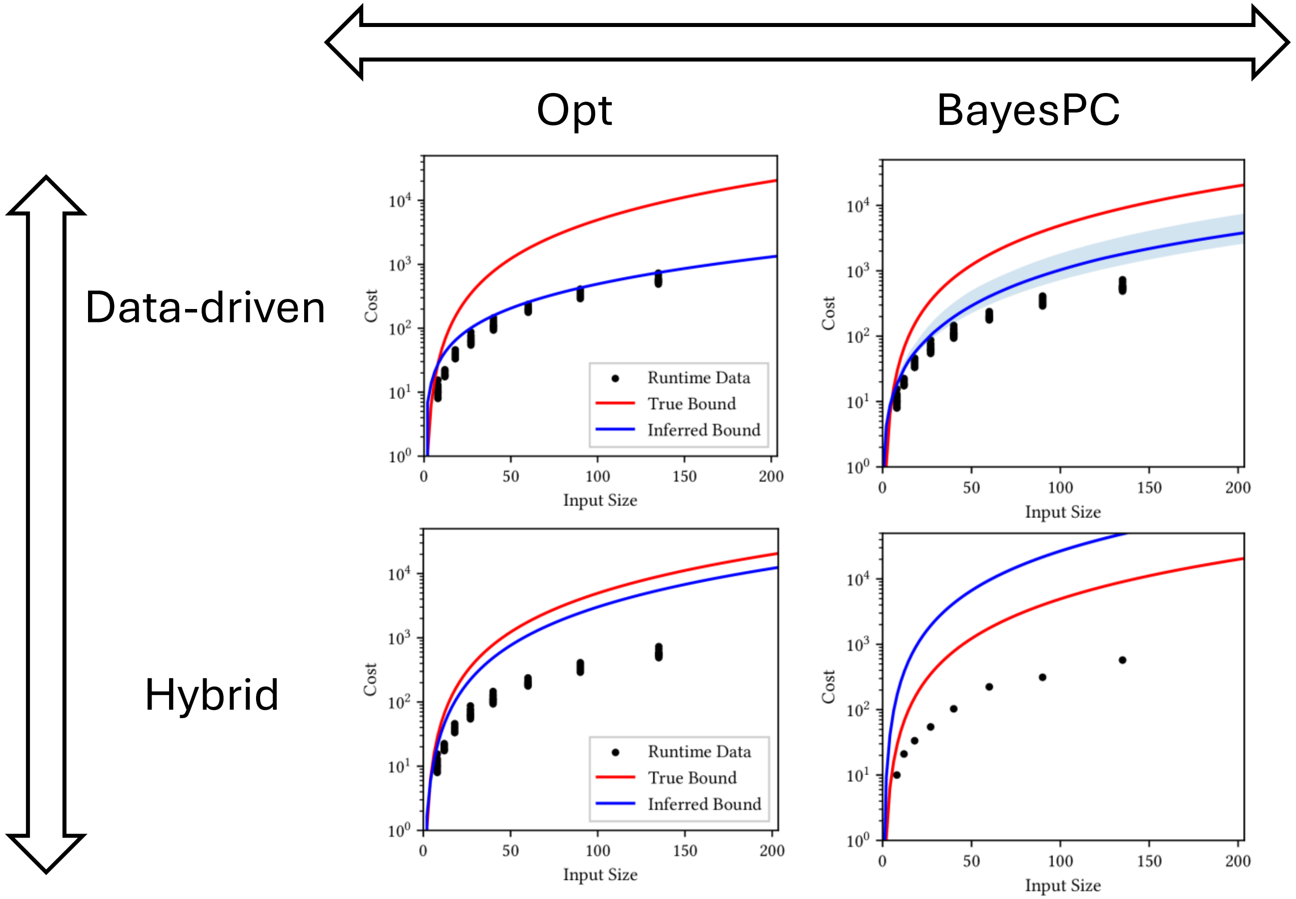
Example Evaluation: Quicksort

Resource metric: comparisons, each of which varies between 0.5 and 1.0

Static analysis is inapplicable to the partition function

1. Bayesian analysis is more accurate than opt.

2. Hybrid analysis is more accurate than data-driven analysis



Outline

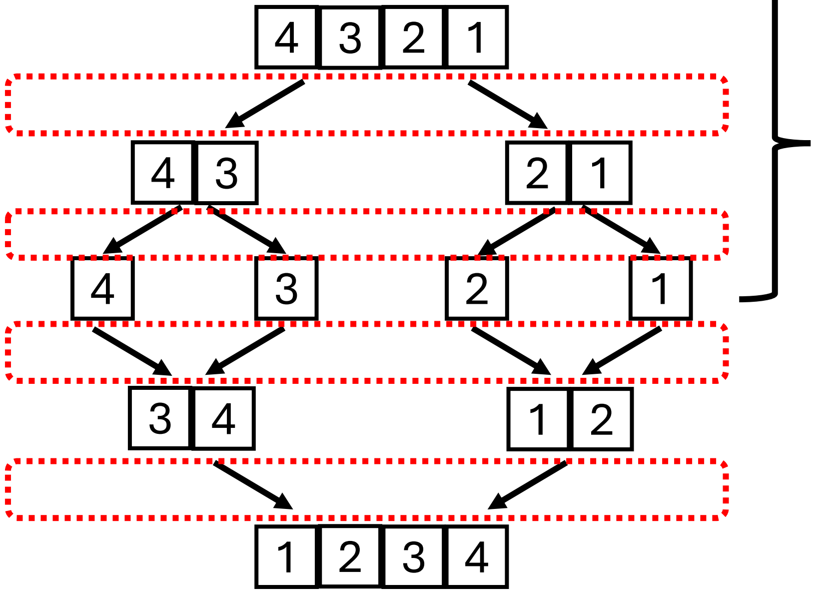
- ☑ Motivation and overview
- ☑ Background on AARA
- ☑ Contribution: Bayesian data-driven analysis
- ☑ Contribution: Hybrid AARA
- ❑ Contribution: resource decomposition
- ❑ Proposed work: inference of program-input generator
- ❑ Timeline and conclusion

Limitations of Hybrid AARA

Two limitations Analysis techniques combined by Hybrid AARA must infer

- Polynomial bounds
- Quantities of the same resource metric

Merge sort



Non-polynomial

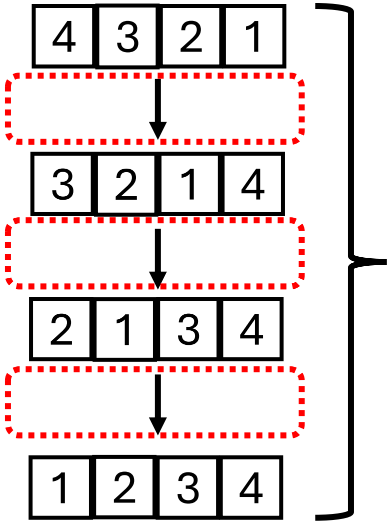
Recursion depth

$\log(n)$

Combined cost at recursion depth

n

Bubble sort



Cannot be statically derived

Recursion depth

n

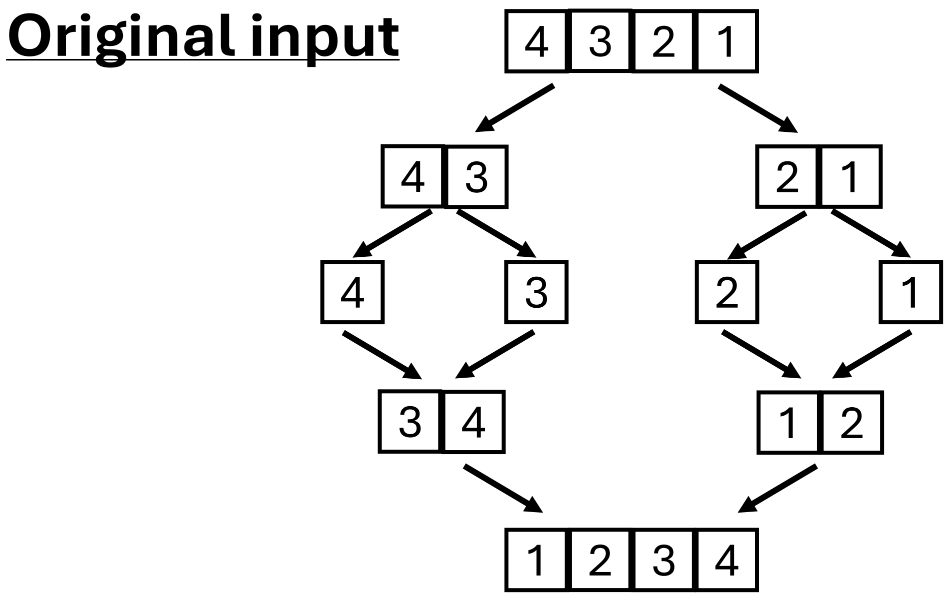
Per-iteration cost

n

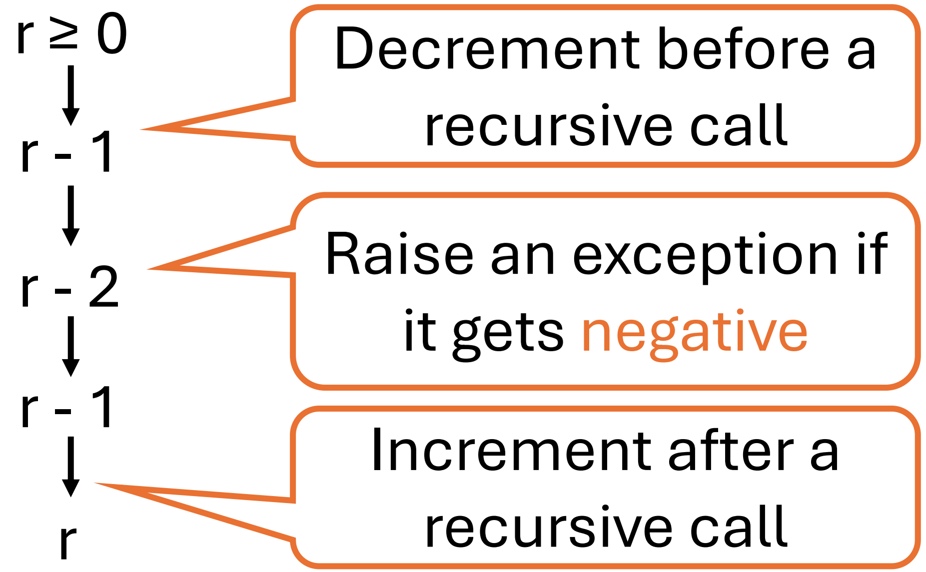
Contribution: Resource Decomposition

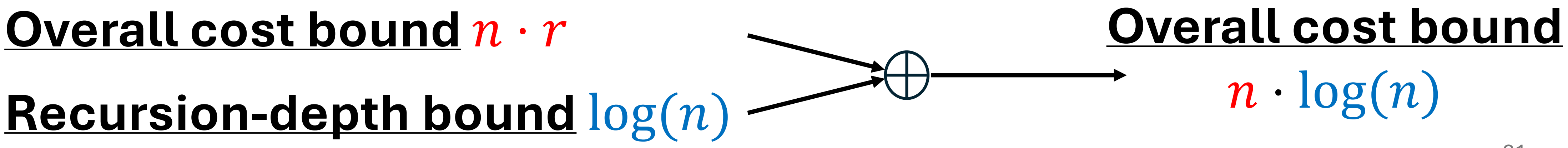
Key idea Extend a program with an extra numeric non-negative variable to represent user-defined quantities (e.g., recursion depths and costs)

Instrumented merge sort

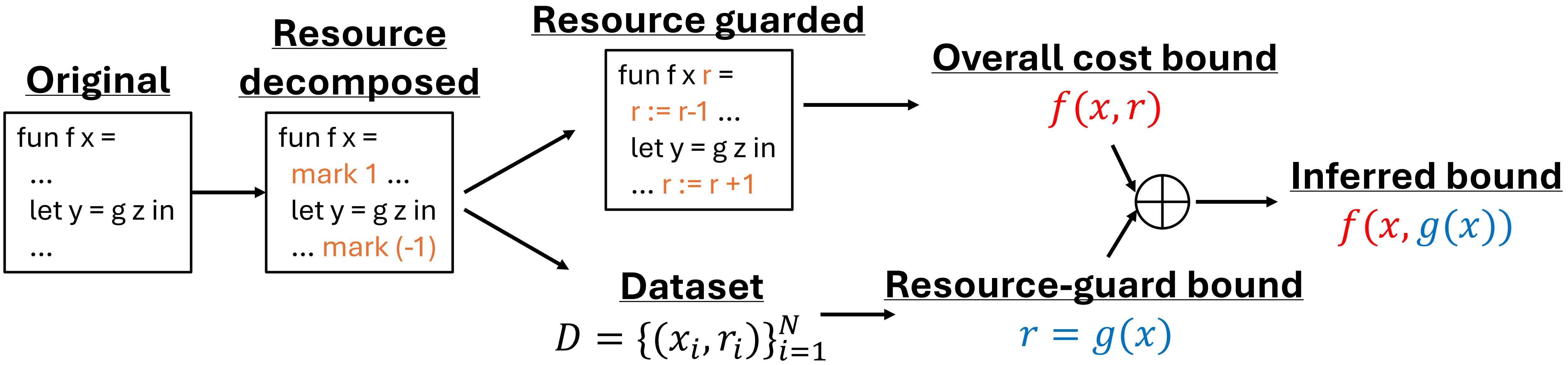


Resource guard





Resource Decomposition: Workflow



1. Manual code annotation: indicate what quantities should be represented by resource guards (e.g., recursion depth, cost of a code fragment, etc.)
2. Automatic transformation: instrument the code with resource guards
3. Conduct Analysis A on the resource-guarded program to infer $f(x, r)$
   Analysis B on the resource-guard to infer $r = g(x)$
4. Substitution: obtain an overall bound $f(x, g(x))$

Resource Decomposition: AARA + Bayesian

Swiftlet Instantiates the resource-decomposition framework with

AARA and Bayesian inference

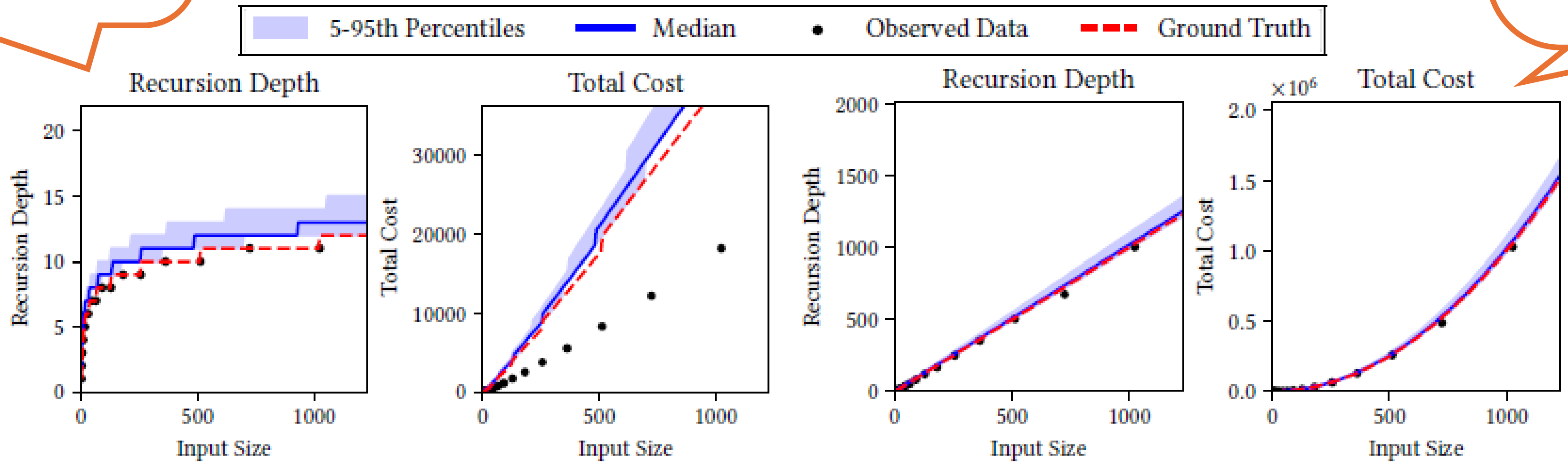
To infer $f(x, r)$

To infer
- $r = c_0 + c_1 n$ or
- $r = c_0 + c_1 \log(1 + c_2 + c_3 n)$

for recursion depths

Correctly infer logarithmic recursion depths

More accurate cost bounds than purely data-driven analysis

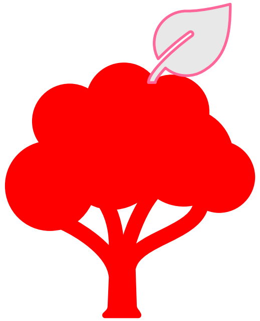


(a) Posterior resource bounds for MergeSort. (b) Posterior resource bounds for BubbleSort.

Comparison between Hybrid Analyses

Hybrid AARA

Interface: potential functions in the input and output



Typing tree

```
let rec f x =
  let y = g z in
  ...
```

$\Gamma_i \vdash g\ \ z : \tau_i$

\+ Precise cost bounds parametric in the input and output sizes

\- Inflexible: only polynomial bounds of the same resource metric

Resource decomposition

Interface: numeric non-negative variable

$f(x, r)$

```
let rec f x r =
  let y = g z in
  ... r := r – 1 ...
```

$r = g(x)$

\+ Flexible: r can represent any quantity with any symbolic bound

\- Resource-guard bounds are only parametric in input sizes

Outline

- ☑ Motivation and overview
- ☑ Background on AARA
- ☑ Contribution: Bayesian data-driven analysis
- ☑ Contribution: Hybrid AARA
- ☑ Contribution: resource decomposition
- ❑ Proposed work: inference of program-input generator
- ❑ Timeline and conclusion

Proposed Work: Program-Input Generator Inference

Issue Existing Bayesian data-driven analysis uses randomly generated program inputs and their cost measurements

1. Program-input generation

$v_1, \ldots, v_N$

2. Cost measurement

$D = \{(v_i, P(v_i), c_i)\}_{i=1}^{N}$

Statistical analysis has no control

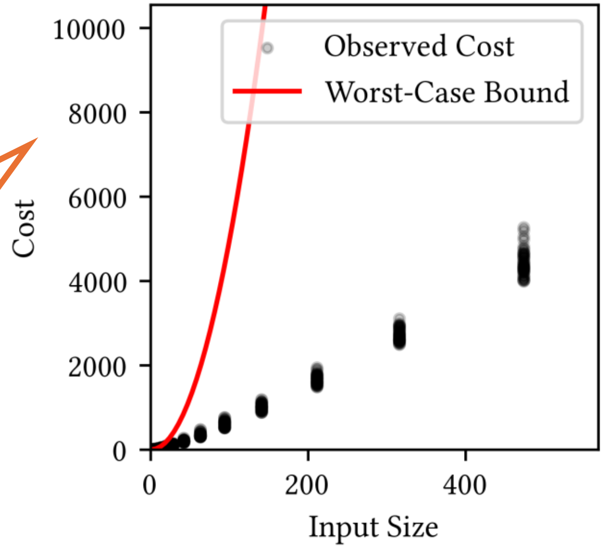
3. Statistical analysis

Analyze D to infer bounds

Quicksort Inputs are generated randomly

Challenge

Huge gap between worst-case complexity and average-case complexity



Proposed Work: Program-Input Generator Inference

Proposal Statistically infer worst-case input generators as well as worst-case cost bounds

1. Statistical analysis
Infer an input generator g

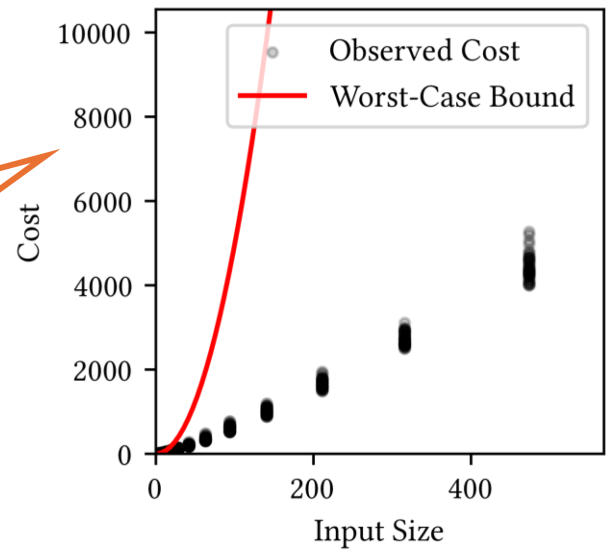
2. Cost measurement
Use g to generate cost dataset D

3. Statistical analysis
Infer a cost bound

Statistical analysis has control

Quicksort Inputs are generated randomly

Solution
Use sorted lists to generate cost measurements



Program-Input Generator Inference: DSL

DSL of program-input generators

1. Probabilistic generators

- The ability to generate all possible values (with some probability) is necessary for a statistical soundness guarantee of data-driven analysis
- More accurate characterization of a class of program inputs

2. Inductively defined by types

$$L = \text{unit} + (\text{int} \times L)$$

```
let rec g_L x =
  let e = g_bool x in
  if e then
    g_unit x
  else
    let v_1 = g_int x in
    let v_2 = g_L x in
    (v_1, v_2)
```

Generate a Boolean

Generate the unit element

Generate a head and tail

Program-Input Generator Inference: Inference

Challenge in generator inference

How do we score generators?

Optimization We can run fuzzing (e.g., based on genetic algorithms) over the space of generators to find a generator with the highest score

Bayesian inference We cannot have a probability distribution where generators with higher scores have higher densities

Outline

- ☑ Motivation and overview
- ☑ Background on AARA
- ☑ Contribution: Bayesian data-driven analysis
- ☑ Contribution: Hybrid AARA
- ☑ Contribution: resource decomposition
- ☑ Proposed work: inference of program-input generator
- ❑ Timeline and conclusion

Timeline

Fall 2024

- Submit the resource-decomposition paper to PLDI 2025

Spring 2025

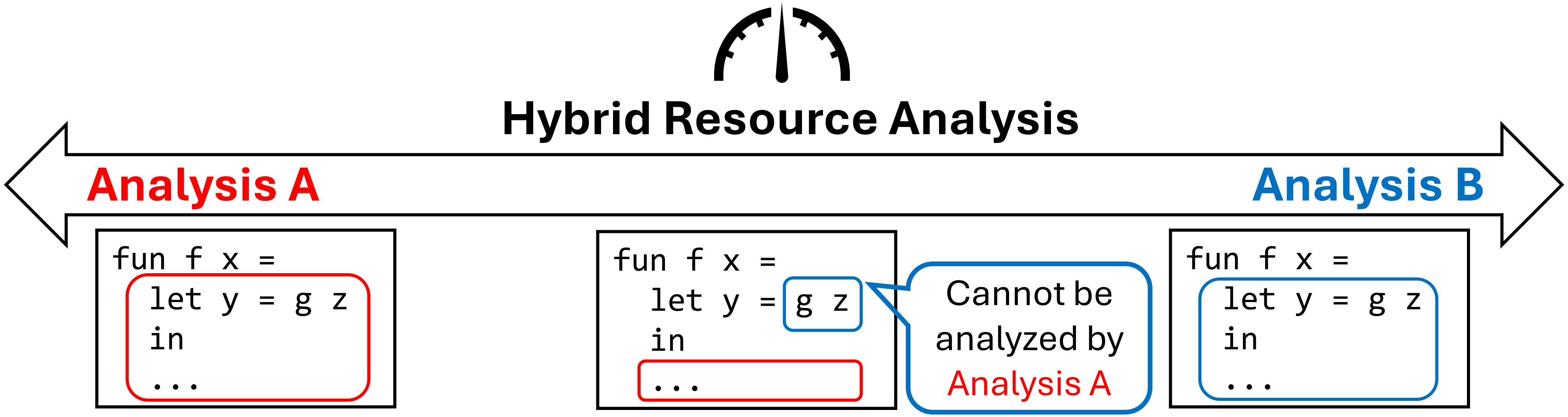
- Thesis proposal
- Complete the design of the DSL for generators and implement a prototype
- Resubmit the resource-decomposition paper (if necessary)
- Complete the speaking-skill requirement
- Complete a thesis draft

Summer 2025

- Thesis defense

Conclusion

1. Hybrid resource analysis integrates complementary analysis techniques to retain their strengths while mitigating their weaknesses



2. Two interfaces: resource-annotated types in Hybrid AARA and numeric program variables in resource decomposition

3. Proposed work: data-driven analysis for inferring worst-case program-input generators as well as worst-case cost bounds

Thank you

Polynomial-time completeness of AARA

Polynomial-time completeness

Typable fragment of AARA contains all polynomial-time functions

Theorem 2.2

Given a polynomial-time Turing machine $M: \mathbb{N} \to \mathbb{N}$, there exists a functional program $P: \mathbb{N} \to \mathbb{N}$ such that

- For every input $n \in \mathbb{N}$, we have $P(n) = M(n)$
- The cost of P is equal to the cost of M
- AARA can infer a polynomial cost bound of P

This idea will be exploited in the design of hybrid resource analysis

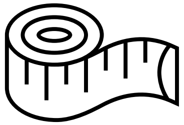
Key idea in the proof

Add an extra program variable to represent the known cost bound

Polynomial-time completeness of AARA

Turing machine M

Tape with input w

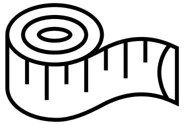


Cost bound $p(|w|)$

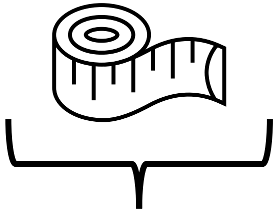
Must know the polynomial bound in advance

Program P

List with input w



List storing potential $\ell_{\text{potential}}$



Length $p(|w|)$

Operation of P

- Create $\ell_{\text{potential}}$ of length $p(|w|)$, where each cell stores one unit of potential
- Every time M moves its head, P simulates the same move, removes one list cell from $\ell_{\text{potential}}$, and runs `tick 1.0`

Resource Decomposition: Soundness

Soundness (Theorem 2.4)

If $f(x, r)$ is a sound overall cost bound of the resource-guarded program $P_{\mathrm{rg}}(x, r)$ and $g(x)$ is a sound bound of a resource guard r, then $f(x, g(x))$ is a sound overall cost bound of the original program $P(x)$.

Proof

By a logical-relation argument

Comparison: Resource Guards and Clocks

Comparison

Usage: resource guards are for decomposing resource analysis, while clocks are for termination proofs. If termination checkers are strong enough, we may no longer need clocks.

Quantities: resource guards can be any quantity that can be defined as a high-water mark cost, while clocks are (at least in the calf paper) recursion depths.

Summary

A clock is a special case of a resource guard for (i) a recursion depth and (ii) decomposing resource analysis into Agda and a termination proof.