From Batch to Stream: Automatic Generation of Online Algorithms

Ziteng Wang[1], Shankara Pailoor[1], Aaryan Prakash[1], Yuepeng Wang[2], Isil Dillig[1]

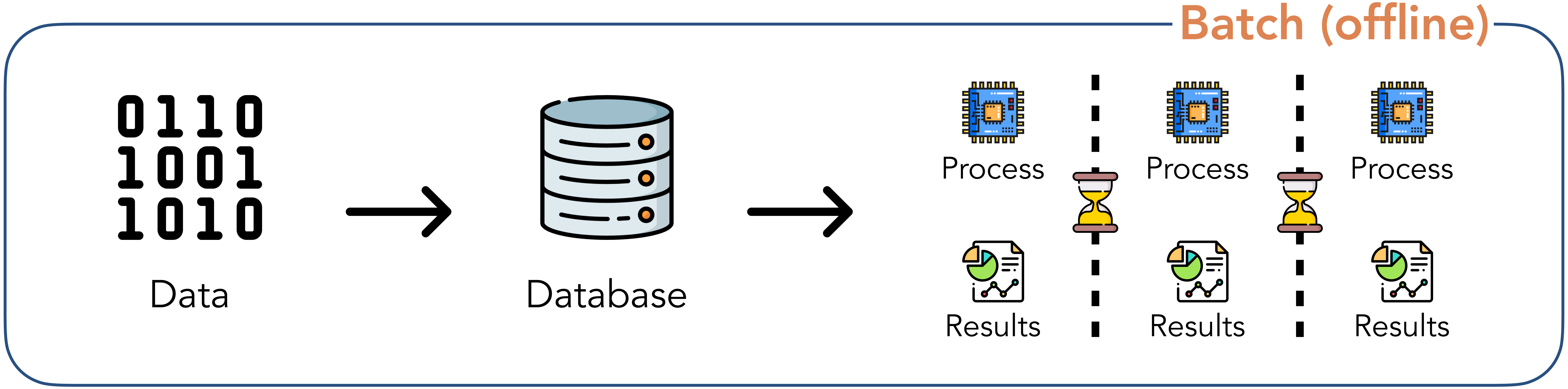
[1] University of Texas at Austin
[2] Simon Fraser University

UToPiA

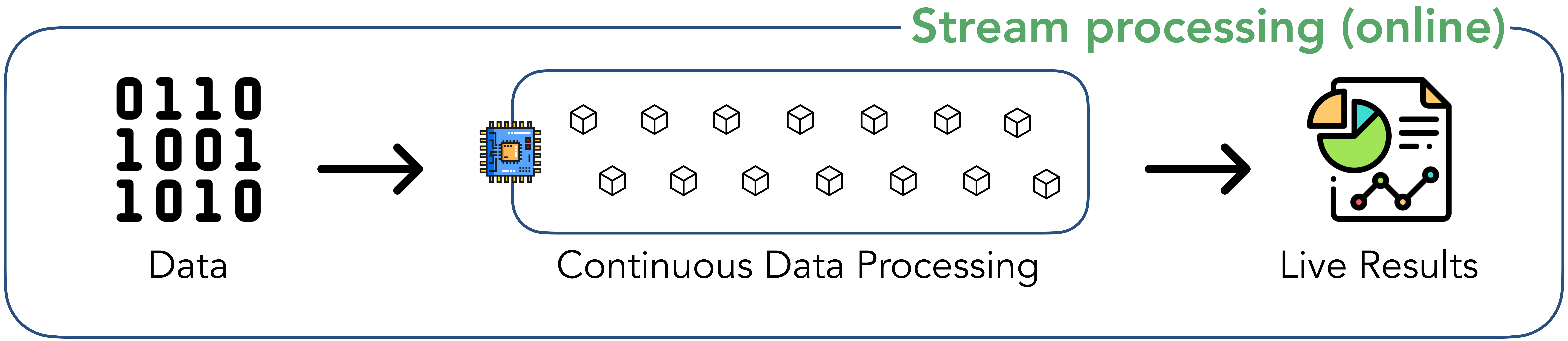
TEXAS
The University of Texas at Austin

why online algorithms?

background



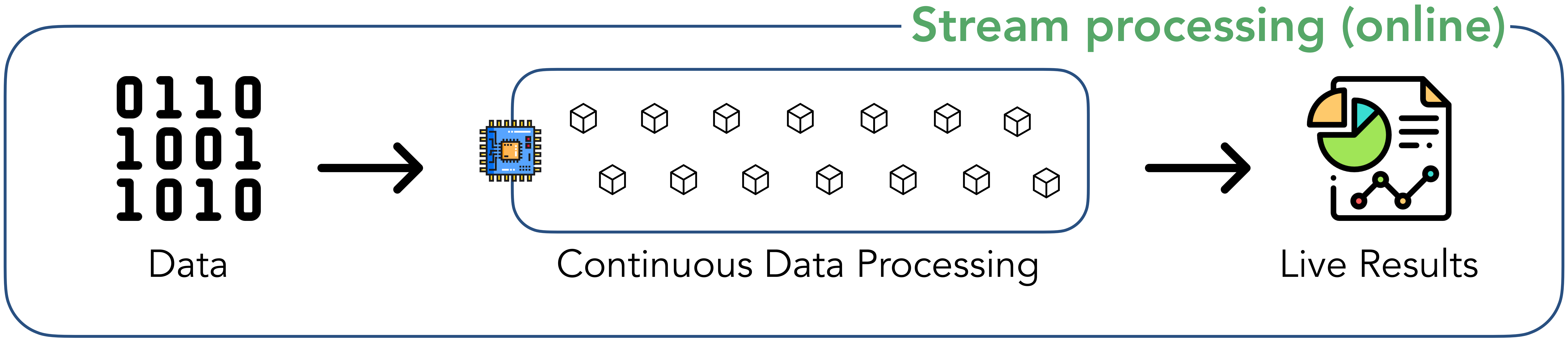
Batch (offline)



Stream processing (online)

why online algorithms?

background



Ideal for time-sensitive computations, e.g.,

- fraud detection
- marketing/sales analytics
- inventory management
- …

Flink

APACHE STORM™

APACHE Spark™

example

background

Offline

```
def mean(xs):
    s = 0
    for x in xs:
        s += x
    return s / len(xs)
```

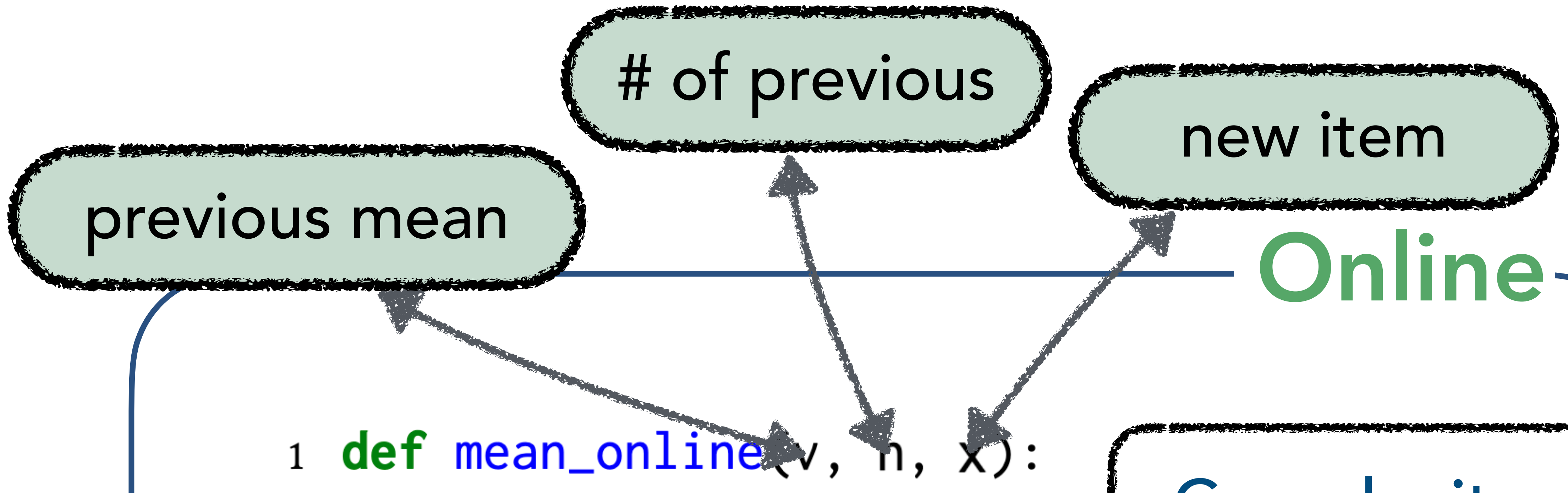
Complexity: $O(n)$

(a) Algorithm for offline sample mean.

```
-- s = 3; len(xs) = 3
mean [0, 1, 2] = 1

-- s = 6; len(xs) = 4
mean [0, 1, 2, 3] = 1.5
```

Online



```
def mean_online(v, n, x):
    new_s = (v * n) + x
    new_n = n + 1
    return new_s / new_n, new_n
```

Complexity: $O(1)$

(b) Algorithm for online sample mean.

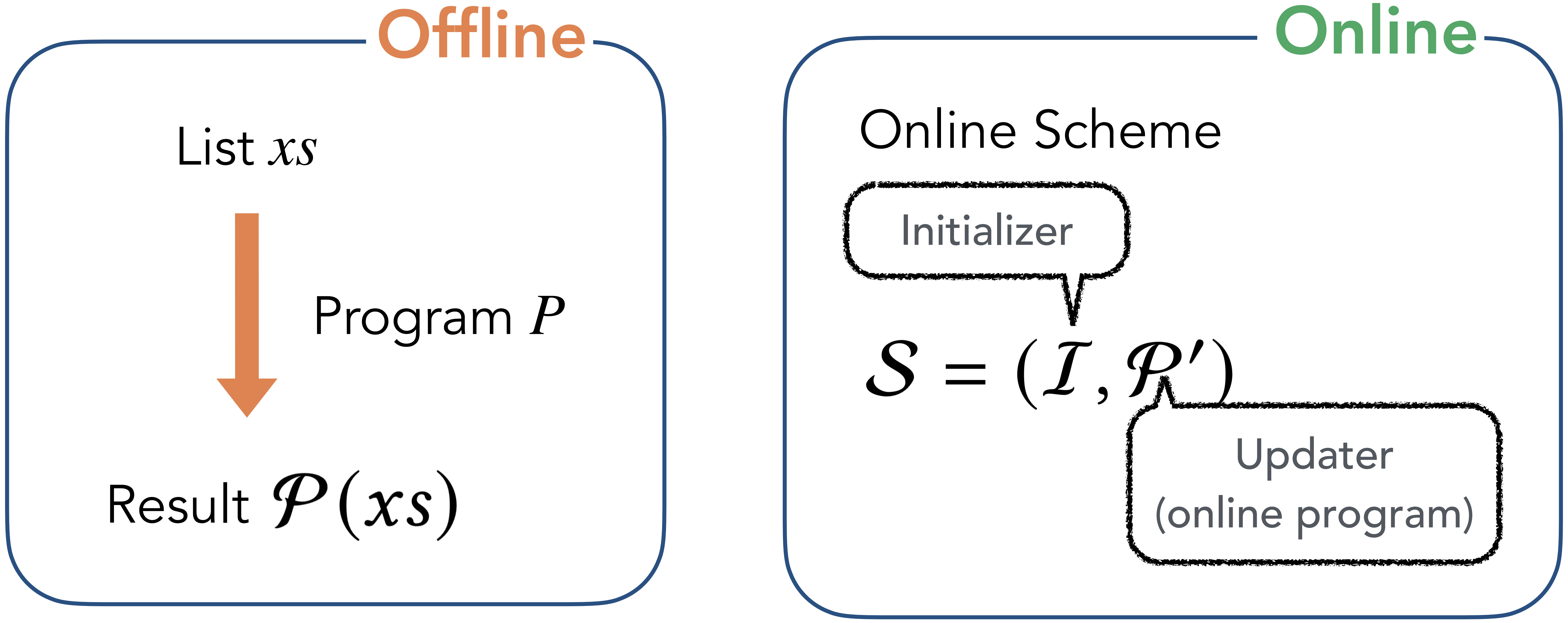
```
Initializer(v, n) = (0, 0)

-- state after processing [0, 1, 2]
v = 1

n = 3

-- new_s = 6; new_n = 4
mean_online v n 3 = 1.5
```

problem statement



Offline

List xs

Program P

Result $\mathcal{P}(xs)$

Online

Online Scheme

Initializer

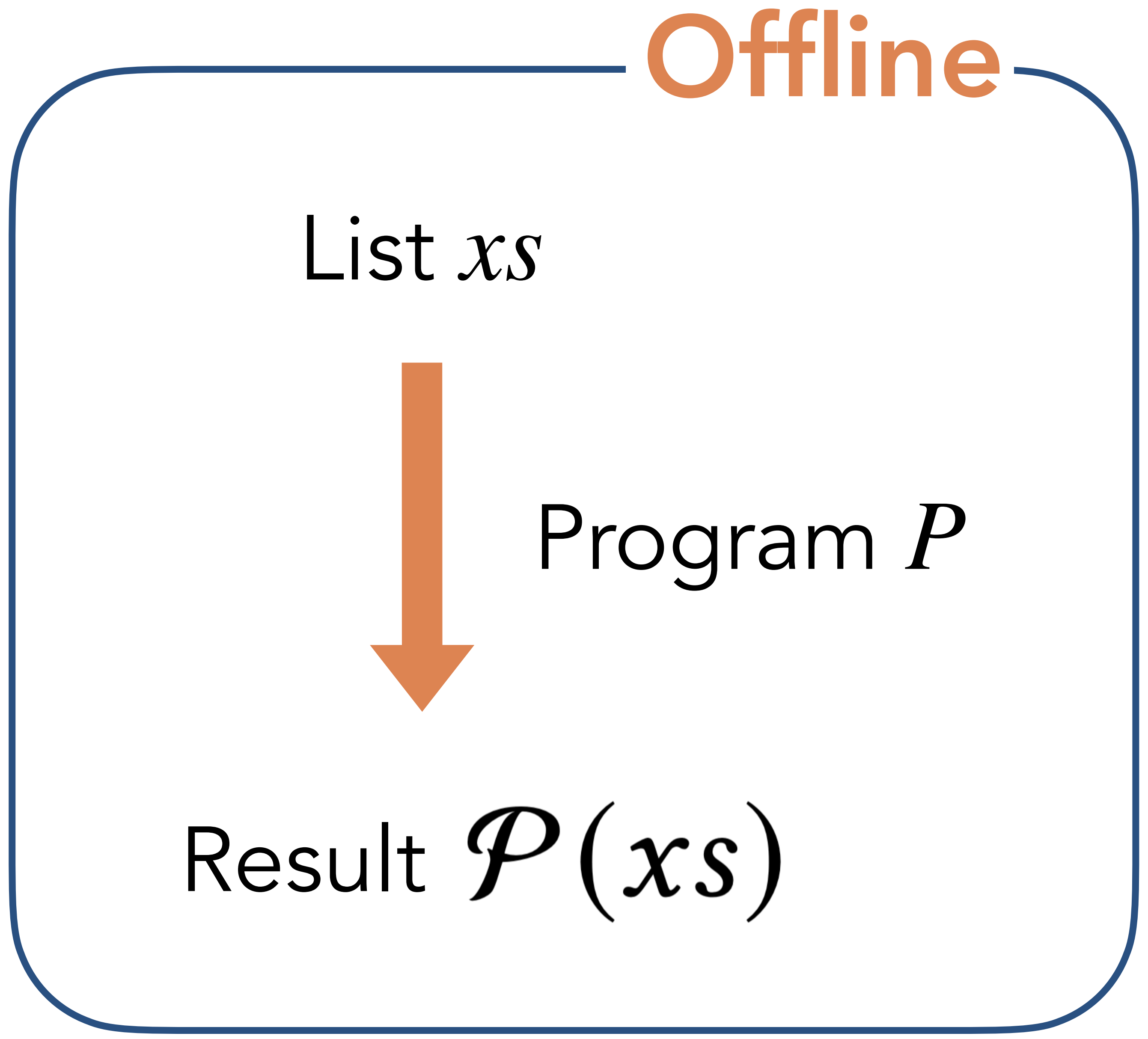
$\mathcal{S} = (\mathcal{I}, \mathcal{P}')$

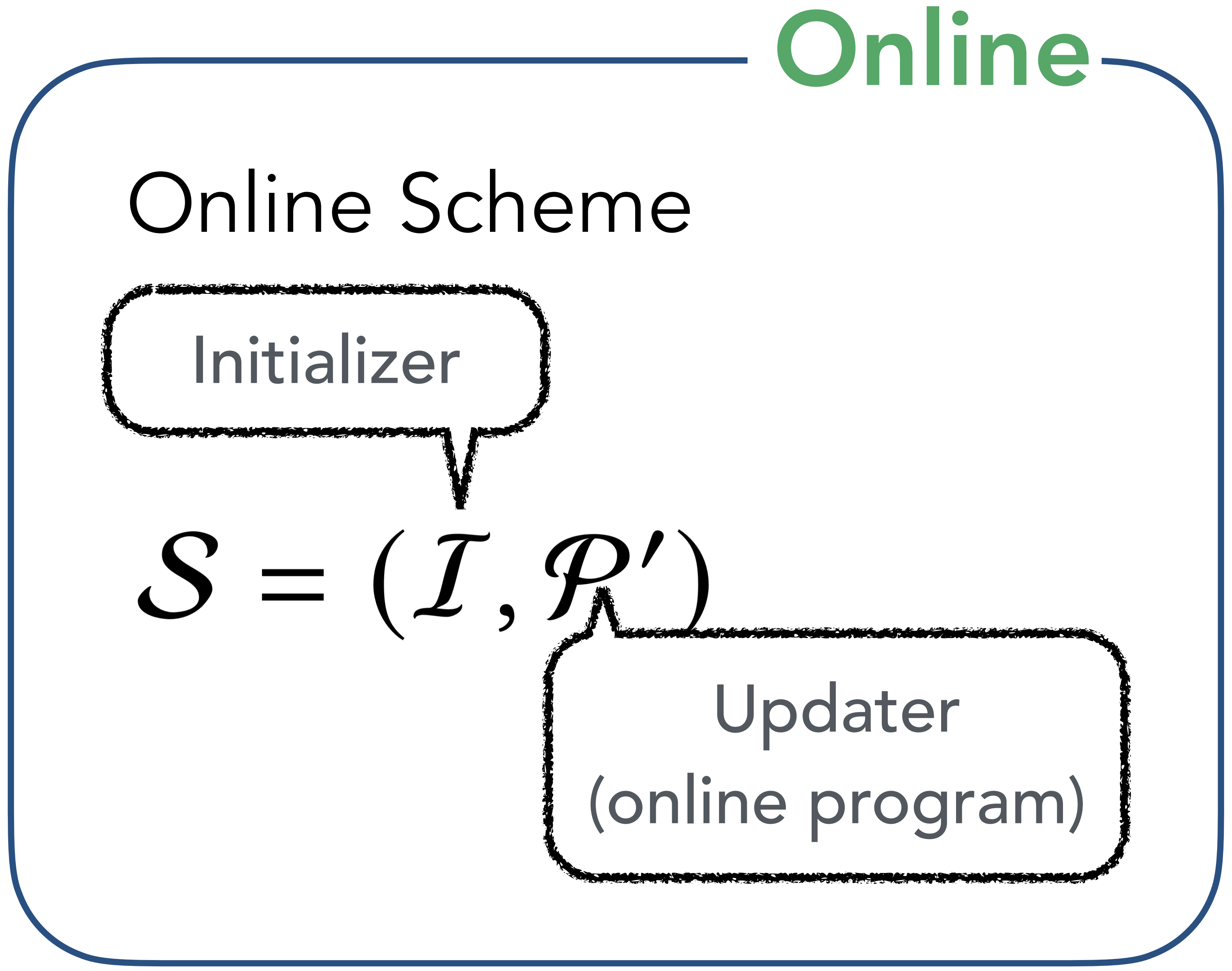
Updater
(online program)

Given an offline program P, find an online scheme
S = (I, P') such that P is **equivalent** to S.

problem statement

background





Given an offline program P, find an online scheme S = (I, P') such that P is **equivalent** to S.

Offline-Online Equivalence

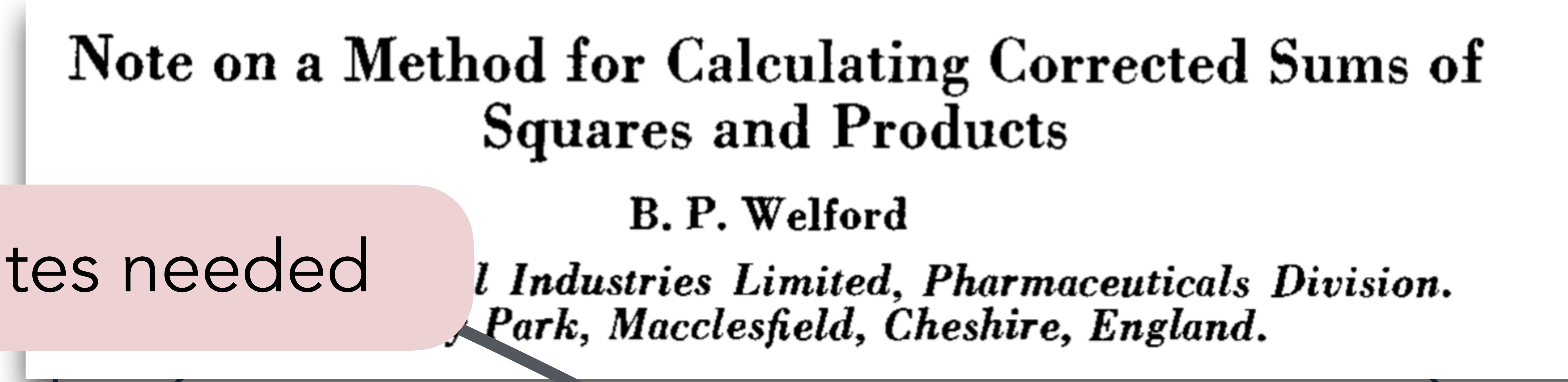
$$\mathcal{P}(xs) = \mathrm{fst}(\mathrm{foldl}(\mathcal{P}', \mathcal{I}, xs))$$

what is the challenge?

```
def variance(xs):
    s = 0
    for x in xs:
        s += x
    avg = s / len(xs)

    sq = 0
    for x in xs:
        sq += (x - avg) ** 2
    return sq / len(xs)
```

Additional states needed



Note on a Method for Calculating Corrected Sums of Squares and Products

B. P. Welford

l Industries Limited, Pharmaceuticals Division.
Park, Macclesfield, Cheshire, England.

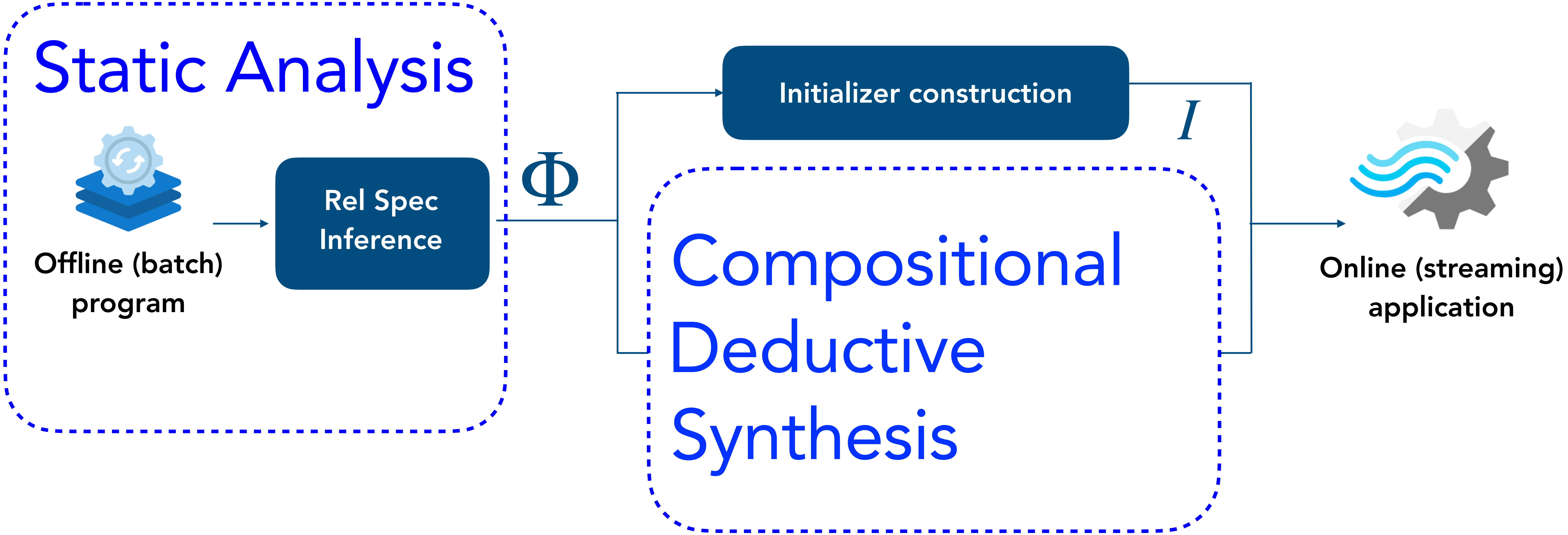
```
def welford(v, s, sq, n, x):
    new_s = s + x
    new_n = n + 1
    avg = new_s / new_n
    tmp = s / n
    new_sq = sq + (x - tmp) * (x - avg)
    new_v = new_sq / new_n
    return new_v, new_s, new_sq, new_n

nitializer(v, s, sq, n) = (0, 0, 0, 0)
```

Complex non-linear expressions

how does opera work?

opera workflow



relational spec

bridging offline and online

Example

```
variance xs =
  let
          s = foldl (+) 0 xs
        avg = s / (length xs)
    f acc x = acc + (x - avg)^2
  in (foldl f 0 xs) / (length xs)
```

Relational Spec

Parameter	Specification
v	`variance xs`
s	`foldl (+) 0 xs`
sq	`foldl (\acc x -> acc + (x - avg)^2) 0 xs`
n	`length xs`

A relational function signature (RFS) Φ consists of:

1. the function signature of the online algorithm
2. what those arguments actually mean

relational spec inference

inferring RFS

Example

```
variance xs =
  let
          s = foldl (+) 0 xs
        avg = s / (length xs)
    f acc x = acc + (x - avg)^2
  in (foldl f 0 xs) / (length xs)
```

Parameter	Specification
〰 v	variance xs
🟨 s	foldl (+) 0 xs
🟩 sq	foldl (\acc x -> acc + (x - avg)^2) 0 xs
🟥 n	length xs

lightweight static analysis

relational spec inference

constructing initializers

Example

```
variance xs =
  let
          s = foldl (+) 0 xs
        avg = s / (length xs)
    f acc x = acc + (x - avg)^2
  in (foldl f 0 xs) / (length xs)
```

Initializer

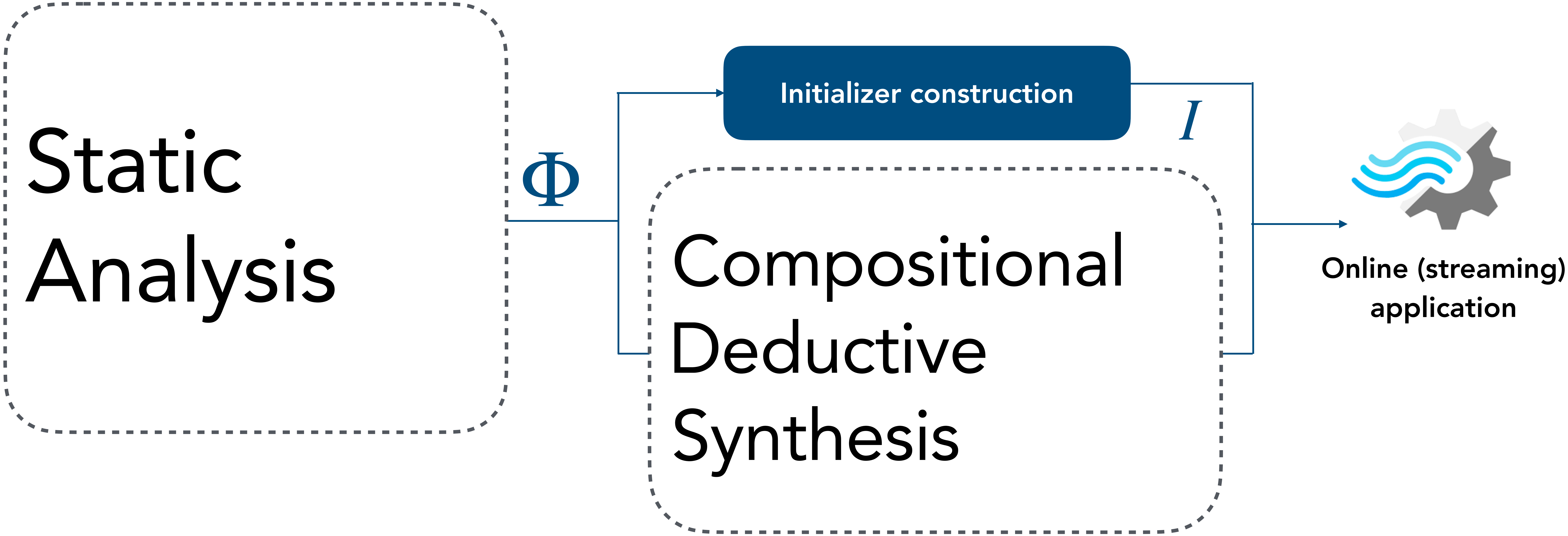
Parameter	Specification
〰 v = 0	`variance xs`
🟨 s = 0	`foldl (+) 0 xs`
🟦 sq = 0	`foldl (\acc x -> acc + (x - avg)^2) 0 xs`
🟥 n = 0	`length xs`

RFS gives us the initializer for free:

construct **initializer** by evaluating expression on an empty list

how does opera work?

opera workflow



Static Analysis

Φ

Initializer construction

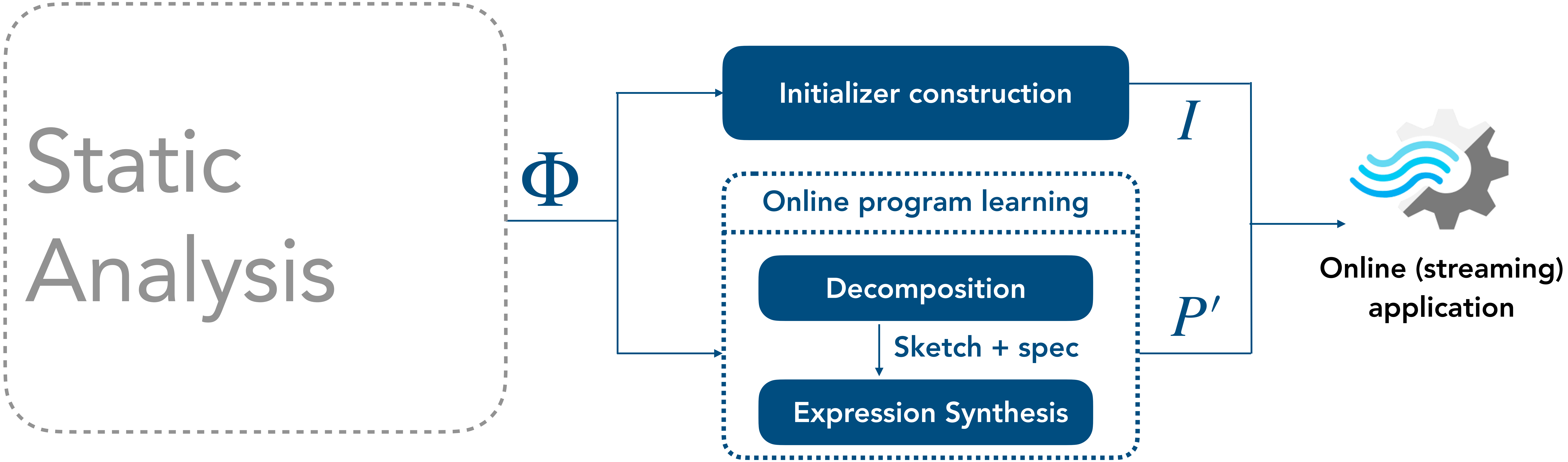
I

Compositional Deductive Synthesis

Online (streaming) application

how does opera work?

opera workflow



Compositional Deductive Synthesis

key idea

inductiveness modulo RFS

💡 Re-define equivalence in terms of inductiveness modulo RFS

Inductiveness modulo RFS, intuitively

an online program P' is inductive modulo RFS if it preserves the RFS

key idea

inductiveness modulo RFS

Re-define equivalence in terms of inductiveness modulo RFS

Inductiveness modulo RFS, formally

An online program $\mathscr{P}'$ is inductive relative to an RFS Φ if the following Hoare triple is valid

$$\{\Phi(xs, y)\} \quad y' := \mathcal{P}'(y, x);\ xs' = xs \mathbin{++} [x] \quad \{\Phi(xs', y')\}$$

equivalence re-defined

Theorem

Let P be an offline program and Φ be an RFS. If an online scheme $S = (I, P')$ satisfies

(1) $I \vDash \Phi(\mathrm{Nil}, I)$ and (2) P' is inductive relative to Φ,

then S is equivalent to the original offline program P.

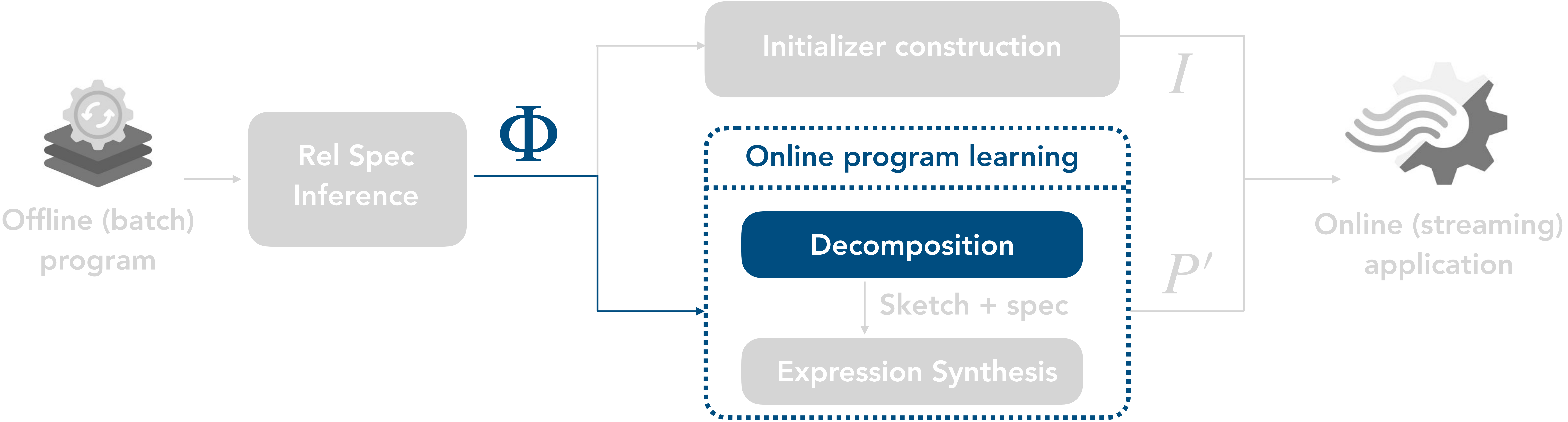
Re-defined Synthesis Task

Given Φ, construct P' that is inductive modulo Φ.

💡 This re-formulation facilitates deductive synthesis

how does opera work?

opera workflow



decomposition

sketch

decomposition

🔍 General high-level structure shared between offline and online

Infer a sketch of online program to reuse structure 💡

🔍 Online expressions are independent to each other

Decompose online synthesis to independently solvable subproblems 💡

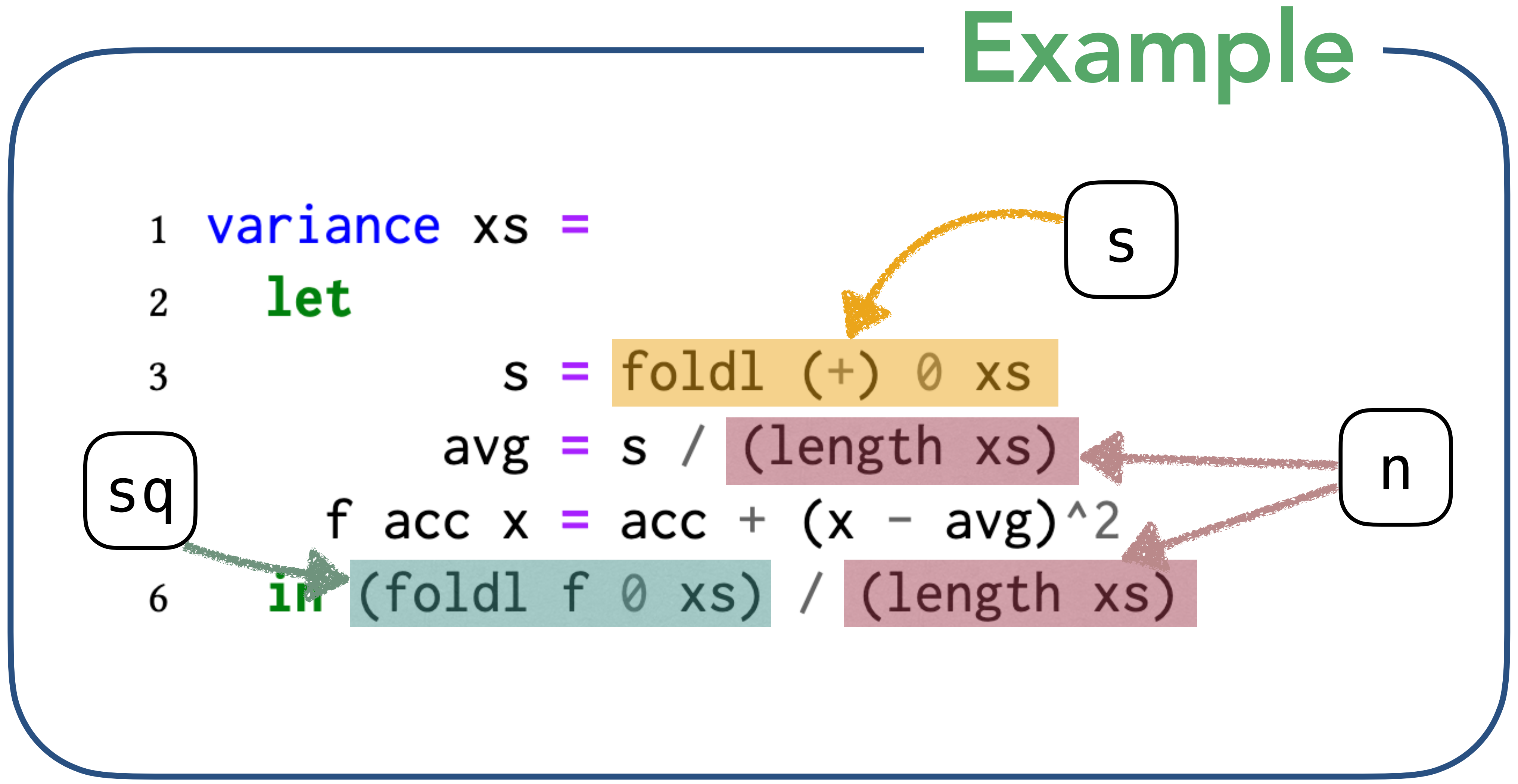
syntax-guided sketch generation

decomposition

replace **offline** expressions that operate over the input list with unknown holes

use the original expression as specification for each hole

Example



```
variance xs =
  let
          s = foldl (+) 0 xs
        avg = s / (length xs)
    f acc x = acc + (x - avg)^2
  in (foldl f 0 xs) / (length xs)
```

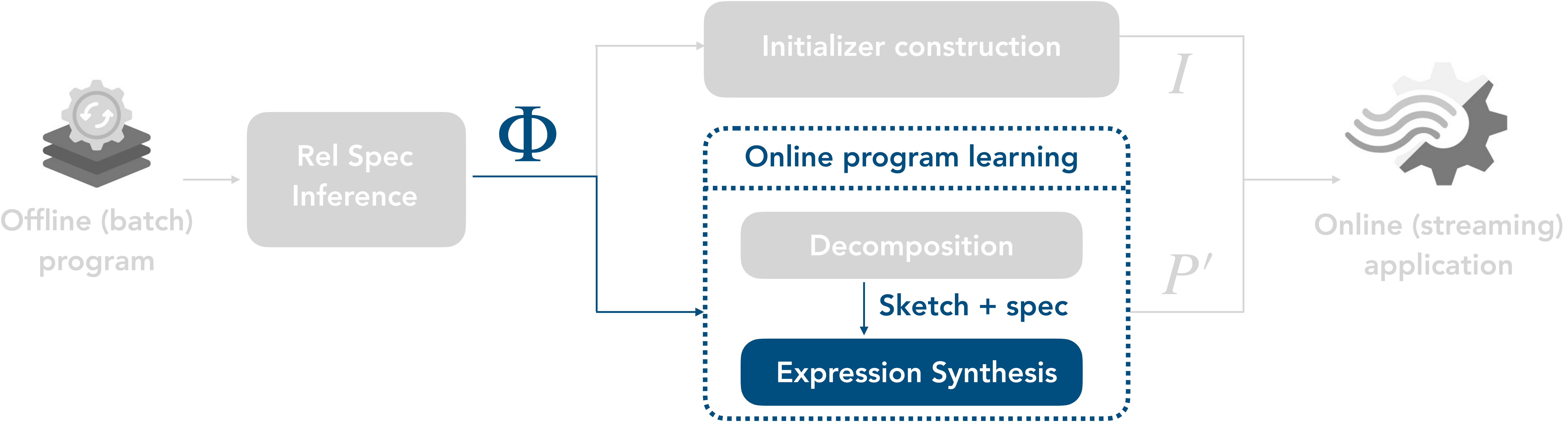
Sketch

```
online_variance (v, s, sq, n) x =
    let new_s = □1
        new_n = □2
        avg = s / new_n
        new_sq = □3
    in (new_sq / new_n, new_s, new_sq, new_n)
```

Unknown	Specification
$\square_1$	`foldl (+) 0 xs`
$\square_2$	`length xs`
$\square_3$	`foldl (\acc x -> acc + (x-avg)^2) 0 xs`

how does opera work?

opera workflow



Offline (batch) program

Rel Spec Inference

Φ

Initializer construction

I

Online program learning

Decomposition

Sketch + spec

Expression Synthesis

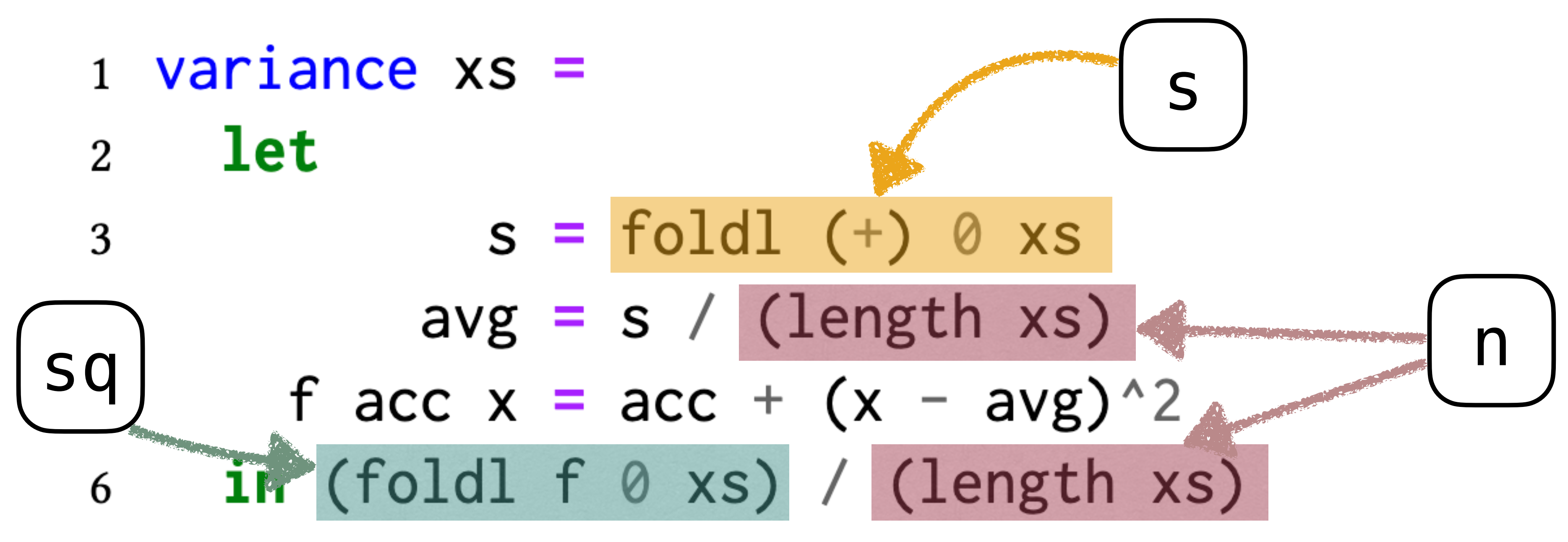
P'

Online (streaming) application

expression synthesis

expression synthesis

Example



```
variance xs =
  let
          s = foldl (+) 0 xs
        avg = s / (length xs)
    f acc x = acc + (x - avg)^2
  in (foldl f 0 xs) / (length xs)
```

Sketch

```
online_variance (v, s, sq, n) x =
    let new_s = □1
        new_n = □2
        avg = s / new_n
        new_sq = □3
    in (new_sq / new_n, new_s, new_sq, new_n)
```

Unknown	Specification
□$_1$	`foldl (+) 0 xs`
□$_2$	`length xs`
□$_3$	`foldl (\acc x -> acc + (x-avg)^2) 0 xs`

Replace offline expressions with equivalent online expressions

expression synthesis

Replace offline expressions with equivalent online expressions

Deductive approach to find such equivalent expressions

expression synthesis

Replace offline expressions with equivalent online expressions

Relational spec enables logical reasoning for sketch completion

Deductive approach to find such equivalent expressions

expression synthesis

how?

Given an expression E in the offline program,

synthesize an expression E' of the online program such that

E and E' are **equivalent modulo RFS**

expression synthesis

equivalence modulo RFS

An online expression E' equivalent to offline expression E modulo the RFS Φ iff

$$\Phi(xs, y) \models E' = E[(xs +\!\!+ [x])/xs]$$

Sketch

```
online_variance (v, s, sq, n) x =
    let new_s = □1
        new_n = □2
        avg = s / new_n
        new_sq = □3
    in (new_sq / new_n, new_s, new_sq, new_n)
```

Unknown	Specification
$\square_1$	`foldl (+) 0 xs`
$\square_2$	`length xs`
$\square_3$	`foldl (\acc x -> acc + (x-avg)^2) 0 xs`

Example: $\square_1$

Claim: $s + x$ is an equivalent expr

$$LHS = E' = s + x$$

$$\begin{aligned} RHS = E[(xs +\!\!+ [x])/xs] &= \mathsf{foldl}(+, 0, xs +\!\!+ [x]) \\ &= \mathsf{foldl}(+, 0, xs) + x \\ &= s + x = LHS \end{aligned}$$

expression synthesis

equivalence modulo RFS

An online expression E' equivalent to offline expression E modulo the RFS Φ iff

$$\Phi(xs, y) \models E' = E[(xs +\!\!+ [x])/xs]$$

💡 Find an implicate of form $\square = E'$ where E' is a term over $x, y_1, \ldots, y_n$

finding implicate

expression synthesis

Example

$$\begin{aligned} \Phi &\equiv y_1 = \mathrm{foldl}(+, 0, xs) \,/\, \mathrm{length}(xs) \wedge y_2 = \mathrm{length}(xs) \\ T &\equiv \square = \mathrm{foldl}(+, 0, xs +\!\!+ [x]) \end{aligned}$$

$$\mathcal{A} \equiv \mathrm{foldl}(+, 0, xs +\!\!+ [x]) = \mathrm{foldl}(+, 0, xs) + x$$

$$\Phi \wedge T \wedge \mathcal{A} \implies \square = (y_1 \times y_2) + x$$

expression synthesis

equivalence module RFS

An online expression E' equivalent to offline expression E modulo the RFS Φ iff

$$\Phi(xs, y) \models E' = E[(xs \mathbin{+\!\!+} [x])/xs]$$

💡 Find an implicate of form $\square = E'$ where E' is a term over $x, y_1, \ldots, y_n$

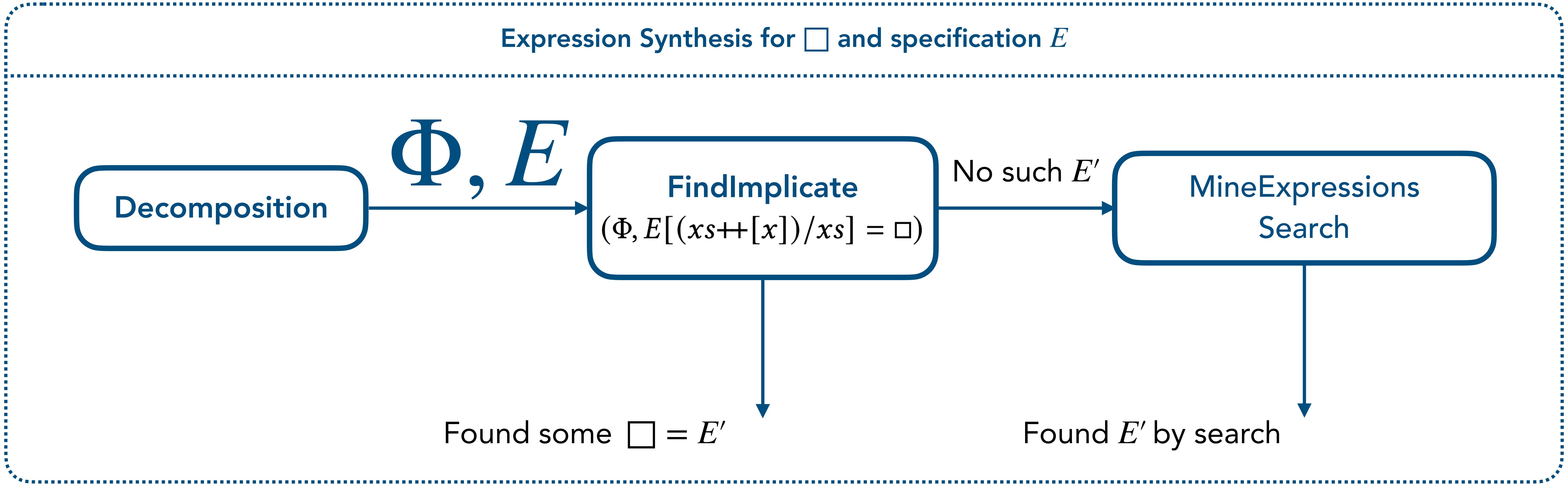
⚠ RFS and offline expression E contain higher-order combinators

expression synthesis

synthesis workflow

An online expression E' equivalent to offline expression E modulo the RFS Φ iff

$$\Phi(xs, y) \models E' = E[(xs +\!\!+ [x])/xs]$$



how well does it work?

benchmark

Source	Description	Example	# of Benchmarks
Statistics	Statistical computations from SciPy and OnlineStats.jl	skewness geometric mean LogSumExp	34
Auctions	Online auction queries from Nexmark	generating bidding reports monitoring new bidders determining top-k bids	18

52 tasks
collected

24
median AST size (offline, statistics)

42
median AST size (offline, auctions)

39
median AST size (online, statistics)

44
median AST size (online, auctions)

baseline comparison

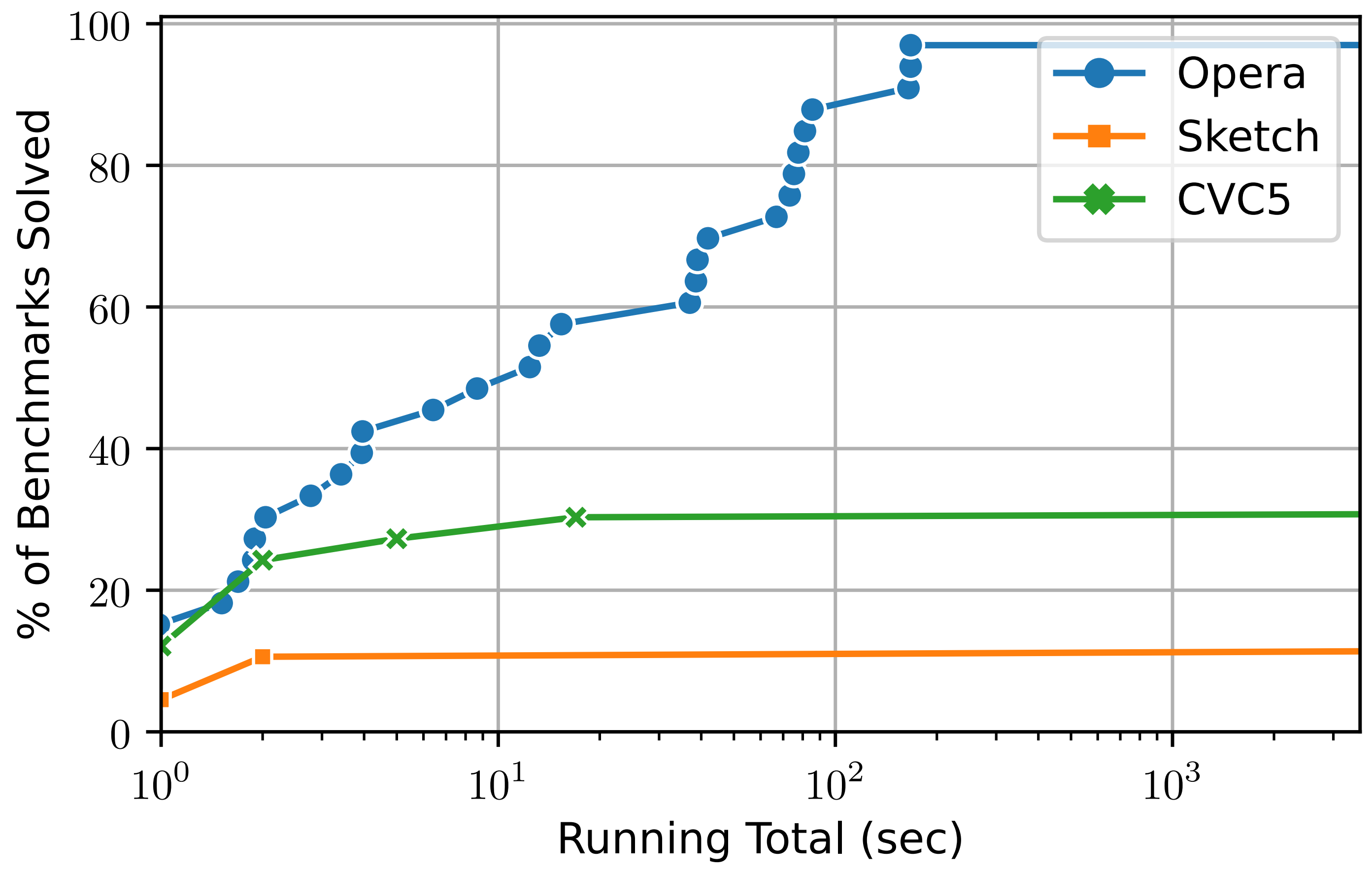
evaluation

Existing Baselines

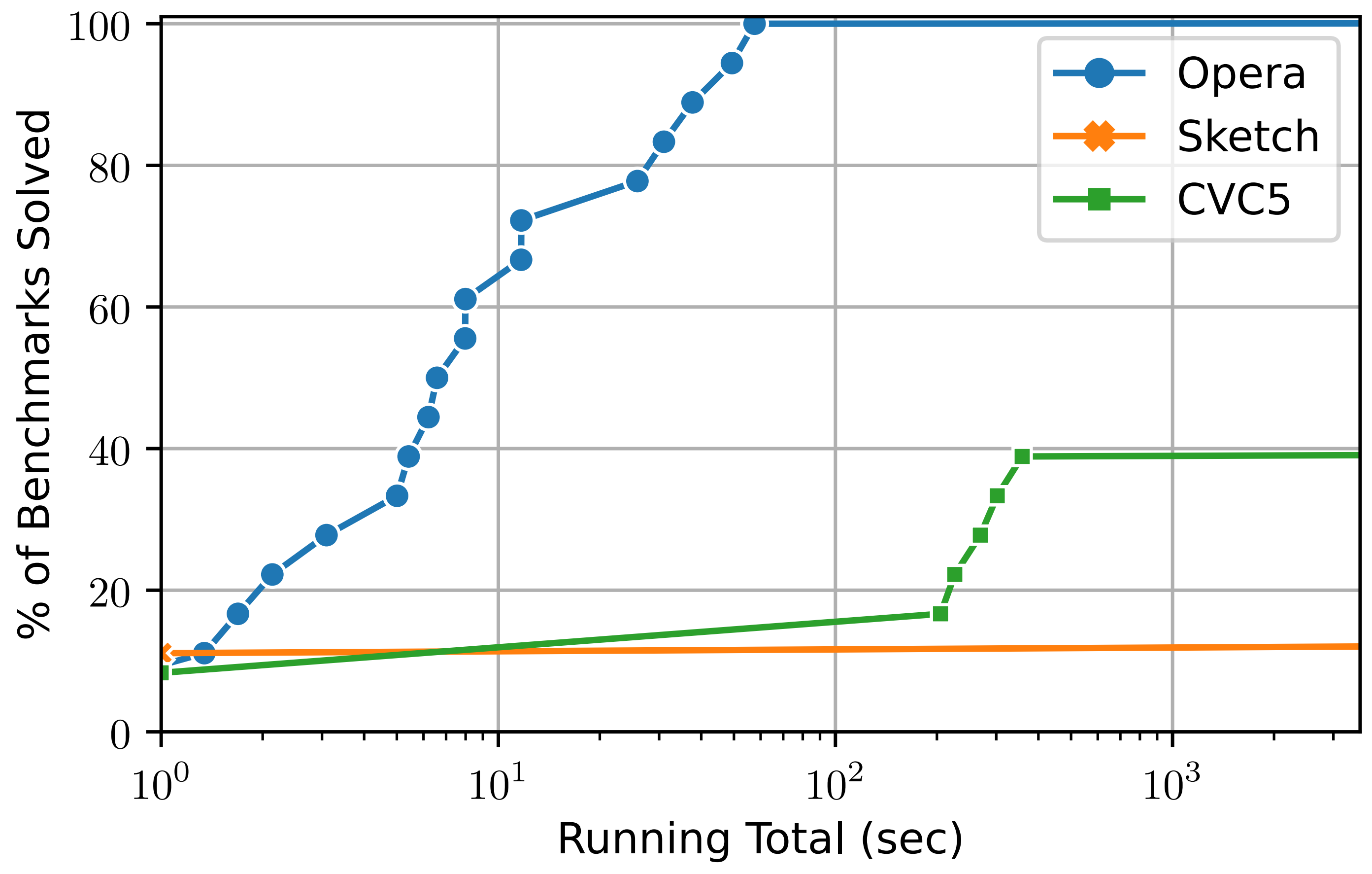
CVC5

Sketch

Statistics



Auctions



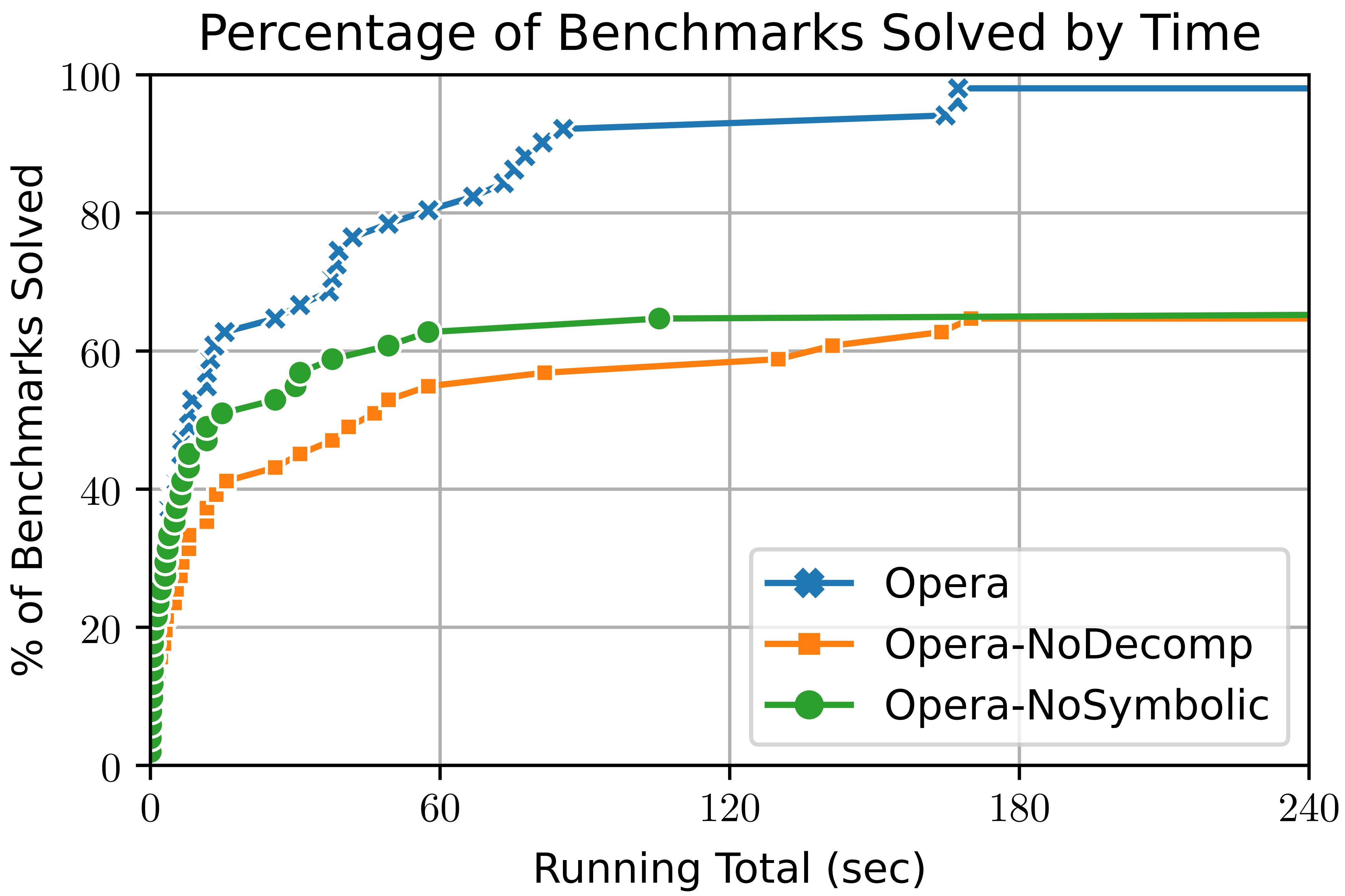
synthesized
50 out of 51
online schemes

average
25.0 s
running time

Opera synthesizes
2.6× more than CVC5
7.2× more than Sketch

ablation study

evaluation



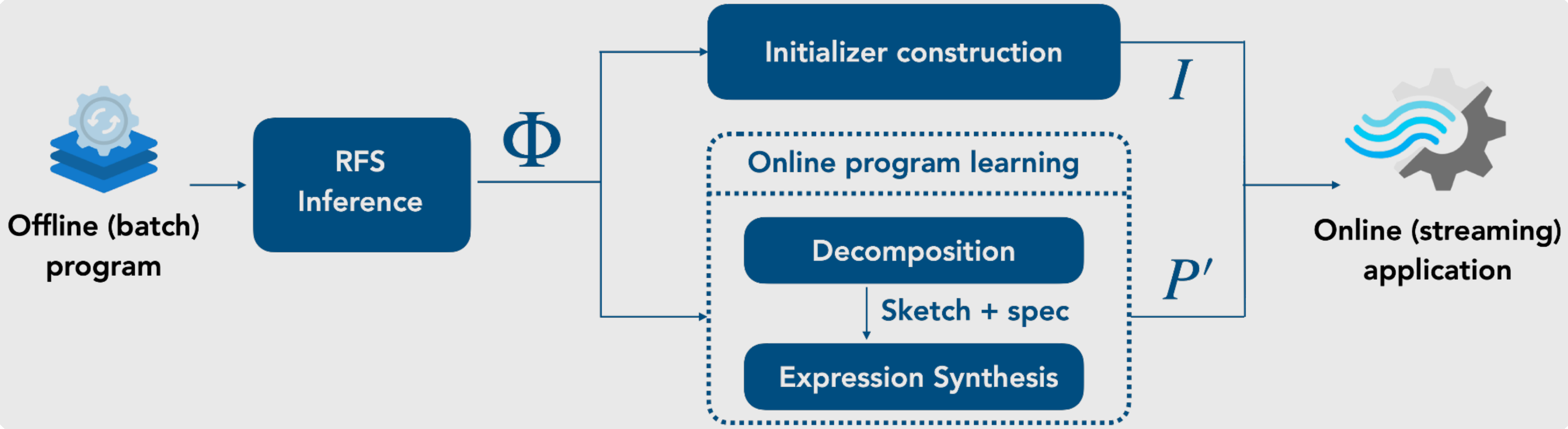
RFS-driven synthesis baselines

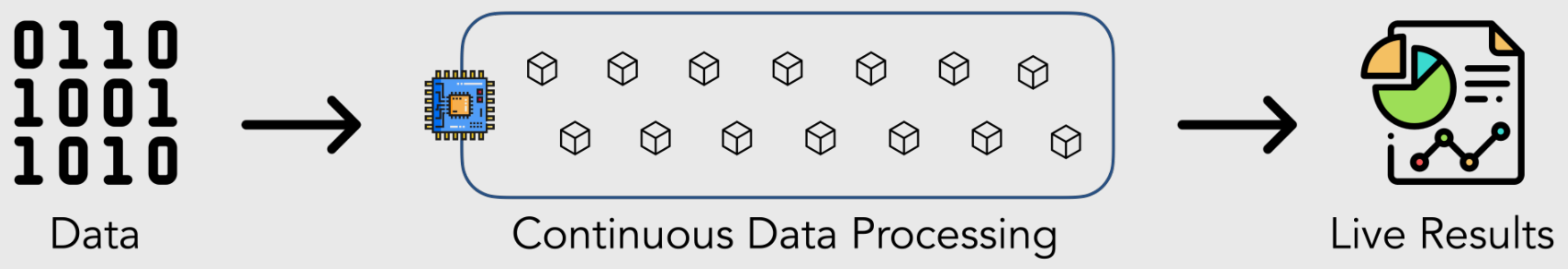
Opera-NoDecomp

compositional synthesis disabled

Opera-NoSymbolic

enumerative search only





Streaming/Online

synthesizes

50 out of 51

online scheme

Try Opera!

Reuse offline program in online synthesis

Replace offline exprs with equivalent online exprs

Relational Spec for Init and Sketch Completion

average

25.0 s

running time

52 benchmarks in 2 domains

2.6× more than CVC5

7.2× more than Sketch

Symbolic Reasoning ✓

Search ✓

Decomposition and **symbolic reasoning** significantly improve the performance of Opera