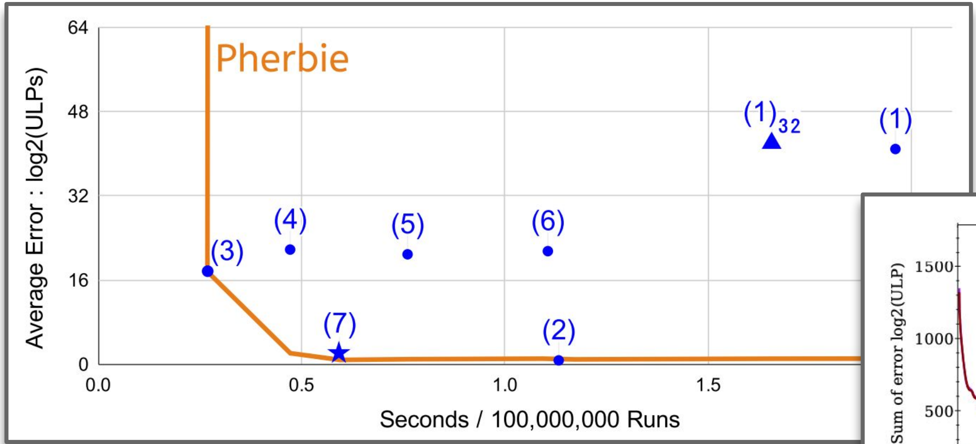
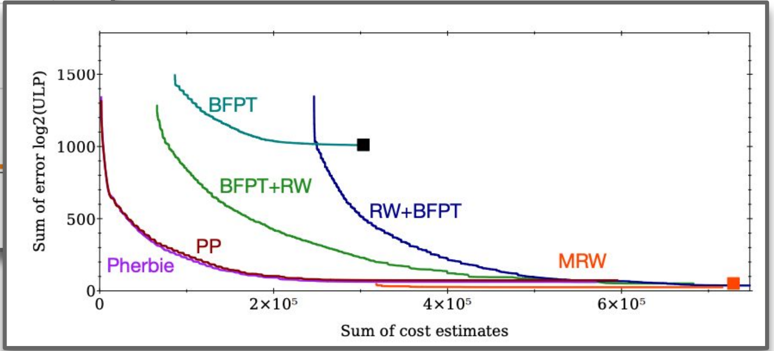
Combining Precision Tuning and Rewriting

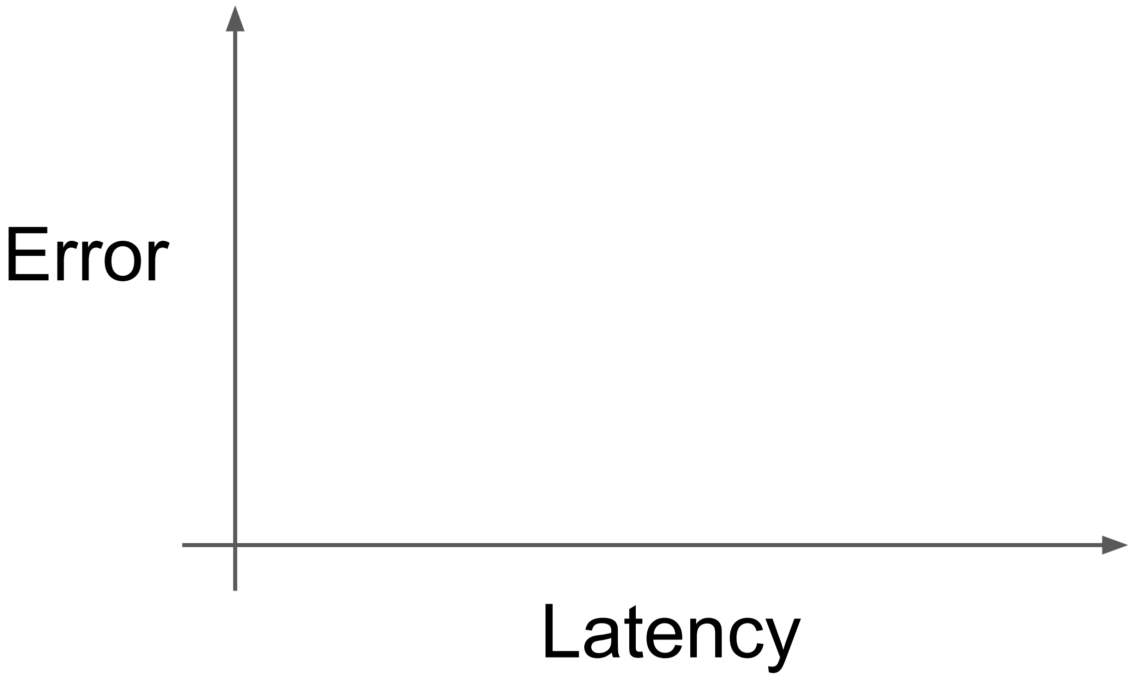


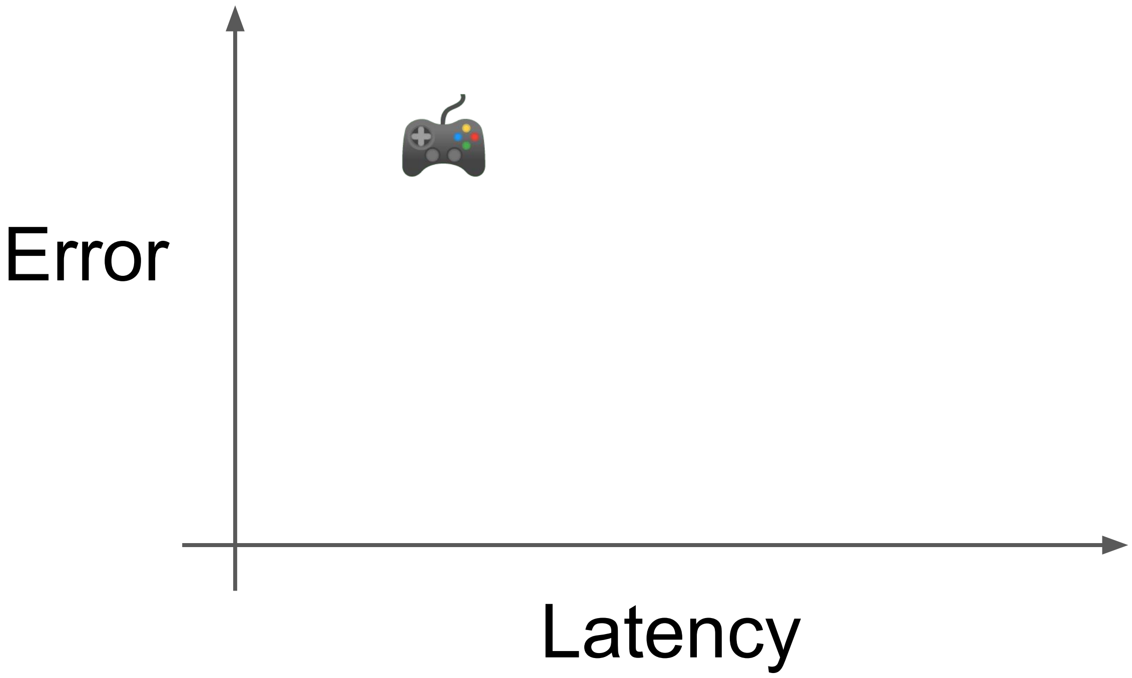


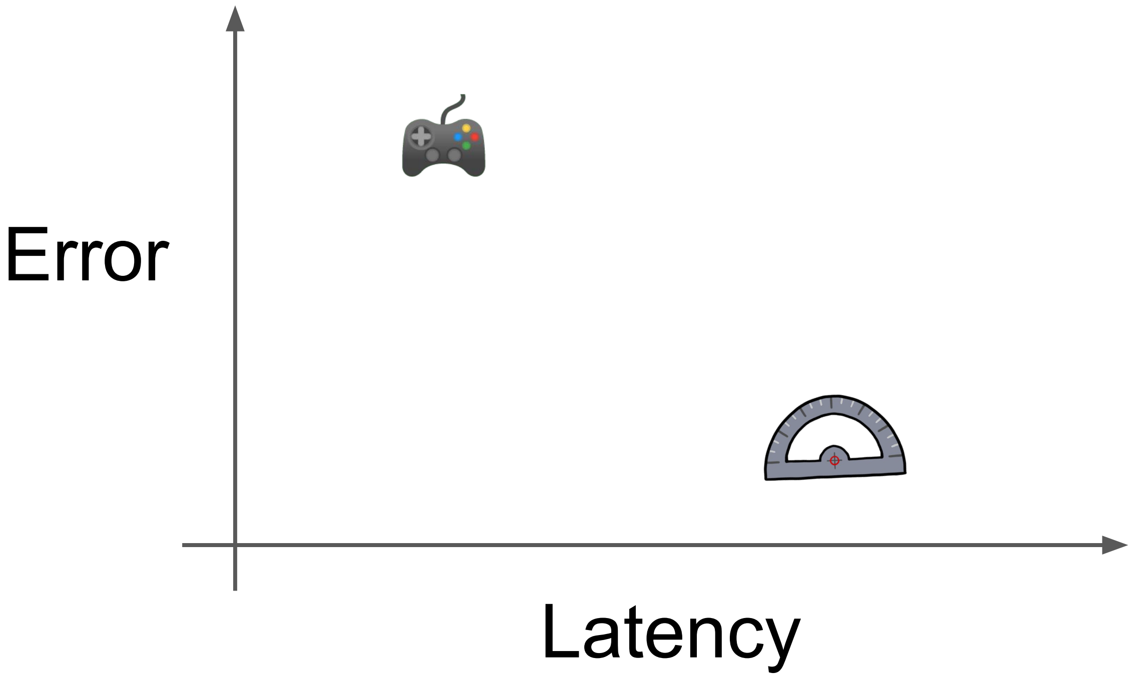
Brett Saiki, Oliver Flatt,

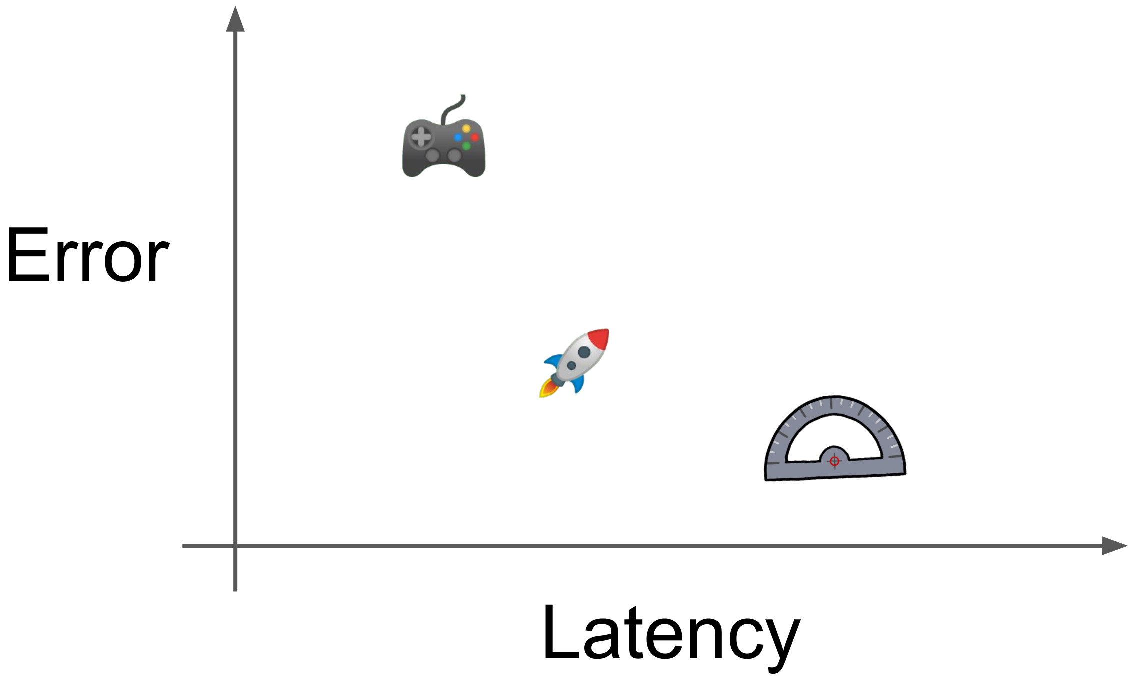
Chandrakana Nandi, Pavel Panchekha, Zachary Tatlock

HERBIE

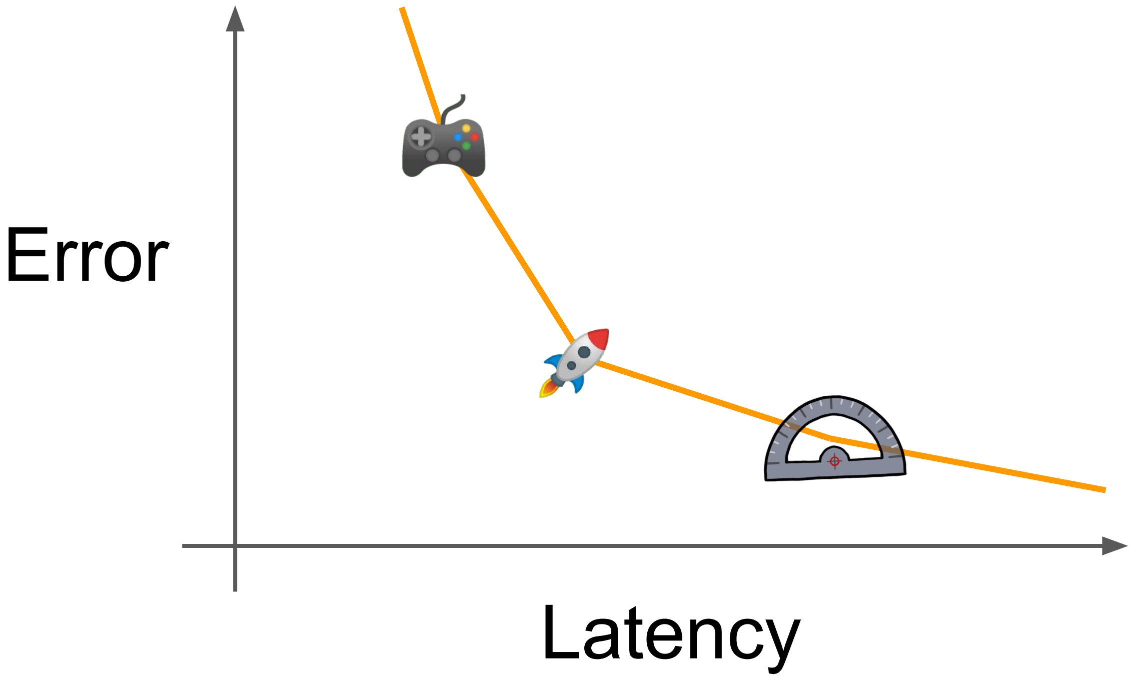

Error
Latency


Error
Latency


Error
Latency

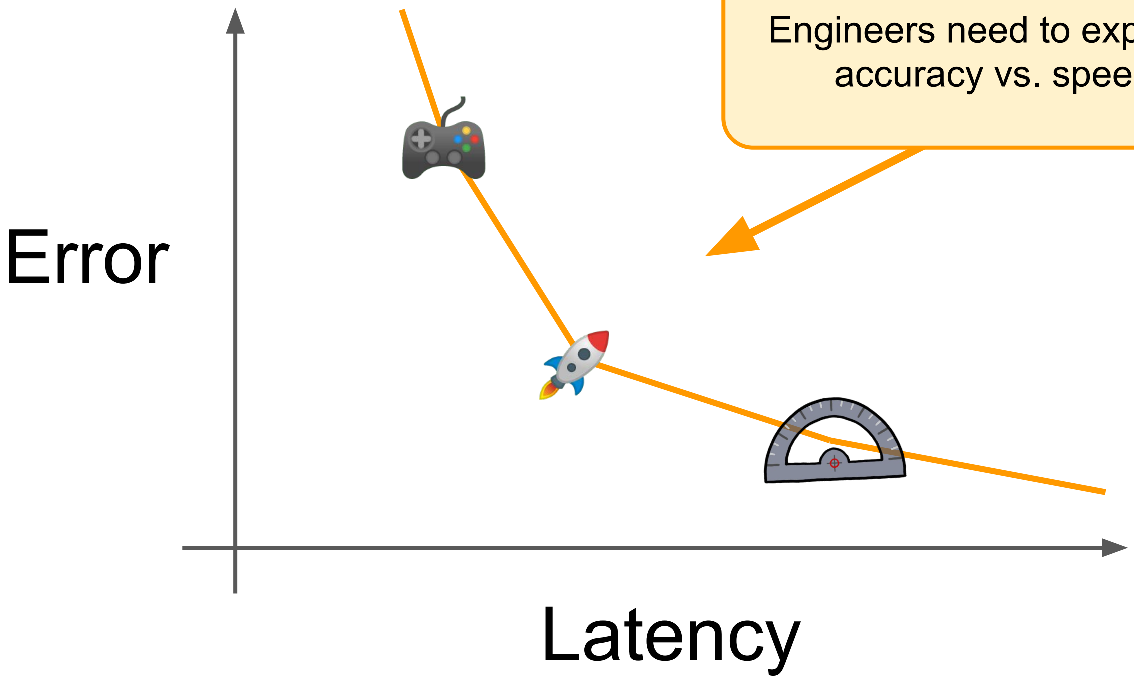


Error
Latency


Error
Latency

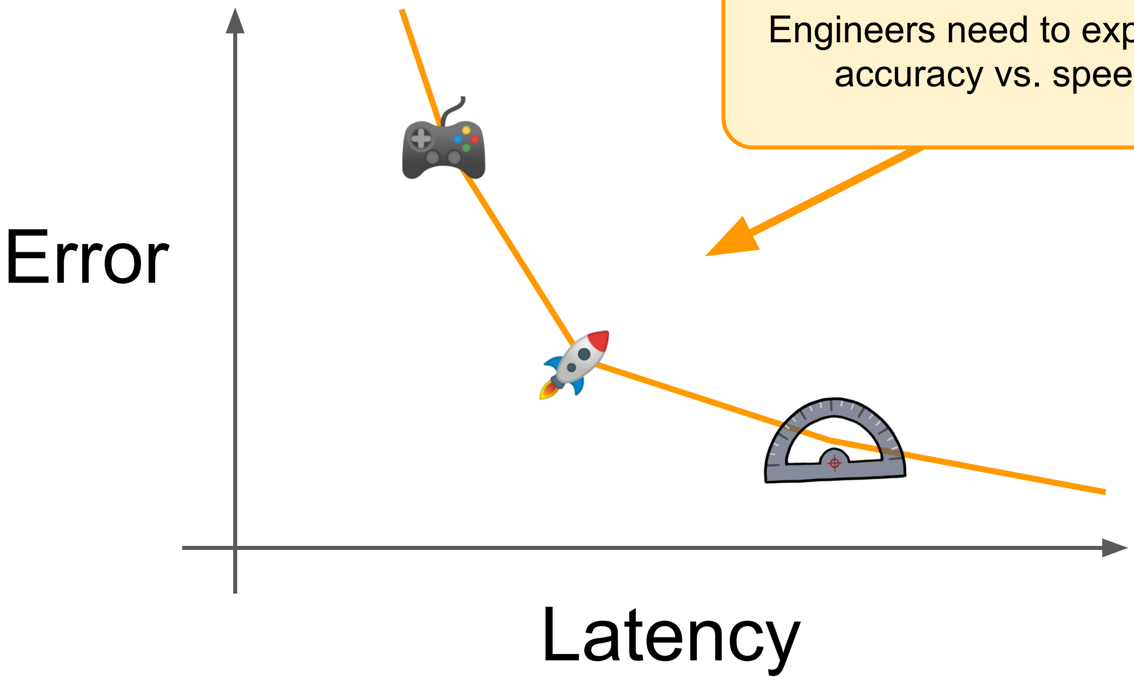
No one tradeoff is right for every application.

Engineers need to explore the Pareto frontier of optimal accuracy vs. speed candidate implementations!



No one tradeoff is right for every application.

Engineers need to explore the Pareto frontier of optimal accuracy vs. speed candidate implementations!

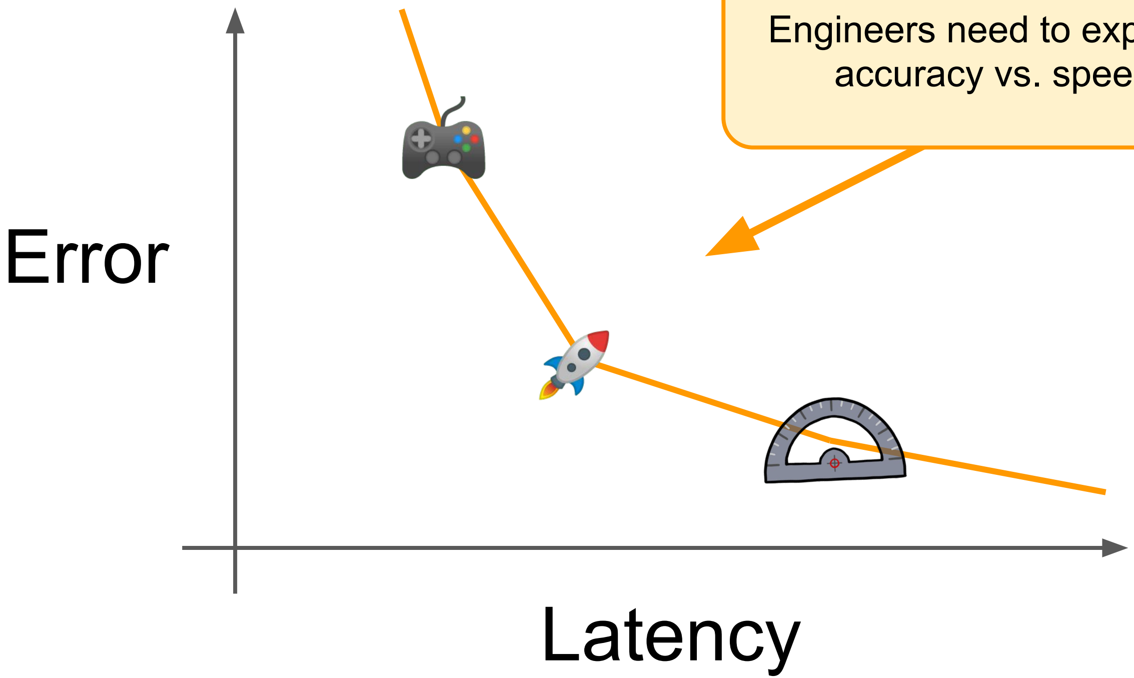


Precision Tuning

e.g., lower 64-bit ⇒ 32-bit

No one tradeoff is right for every application.

Engineers need to explore the Pareto frontier of optimal accuracy vs. speed candidate implementations!



Precision Tuning

e.g., lower 64-bit ⇒ 32-bit

Program Rewriting

e.g., take series expansion

Precision Tuning

Program Rewriting

Precision Tuning

Program Rewriting

Lower bitwidth ⇒ higher throughput

- Major barrier: the memory wall!
- Enable more vectorization, etc.

Difficult to tell where lowering is safe

- Accums. large, but elts small?
- Past work adapts *delta debugging*
    - [Khalifa et al. FTSCS '19]
    - [Rubio-González et al. SC '13]

Precision Tuning

Lower bitwidth ⇒ higher throughput

- Major barrier: the memory wall!
- Enable more vectorization, etc.

Difficult to tell where lowering is safe

- Accums. large, but elts small?
- Past work adapts *delta debugging*
    - [Khalifa et al. FTSCS '19]
    - [Rubio-González et al. SC '13]

Program Rewriting

Avoid pitfalls and/or use coarser approx

- Avoid cancellation, intro series
- e.g., generally want `(x + 1) - x ⇒ 1`

Difficult to find / carry out good rewrites

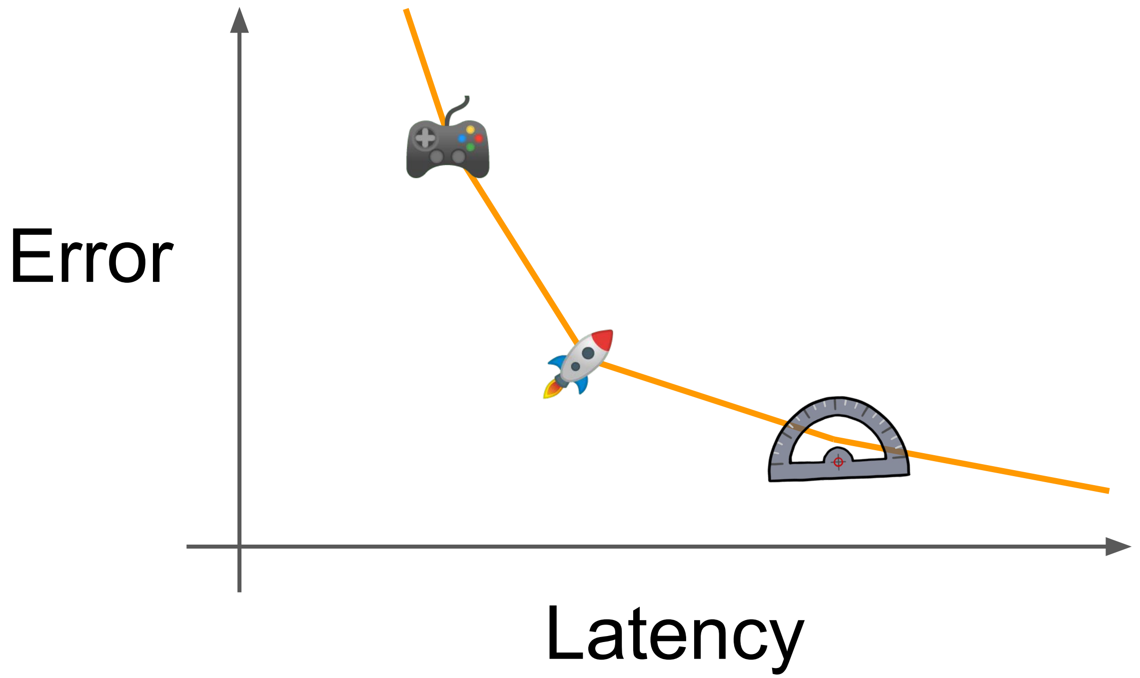
- Need to guide rewrite search
- Past work applies PL *synthesis*
    - [Schkufza et al. PLDI '14]
    - [Panchekha et al. PLDI '15]

How to optimize?

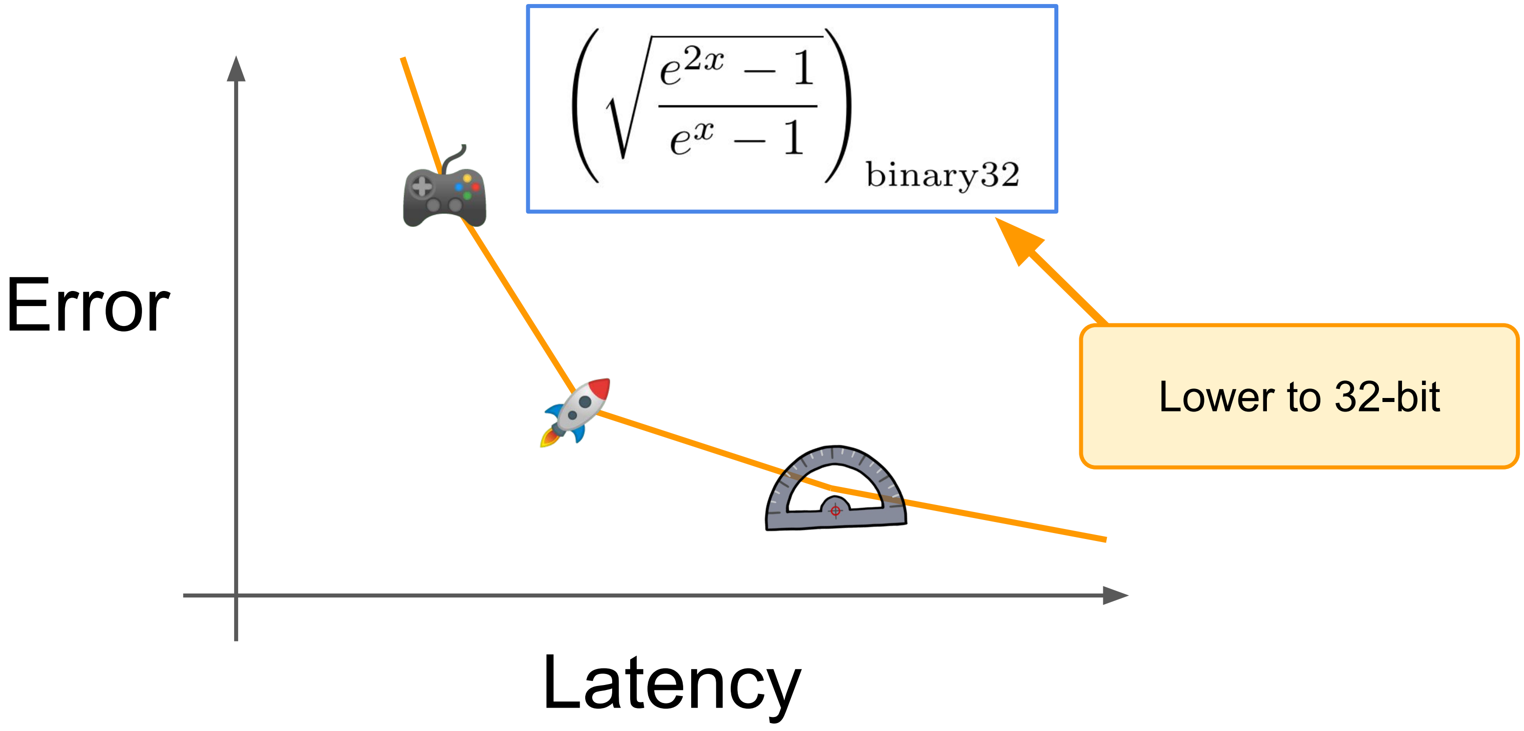
$$\sqrt{\frac{e^{2x} - 1}{e^x - 1}}$$

How to optimize $\sqrt{\frac{e^{2x} - 1}{e^x - 1}}$ via precision tuning?

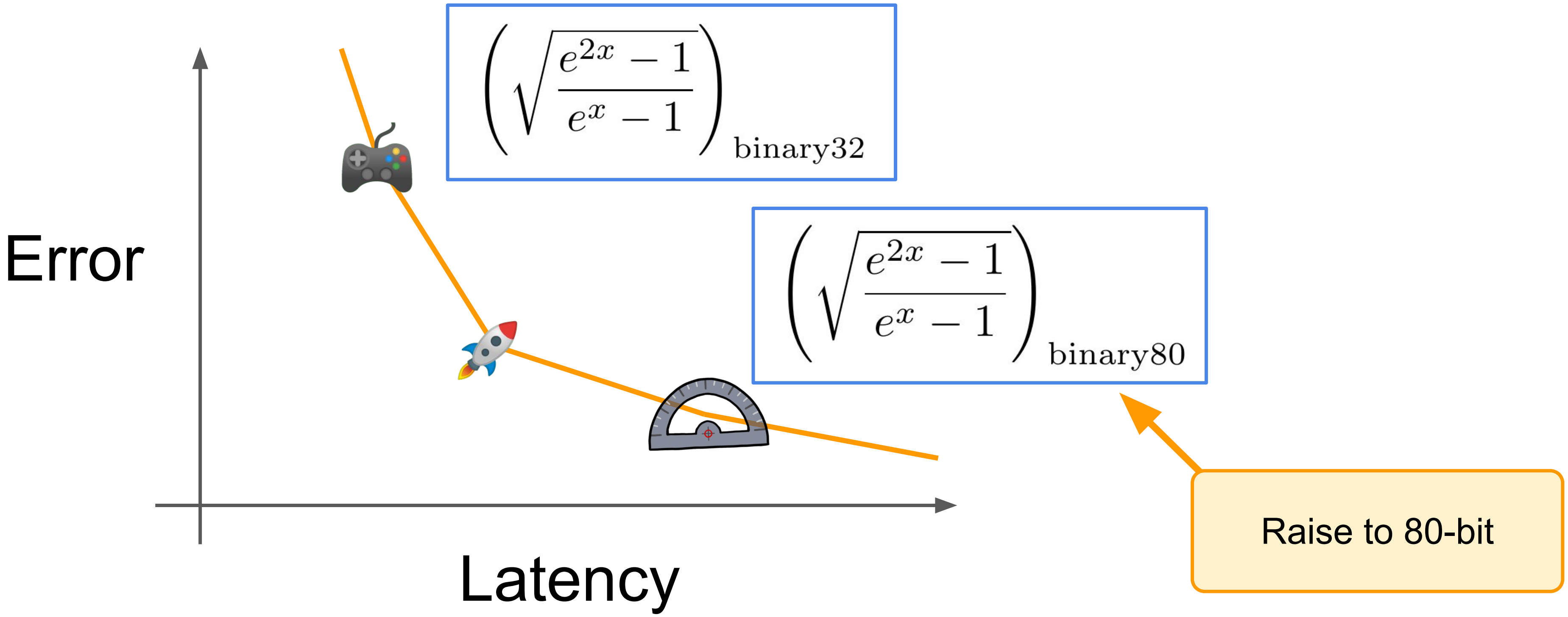
How to optimize $\sqrt{\frac{e^{2x}-1}{e^{x}-1}}$ via precision tuning?



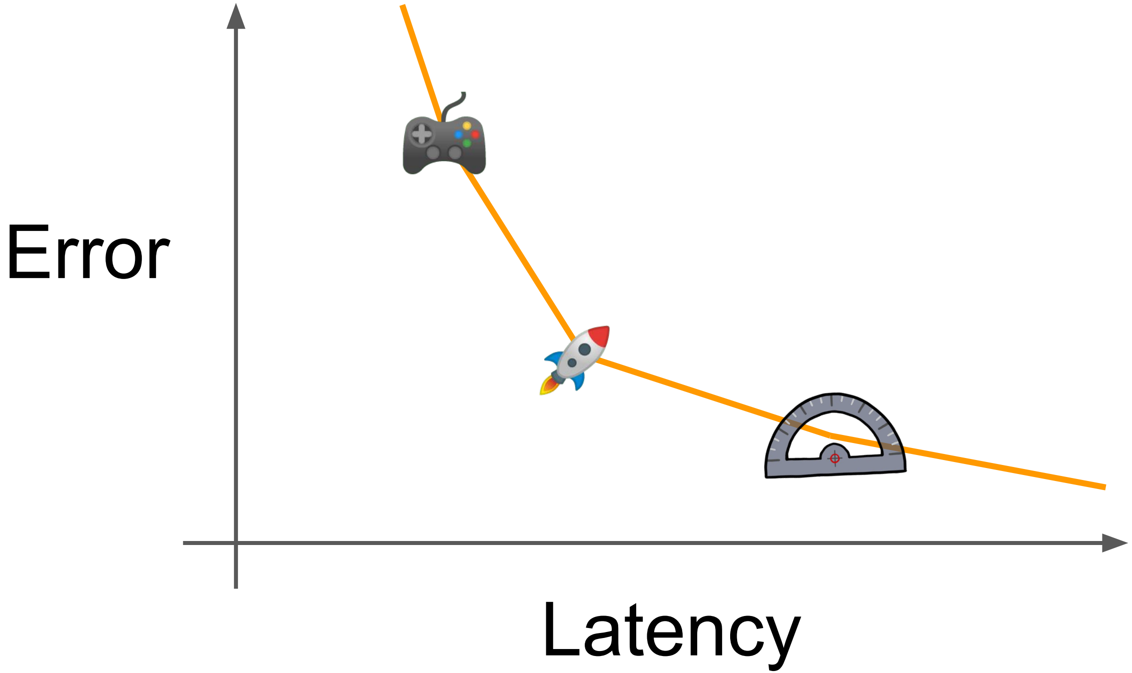
How to optimize $\sqrt{\frac{e^{2x}-1}{e^{x}-1}}$ via precision tuning?



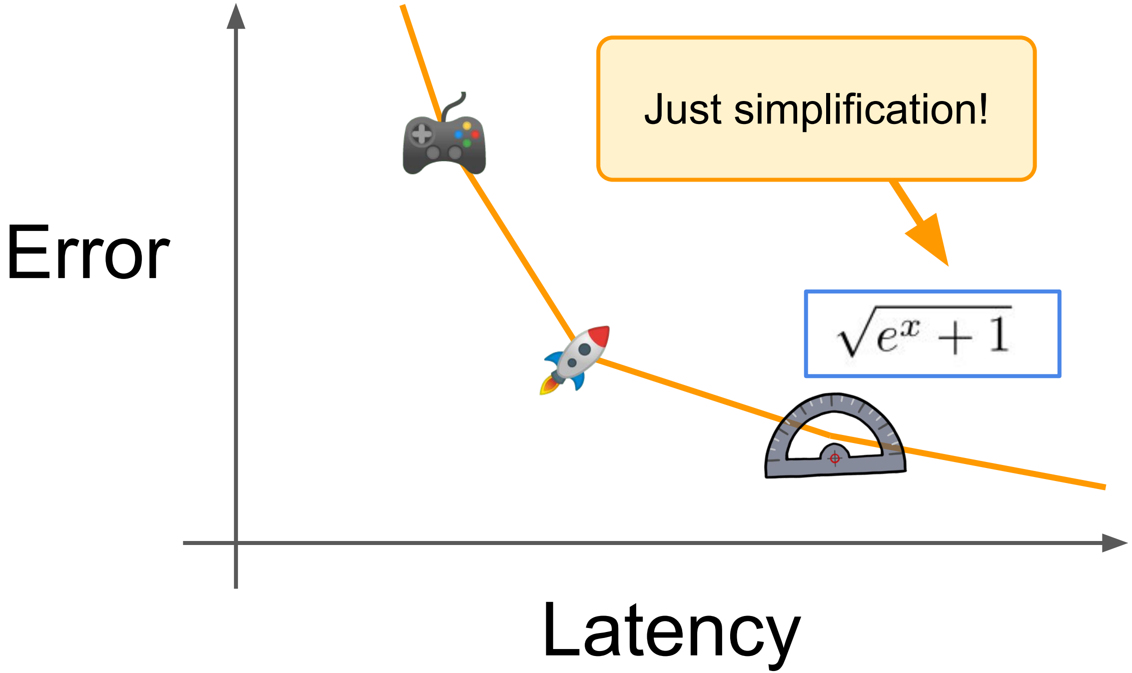
How to optimize $\sqrt{\frac{e^{2x}-1}{e^x-1}}$ via precision tuning?



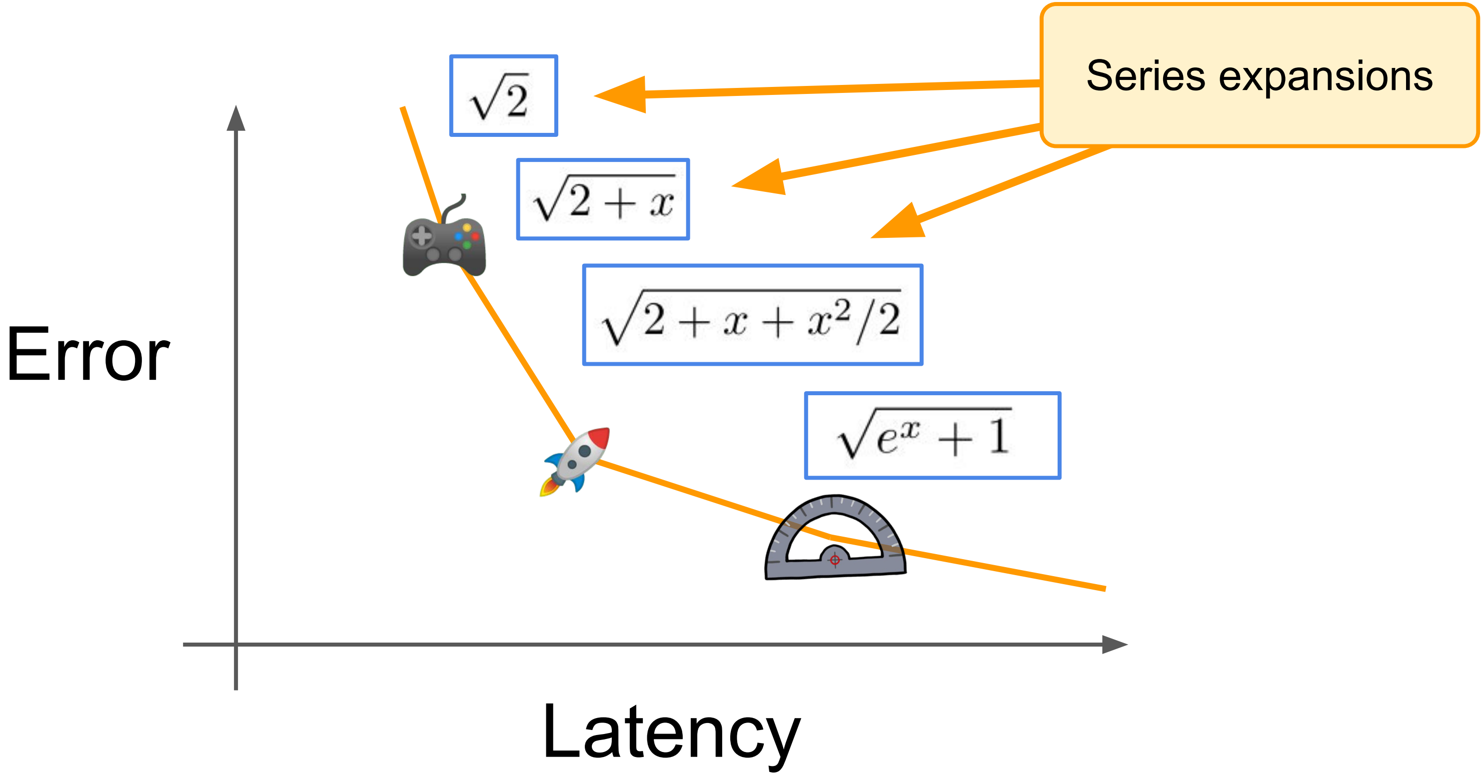
How to optimize $\sqrt{\dfrac{e^{2x} - 1}{e^{x} - 1}}$ via rewriting?



How to optimize $\sqrt{\frac{e^{2x}-1}{e^x-1}}$ via rewriting?



How to optimize $\sqrt{\frac{e^{2x}-1}{e^x-1}}$ via rewriting?



Optimizing in general: precision tuning **OR** rewriting ?

Optimizing in general: precision tuning **OR** rewriting ?

When and how to use?

- Tune then rewrite?
- Rewrite then tune?
- Alternate? Run to fixpoint?
- Share accuracy analyses?

Optimizing in general: precision tuning **AND** rewriting **!**

How to optimize $\sqrt{\frac{e^{2x}-1}{e^x-1}}$ via precision tuning **AND** rewriting !

$$
\begin{aligned}
&\textbf{if} \;\; |x| \leq 0.05 : \\
&\quad \sqrt{2+x} \\
&\textbf{else} : \\
&\quad \sqrt{\langle (e^x+1)_{\text{binary32}} \rangle_{\text{binary64}}}
\end{aligned}
$$

How to optimize $\sqrt{\frac{e^{2x}-1}{e^{x}-1}}$ via precision tuning **AND** rewriting !

Different techniques for different inputs

Sometimes just rewrite

$$
\begin{aligned}
&\textbf{if } |x| \leq 0.05 : \\
&\quad \sqrt{2+x} \\
&\textbf{else} : \\
&\quad \sqrt{\langle (e^{x}+1)_{\text{binary32}} \rangle_{\text{binary64}}}
\end{aligned}
$$

Sometimes rewrite + tune

Optimizing in general: precision tuning **AND** rewriting **!**

Our Result

Combine precision tuning and rewriting to produce a rich set of Pareto-optimal accuracy versus speed trade-offs.

Optimizing in general: precision tuning **AND** rewriting !

Our Result

Combine precision tuning and rewriting to produce a rich set of Pareto-optimal accuracy versus speed trade-offs.

Key Insights:

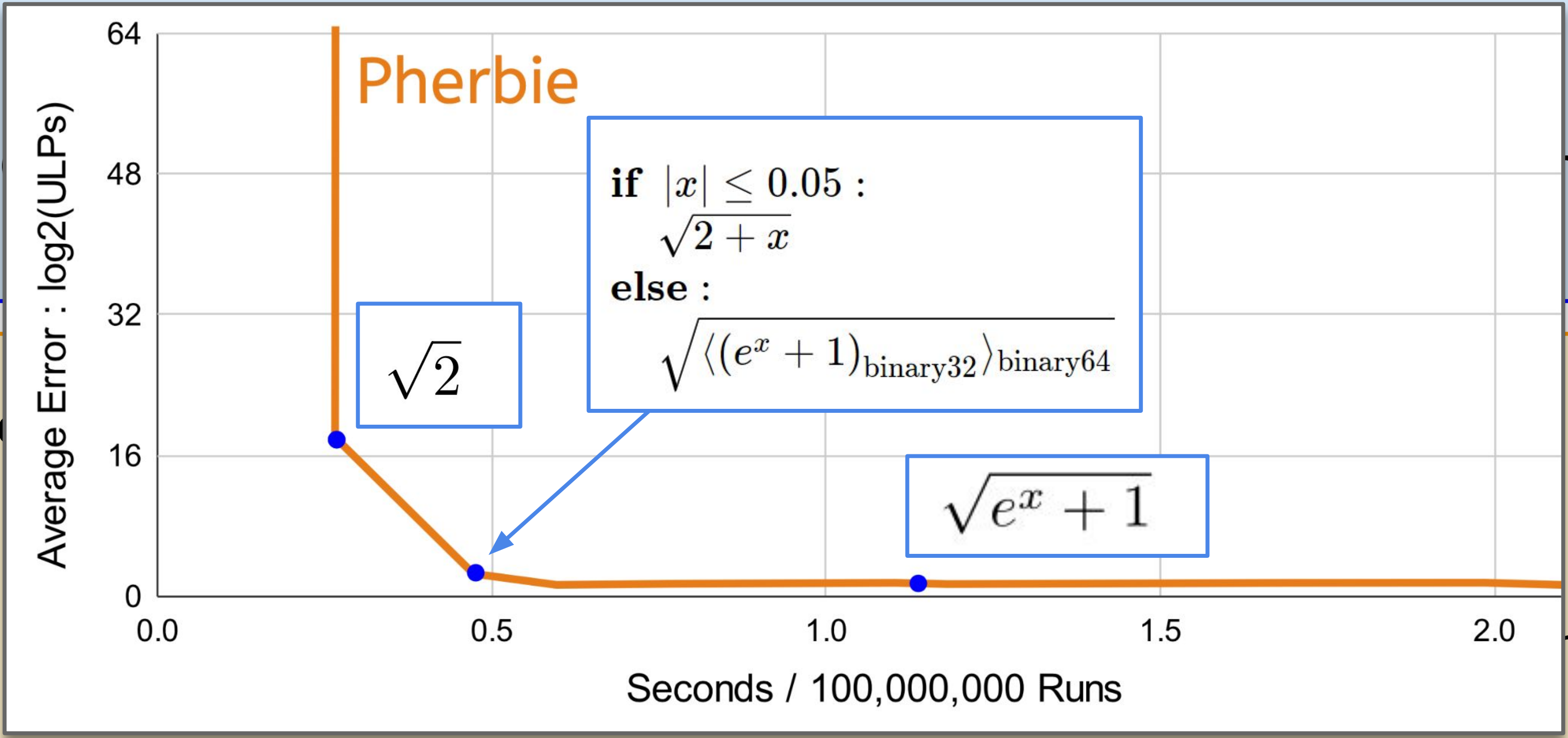
- Finer-grained interleavings ⇒ better Pareto frontiers
- Precision tuning can be rephrased as a rewriting problem
- “Local Error Analysis” helps both precision tuning and rewriting

Optimizing in general: precision tuning **AND** rewriting !

Ke

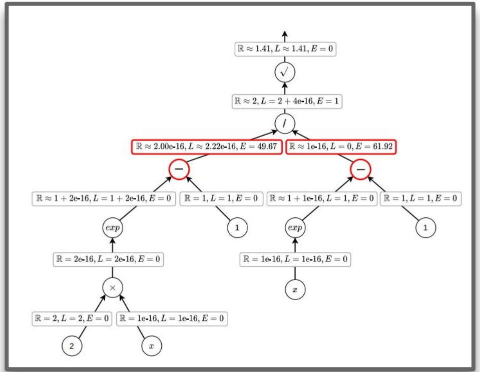
n

ng

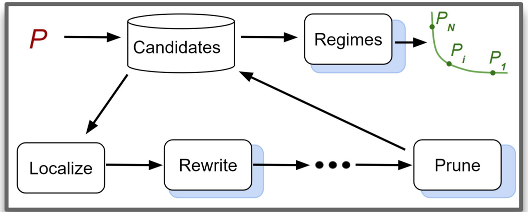


Outline

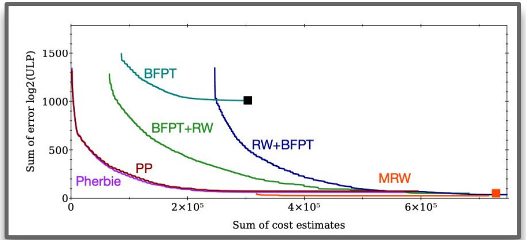
- Herbie: Improving Accuracy via Rewriting
  - Key Insight: local error guides rewriting



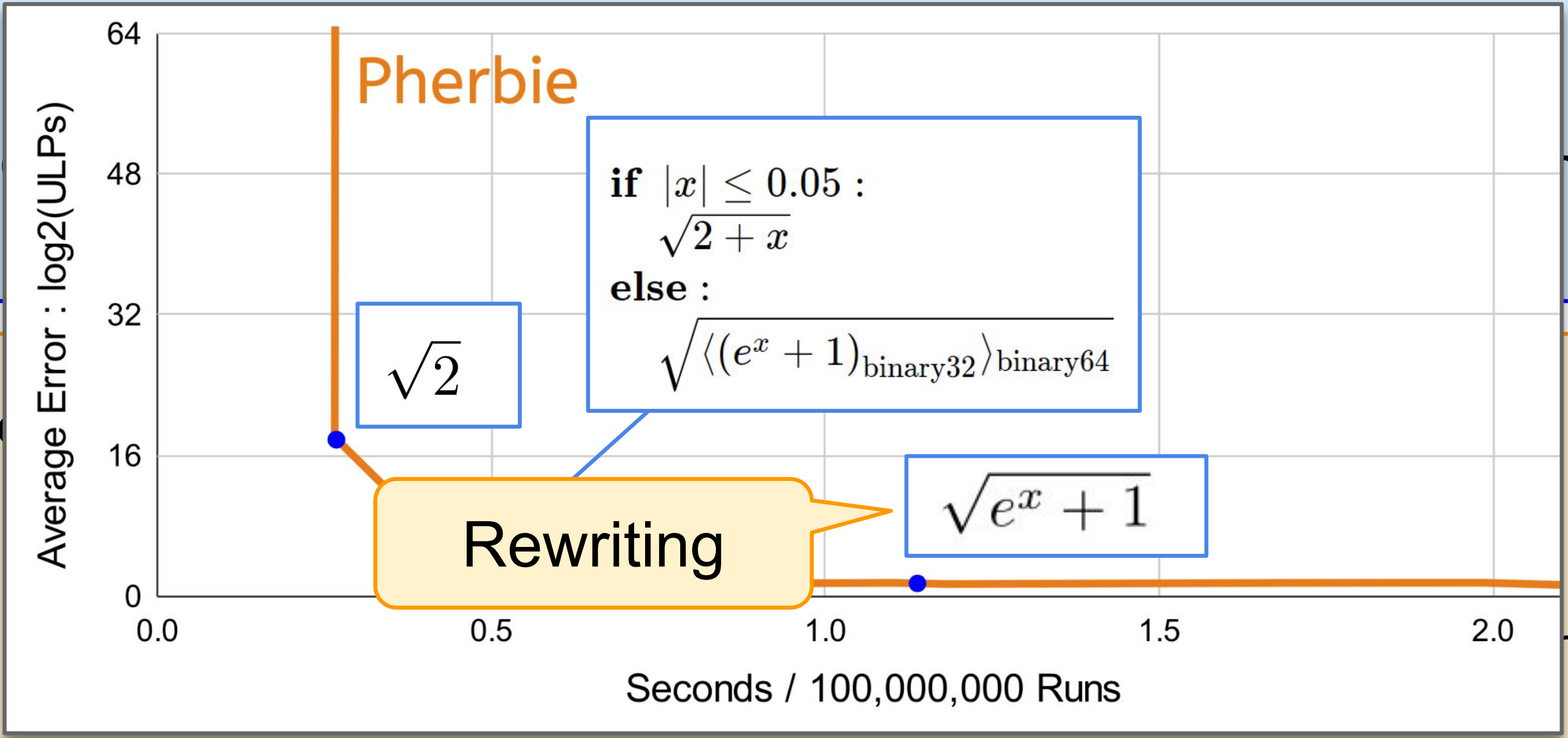
- Pherbie: Extending Herbie with Precision Tuning
  - Key Insight: local error also guides precision tuning!



- Evaluation: Applying Pherbie to Classics + Graphics
  - Key Insight: Finer-grained interleaving → better optimization!



Optimizing in general: precision tuning **AND** rewriting !



Developed continuously since 2015

Improves Accuracy Automatically

Rewriting Only

Input

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

Rewriting →

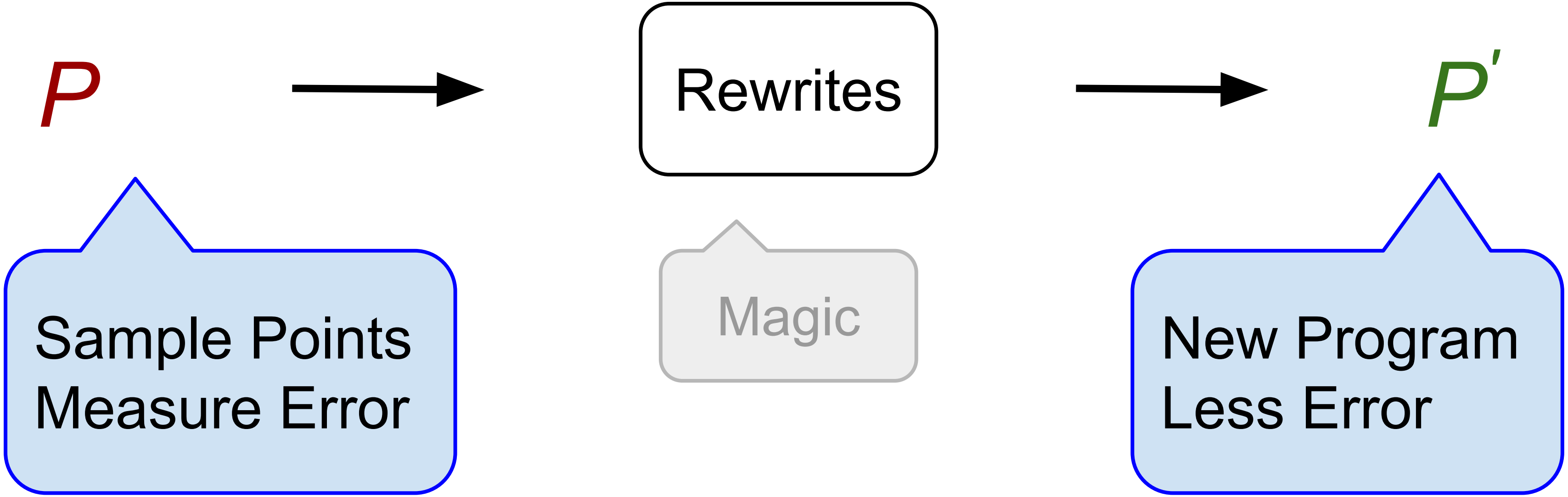
Output

$$\sqrt{e^{x} + 1}$$

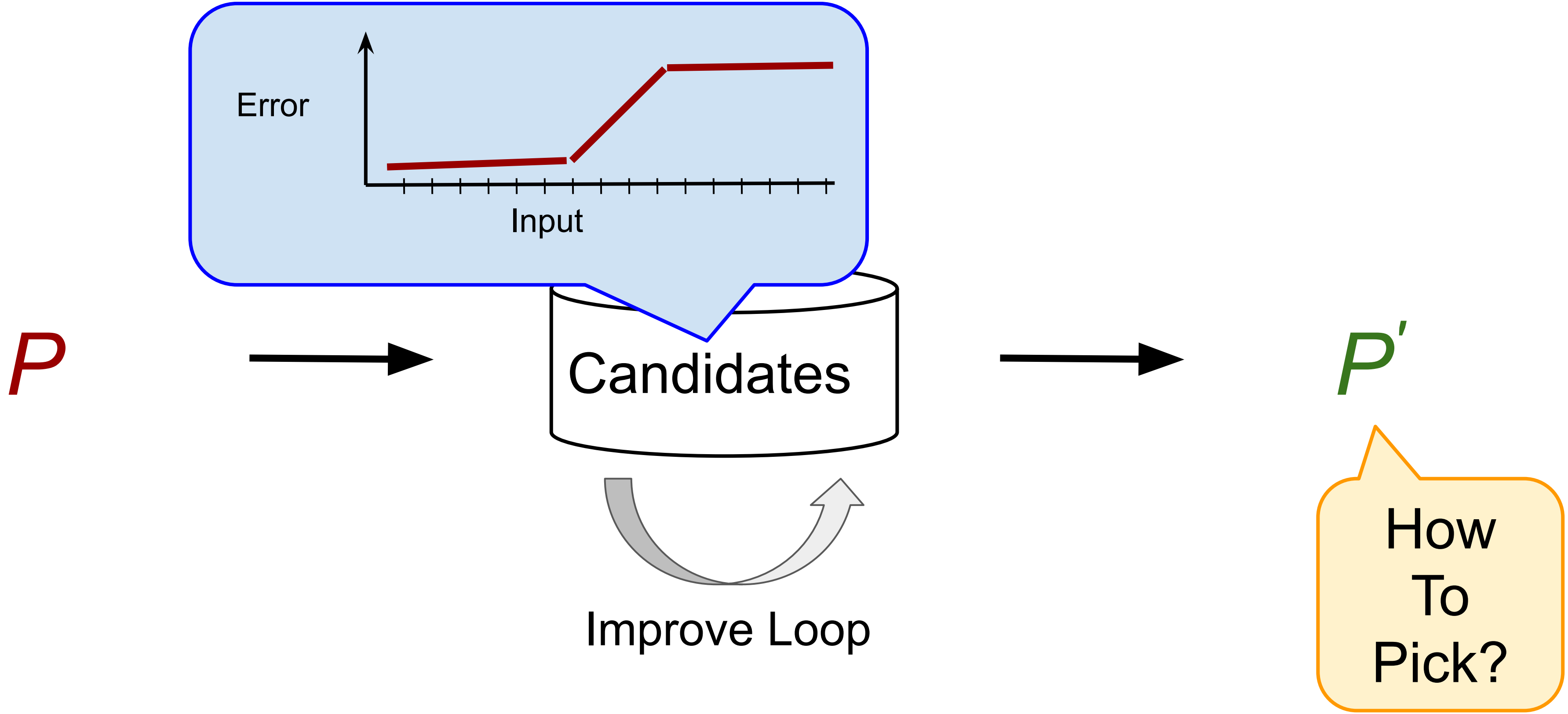
Developed continuously since 2015

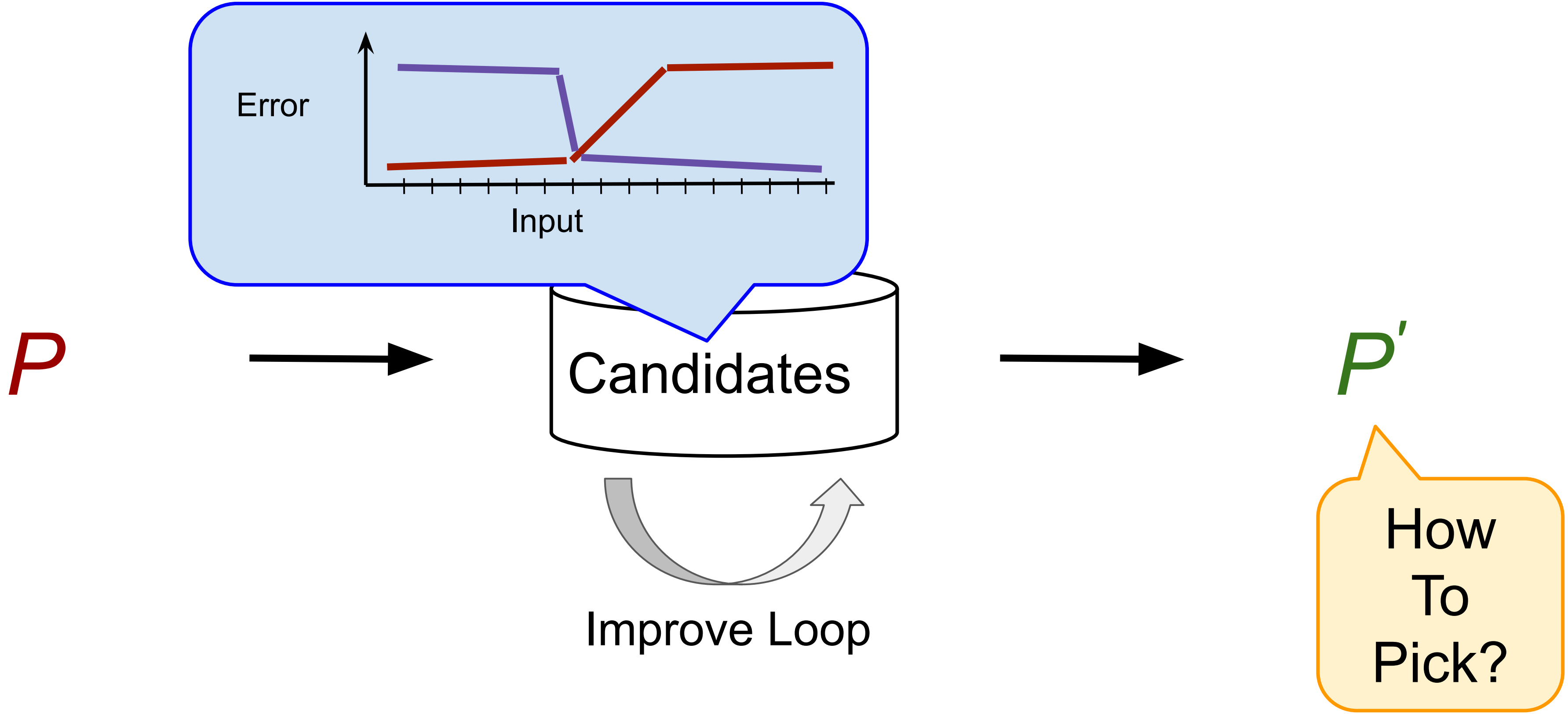
Improves Accuracy Automatically

Rewriting Only

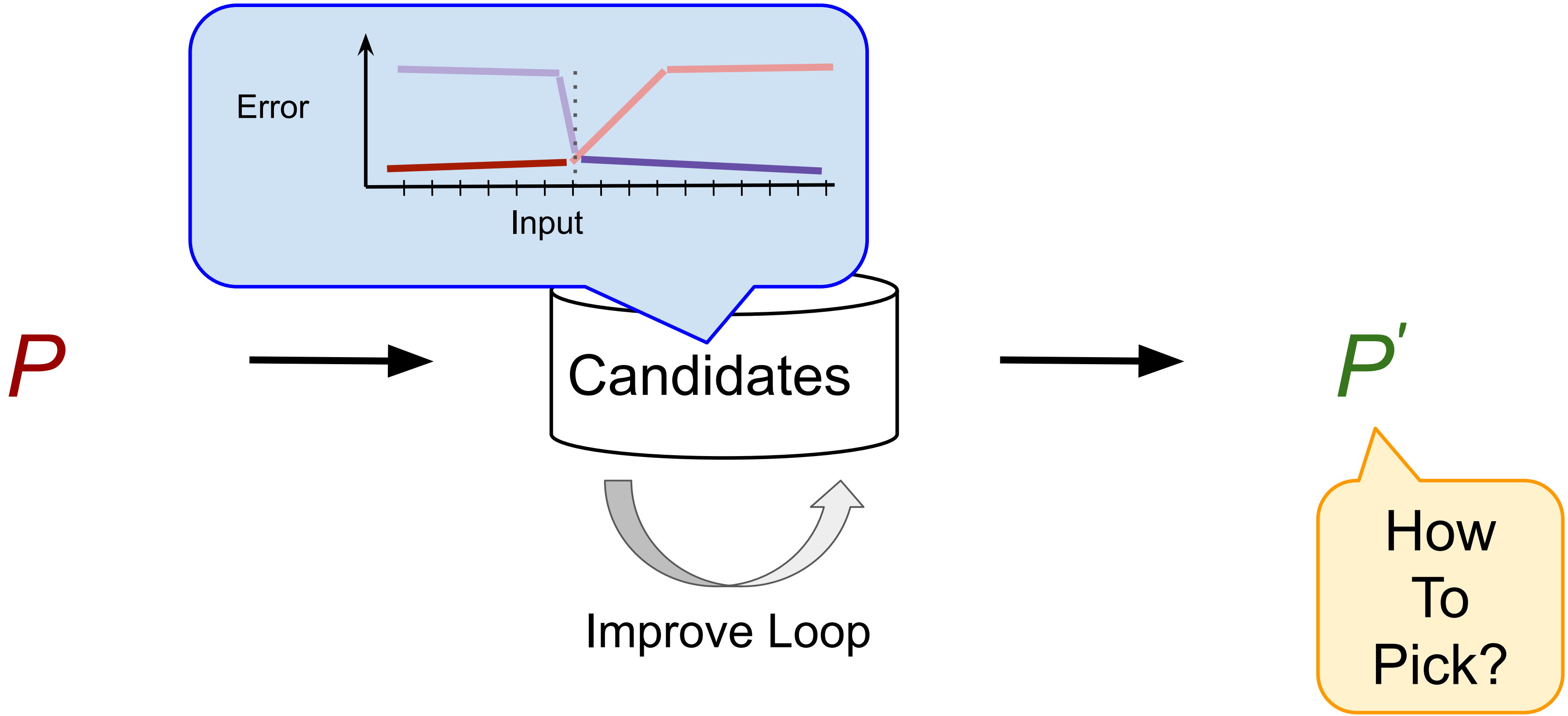


P
Rewrites
P'
Sample Points
Measure Error
Magic
New Program
Less Error

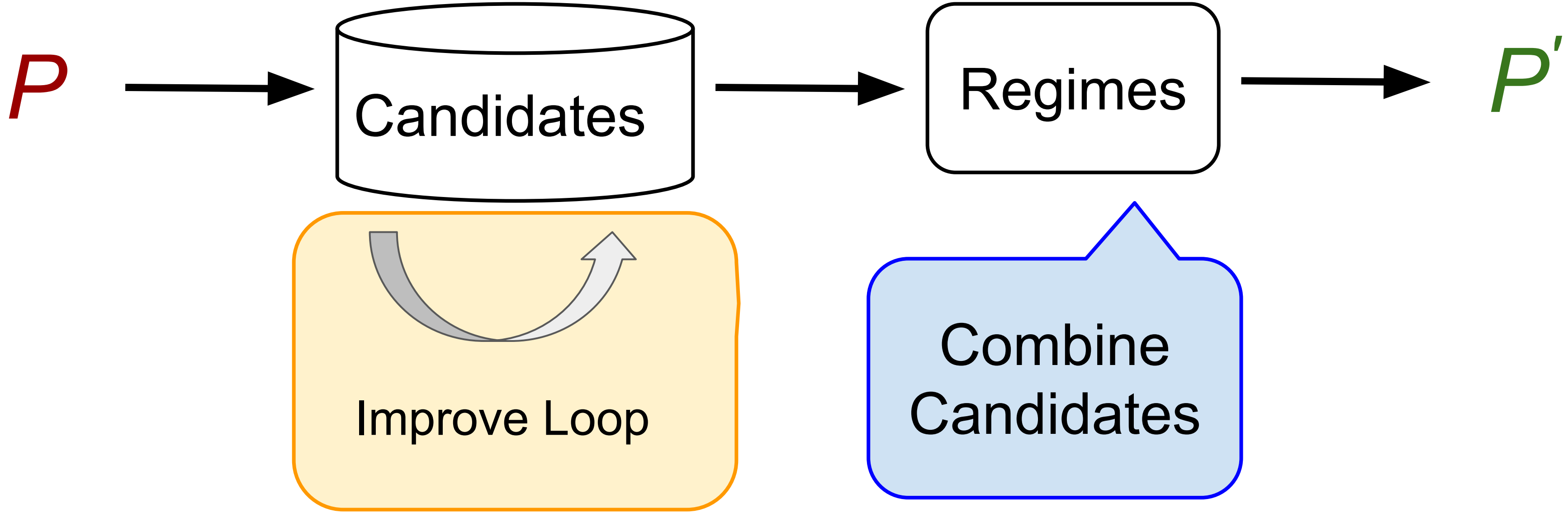

Error
Input
P
Candidates
P'
Improve Loop
How To Pick?

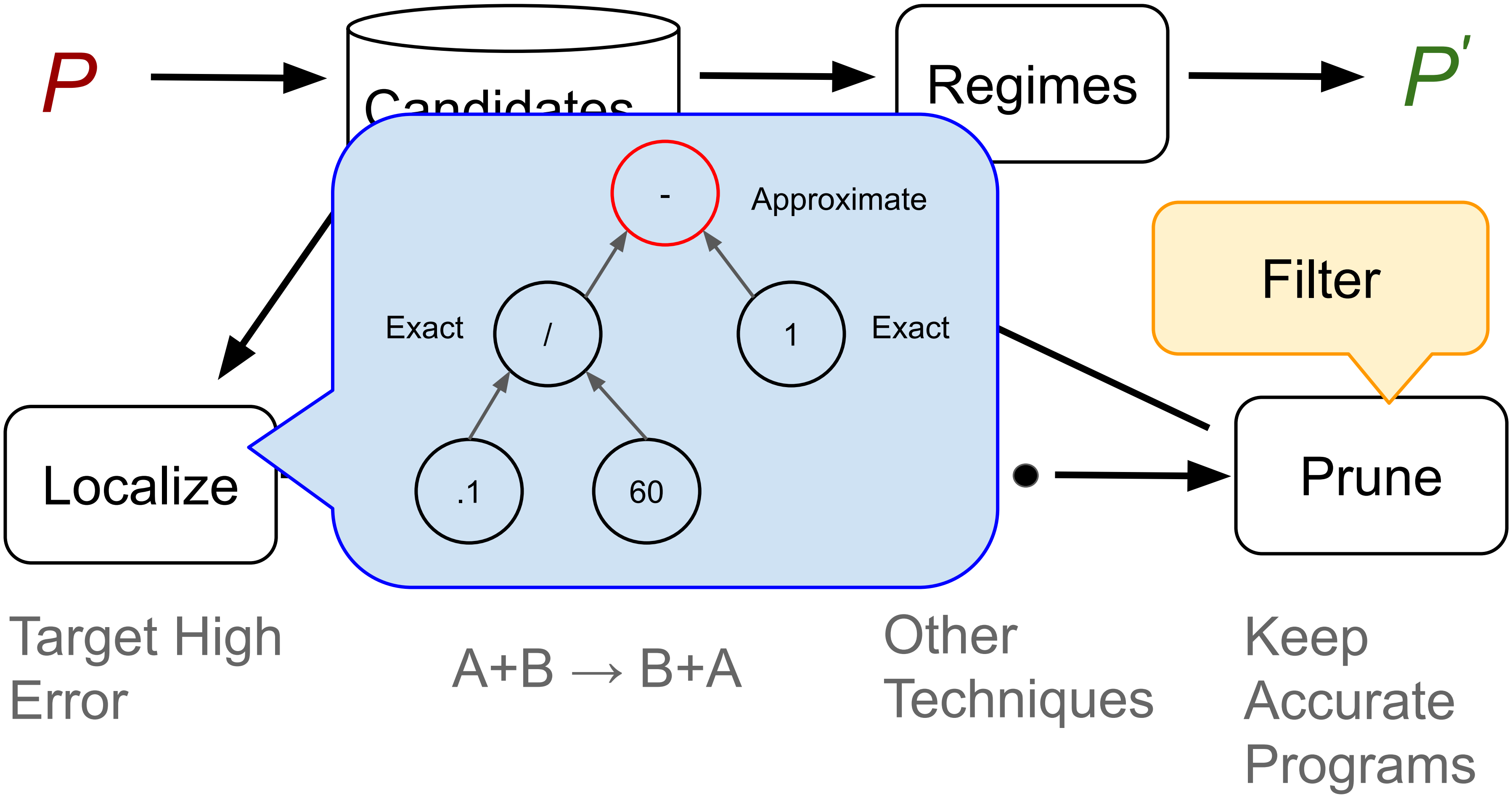


Error
Input
P
Candidates
P′
Improve Loop
How
To
Pick?

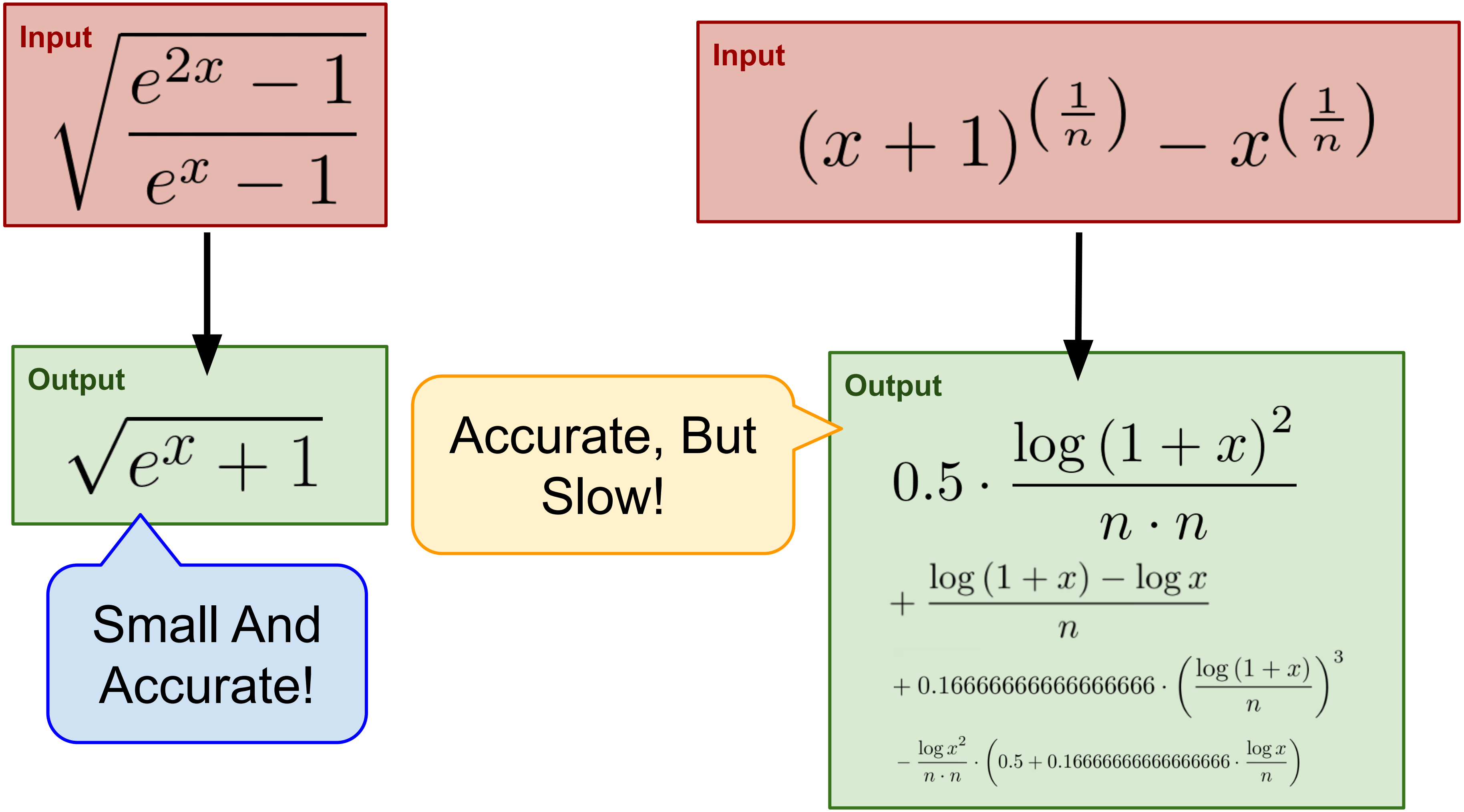


Error
Input
P
Candidates
P′
Improve Loop
How To Pick?


P
Candidates
Regimes
P'
Improve Loop
Combine
Candidates



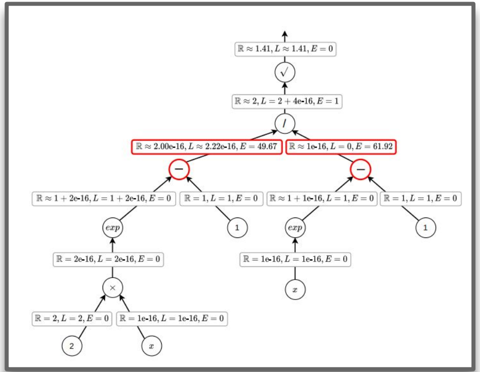
P
Candidates
Regimes
P′
Approximate
-
Exact
/
1
Exact
.1
60
Filter
Localize
Prune
Target High Error
A+B → B+A
Other Techniques
Keep Accurate Programs


Input
$\sqrt{\frac{e^{2x} - 1}{e^x - 1}}$
Output
$\sqrt{e^x + 1}$
Small And Accurate!
Input
$(x+1)^{\left(\frac{1}{n}\right)} - x^{\left(\frac{1}{n}\right)}$
Accurate, But Slow!
Output
$0.5 \cdot \frac{\log{(1+x)}^2}{n \cdot n} + \frac{\log{(1+x)} - \log x}{n} + 0.16666666666666666 \cdot \left(\frac{\log{(1+x)}}{n}\right)^3 - \frac{\log x^2}{n \cdot n} \cdot \left(0.5 + 0.16666666666666666 \cdot \frac{\log x}{n}\right)$

Outline

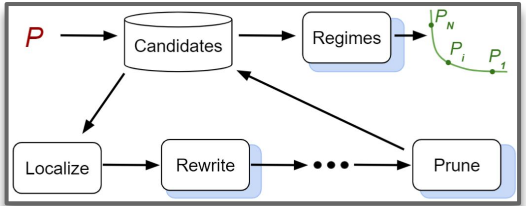
✓ Herbie: Improving Accuracy via Rewriting

- Key Insight: local error guides rewriting



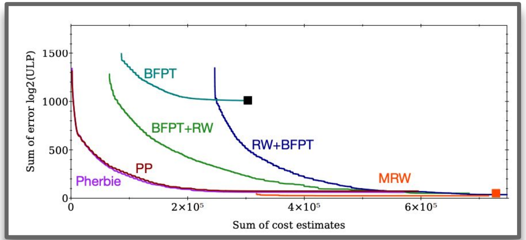
❏ Pherbie: Extending Herbie with Precision Tuning

- Key Insight: local error also guides precision tuning!

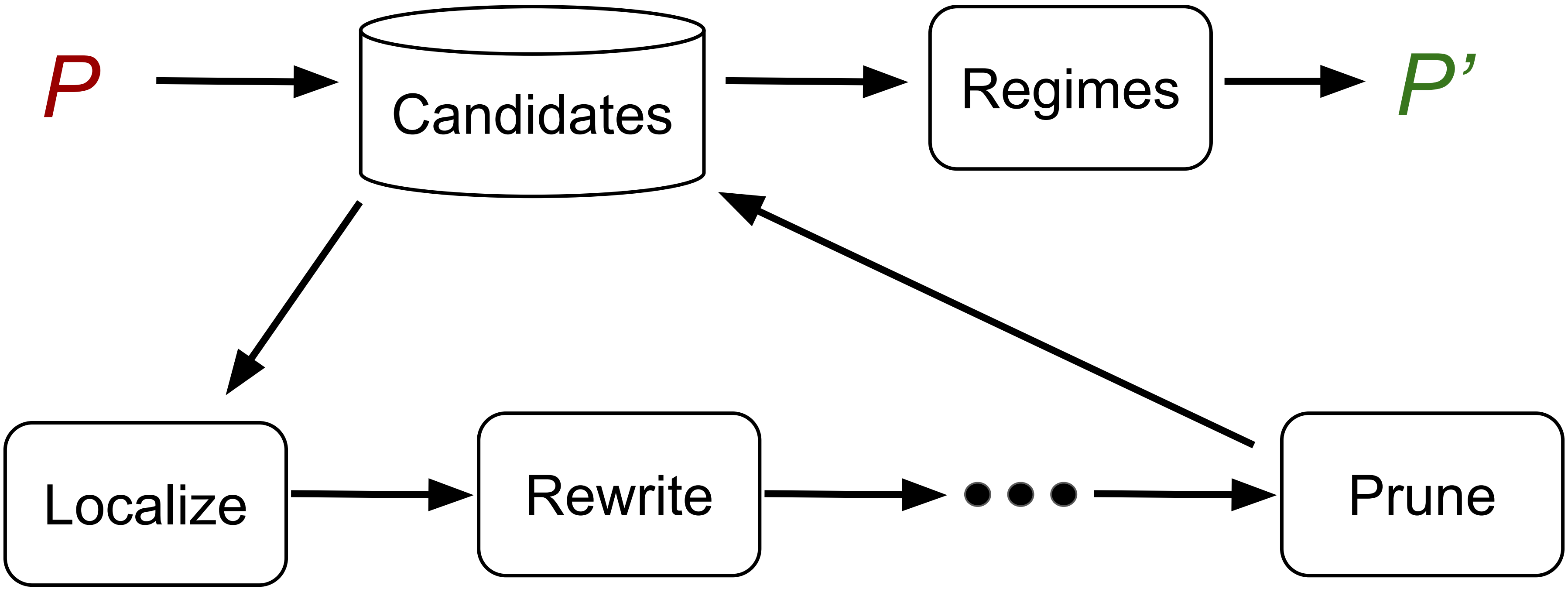


❏ Evaluation: Applying Pherbie to Classics + Graphics

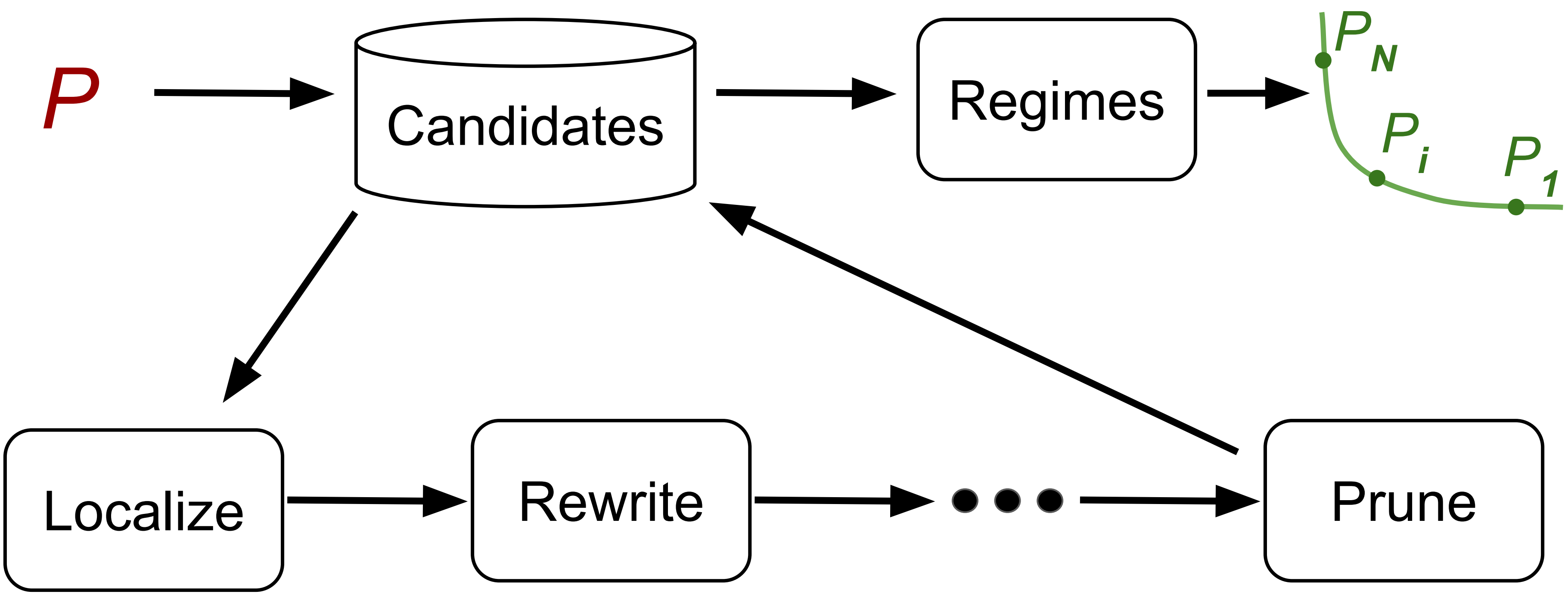
- Key Insight: Finer-grained interleaving → better optimization!



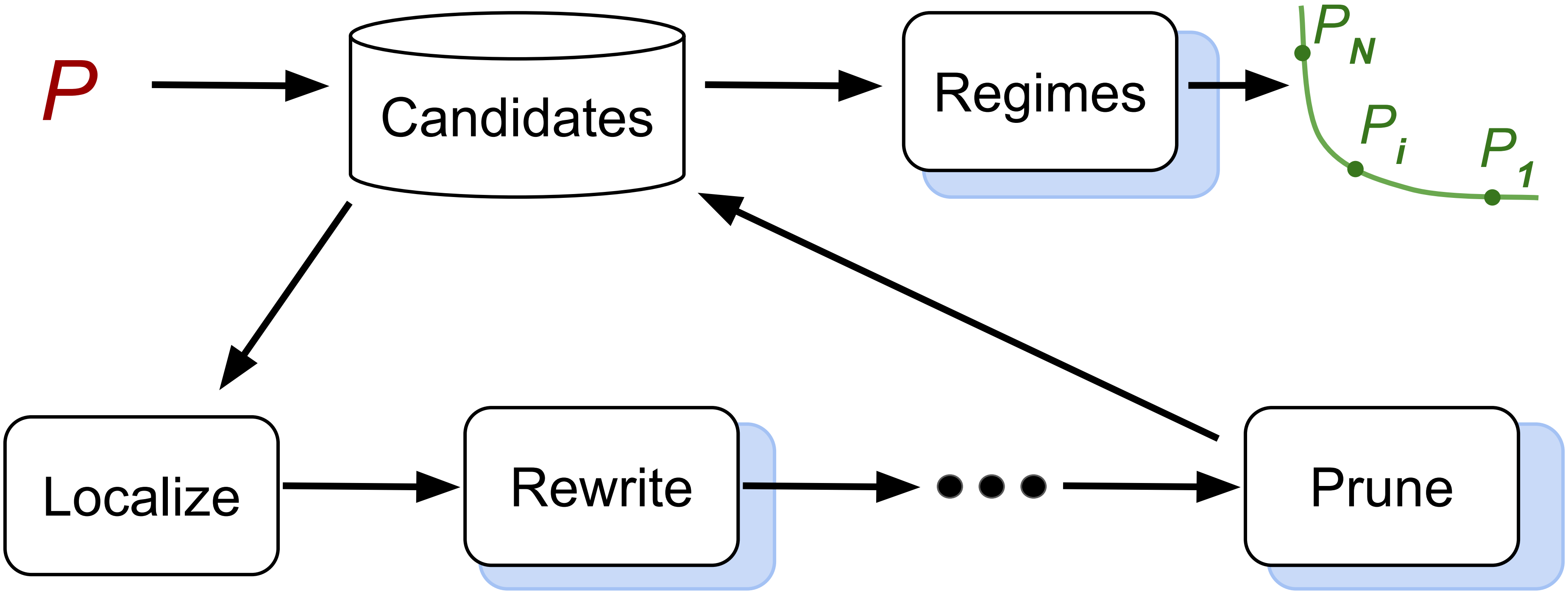
Pherbie Starting Point: Herbie



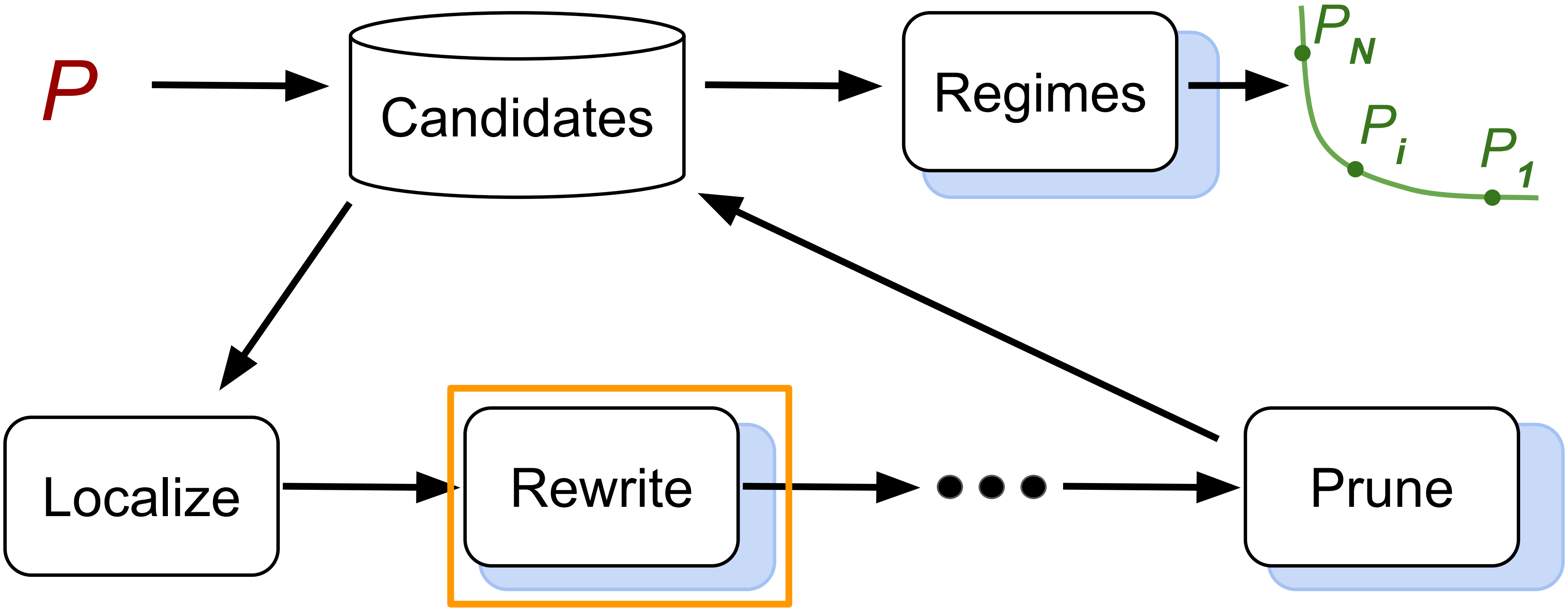
Pherbie: Extending Herbie to Combine Tuning + Rewriting



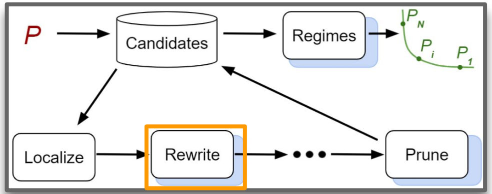
Pherbie: Extending Herbie to Combine Tuning + Rewriting



Pherbie: Extending Herbie to Combine Tuning + Rewriting



Pherbie: Precision Rewrites

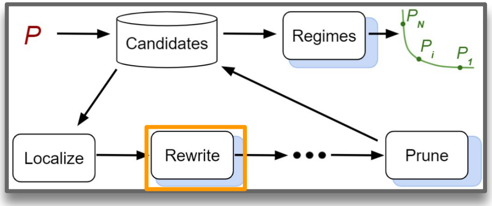


Single global precision

Herbie $$(a + b) \;/\; c \;\Rightarrow\; (a \;/\; c) + (b \;/\; c)$$

Pherbie

Pherbie: Precision Rewrites

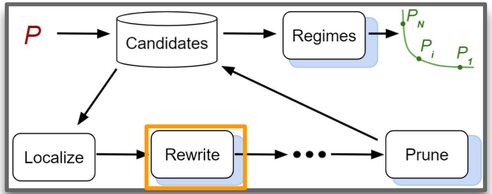


Herbie $(a + b)\ /\ c \Rightarrow (a\ /\ c) + (b\ /\ c)$

Single global precision

Pherbie $(a +_{\mathrm{f64}} b)\ /_{\mathrm{f64}}\ c \Rightarrow (a\ /_{\mathrm{f64}}\ c) +_{\mathrm{f64}} (b\ /_{\mathrm{f64}}\ c)$

Precision-specific operators



Pherbie: Precision Rewrites

Herbie $$(a + b) \ / \ c \ \Rightarrow \ (a \ / \ c) + (b \ / \ c)$$

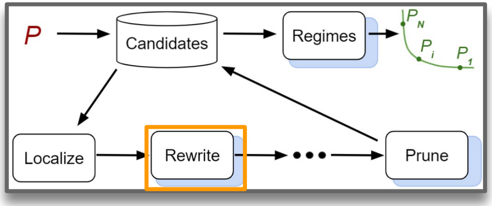
Single global precision

Pherbie $$(a +_{\mathrm{f64}} b) \ /_{\mathrm{f64}} \ c \ \Rightarrow \ (a \ /_{\mathrm{f64}} \ c) +_{\mathrm{f64}} (b \ /_{\mathrm{f64}} \ c)$$

Precision-specific operators

$$(x)_p \ \Rightarrow \ \mathrm{cast}_p \, (x)_q$$

Precision rewrites



Pherbie: Precision Rewrites

Herbie $$(a + b) \;/\; c \;\Rightarrow\; (a \;/\; c) + (b \;/\; c)$$

Rewriting

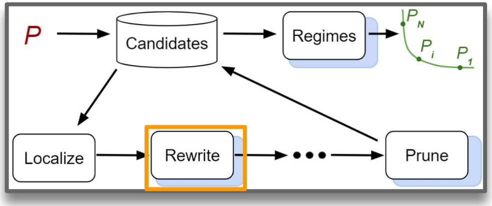
Pherbie $$(a +_{\mathrm{f64}} b) \;/_{\mathrm{f64}}\; c \;\Rightarrow\; (a \;/_{\mathrm{f64}}\; c) +_{\mathrm{f64}} (b \;/_{\mathrm{f64}}\; c)$$

Rewriting

$$(x)_p \;\Rightarrow\; \mathrm{cast}_p\,(x)_q$$

Precision tuning

Pherbie: Precision Rewrites



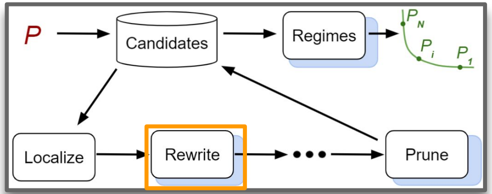
Herbie $$(a + b) \;/\; c \;\Rightarrow\; (a \;/\; c) + (b \;/\; c)$$

Pherbie

Pherbie can use the same rewriting machinery as Herbie!

$$(x)_p \;\Rightarrow\; \mathrm{cast}_p\,(x)_q$$

Pherbie: Precision Rewrites



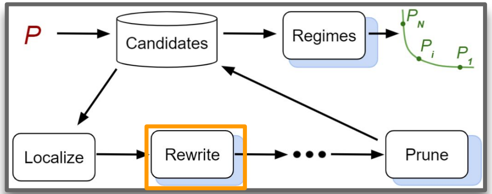
Herbie $$(a + b) \,/\, c \;\Rightarrow\; (a \,/\, c) + (b \,/\, c)$$

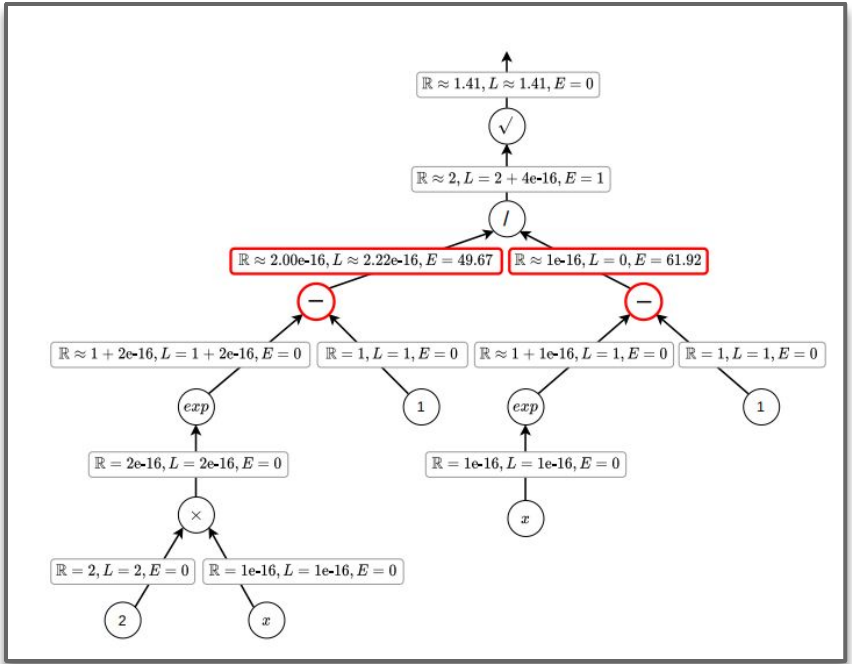
Pherbie

Pherbie can use the same rewriting machinery as Herbie!

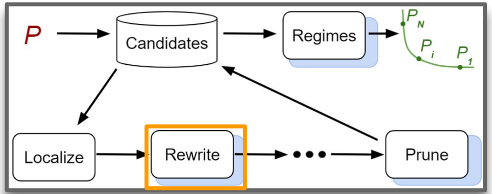
But where should Pherbie apply precision rewrites?

Pherbie: Guide Tuning w/ Local Error

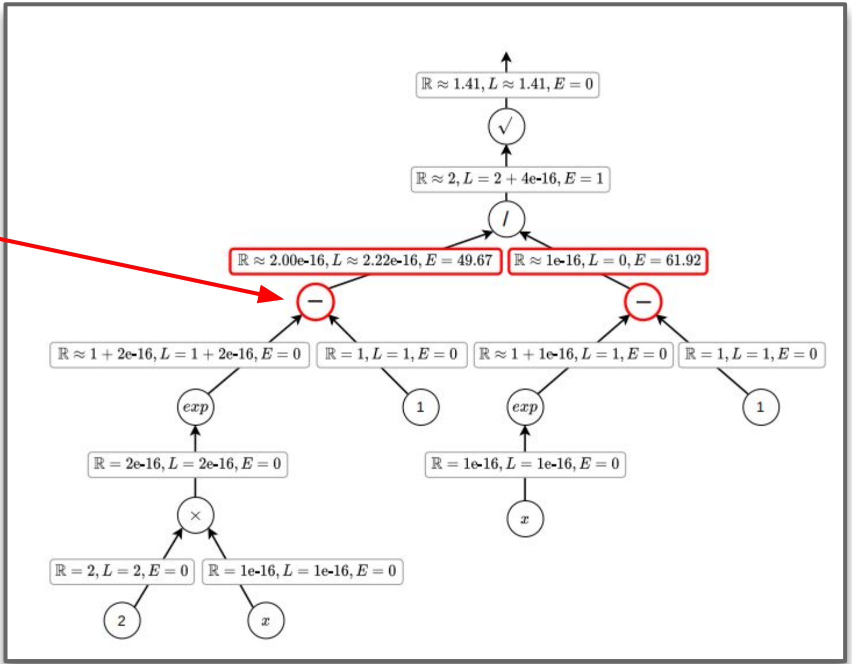




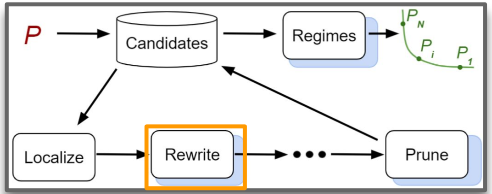
Pherbie: Guide Tuning w/ Local Error



- Rewriting to increase precision at locations w/ *high local error* improves accuracy.

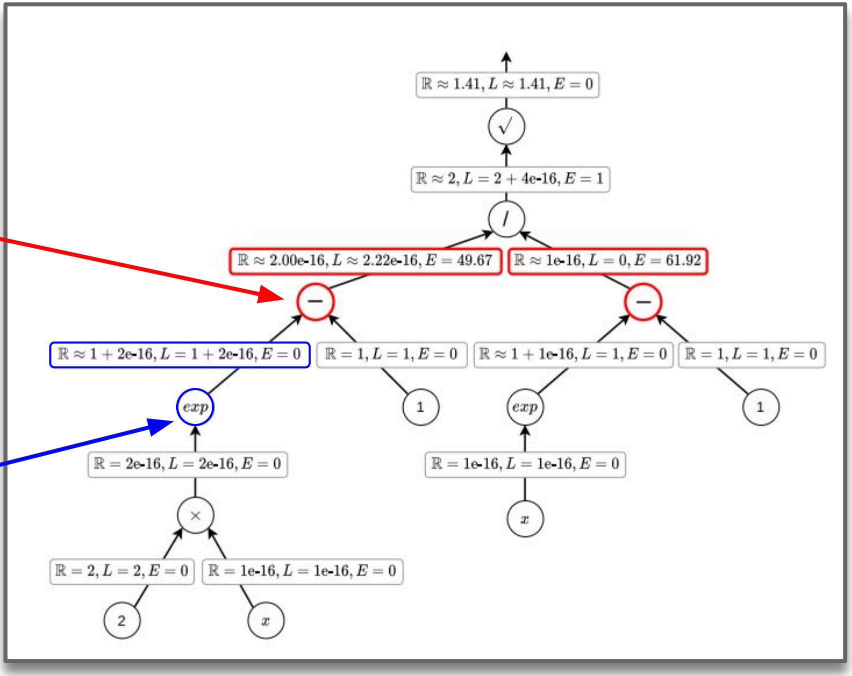


Pherbie: Guide Tuning w/ Local Error

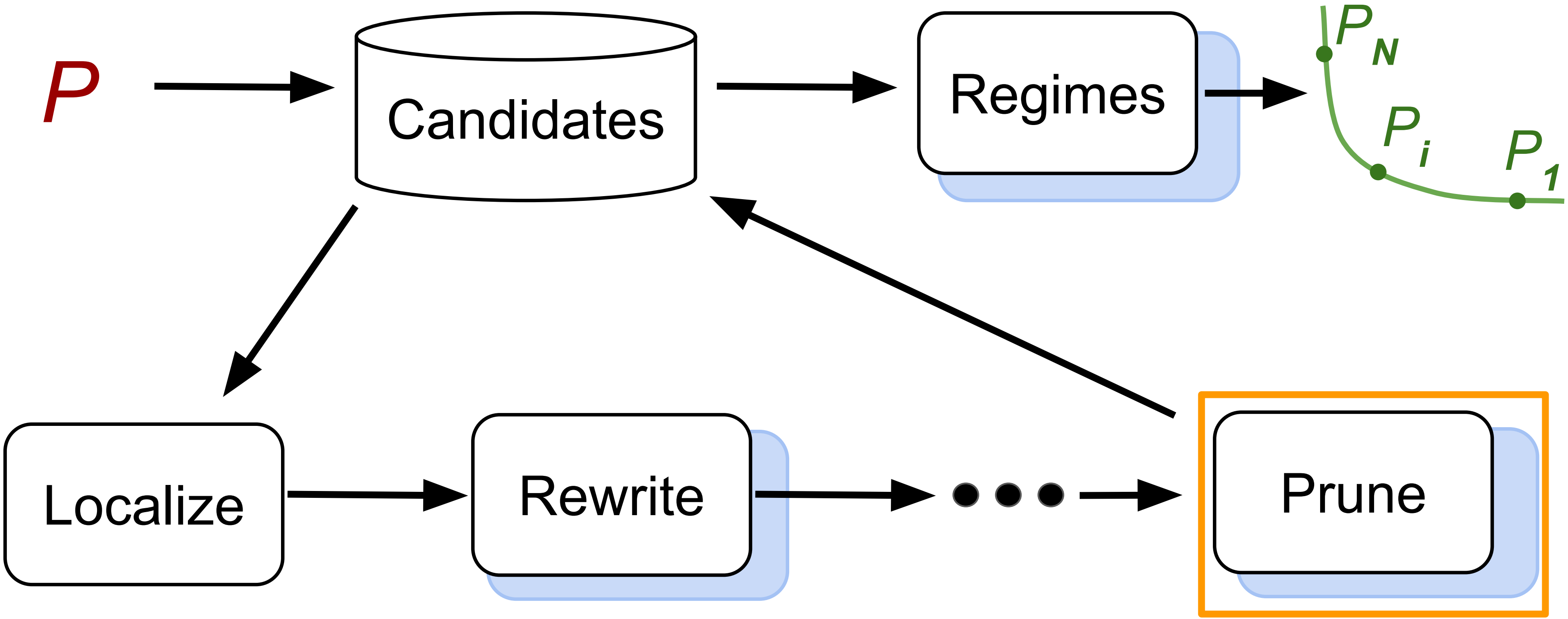


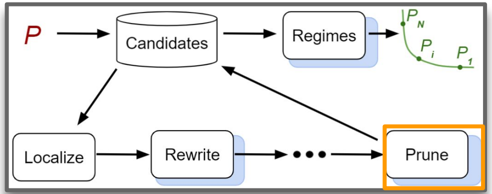
- Rewriting to increase precision at locations w/ *high local error* improves accuracy.

- Rewriting to decrease precision at locations w/ *low local error* improves speed.



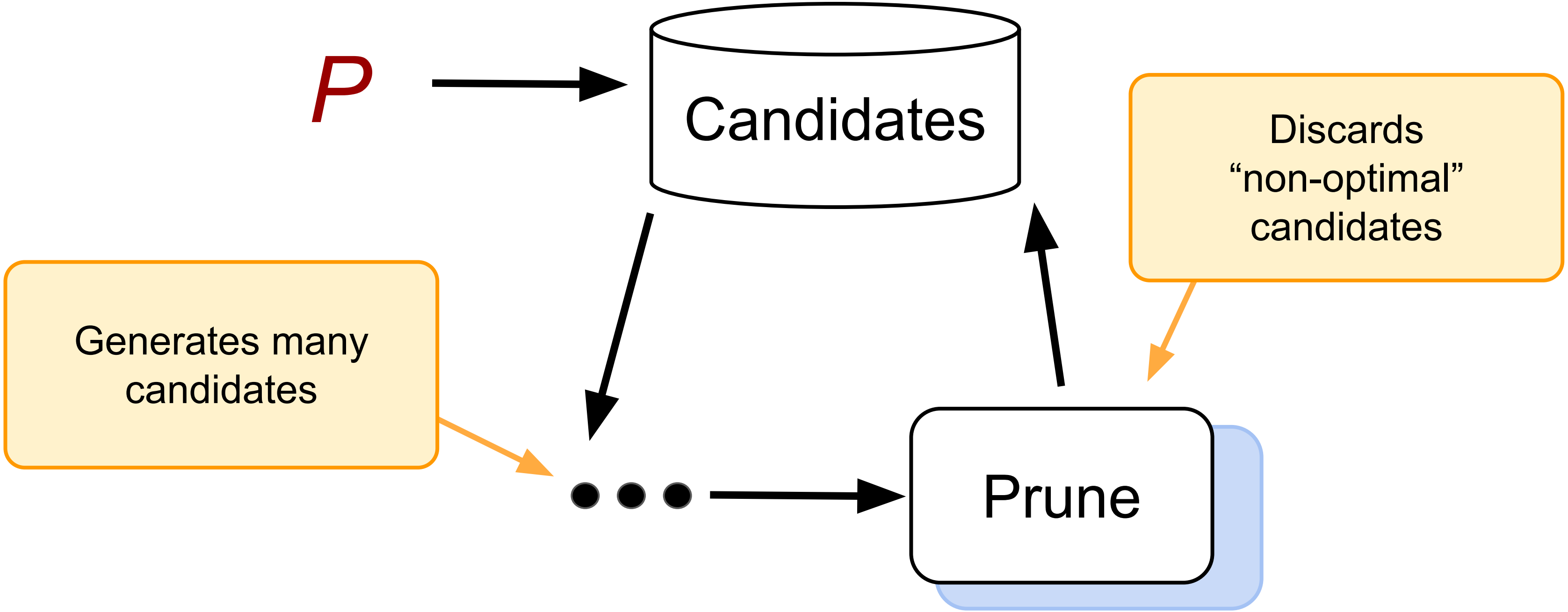
Pherbie: Extending Herbie to Combine Tuning + Rewriting



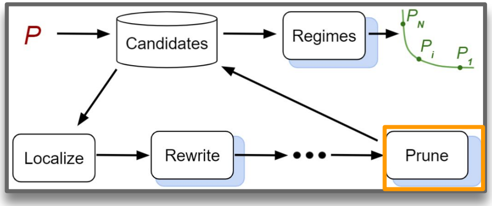


Pherbie: Pruning

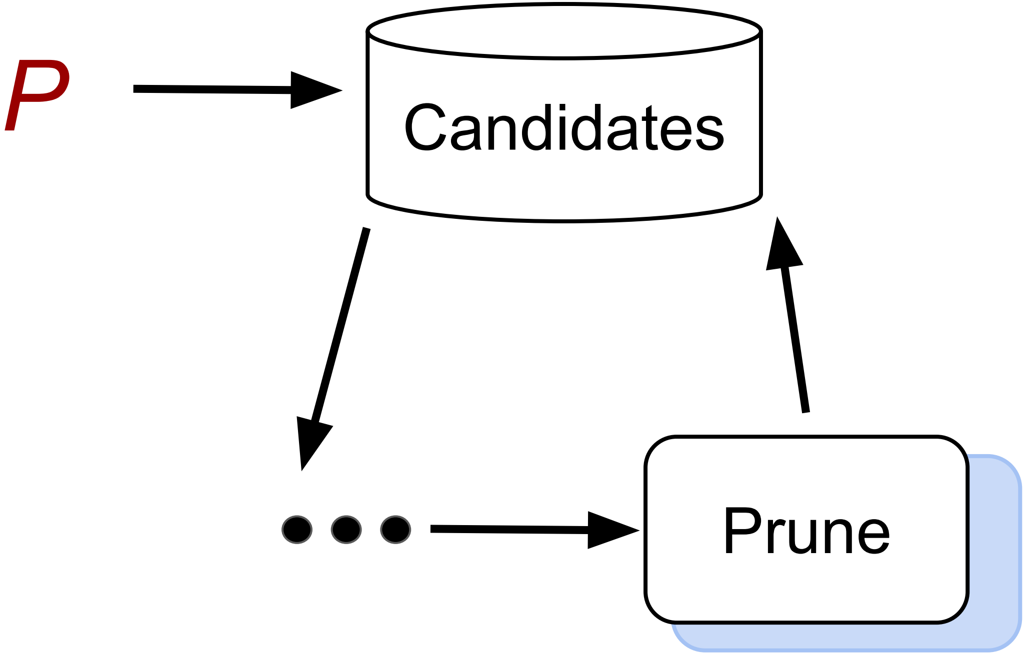
Pruning in general



Pherbie: Pruning



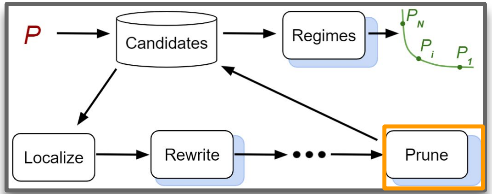
Pruning in Herbie:



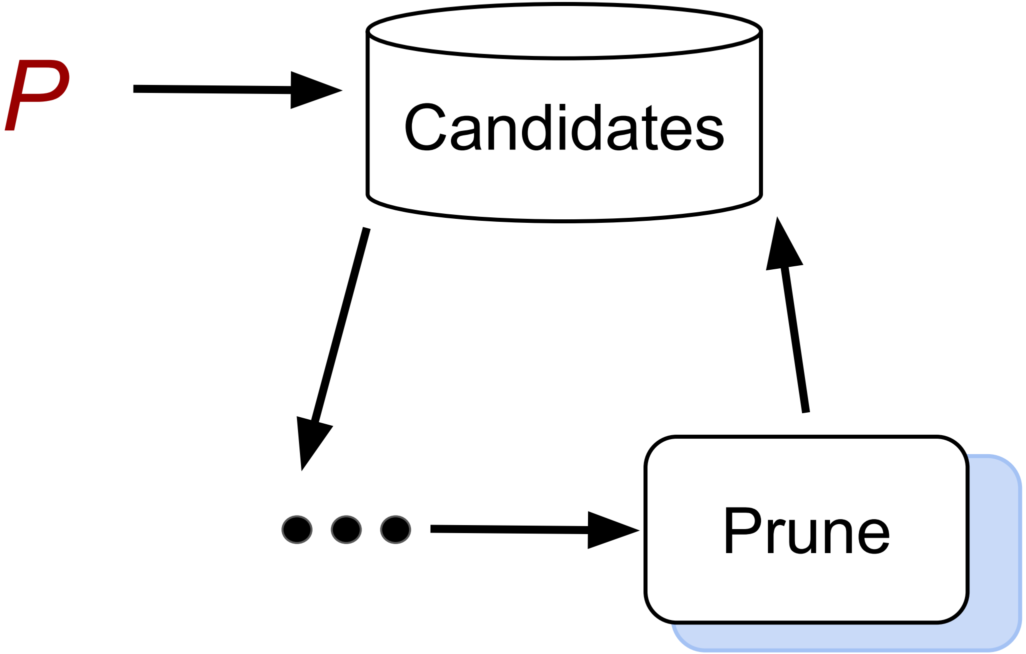
Criteria

Must be more accurate than every other expression on at least one sampled point

Pherbie: Pruning



Pruning in Herbie:



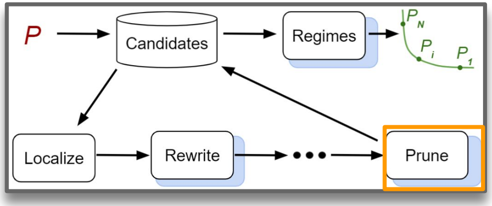
Criteria

Must be more accurate than every other expression on at least one sampled point

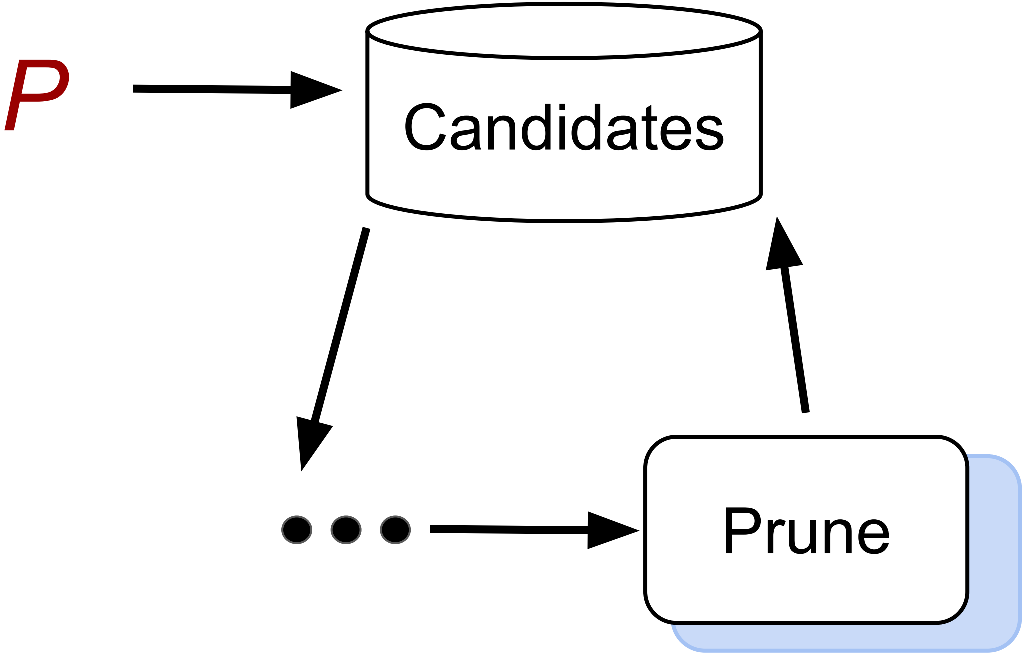
Use in Pherbie?

Accuracy only ⇒ slow expressions

Pherbie: Pruning



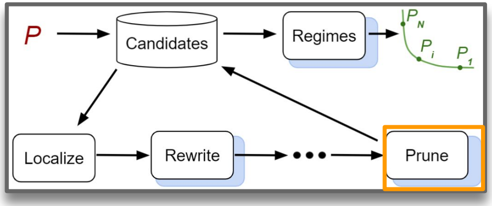
Pruning in Pherbie:



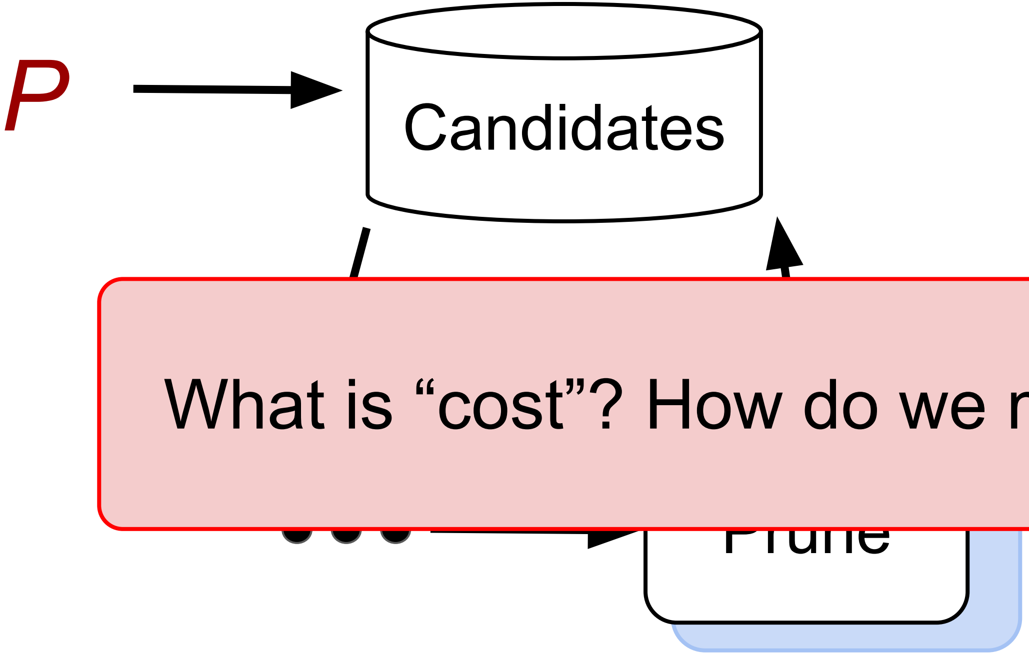
Criteria

Must be more accurate on at least one sampled point than every other expression *at or below the cost of the candidate*

Pherbie: Pruning



Pruning in Pherbie:

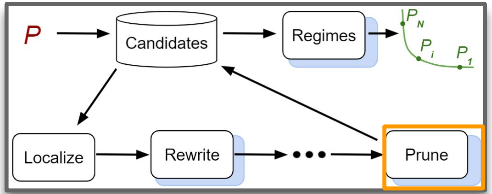


Criteria

Must be more accurate on at least one sampled point than every other expression *at or below the cost of the candidate*

What is “cost”? How do we measure it?

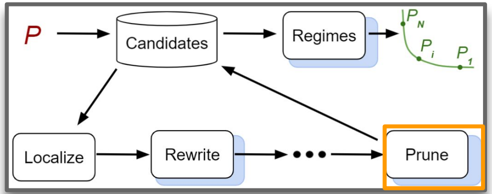
Pherbie: Pruning



What is "cost"? How do we measure it?

Too expensive to measure precise latency of each candidate

- Need to evaluate candidate many times to get accurate estimator
- Pherbie produces thousands of candidates

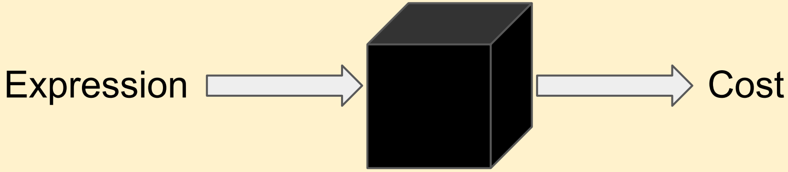


Pherbie: Pruning

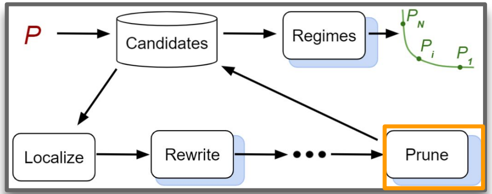
What is “cost”? How do we measure it?

Key Insight: Only need relative speed comparison → use a simple cost model!

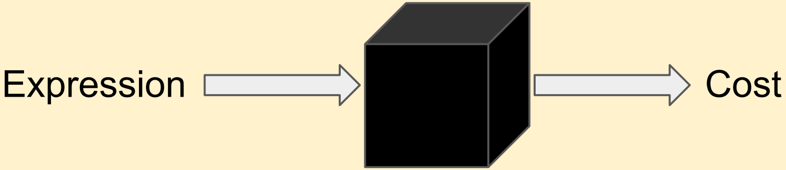
- Quickly estimates latency
- Sufficient for relative ordering of candidates



Pherbie: Pruning



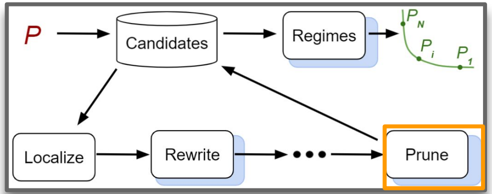
What is “cost”? How do we measure it?



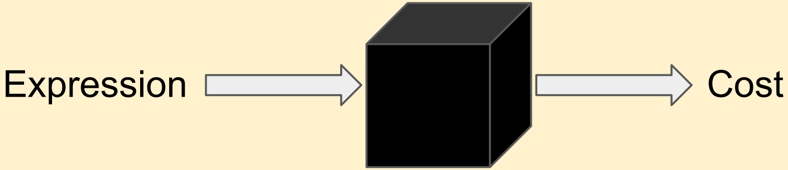
Example cost model:

- Operators assigned a cost:
  - Arithmetic: low number (1)
  - Library functions: large number (100)
- Multiply operator cost by bitwidth of representation
- Conditionals: branch conditions cost + largest branch cost

Pherbie: Pruning



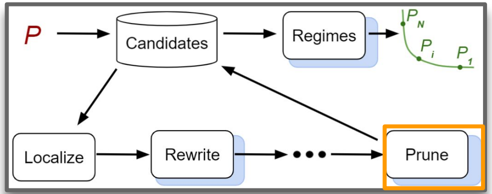
What is “cost”? How do we measure it?



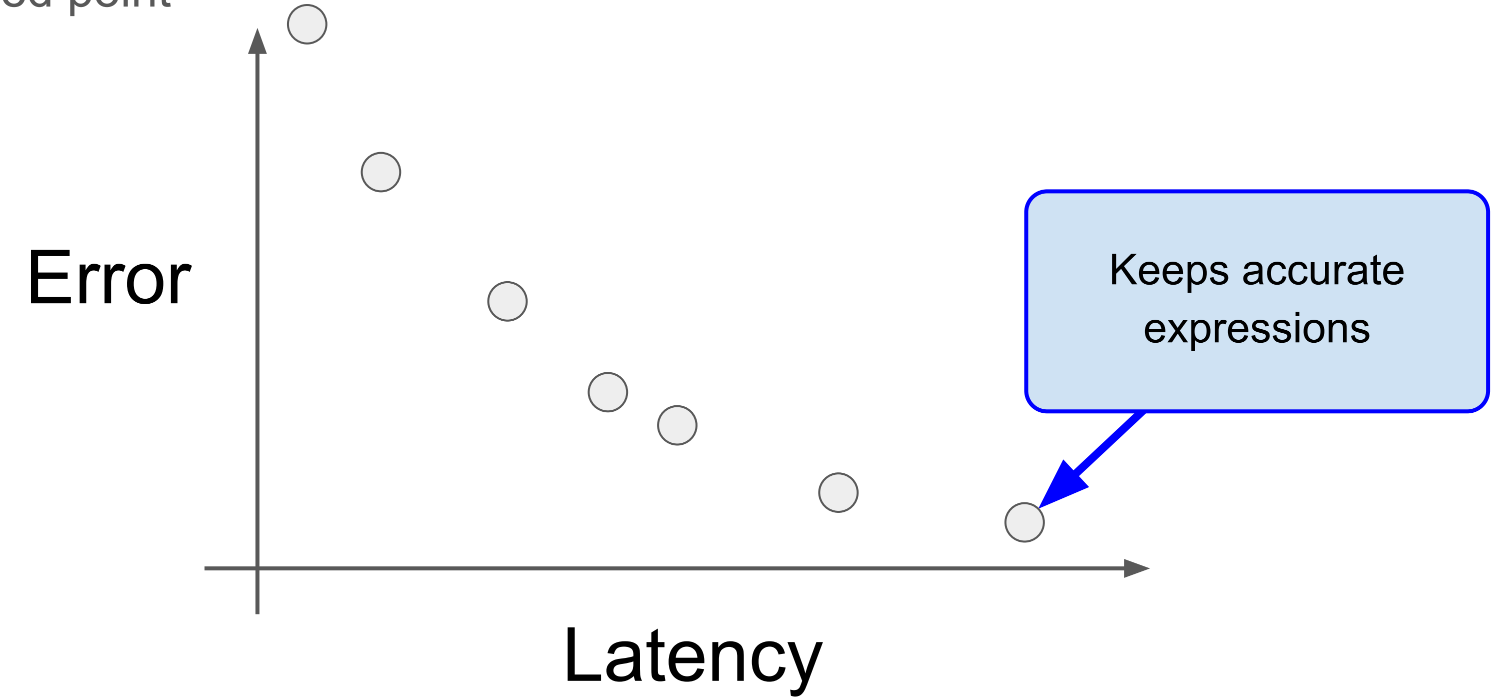
Cost models in general

- Simple cost models are good enough
- Better cost models exist
- Pherbie is modular, so users can plug and play

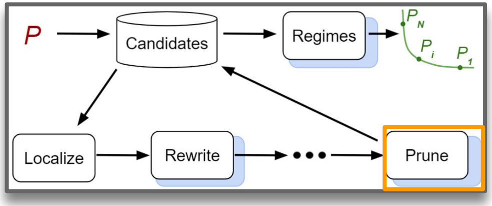
Pherbie: Pruning



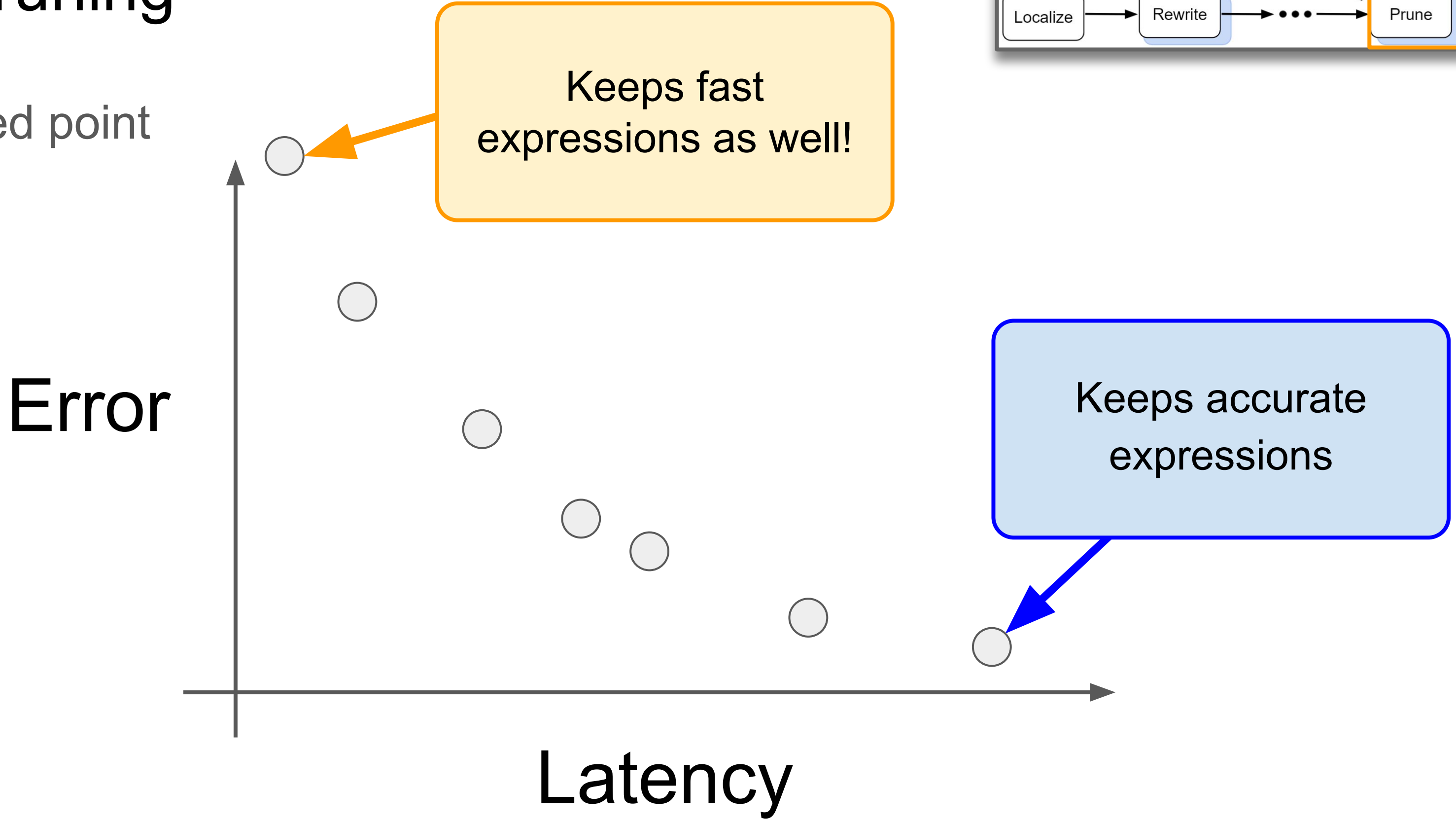
At each sampled point



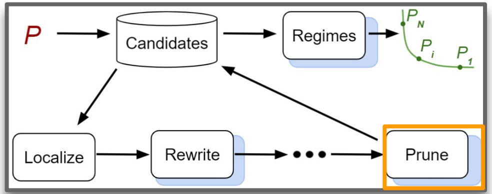
Pherbie: Pruning



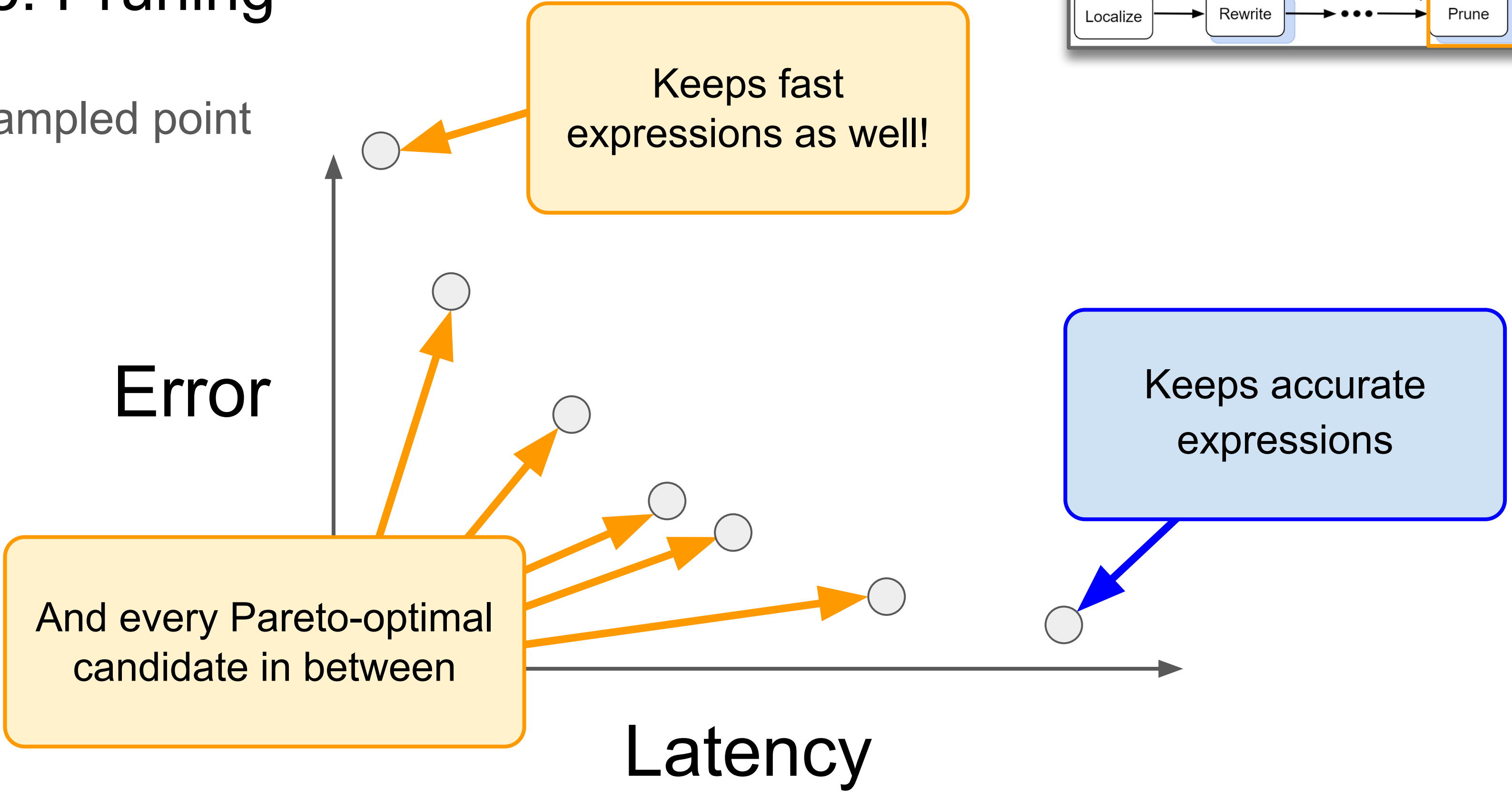
At each sampled point



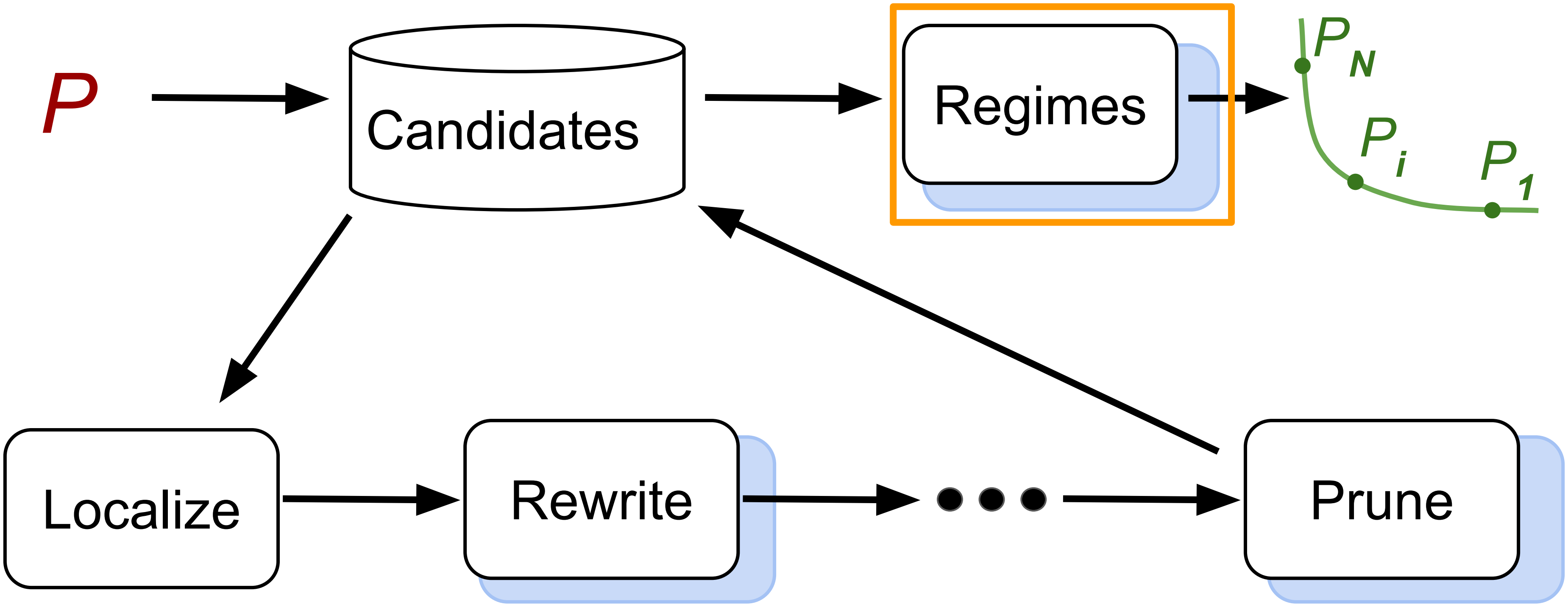
Pherbie: Pruning

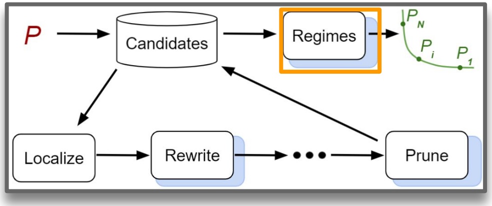


At each sampled point



Pherbie: Extending Herbie to Combine Tuning + Rewriting

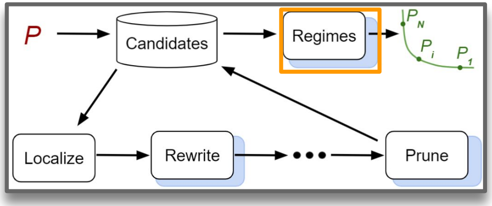




Pherbie: Regimes

Pherbie: accuracy and cost

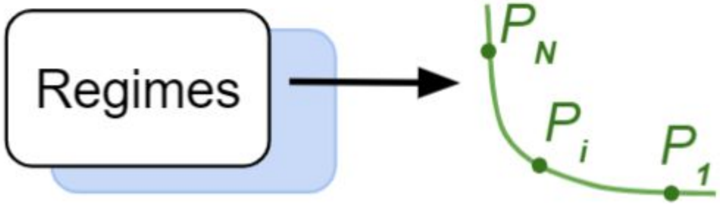
- Need to produce a Pareto frontier!
- Iteratively run Herbie's regimes algorithm on subset of candidates



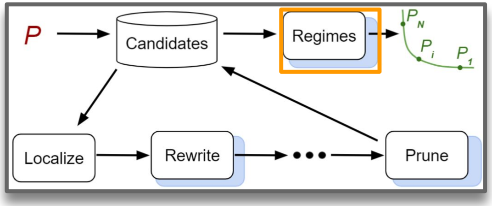
Pherbie: Regimes

Pherbie regimes algorithm

1. Run Herbie regimes algorithm on subset cheaper than cost bound

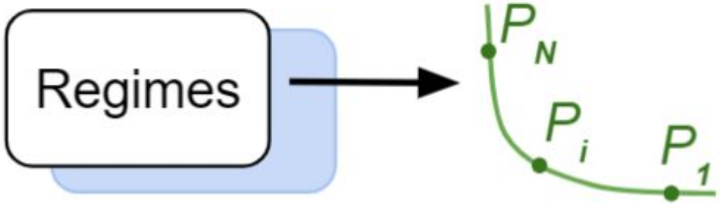


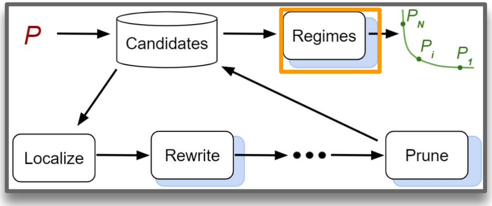
Pherbie: Regimes



Pherbie regimes algorithm

1. Run Herbie regimes algorithm on subset cheaper than cost bound
2. Decrease cost bound so next iteration produces *different* candidate

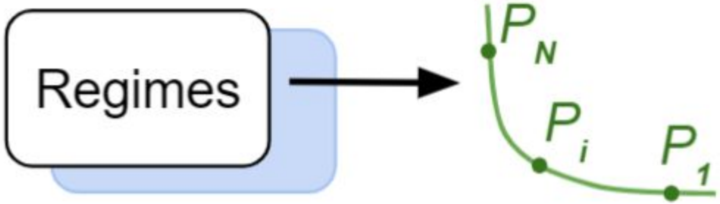




Pherbie: Regimes

Pherbie regimes algorithm

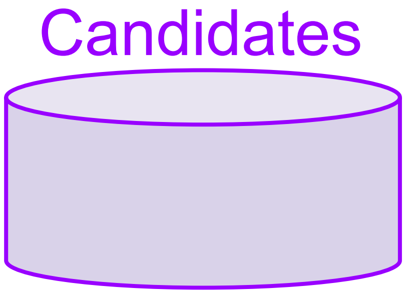
1. Run Herbie regimes algorithm on subset cheaper than cost bound
2. Decrease cost bound so next iteration produces *different* candidate
3. Repeat until no candidate is below cost bound

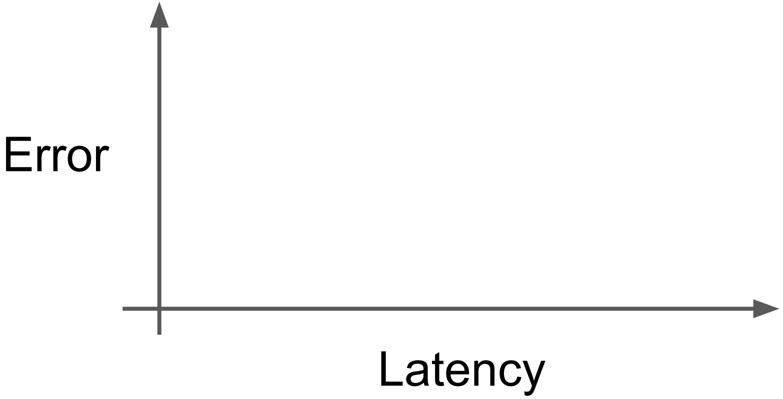


Pherbie Regimes Example : Iter 1 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```



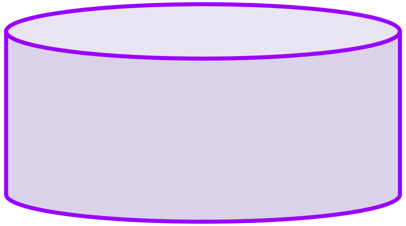


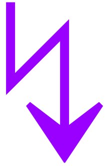
Pherbie Regimes Example : Iter 1 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

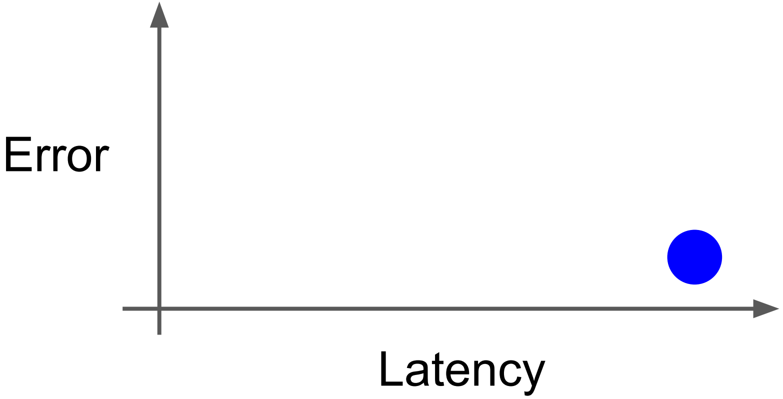
```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates





$$\sqrt{\frac{1 + \left(e^{x}\right)^{1.5} \cdot \left(e^{x}\right)^{1.5}}{1 + \left(e^{x} + \left(e^{x}\right)^{1.5}\right) \cdot \left(\sqrt{e^{x}} - 1\right)}}$$

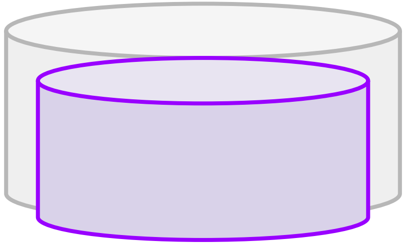


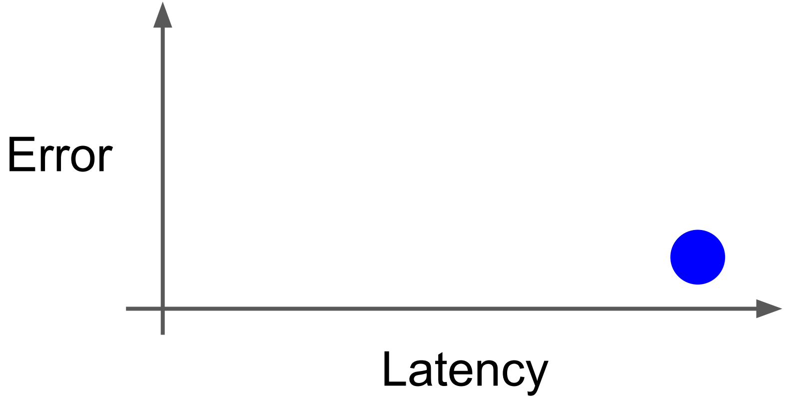
Pherbie Regimes Example : Iter 1 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates





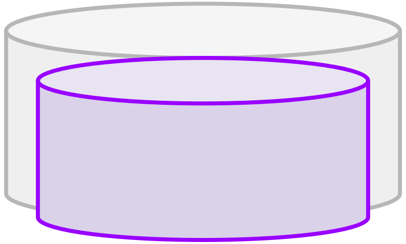
$$\sqrt{\frac{1 + \left(e^{x}\right)^{1.5} \cdot \left(e^{x}\right)^{1.5}}{1 + \left(e^{x} + \left(e^{x}\right)^{1.5}\right) \cdot \left(\sqrt{e^{x}} - 1\right)}}$$

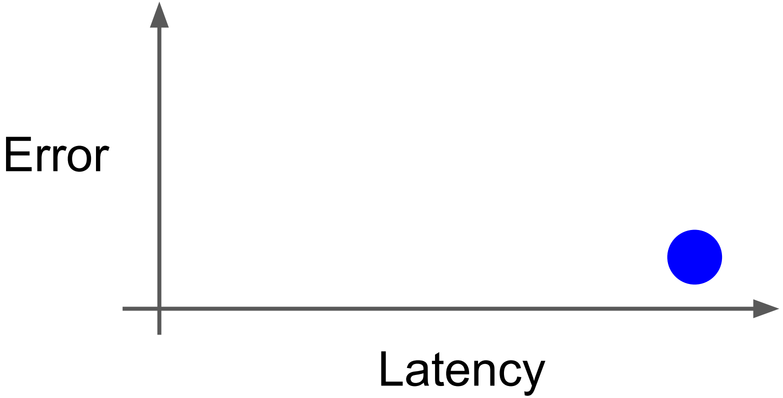
Pherbie Regimes Example : Iter 2 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates

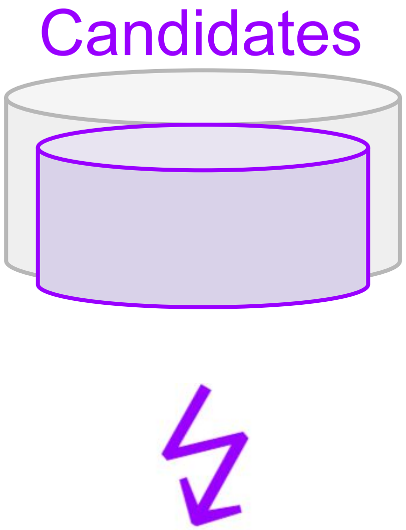




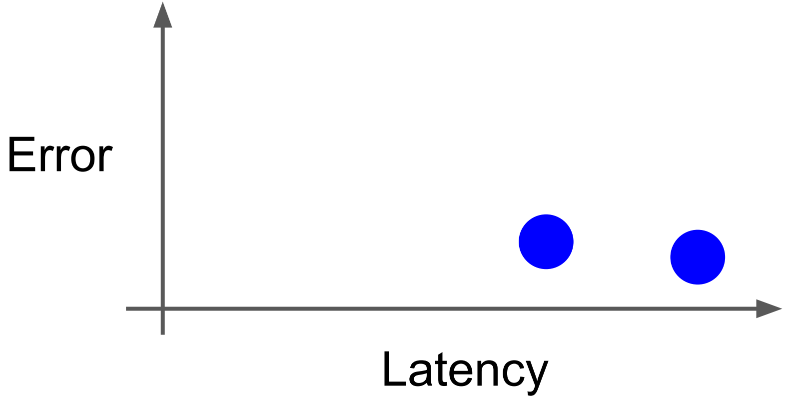
Pherbie Regimes Example : Iter 2 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```



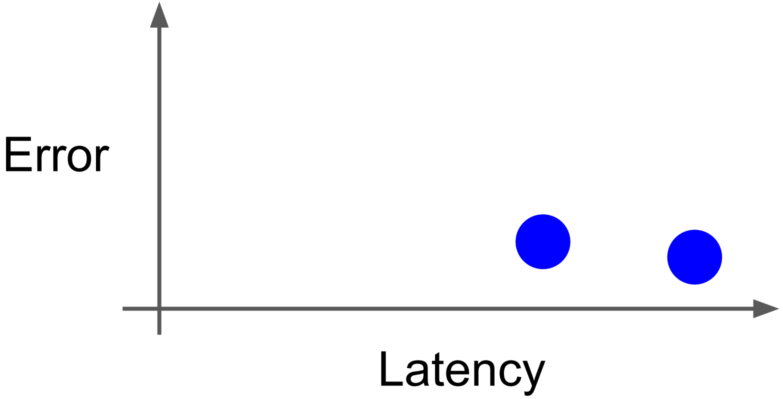
$$\sqrt{1 + e^{x}}$$



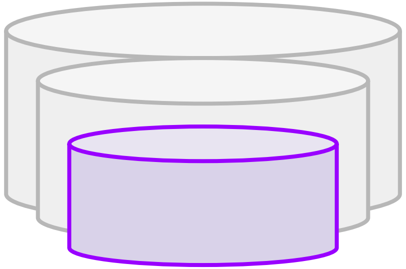
Pherbie Regimes Example : Iter 2 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```



Candidates



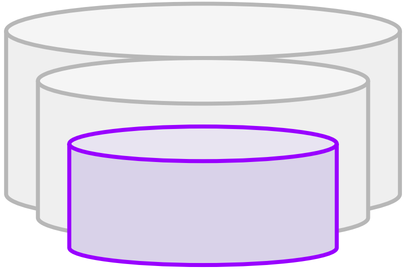
$$\sqrt{1 + e^{x}}$$

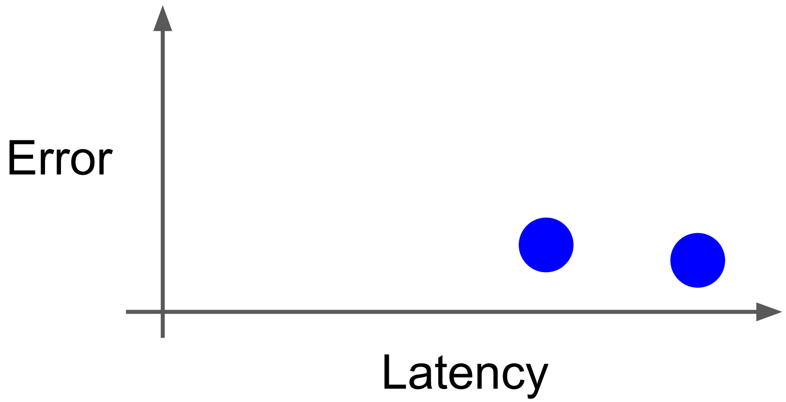
Pherbie Regimes Example : Iter 3 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates



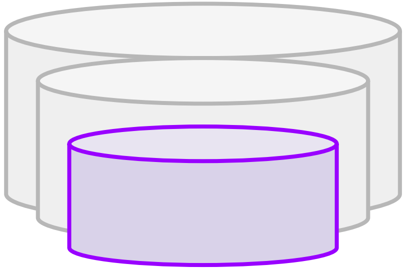


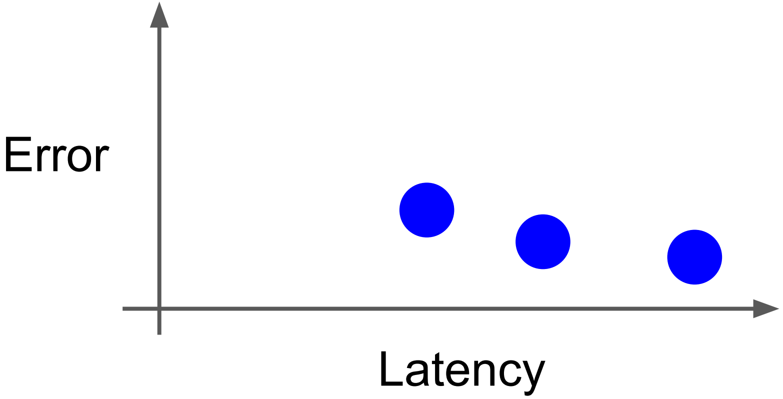
Pherbie Regimes Example : Iter 3 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates





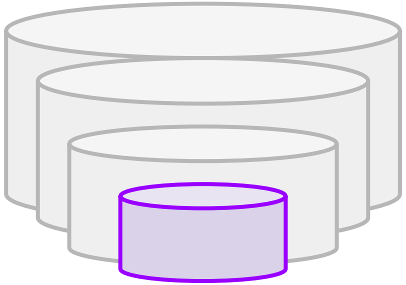
$$\begin{array}{l}\textbf{if } x \leq -0.10591501462198885 : \\ \quad \langle (\sqrt{1 + e^{x}})_{(\text{float } 5\ 16)} \rangle_{\text{binary64}} \\ \textbf{else} : \\ \quad \sqrt{2 + (x + x \cdot (x \cdot 0.5))}\end{array}$$

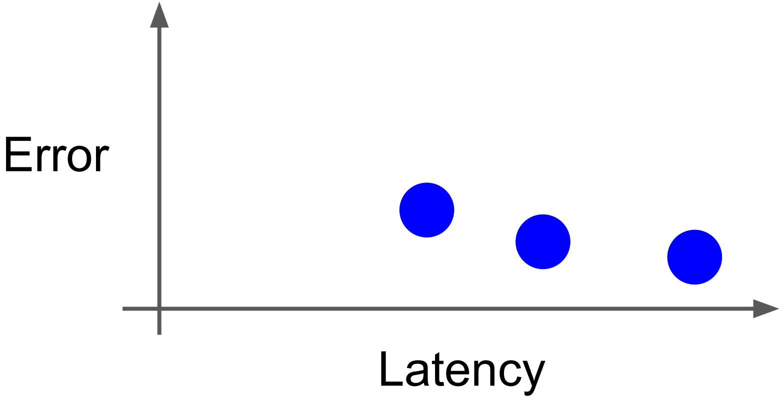
Pherbie Regimes Example : Iter 3 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates



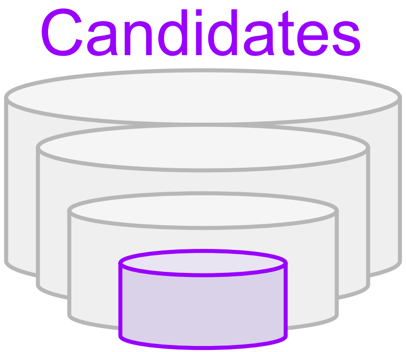


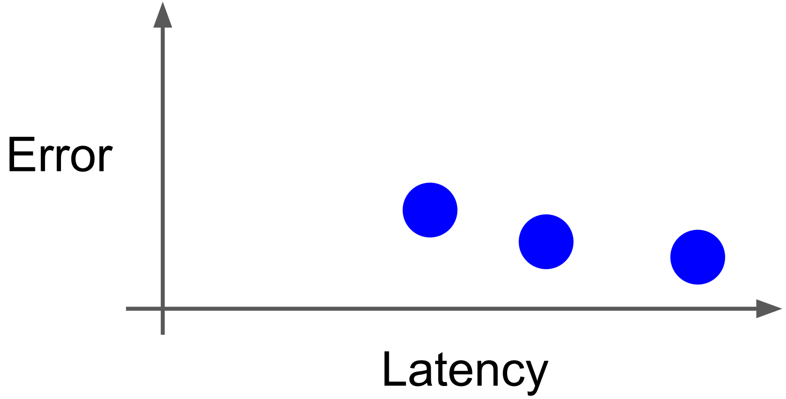
$$\begin{array}{l}\textbf{if } x \leq -0.10591501462198885 : \\ \quad \langle (\sqrt{1 + e^{x}})_{(\text{float } 5\ 16)} \rangle_{\text{binary64}} \\ \textbf{else} : \\ \quad \sqrt{2 + (x + x \cdot (x \cdot 0.5))}\end{array}$$

Pherbie Regimes Example : Iter 4 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^x - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```



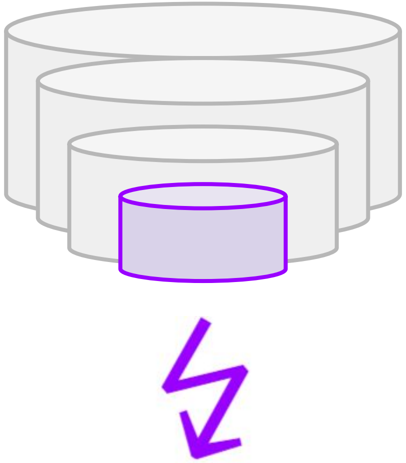


Pherbie Regimes Example : Iter 4 / 5

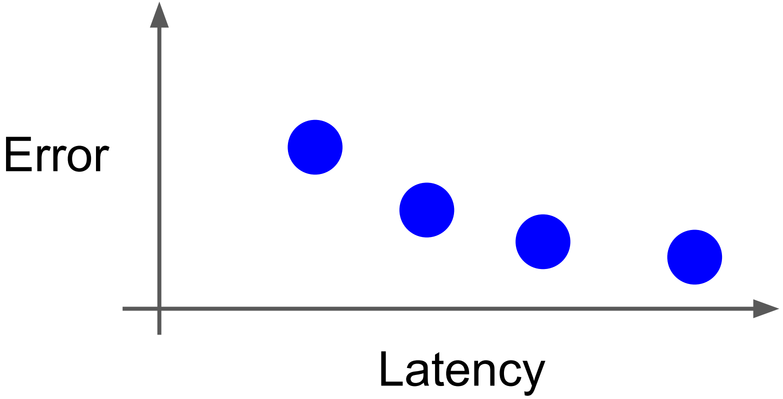
$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates



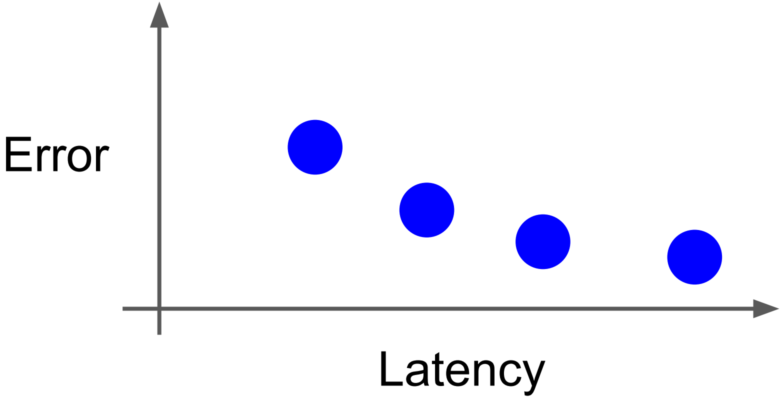
$$\sqrt{2 + (x + x \cdot (x \cdot 0.5))}$$

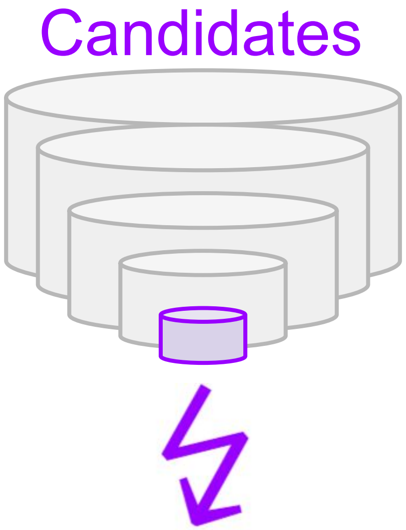


Pherbie Regimes Example : Iter 4 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```





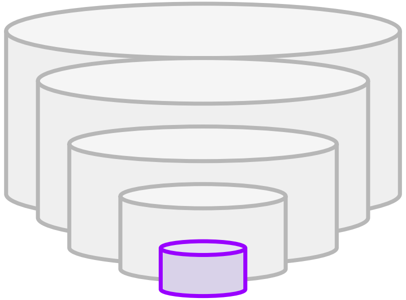
$$\sqrt{2 + (x + x \cdot (x \cdot 0.5))}$$

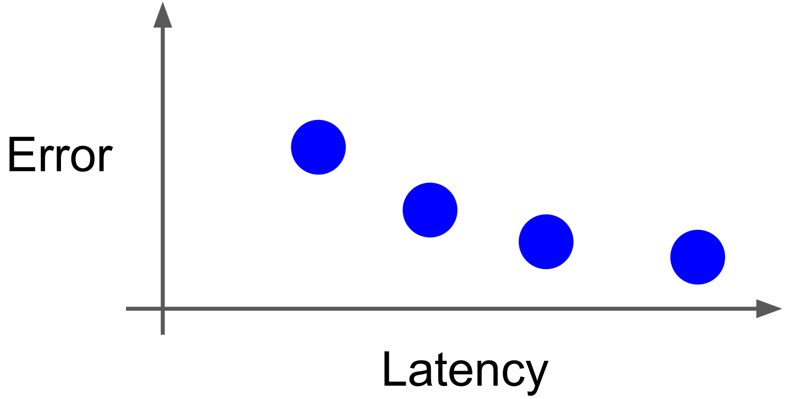
Pherbie Regimes Example : Iter 5 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates

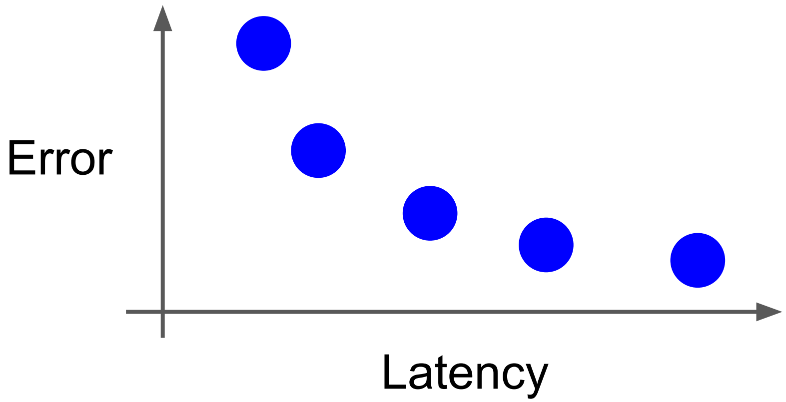




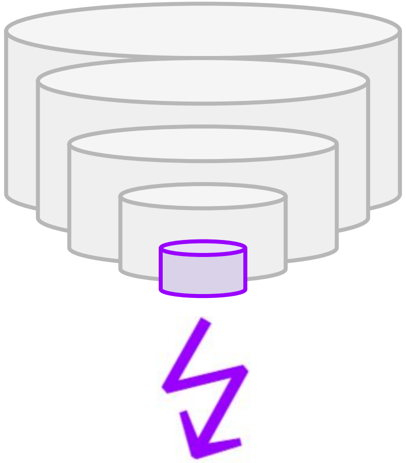
Pherbie Regimes Example : Iter 5 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```



Candidates



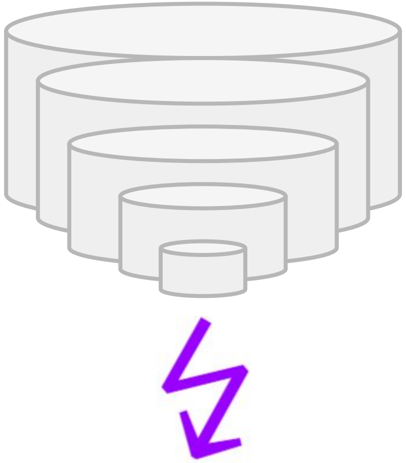
$$\sqrt{2}$$

Pherbie Regimes Example : Iter 5 / 5

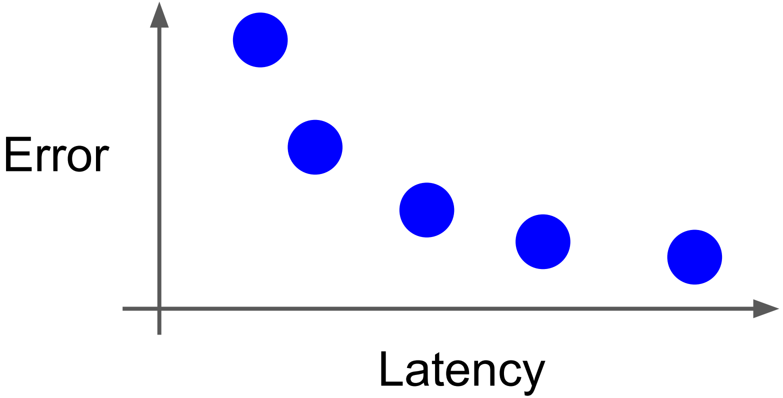
$$\sqrt{\frac{e^{2x}-1}{e^{x}-1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

Candidates



$$\sqrt{2}$$

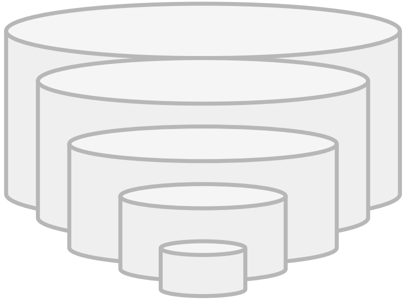


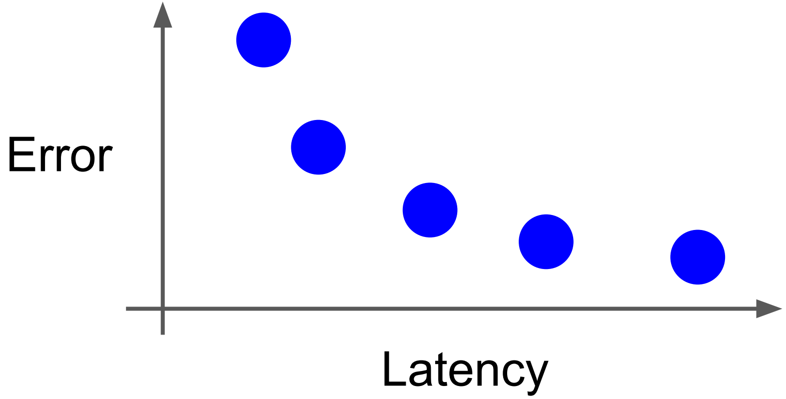
Pherbie Regimes Example : Iter 5 / 5

$$\sqrt{\frac{e^{2x} - 1}{e^{x} - 1}}$$

```
while 0 < |Candidates| :
    p = ExtractMinError(Candidates)
    Candidates.removeAboveCost(p)
```

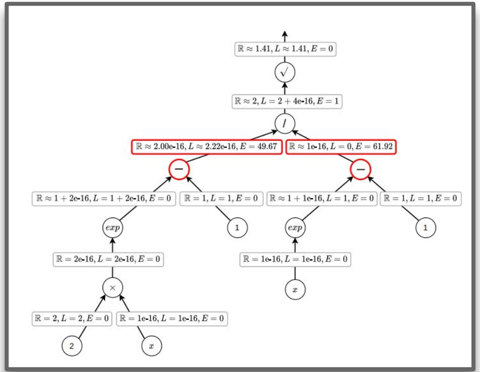
Candidates





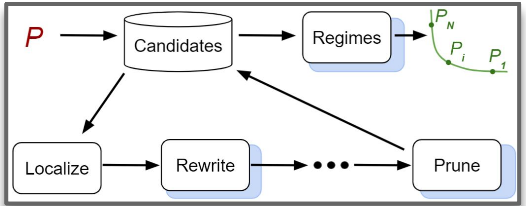
Outline

✓ **Herbie**: Improving Accuracy via Rewriting



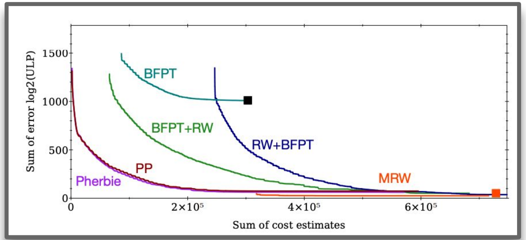
- Key Insight: local error guides rewriting

✓ **Pherbie**: Extending Herbie with Precision Tuning



- Key Insight: local error also guides precision tuning!

❏ **Evaluation**: Applying Pherbie to Classics + Graphics



- Key Insight: Finer-grained interleaving → better optimization!

Evaluation: Benchmark Suites

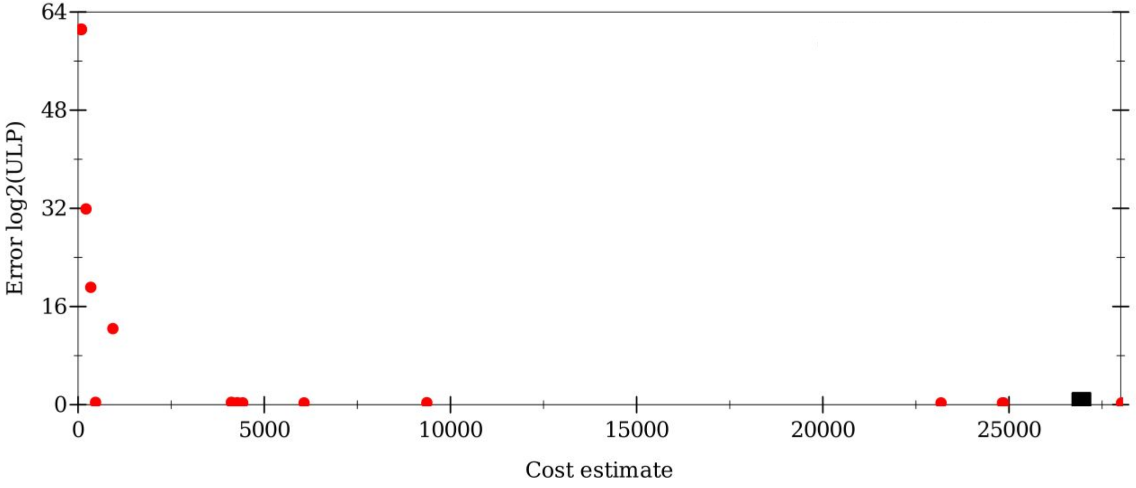
- NMSE - *Numerical Methods for Scientists and Engineers* (Hamming, 1986)
  - Standard textbook on numerical analysis
- PBRT - *Physically Based Rendering* (Pharr et. al, 2016)
  - Open-source textbook describing rendering photorealistic scenes

Evaluation

Pherbie produces Pareto-optimal implementations

Curve Intersection (PBRT)

$$\left(\sin\left(\left(1-u\right)\cdot normAngle\right)\cdot\frac{1}{\sin normAngle}\right)\cdot n0_i+\left(\sin\left(u\cdot normAngle\right)\cdot\frac{1}{\sin normAngle}\right)\cdot n1_i$$

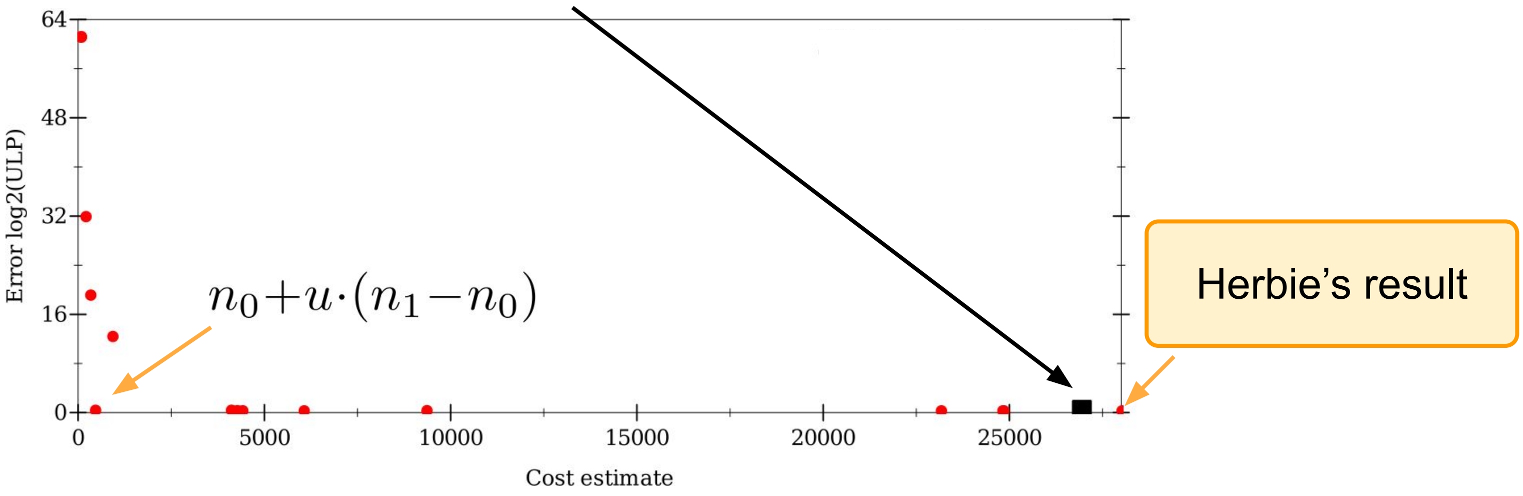


Evaluation

Pherbie produces Pareto-optimal implementations

Curve Intersection (PBRT)

$$\left(\sin\left((1-u)\cdot normAngle\right)\cdot\frac{1}{\sin normAngle}\right)\cdot n0_i+\left(\sin\left(u\cdot normAngle\right)\cdot\frac{1}{\sin normAngle}\right)\cdot n1_i$$

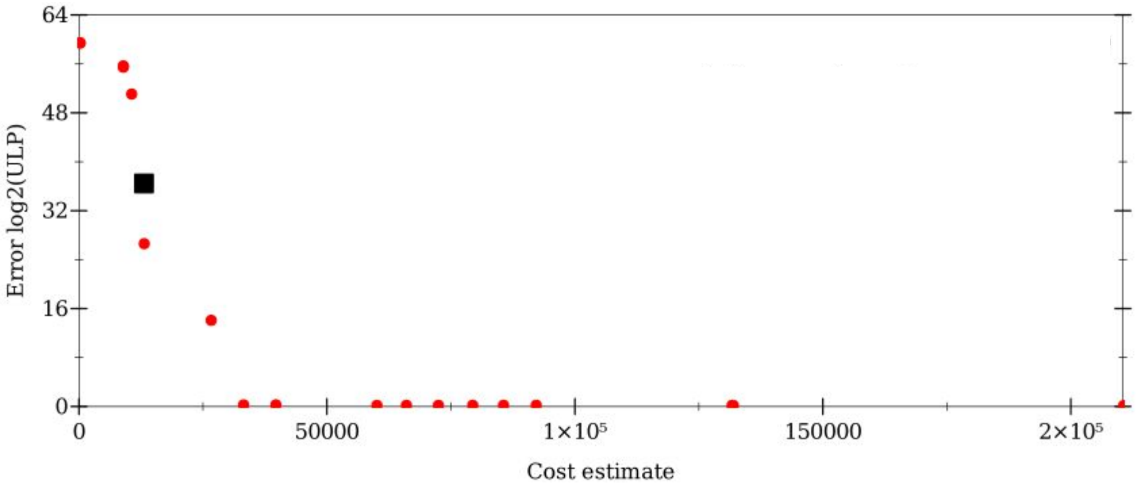


Evaluation

Pherbie produces Pareto-optimal implementations

Nearby Tangent Difference (NMSE)

$$\tan(x + \epsilon) - \tan x$$

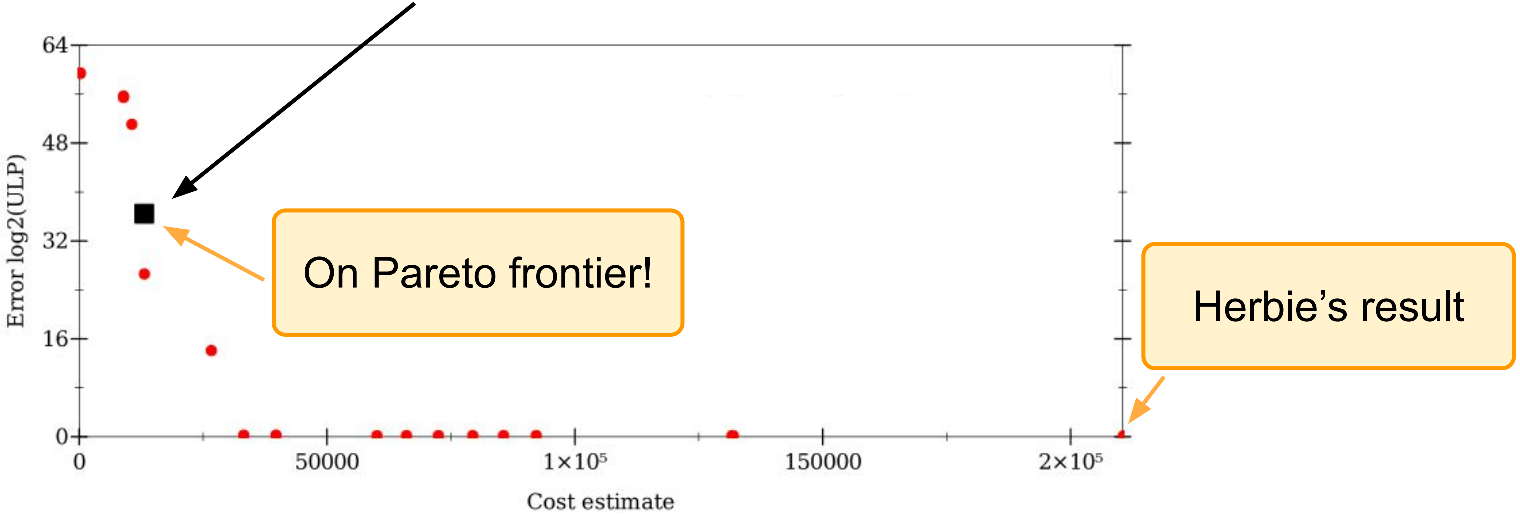


Evaluation

Pherbie produces Pareto-optimal implementations

Nearby Tangent Difference (NMSE)

$$\tan(x + \epsilon) - \tan x$$

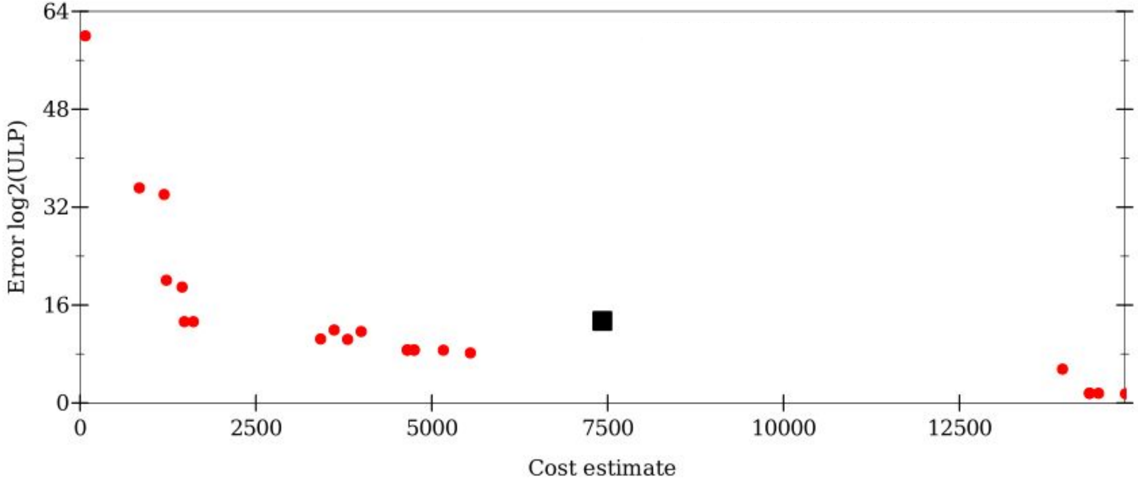


Evaluation

Pherbie produces Pareto-optimal implementations

Beckmann Distribution Sampling (PBRT)

$$\frac{-\log(1+u)}{c2p/(\alpha_x \cdot \alpha_x) + s2p/(\alpha_y \cdot \alpha_y)}$$

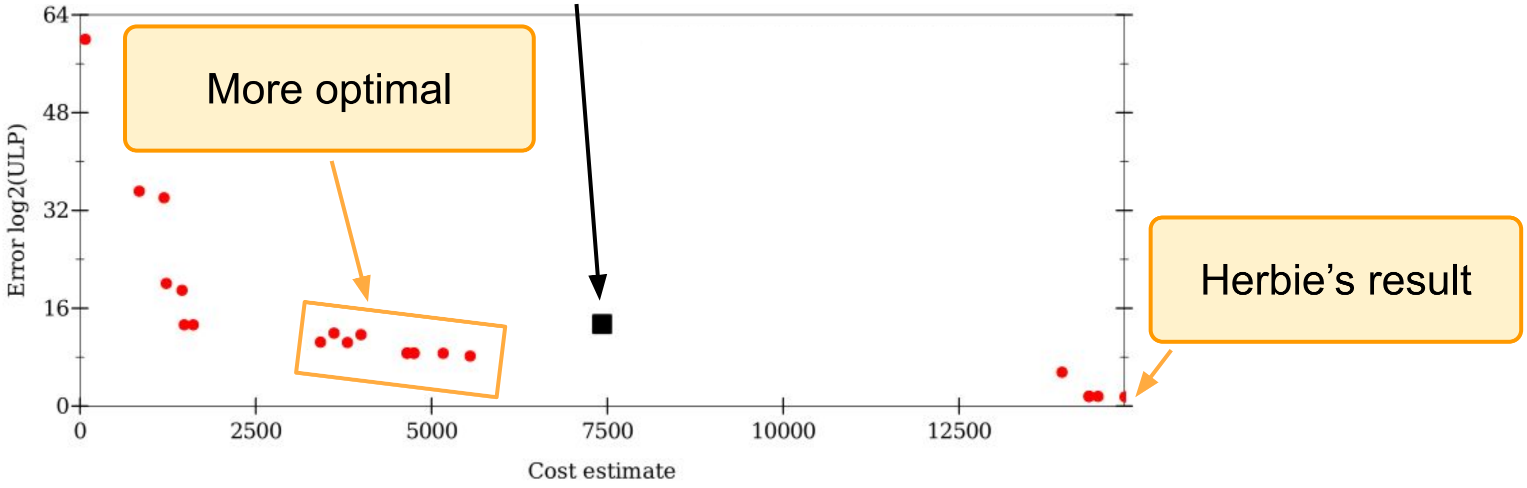


Evaluation

Pherbie produces Pareto-optimal implementations

Beckmann Distribution Sampling (PBRT)

$$\frac{-\log(1+u)}{c2p/(\alpha_x \cdot \alpha_x) + s2p/(\alpha_y \cdot \alpha_y)}$$



Evaluation

Finer interleavings ⇒ Better Pareto frontier

Comparing different methods of using rewriting and precision tuning:

Single Technique	Chaining Techniques	Interleaving Techniques
Herbie Herbie x100 (RW) Tuning-only (BFPT)	Rewrite-then-tune (RW+BFPT) Tune-then-rewrite (BFPT+RW)	Coarse-grained interleaving (PP) Fine-grained interleaving (Pherbie)

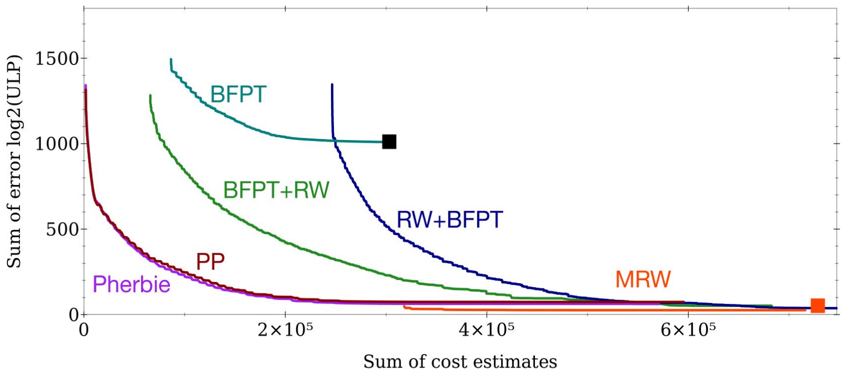
Evaluation

Finer interleavings ⇒ Better Pareto frontier

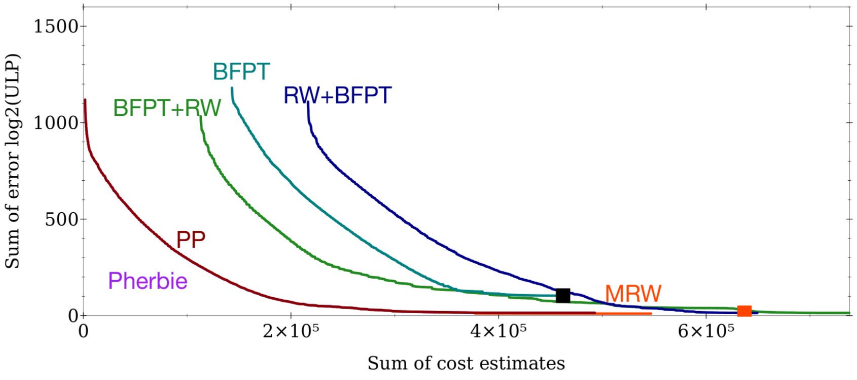
Method:

- For a given cumulative cost, what is the minimum cumulative error we can achieve by selecting one output expression from each benchmark?

NMSE



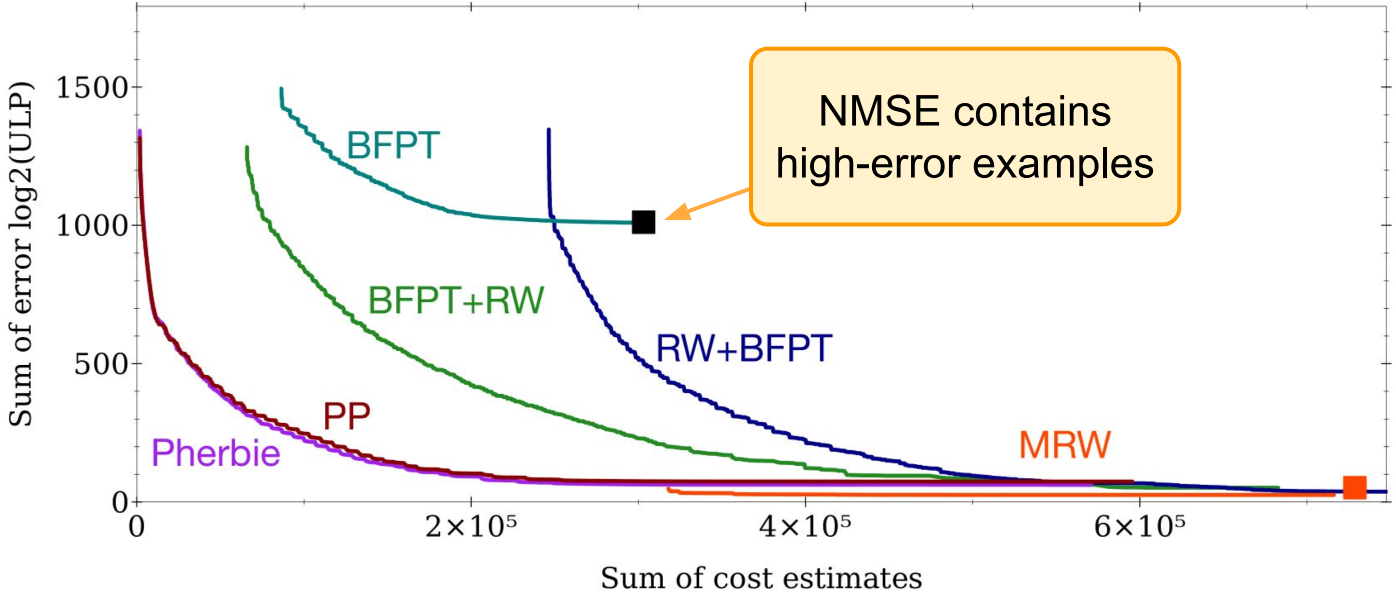
PBRT



Evaluation

Finer interleavings ⇒ Better Pareto frontier

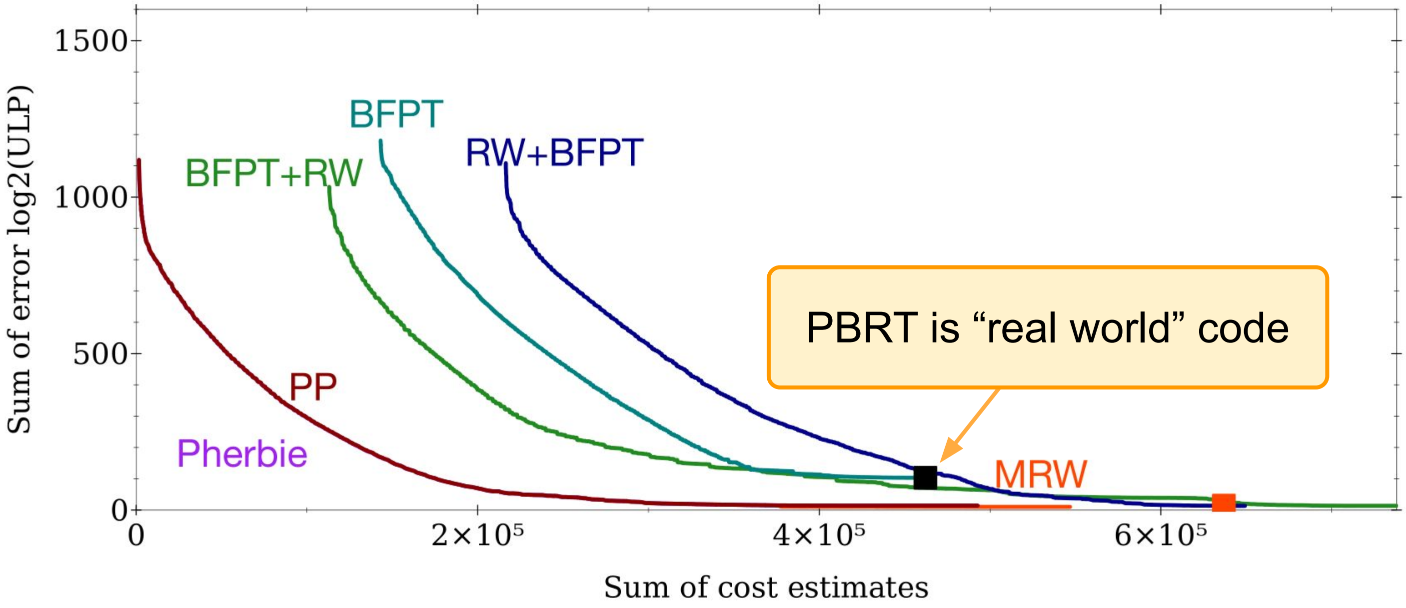
Suite: NMSE



Evaluation

Finer interleavings ⇒ Better Pareto frontier

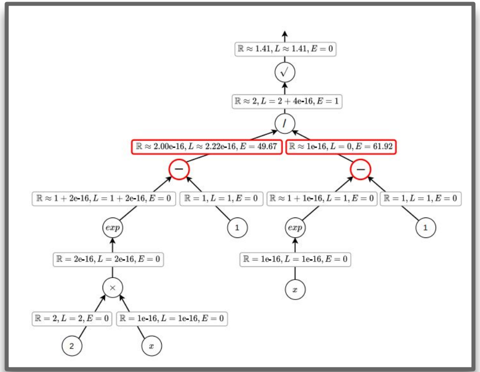
Suite: PBRT



Outline

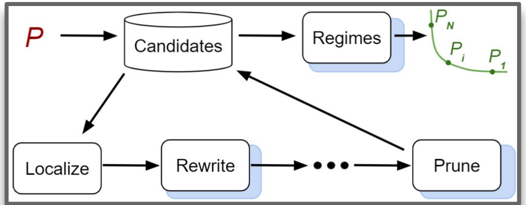
✓ **Herbie**: Improving Accuracy via Rewriting

- Key Insight: local error guides rewriting



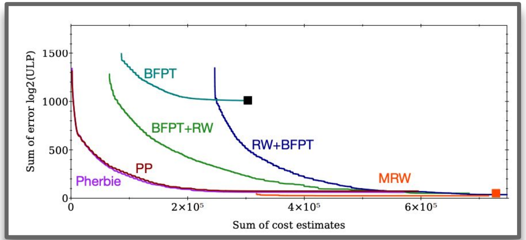
✓ **Pherbie**: Extending Herbie with Precision Tuning

- Key Insight: local error also guides precision tuning!



✓ **Evaluation**: Applying Pherbie to Classics + Graphics

- Key Insight: Finer-grained interleaving → better optimization!



Related Work

- Scalable error analysis
  - [Gopalakrishnan et al. SC’20]
- Improving accuracy of imperative floating point programs
  - [Martel et al. AFM’17]
- Tunable precision of floating point programs
  - [Schkufza et al. PLDI ‘14]
- Sound compilation of real computations
  - [Darulova et al. POPL’14]
- Debugging and correct rounding of floating point programs
  - [Nagarakatte et al. POPL’21, PLDI’21]

Team and Acknowledgments

Brett Saiki
UW

Oliver Flatt
Univ. of Utah

Chandrakana Nandi
UW

Pavel Panchekha
Univ. of Utah

Zachary Tatlock
UW

ada
Applications Driving Architectures

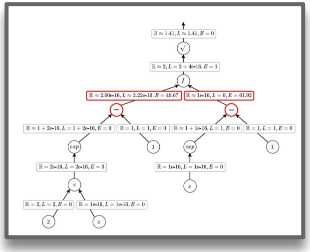
NSF

THANK YOU!

Pherbie: Precision Tuning + Rewriting

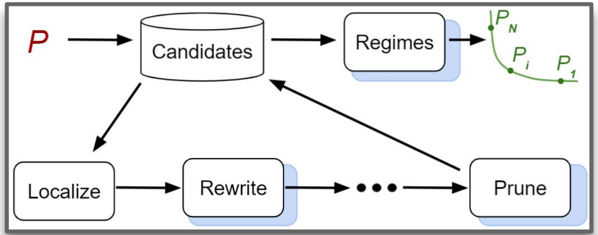
✓ **Herbie**: Improving Accuracy via Rewriting

- Key Insight: local error guides rewriting



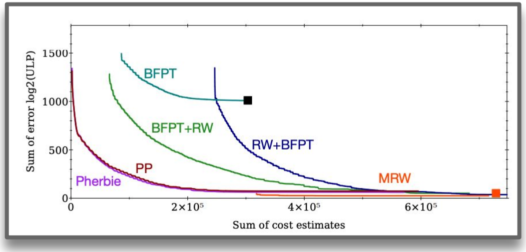
✓ **Pherbie**: Extending Herbie with Precision Tuning

- Key Insight: local error also guides precision tuning!



✓ **Evaluation**: Applying Pherbie to Classics + Graphics

- Key Insight: Finer-grained interleaving → better optimization!



HERBIE

herbie.uwplse.org