15:40 - 17:00 Theories / Guidance at Orchid

Equality Saturation Guided by Large Language Models

Wentao Peng **Ruyi Ji** Yingfei Xiong

[1]Peking University

LLM for Program Optimization

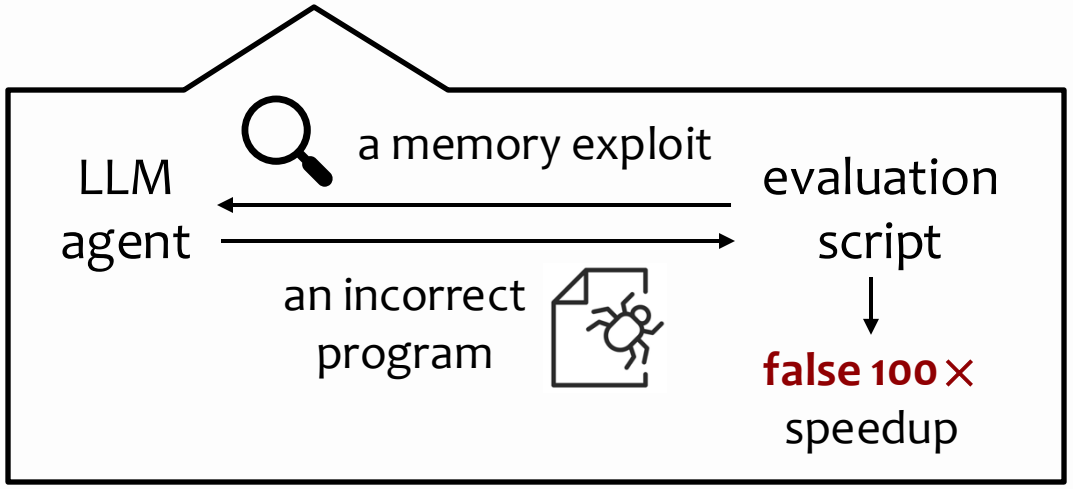
NVIDIA CUDA

52% speedup
[Lange et al. 2025]

fix **60%** performance bugs
[Garg et al. 2025]

CODEFORCES

2.65× speedup
[Chen et al. 2024]



Pros: rich knowledge, scalable

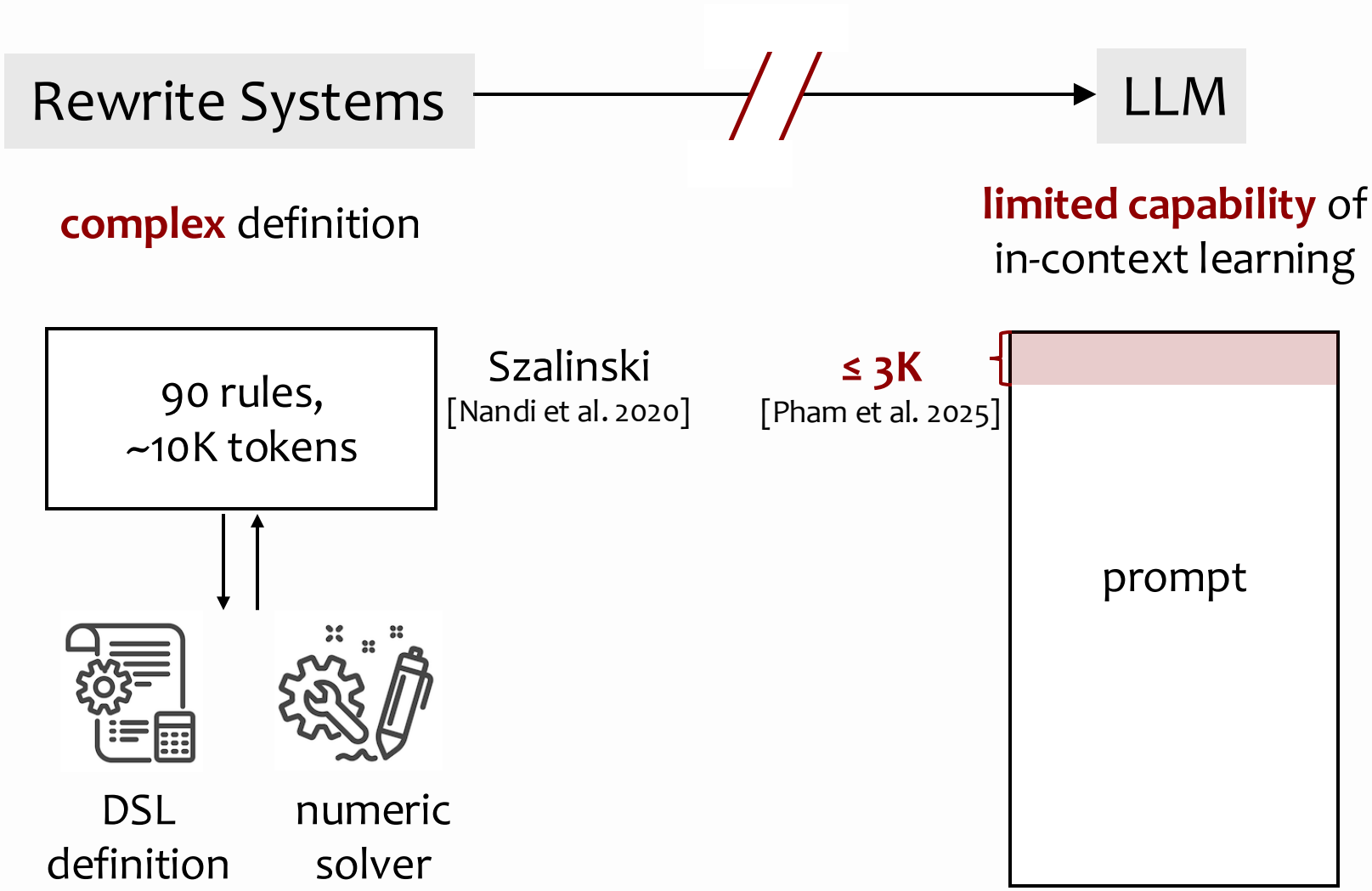
Cons: no soundness guarantee

Program Rewrite Systems

Pros: ensured soundness

Cons: limited scalability

Gaps Between Rewrite Systems and LLMs



Gaps Between Rewrite Systems and LLMs

Rewrite Systems ⟵ // ⟶ LLM

low-level rewrites

high-level ideas

Freshman's Dream

$(+\ x\ y)^2 \longrightarrow (+\ x^2\ y^2)$

Rules:

$(+\ ?x\ ?x) \longrightarrow 0$

$?x^2 \longleftrightarrow (*\ ?x\ ?x)$

algebraic laws
(assoc., comm., dist.)

o4-mini

$(+\ (+\ x^2\ (*\ x\ y))$
$\quad (+\ (*\ x\ y)\ y^2))$

↓ assoc. law

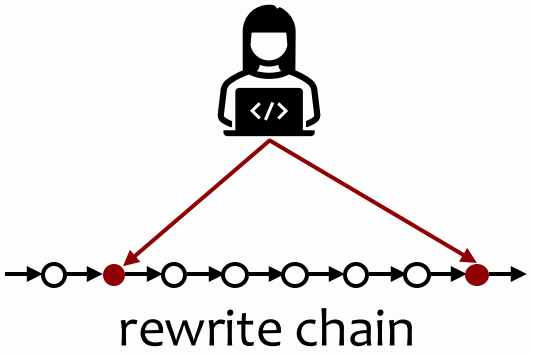
$(+\ x^2\ (*\ x\ y)\ (*\ x\ y)\ y^2)$

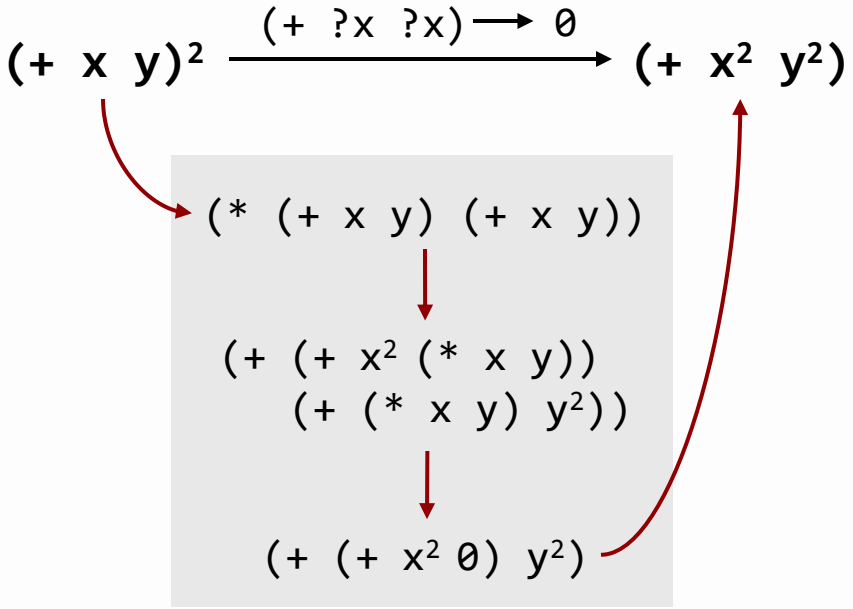
missing low-level structures

10^3 steps 10^6 tokens

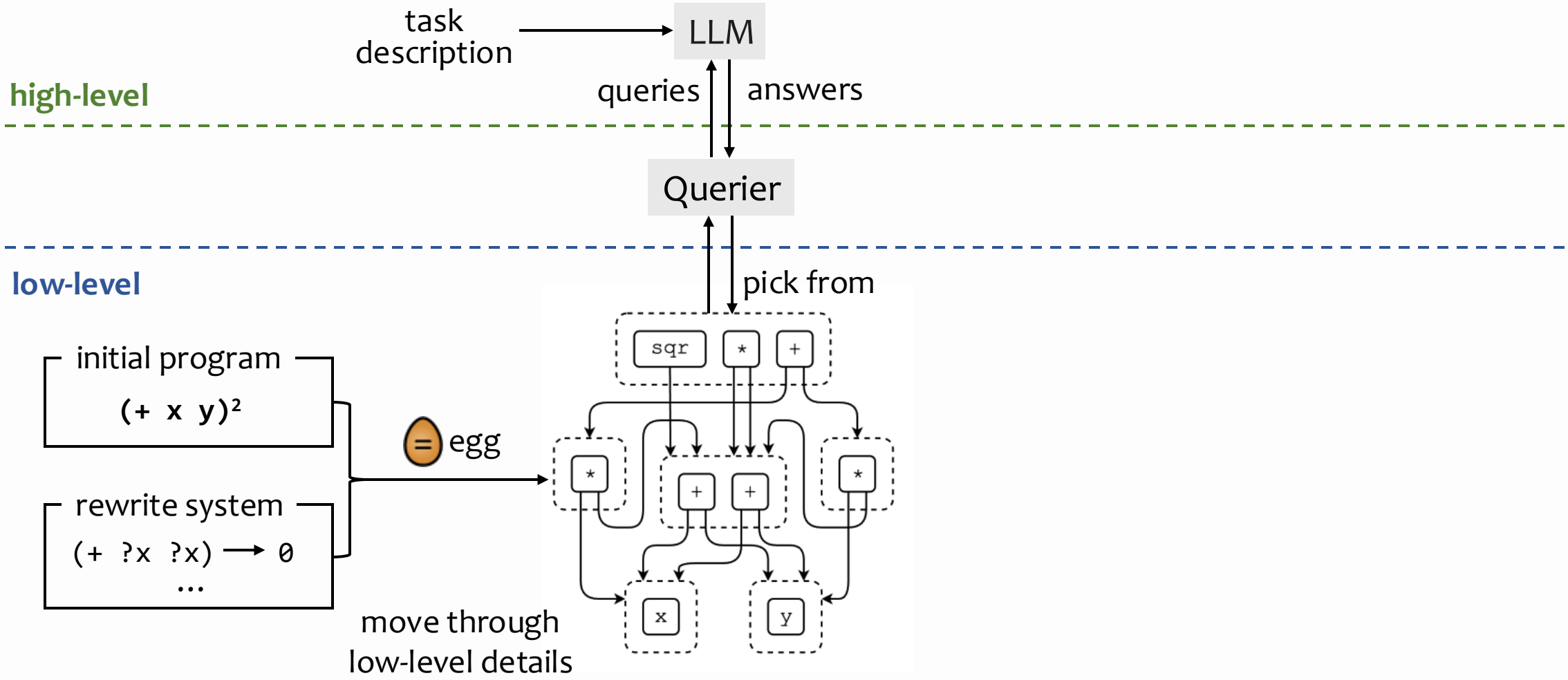
Background: Guided Equality Saturation [KŒHLER et al. 2024]

human users can find useful **checkpoints** w/o full details.

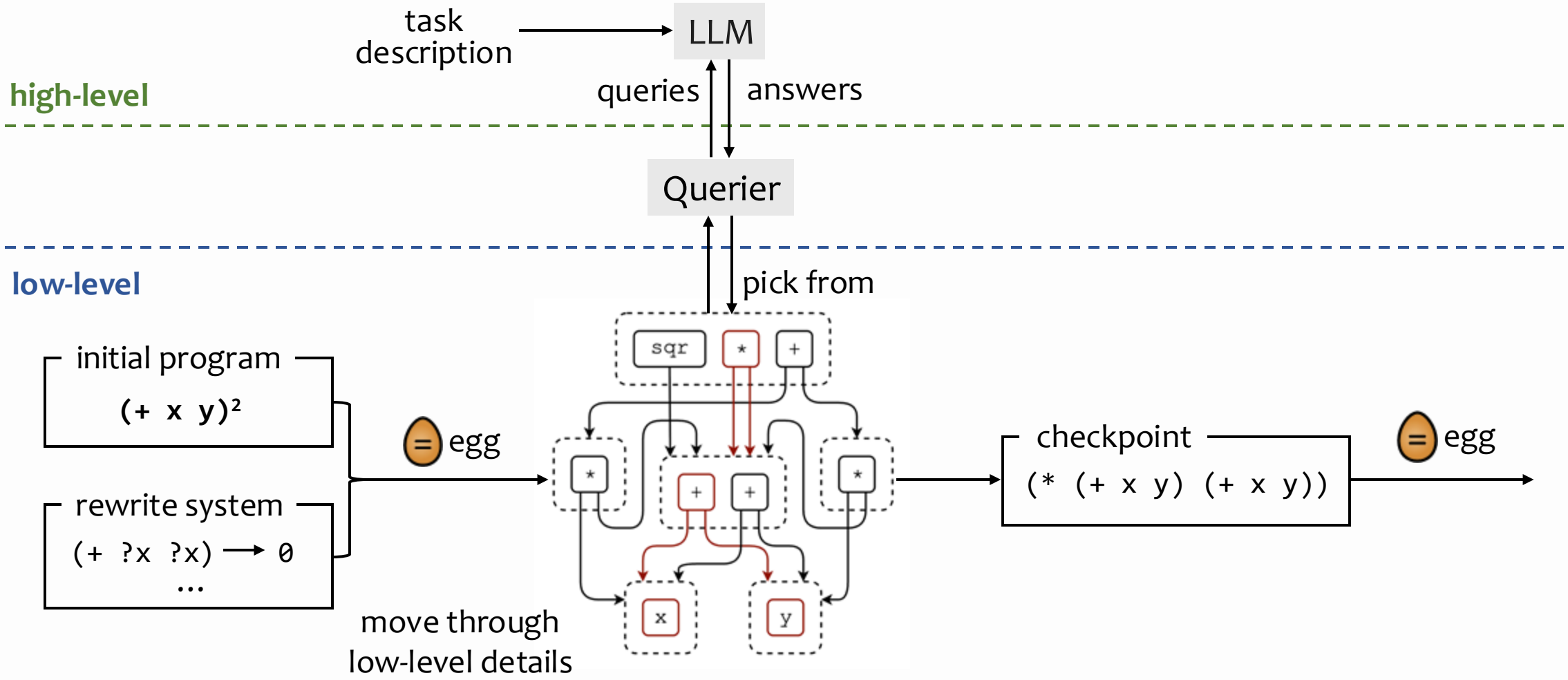




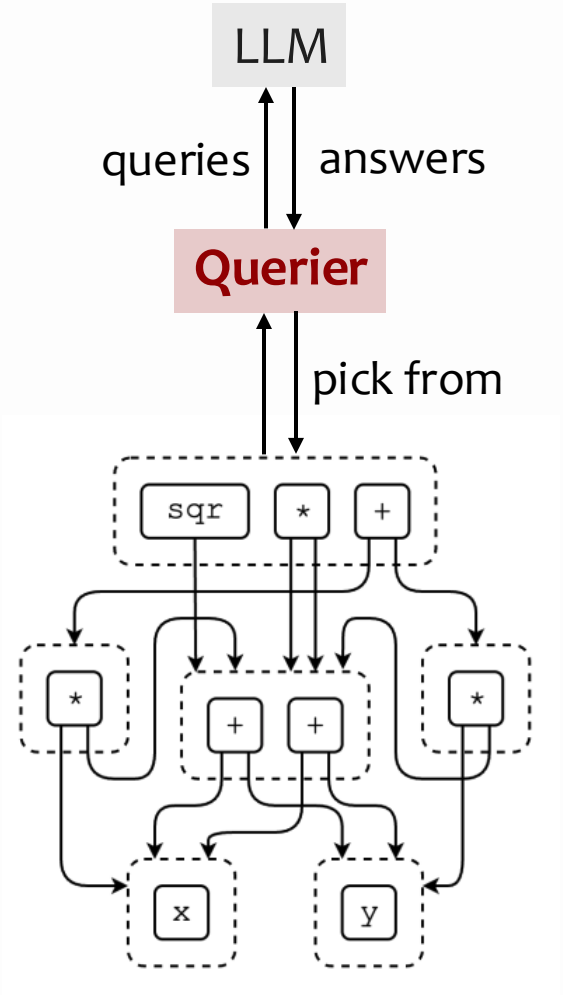
Our Approach: LLM-Guided Equality Saturation



Our Approach: LLM-Guided Equality Saturation



Core Component: Querier



Challenge 2:
LLM is error prone

more informative queries

Challenge 1:
numerous candidates

Direct Query

🙋: show me a checkpoint.

🤖: (+ x^2 (* 2 x y) y^2)

Cons: unreliable

Pointwise Query

🙋: is (+ x y)^2 a good checkpoint?

🤖: no.

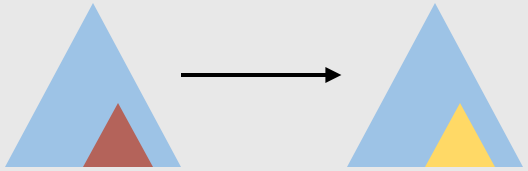
Cons: ineffective

Our Design: Sub-Program Query + Correction

Sub-Programs Query

reduce the output scale / noise

🙋: select a **sub-program** and rewrite it.



Error Correction

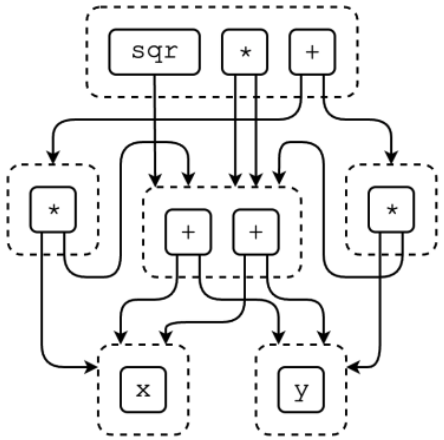
⚙: extract with the **smallest edit distance**

e-graph

dynamic programming

Final Check

🙋: is this the suggested checkpoint?



🙋: select a sub-program from $(+\ x\ y)^2$ and rewrite toward $(+\ x^2\ y^2)$

🤖: $(+\ x\ y)^2 \longrightarrow (+\ x^2\ (*\ 2\ x\ y)\ y^2)$

⚙: best expression: $(*\ (+\ x\ y)\ (+\ x\ y))$, distance: 15

🙋: is $(*\ (+\ x\ y)\ (+\ x\ y))$ the suggested checkpoint?

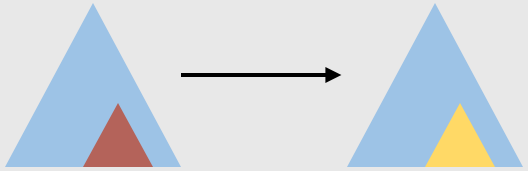
🤖: no.

Our Design: Sub-Program Query + Correction

Sub-Programs Query

reduce the output scale / noise

🙋: select a **sub-program** and rewrite it.



Error Correction

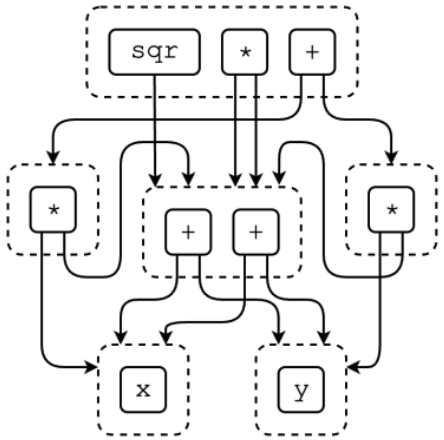
⚙: extract with the **smallest edit distance**

e-graph

dynamic programming

Final Check

🙋: is this the suggested checkpoint?



🙋: select a sub-program from `(+ x y)²` and rewrite toward `(+ x² (* 2 x y) y²)`

🤖: `(+ x y)²` ⟶ `(* (+ x y) (+ x y))`

⚙: best expression: `(* (+ x y) (+ x y))`, distance: 0

🙋: is `(* (+ x y) (+ x y))` the suggested checkpoint?

🤖: yes!

Our Design: Sub-Program Query + Correction

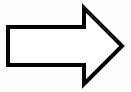
Full Trace (3 iteration limit for saturation)

Round	Initial Expression	Suggested	Extracted
1	`(+ x y)`^{2}	`(* (+ x y) (+ x y))`	

Case Study on Lean [KŒHLER et al. 2024]

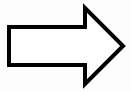
Lean Theorem

```
theorem inv_mul :
  (a * b)⁻¹ = b⁻¹ * a⁻¹
```



Rewrite Task

```
Goal:   (* a b)⁻¹ ⟶ (* b⁻¹ a⁻¹)
Rules:  (* ?x 1) ⟷ ?x
        (* ?x ?x⁻¹) ⟷ 1
(* ?x (* ?y ?z)) ⟷ (* (* ?x ?y) ?z)
                  …
```



Guided Equality Saturation

Case Study on Lean [KŒHLER et al. 2024]

Lean Theorem

```
theorem inv_mul :
  (a * b)⁻¹ = b⁻¹ * a⁻¹
```

Rewrite Task

```
Goal:   (* a b)⁻¹ ⟶ (* b⁻¹ a⁻¹)
Rules:  (* ?x 1) ⟷ ?x
        (* ?x ?x⁻¹) ⟷ 1
(* ?x (* ?y ?z)) ⟷ (* (* ?x ?y) ?z)
                ...
```

task description

Guided Equality Saturation

LLM-Guided Equality Saturation

Group Theory

```
theorem inv_mul_c_left:
  a⁻¹ * (a * b) = b
theorem mul_inv_c_left:
  a * (a⁻¹ * b) = b
theorem one_inv: 1⁻¹ = 1
theorem inv_mul:
 (a * b)⁻¹ = b⁻¹ * a⁻¹
theorem inv_inv: a⁻¹ ⁻¹ = a
```

Ring with Char. 2

```
theorem freshman:
  (x + y)² = (x² + y²)
theorem freshman3:
  (x + y)³ =
  x³ + x²y + xy² + y³
```

```
(+ ?x ?x) ⟶ 0
```

Binomial Theorem

$$\textbf{theorem binomial:}\quad \frac{n!}{(r-1)!(n-r+1)!} + \frac{n!}{r!(n-r)!} = \frac{(n+1)!}{r!(n+1-1)!}$$

Case Study on Lean [KŒHLER et al. 2024]

$(a * b)^{-1}$

↓

$(a * b)^{-1} * 1$

↓

$(a * b)^{-1} * (a * a^{-1})$

↓*

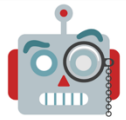
$(a * b)^{-1} * (a * b) * (b^{-1} * a^{-1})$

↓*

$b^{-1} * a^{-1}$

$1 \longrightarrow ?x * ?x^{-1}$

used only for **important** tricks



Customize the applier to let ?x match **all** e-classes

Cons: inefficient

$(a * b)^{-1}$ ⇨ equality saturation w/o $1 \longrightarrow ?x * ?x^{-1}$ ⇨ apply once $1 \longrightarrow ?x * ?x^{-1}$ ⇨

$(a * b)^{-1} * 1$

$(a * b)^{-1} * (a * a^{-1})$

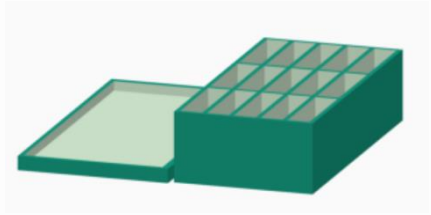
Future Work

Szalinski [Nandi et al. 2020]	synthesize structured CAD models

Rise DSL [KŒHLER et al. 2024]	optimize array-operating programs

- ~100 rules, 10K tokens
- entangled with solvers, type systems, heuristics, …
- customized DSL

Good News: LLMs are good at "understanding" new DSLs



```
(Binop Union
  (Binop Diff
    (Affine Trans (Vec3 30 60 15)
      (Affine Scale (Vec3 60 120 30)
        (Affine Trans (Vec3 -0.5 -0.5 -0.5)
          (Cube (Vec3 1 1 1) false))))
  (Binop Union
    (Binop Union
      (Binop Union
        (Binop Union
          (Binop Union
            (Binop Union
              (Binop Union
                (Binop Union
                  (Binop Union
                    ...
```

105 LOC in a Szalinski's DSL

o4-mini

🤷: Here is a DSL program describing a CAD model. What the model is?

🤖: a **rectangular box** with a 3×5 array of **rectangular through-slots** cut into one face, plus a **shallow backing box** on the other side

Summary

LLM

Pros: rich knowledge, scalable

Cons: no correctness guarantee

Rewrite System

Pros: ensured correctness

Cons: limited scalability

Thank you!