

# Scalable & Flexible Equality Saturation

Jakub Bachurski

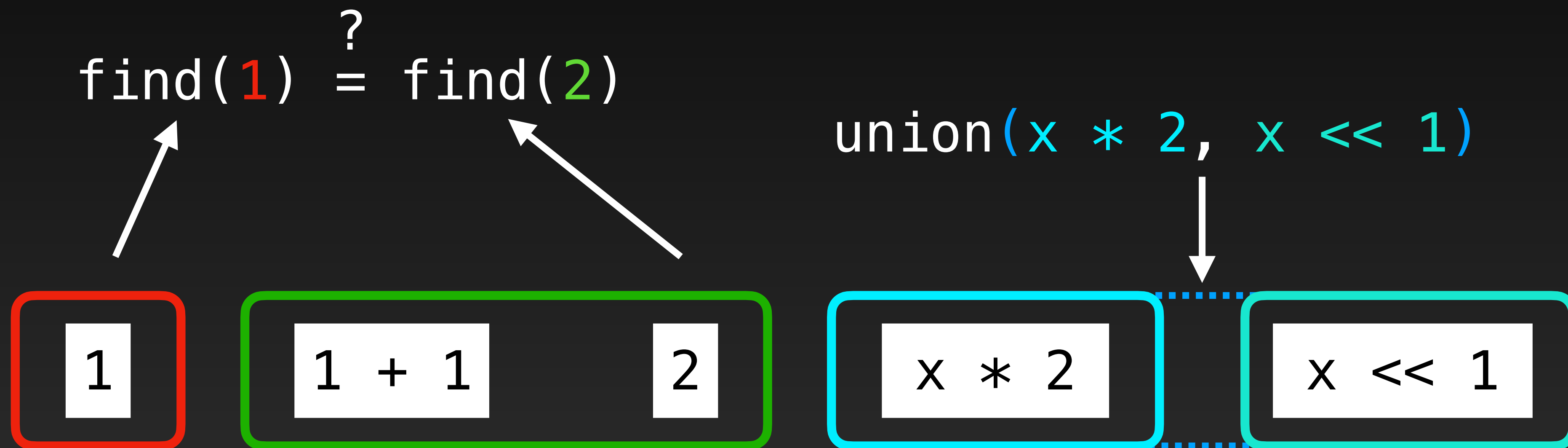
# Equality Saturation

## *Highlights*

- Promising approach for optimising compilers [Willsey, 2021] — egg SOTA
  - **Rewrite**-based optimisation, **non-destructively**
  - Solves (?!) **phase ordering**, can replace backtracking (TENSAT)
- Problem: *saturation* is a **lie**
- Key: **e-graphs**

# Equivalence

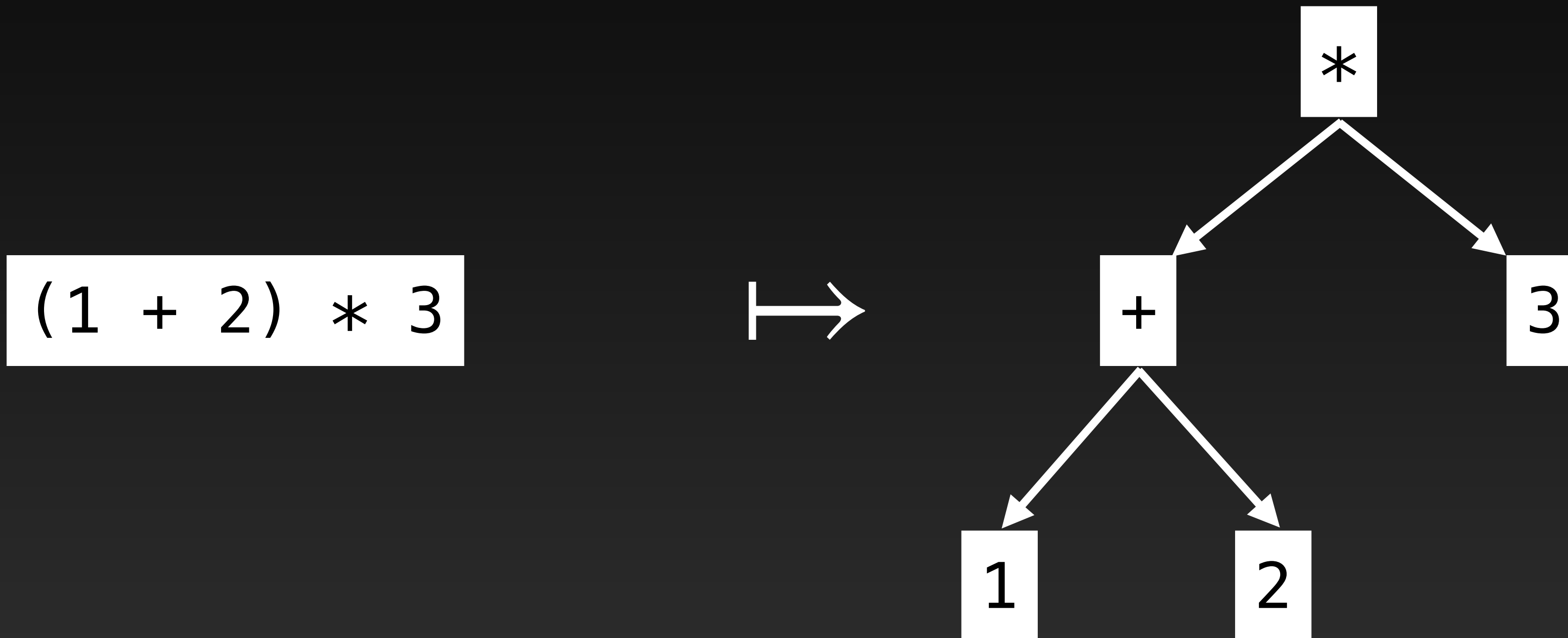
Disjoint Sets / Find & Union [Galler and Fischer, 1964]



$$\begin{cases} e = e & \text{(refl)} \\ e = e' \implies e' = e & \text{(sym)} \\ e = e' \wedge e' = e'' \implies e = e'' & \text{(trans)} \end{cases}$$

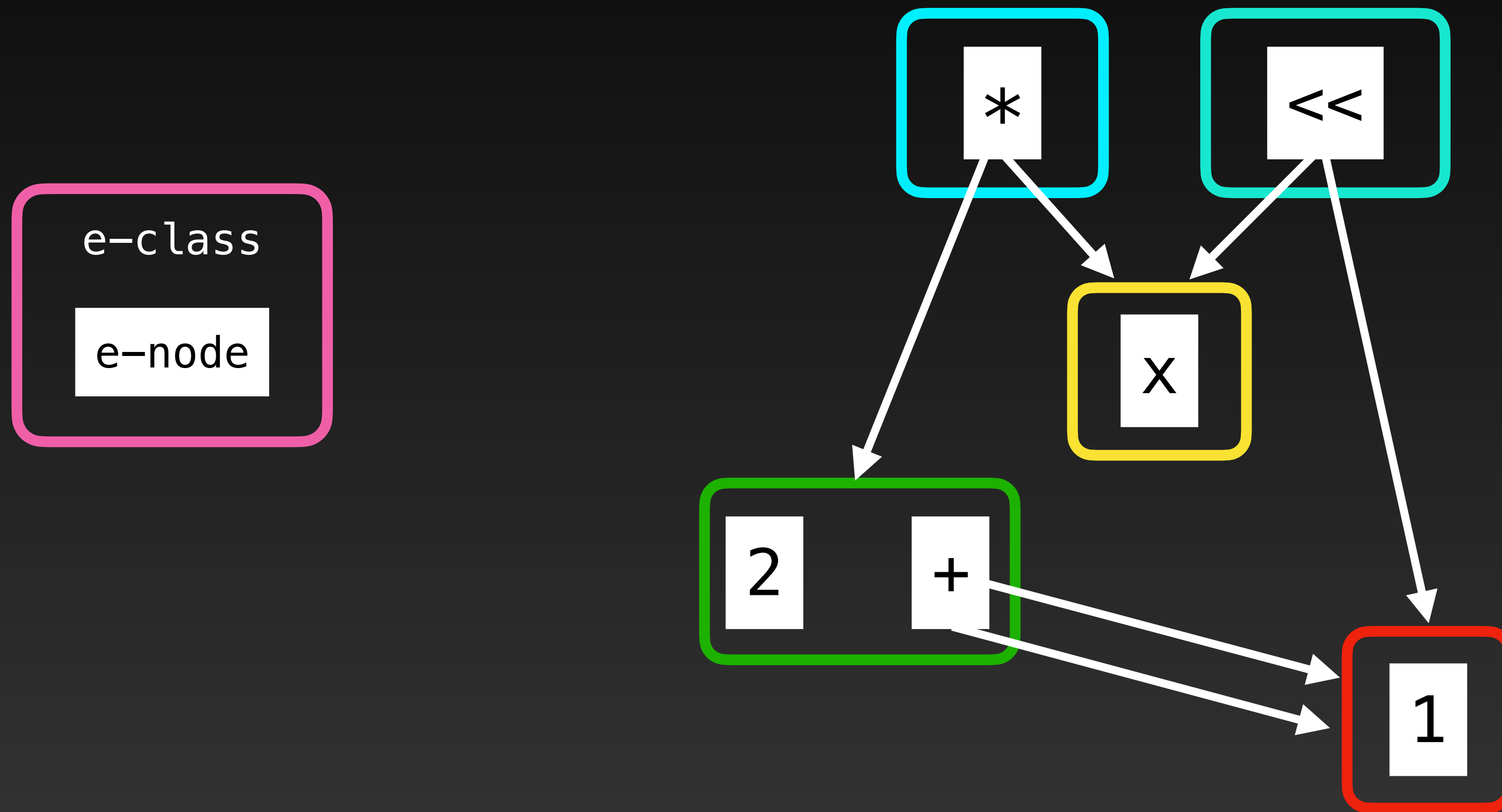
# Terms

*everything is a graph*



# Congruence

## E-Graphs [Nelson, 1980]



$2 = 1 + 1$   
 thus  
 $x * 2 = x * (1 + 1)$

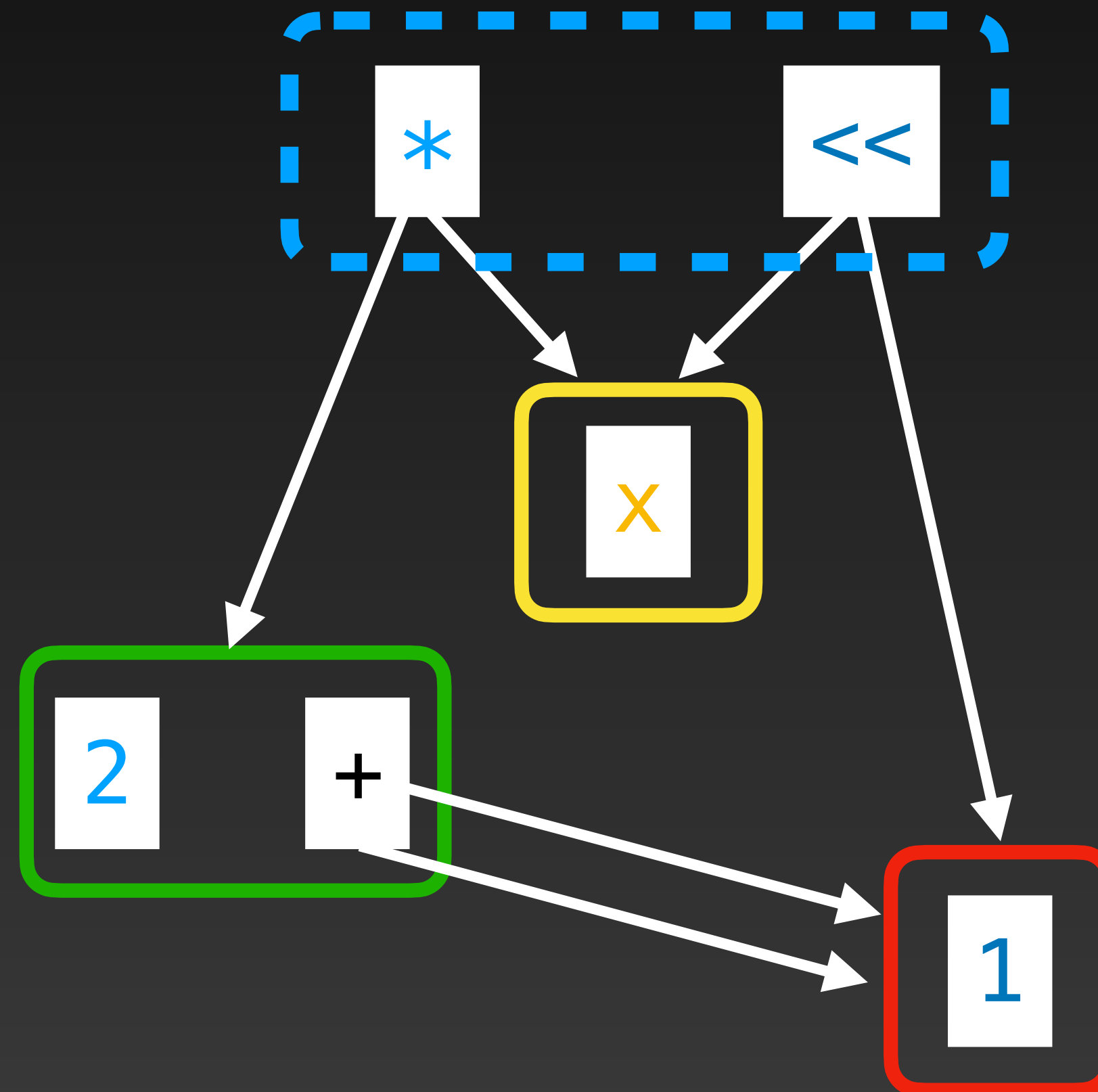
$$e = e' \implies \mathcal{C}[e] = \mathcal{C}[e']$$

$$(1 + 1 = 2 \implies x * (1 + 1) = x * 2)$$

# Rewrites

## E-Graph Matching

$(x * 2) \Rightarrow (x \ll 1)$



$= x * 2$   
 $= x * (1 + 1)$   
 $= x \ll 1$

# So what?

## Equality saturation

- **Equality saturation** uses the **e-graph** to find all **reachable programs**
- *No more rewrites— saturation*
- Afterwards, **e-graph extraction**
  - most **concise**, most **precise**, most **performant** program



**Thank you!**