Compiler Design

An Introduction to Equality Saturation and Its Applications

March 28, 2025

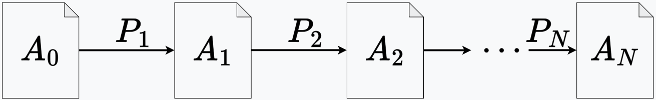
Abd-El-Aziz Zayed

McGill

Table Of Contents

Pass Sequences

An optimizer performs a sequence of passes (a.k.a. transformations, rewrites) on the program to find the optimal version.



Optimization Passes

- Constant folding
- Dead code elimination
- Common subexpression elimination
- Loop unrolling
- Loop fusion
- Loop tilling
- Loop vectorization

Phase Ordering Problem

The order in which passes are applied affects the final program and its performance.

Phase Ordering Problem

The order in which passes are applied affects the final program and its performance.

- P_1. $xy/z \to x(y/z)$
- P_2. $x \cdot 2 \to x \ll 1$
- P_3. $xy \to yx$
- P_5. $x/x \to 1$
- P_6. $x \cdot 1 \to x$

Phase Ordering Problem

The order in which passes are applied affects the final program and its performance.

- P_1. $xy/z \to x(y/z)$
- P_2. $x \cdot 2 \to x \ll 1$
- P_3. $xy \to yx$
- P_5. $x/x \to 1$
- P_6. $x \cdot 1 \to x$

- $(a \cdot 2)/2 \xrightarrow{P_2} (a \ll 1)/2$
- $(a \cdot 2)/2 \xrightarrow{P_1} a \cdot (2/2) \xrightarrow{P_5} a \cdot 1 \xrightarrow{P_6} a$

Phase Ordering Problem

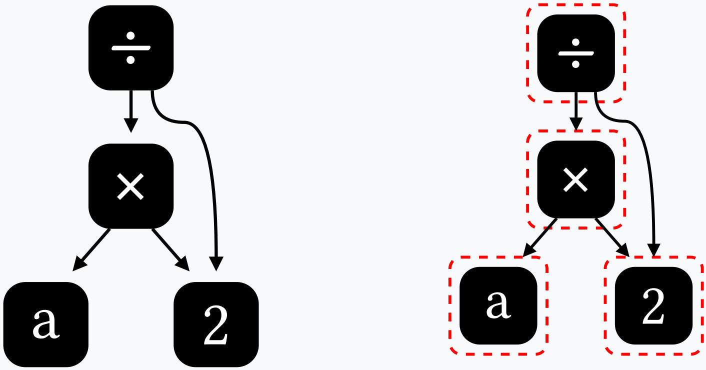
- $(a \cdot 2)/2 \xrightarrow{P_2} (a \ll 1)/2$
- $(a \cdot 2)/2 \xrightarrow{P_1} a \cdot (2/2) \xrightarrow{P_5} a \cdot 1 \xrightarrow{P_6} a$

Problems:

1. The compiler gets stuck in a local optimum if it applies P_2 before P_1.
2. The passes are destructive: we lose the original and intermediate versions of the program.

Equivalence Graphs

We build an **e**-graph to store all defined equivalent versions of the program.

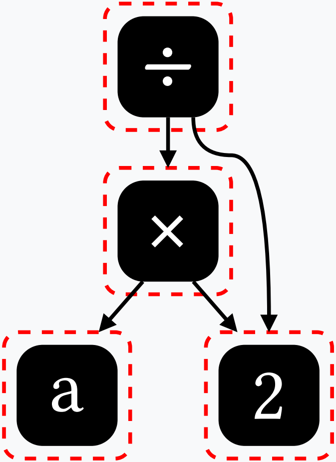


$$(a \cdot 2)/2$$

Equivalence Graphs

Store many equivalent versions of a program.

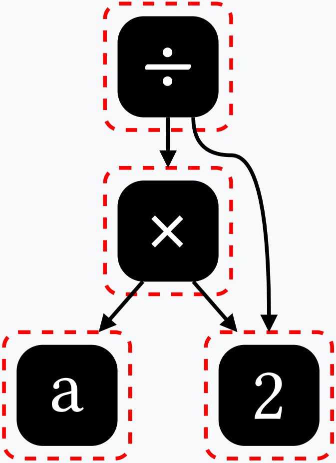
1. **e-node**: Operation node.
2. **e-class**: Set of equivalent e-nodes.
3. **e-graph**: Set of e-classes.



Equality Saturation

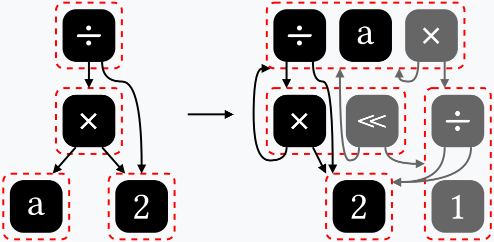
Process:

1. Build an **e**-graph of $(a \cdot 2)/2$.
2. Apply the rewrite rules to the e-graph until a fixed-point.
3. Extract the optimized expression via a cost model.

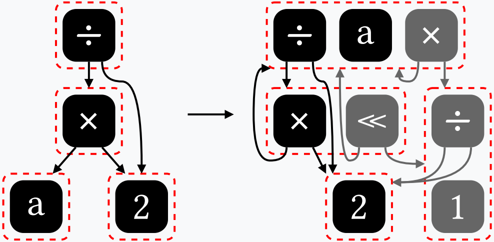


Equality Saturation

- P_1. $xy/z \to x\,(y/z)$
- P_2. $x \cdot 2 \to x \ll 1$
- P_3. $xy \to yx$
- P_5. $x/x \to 1$
- P_6. $x \cdot 1 \to x$



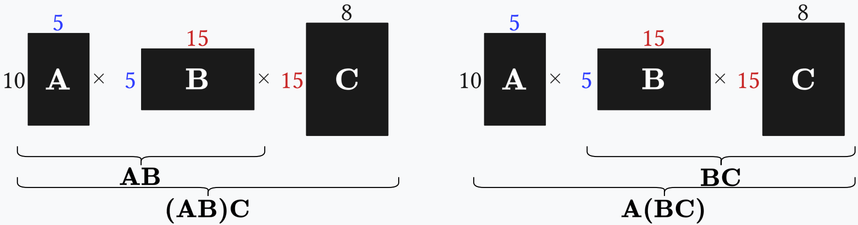
Equality Saturation



Problems are solved:

1. The global optimum is found by extracting the optimal expression via a cost model.
2. The original and intermediate versions of the program are preserved.

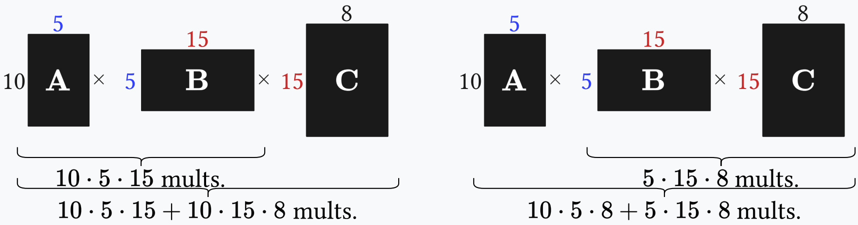
MatMul Associativity



$$\mathbf{(AB)\,C} = \mathbf{A\,(BC)}$$

How do we find out which one is more efficient?

The Order of Operations Matters

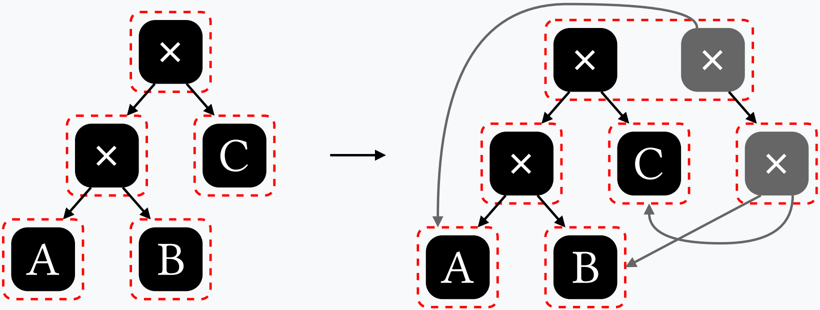


`cost` **(AB) C** = 1950 multiplications.

`cost` **A (BC)** = 1000 multiplications.

MatMul Associativity

$$\mathbf{(AB)\,C = A\,(BC)}$$



Polynomial Evaluation: Horner's Method

Reduce from $O(n^2)$ to $O(n)$ multiplications.

$$
\begin{aligned}
P(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \\
&= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n)))
\end{aligned}
$$

Polynomial Evaluation: Horner's Method

Reduce from $O(n^2)$ to $O(n)$ multiplications.

$$\begin{aligned} P(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \\ &= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n))) \end{aligned}$$

- Exponentiation: $\underline{x^0 \Rightarrow 1}$ and $\underline{x^n \Rightarrow x \cdot x^{n-1}}$

Polynomial Evaluation: Horner's Method

Reduce from $O(n^2)$ to $O(n)$ multiplications.

$$\begin{aligned} P(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \\ &= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n))) \end{aligned}$$

- Exponentiation: $\underline{x^0 \Rightarrow 1}$ and $\underline{x^n \Rightarrow x \cdot x^{n-1}}$
- Commutativity: $\underline{x + y \Leftrightarrow y + x}$ and $\underline{x \cdot y \Leftrightarrow y \cdot x}$
- Associativity: $\underline{(x + y) + z \Leftrightarrow x + (y + z)}$ and $\underline{(x \cdot y) \cdot z \Leftrightarrow x \cdot (y \cdot z)}$
- Distributivity: $\underline{x \cdot (y + z) \Leftrightarrow x \cdot y + x \cdot z}$
- Identity: $\underline{x \cdot 1 \Rightarrow x}$

References

- Equality Saturation: a New Approach to Optimization. [Tate et al., 2009]
- DialEgg: Dialect-Agnostic MLIR Optimizer using Equality Saturation with Egglog. [Zayed and Dubach, 2025]
- Latent Idiom Recognition for a Minimalist Functional Array Language using Equality Saturation. [der Cruysse and Dubach, 2023]

der Cruysse, J. V. and Dubach, C. (2023).

Latent idiom recognition for a minimalist functional array language using equality saturation.

Tate, R., Stepp, M., Tatlock, Z., and Lerner, S. (2009).

Equality saturation: a new approach to optimization.

In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '09, page 264–276, New York, NY, USA. Association for Computing Machinery.

Zayed, A.-E.-A. and Dubach, C. (2025).

Dialegg: Dialect-agnostic mlir optimizer using equality saturation with egglog.

In *Proceedings of the 23rd ACM/IEEE International Symposium on Code Generation and Optimization*, CGO '25, page 271–283, New York, NY, USA. Association for Computing Machinery.