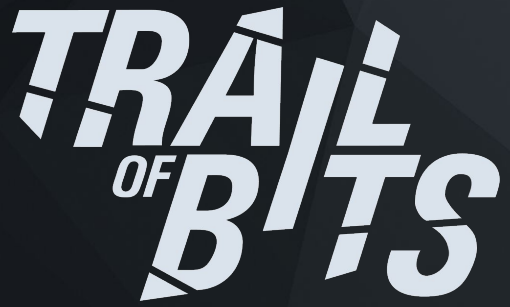

TRAIL
OF
BITS

On the Optimization of Equivalent Concurrent Computations*

Henrich Lauko, Lukáš Korenčik & Peter Goodman

*Research funded by DARPA SIEVE program.

On the Optimization of ~~Equivalent Concurrent Computations~~ Common Subexpressions*

Henrich Lauko, Lukáš Korenčik & Peter Goodman

*Research funded by DARPA SIEVE program.

Problem Statement

- Find common subexpressions independent of their arguments
- Factor out all of them or none to a separate function

$$f(e(a_1, a_2)) \wedge h(e(b_1, b_2))$$

$$\downarrow$$

$$f(g(a_1, a_2)) \wedge h(g(b_1, b_2)) \wedge g(c_1, c_2) = e(c_1, c_2)$$

Pattern Extraction Example

$$v_1 = a_1 \times (a_2 \times 2) \land v_2 = b_1 \times (b_2 \ll 1)$$

$$\downarrow$$

extract pattern of form $c_1 \times (c_2 \times 2)$

$$\downarrow$$

$$v_1 = g(a_1, a_2) \land v_2 = g(b_1, b_2) \land g(c_1, c_2) = c_1 \times (c_2 \times 2)$$

Applications

- **Program refactoring**
- **Term simplification**

Optimization of arithmetic circuits for ZK proofs

- **Application targets optimization of circuits for the ZK proofs**
- **Extraction of common arithmetic logic units from generated circuit**

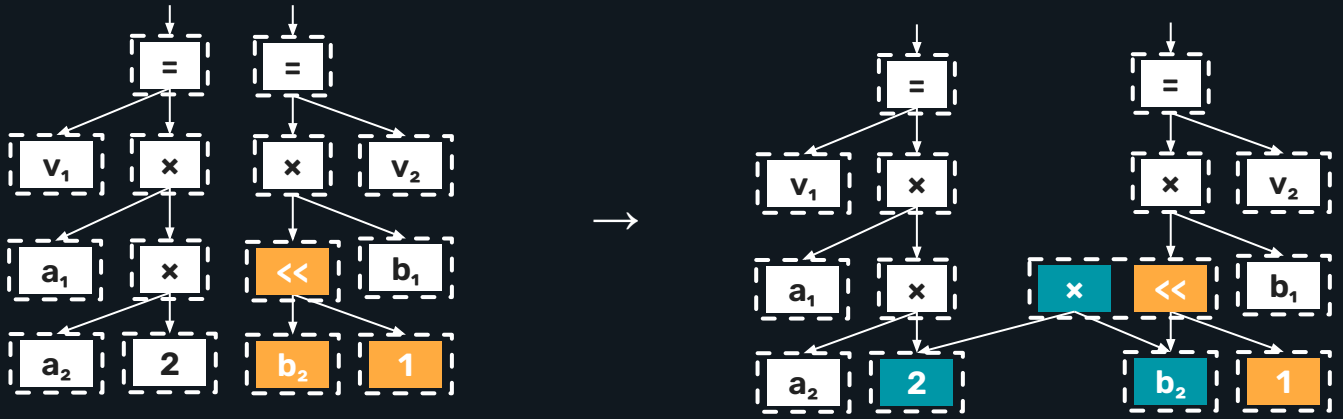
Problems to Solve

1. Keep the equality saturation algorithm
2. Extend the ematch to find common subexpressions
3. Represent subexpression relation in the e-graph
4. Extract refactored structure

1. Apply rewrite rules as usual

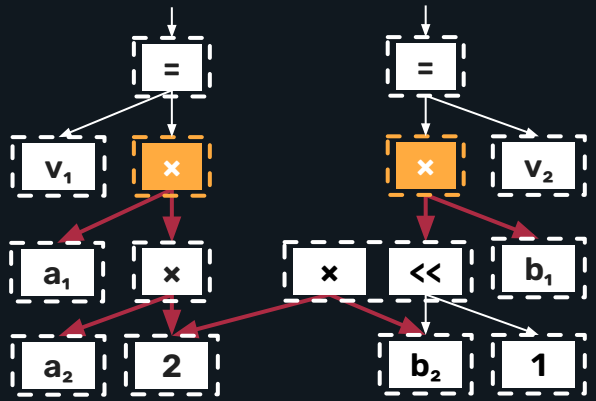
Original formula: $v_1 = a_1 \times (a_2 \times 2) \wedge v_2 = b_1 \times (b_2 \ll 1)$

Rewrite rule: $? \ll 1 \rightarrow ? \times 2$



2. Find common subexpression

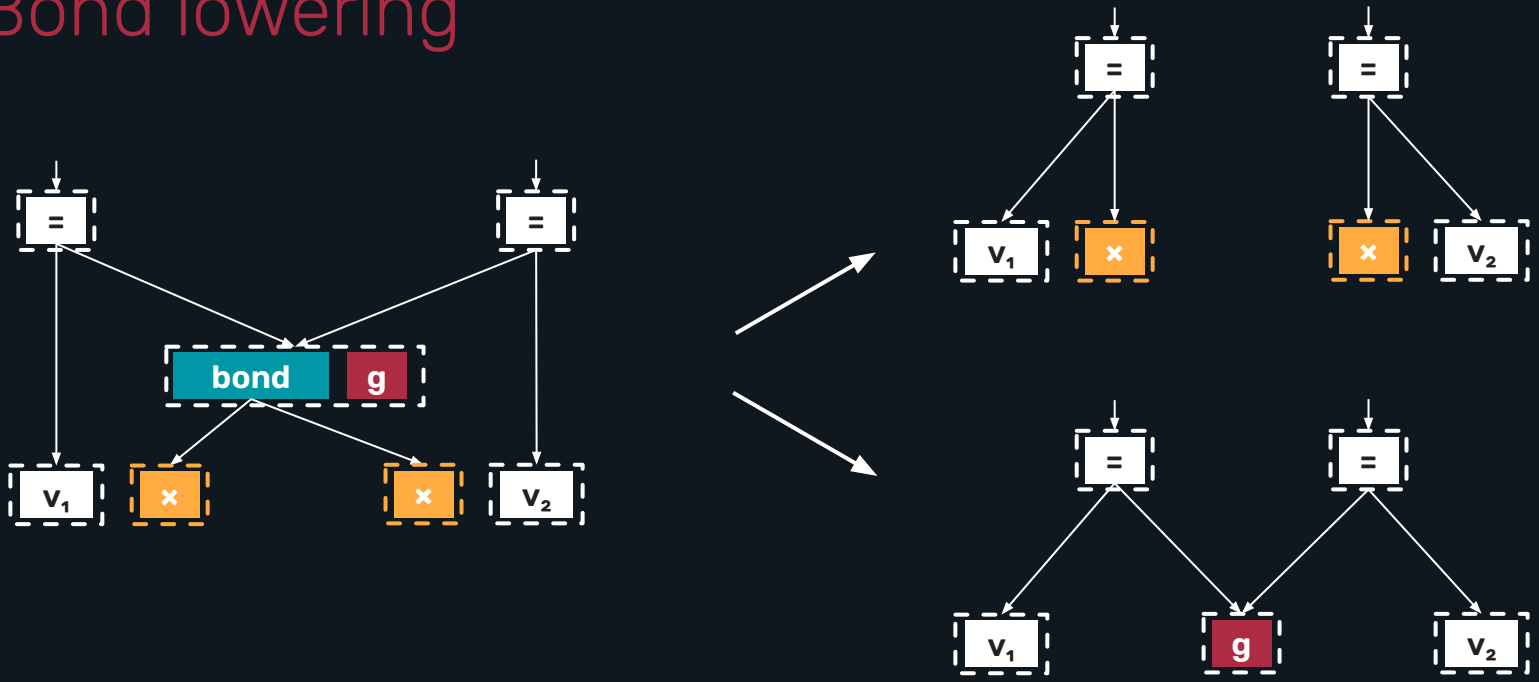
- **New ematch rule:** (let E ($?_1 \times (?_2 \times 2)$)) (match E…) →
- Eagerly match all patterns of a given form



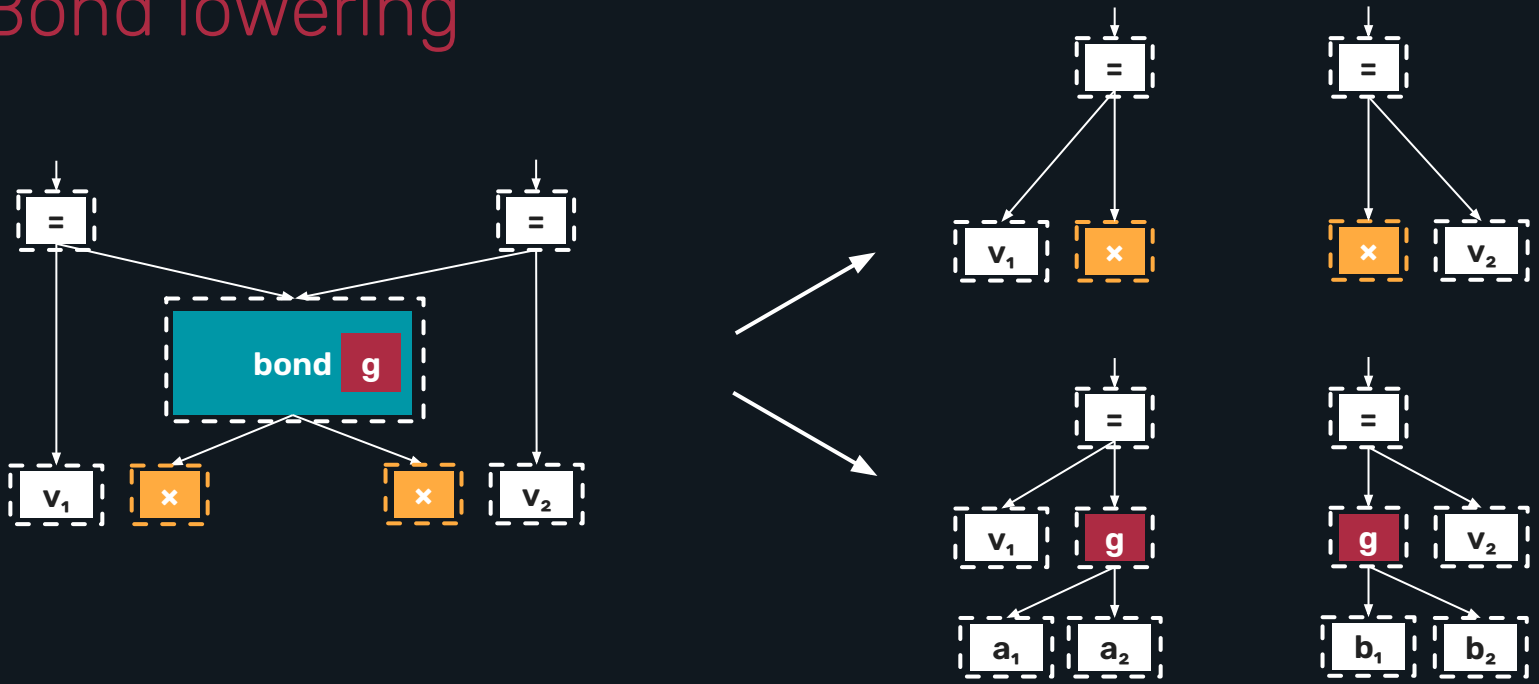
3. Relate common subexpressions

- Extend E-GRAPH by a new type of node – b-node (bond node)
- Represents relation between multiple e-classes
- Keeps relation between parent and children e-classes
- New rule action: (let E ($?_1 \times (?_2 \times 2)$)) (match E…) → (bond E…)

Bond lowering



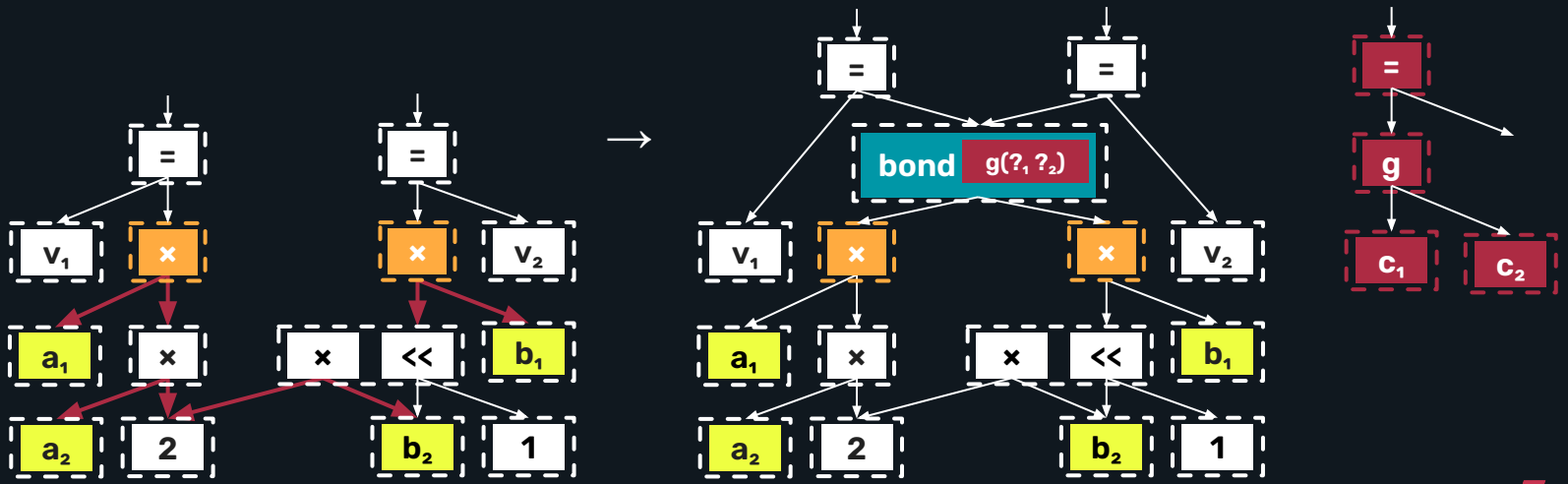
Bond lowering



The final bonding rule application

(let E ($?_1$ × ($?_2$ × 2))) (match E… with vars $?_1$ $?_2$) →

((let G (g(c_1, c_2) = c_1 × (c_2 × 2))) (bond E… with g($?_1$,$?_2$)))



Experiments

Benchmark	Circuit Size		AND Gates		Multiplications	
	UNOPTIMIZED	EQSAT	UNOPTIMIZED	EQSAT	UNOPTIMIZED	EQSAT
x86 mul-forms	94,286	**71,102**	42,591	**31,043**	25	**5**
3D toolkit	124,795	**97,881**	61,783	**49,590**	12	**2**
Router sim.	109,596	**86,585**	55,242	**40,757**	13	**2**
LAN simulator	126,657	**104,184**	63,430	**49,232**	19	**3**

Implementation

- Inspired by egg
- Soon to be released as C++ library:

  https://github.com/lifting-bits/eqsat
- Extends the language of patterns and e-graph modifiers
- Used as optimization ZK circuit compiler in the tool circuitous

Implementation

- Inspired by egg
- Soon to be released as C++ library:

  https://github.com/lifting-bits/eqsat
- Extends the language of patterns and e-graph modifiers
- Used as optimization ZK circuit compiler in the tool circuitous

Thank You