Abstract Interpretation on E-Graphs

Samuel Coward
Electrical and Electronic Engineering
Imperial College London
s.coward21@imperial.ac.uk

George A. Constantinides
Electrical and Electronic Engineering
Imperial College London
g.constantinides@imperial.ac.uk

Theo Drane
Intel Corporation
Folsom, USA
theo.drane@intel.com

1 Introduction

Recent e-graph applications have typically considered concrete semantics of expressions, where the notion of equivalence stems from concrete interpretation of expressions [10, 11]. However, equivalences that hold over one interpretation may not hold in an alternative interpretation. Such an observation can be exploited. We consider the application of abstract interpretation to e-graphs, and show that within an e-graph, the lattice meet operation associated with the abstract domain has a natural interpretation for an e-class, leading to improved precision in over-approximation. In this extended abstract, we use Interval Arithmetic (IA) [4, 9] to illustrate this point.

IA is commonly used to provide tight bounds on expressions or numerical program outputs. This is useful across numerical hardware/software design and verification, providing guarantees that exceptional behaviour is never encountered and enabling deeper optimizations. In IA, every numerical expression is associated with an (real or floating-point) interval rather than a single numerical value and for each operation \( f \) in the arithmetic, the natural interval extension operation is associated, where we assume that all infimums and supremums exist:

\[
f(X_1, \ldots, X_k) = \left[ \inf_{x_i \in X_i} f(x_1, \ldots, x_k), \sup_{x_i \in X_i} f(x_1, \ldots, x_k) \right]
\]

(1)

Let \([e]\) denote the natural interval extension of an expression, produced by structural induction on the expression syntax.

A key limitation of IA is the dependency problem, as illustrated through the following example. For \( x \in [0, 1] \) a standard interpretation gives:

\[
\left[ x - x \right] = [0, 1] - [0, 1] = [0 - 1, 1 - 0] = [-1, 1].
\]

(2)

Implementing IA using e-graphs, helps to mitigate the effect of the dependency problem as we shall see in §2.

Among existing tools using IA to obtain tight bounds [3, 6], Gappa [2], a tool for fixed and floating-point error analysis, is particularly relevant as it deploys a set of term rewrites in order to tackle the dependency problem.

2 Approach

We implement IA for real arithmetic expressions on top of the extensible egg library [12] as an e-class analysis. Following a standard approach, we represent real intervals by pairs of floating-point values, conservatively approximating real operations by rounding away from zero, a technique known as ‘outwardly rounded IA’ [5, 7], however our examples in this abstract are presented as real numbers for simplicity.

The key insight of our work is that expressions that are equivalent in the concrete interpretation, and hence can belong to the same e-class in an e-graph, may differ in their abstract interpretation. Despite this difference in abstract interpretation, the soundness of the two or more different abstract interpretations of concrete-equivalent expressions, implies that they may be combined via the meet operation associated with the abstract lattice [1], producing a more precise approximation.

A trivial example of this process derives from Eqn. 2. Consider the expression \( x - x \), together with the rewrite rule \( x - x \rightarrow 0 \). This implies that \( x \neq x \approx 0 \), where \( \approx \) denotes concrete-equivalence. In an e-graph, an e-class for this expression would contain two nodes, corresponding to the equivalent expressions. The interval interpretation of these two expressions is \([-1, 1]\) and \([0, 0]\), respectively, and as a result we may conclude that both expressions lie in \([-1, 1] \cap [0, 0]\), where intersection is the meet operation of the interval lattice.

To describe interval propagation throughout the e-graph, define \( C \) to be the set of e-classes, and \( \mathcal{N}_c \) the set of e-nodes contained in \( c \in C \). With each e-class, we associate a pair of floating-point values \((\underline{X}, \overline{X})\) to represent a real interval, which we denote \([c] = [\underline{X}, \overline{X}]\).

Similarly interpret a \( k \)-arity e-node \( n \) of function \( f \) with children classes \( c_{n,1}, \ldots, c_{n,k} \), as:

\[
[c] = f ([c_{n,1}], \ldots, [c_{n,k}])
\]

(3)

via the natural interval extension of the function of the e-node \( n \), as per Eqn. 1. A-arity e-nodes represent constants, associated with degenerate intervals containing a single value, or variables, accompanied by user specified intervals.

For acyclic e-graphs, it is trivial to propagate the known intervals upwards through the e-graph using Eqn. 3 together with the following novel tightening relationship, where meet ⋂ is intersection for intervals:

\[
[c] = \bigcap_{n \in \mathcal{N}_c} [n]
\]

(4)
We also note that this property allows for computation with bounds on the simple expressions presented in Table 1. On including two operator specific rewrites. It may be further possible to exploit cyclic e-graphs, which has two key advantages. Due to constructive rewrite rule inference [8] to explore the space of bound tightening rules on relational domains. We will apply rewrite application deciding which rewrites to apply and in which order is not a concern in the e-graph, useful if the route to tightly bounding expressions is non-obvious. There is also no constraint on the number of expressions that can provide relevant information in a given interpretation.

Further work will explore more complex problems, along with comparisons against existing tools such as Gappa [2], and results on relational domains. We will apply rewrite rule inference [8] to explore the space of bound tightening rules. It may be further possible to exploit cyclic e-graphs on abstract domains by interpreting the fixpoint equations as defining an iterative numerical method such as the Krawczyk method [7] which may then be extracted from the e-graph. Incorporating the technique into tools, such as Herbie [10], where bounds can be exploited, would demonstrate its value.

| Expression                | Initial | Improved | Width Change |
|---------------------------|---------|----------|--------------|
| $x^2 - 2x + 1$            | [-2.3]  | [0.1]    | -80%         |
| $x(2 - xy) - \frac{1}{y}$ | [-5, 1.5] | [-4.5, 0] | -31%         |
| $\sqrt{x + 1} - \sqrt{x}$ | [0, \sqrt{2}] | $\left[ \frac{1}{\sqrt{3 - \sqrt{2}}}, \frac{1}{1 + \sqrt{2}} \right]$ | -93%         |
| $\frac{x}{xy}$           | $\left[ \frac{1}{4}, 1 \right]$ | $\left[ \frac{3}{4}, \frac{3}{4} \right]$ | -33%         |

Table 1. Interval tightening via e-graphs, for $x, y \in [1, 2]$. Width change=(improved width - initial width)/initial width.
References

[1] Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: ‘A’ unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Conference Record of the Annual ACM Symposium on Principles of Programming Languages, Vol. Part F130756. https://doi.org/10.1145/512950.512973

[2] Marc Daumas and Guillaume Melquiond. 2010. Certification of bounds on expressions involving rounded operators. ACM Trans. Math. Software 37, 1 (2010). https://doi.org/10.1145/1644001.1644003

[3] Marc Daumas, Guillaume Melquiond, and César Muñoz. 2005. Guaranteed proofs using interval arithmetic. In Proceedings - Symposium on Computer Arithmetic. https://doi.org/10.1109/arith.2005.25

[4] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. 2002. Applied interval analysis. Choice Reviews Online 39, 06 (2002). https://doi.org/10.5860/choice.39-3424

[5] Ulrich Kulisch. 1981. Computer Arithmetic in Theory and Practice. https://doi.org/10.1016/c2013-0-11018-5

[6] Érik Martin-Dorel and Guillaume Melquiond. 2016. Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq. Journal of Automated Reasoning 57, 3 (2016). https://doi.org/10.1007/s10817-015-9350-4

[7] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. 2009. Introduction to Interval Analysis. https://doi.org/10.1137/1.9780898717716

[8] Chandrakana Nandi, Max Willsey, Amy Zhu, Yisu Remy Wang, Brett Saiki, Adam Anderson, Adriana Schulz, Dan Grossman, and Zachary Tatlock. 2021. Rewrite rule inference using equality saturation. Proceedings of the ACM on Programming Languages 5, OOPSLA (2021). https://doi.org/10.1145/3485496

[9] A. Neumaier. 1991. Interval Methods for Systems of Equations. https://doi.org/10.1017/cbo9780511526473

[10] Pavel Panchekha, Alex Sanchez-Stern, James R Wilcox, and Zachary Tatlock. 2015. Automatically improving accuracy for floating point expressions. ACM SIGPLAN Notices 50, 6 (2015), 1–11.

[11] Yisu Remy Wang, Shana Hutchison, Jonathan Leang, Bill Howe, and Dan Suciu. 2020. SPORES: Sum-Product Optimization via Relational Equality Saturation for Large Scale Linear Algebra. (2 2020). http://arxiv.org/abs/2002.07951

[12] Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. 2021. Egg: Fast and extensible equality saturation. Proceedings of the ACM on Programming Languages 5, POPL (2021), 1–29.