Logging an Egg:

Datalog on E-Graphs

Philip Zucker
Draper Laboratory
pzucker@draper.com

Abstract

There are many striking similarities between *equality saturation* [11] and *Datalog* [4]. This talk is about a prototype implementation called Egglog0[1] built upon the e-graph framework library egg [11] extended with *multipatterns* and the modeling power equality saturation gains with the addition of logic programming constructs.

Keywords: e-graphs, Datalog, theorem proving

1 Syntax and Semantics of Egglog0

Egglog0 is a prototype system built around the egg equality saturation library. It extends egg with an implementation of *multipatterns* and is designed bring to the forefront the analogy between equality saturation and datalog.

Similar to datalog, ground terms can be asserted as facts.

```
edge(a,b).
edge(b,c).
```

Operationally this inserts the ground term into the backing e-graph. To simplify the implementation, there is no distinction between the storage of predicates and terms. Hence, facts can also be used to initialize the e-graph with ground terms for equality saturation.

```
add(x, succ(succ(zero))).
```

Clauses are represented using prolog convention of capitalized names for variables.

```
path(X, Y), path(Y, X) :- edge(X, Y).
path(X, Y) :- path(X, Z), edge(Z, Y).
```

A term that appears on the right is a multipattern to seek in the e-graph. Multipatterns are a slight step away from the pure rewriting paradigm egg was initially conceived and have been upstreamed to egg. A ground term is produced and inserted into the e-graph by instantiating the head multipattern on the left.

The features so far have not used the e-graph in any intrinsic way. This subset of the language, "hashlog", could instead be implemented by a backing hash cons data structure and is available in a Datalog that supports algebraic datatypes.

Egglog0 also possesses a special built in relation _=_ backed by e-graph equality. Facts, clause heads, and clause bodies may use _=_. In the body, `p1 = p2` corresponds to a check that the patterns `p1` and `p2` were found in the same e-class. In a fact or in the head of a clause it correspond to calling `union` on the two e-classes of the instantiated terms.

Pattern variables are best not thought of as binding to terms, but instead to e-classes. Rewrite rules can be simply encoded in Horn clauses.

```
add(Y,X) = E :- add(X,Y) = E.
```

Because unidirectional and bidirectional rewriting is a common use case, there is syntax sugar for them.

```
add(Y,X) <- add(X,Y).
add(X,add(Y,Z)) <-> add(add(X,Y),Z).
```

Queries recover information out of the e-graph.

```
?- add(succ(zero),succ(Y)) = Z.
```

Queries are run after (possibly early) termination of the equality saturation process. They are implemented as multipatterns followed by extraction. A substitution dictionary of pattern variables to e-classes is returned from the multipattern. To print out a concrete ground term, a small representative term is extracted out of each variable's e-class in this dictionary.

2 Applications

Why bother with this extra expressivity? In this section we demonstrate a number of applications that are not easily expressible as equality saturation or Datalog on their own. With the upstreaming of multipatterns these examples are expressible in mainline egg, albeit with a different syntax.

2.1 Injectivity

One simple application is being able to express the axiom of injectivity of a constructor. This axiom is in some respects the dual of the congruence axiom which is applied automatically in the e-graph.

```
X = Y, Xs = Ys :- cons(X,Xs) = cons(Y,Ys).
```

[1] https://github.com/philzook58/egglog

PL'18, January 01–03, 2018, New York, NY, USA
2018.

2.2 Equation Solving Rules

A common manipulation in algebraic reasoning is to manipulate equations by applying the same operation to both sides. This is often used for the purposes of variable isolation and then substitution to achieve variable elimination. The e-graph by its very nature immediately understands the substitution, but we must show it how to manipulate equations. As an example, the following axiom isolates the Y variable using the known equation on the right.

```
sub(Z,X) = Y :- add(X,Y) = Z
```

It is possible to extract from the e-graph a term containing the least number of undesirable variables that are in the same e-class as a term `x`, although Egglog0 does not currently implement this extraction.

2.3 Reflection and Equational Logic

Like Datalog, the presence of predicate "terms" in the e-graph database can be considered as an assertion of their truth. An alternative convention is to model predicates as functions into `Bool`, allowing one to manipulate and talk about predicates in the absence of knowledge of their truth. Instead, a predicate term being in the same e-class as the term **`true`** can be seen as evidence of its truth. More interestingly, the lack of a predicate being in the e-graph cannot be seen as evidence of it's falsehood, whereas being in the same e-class as `false` can be seen as evidence that the predicate has been determined to be definitely false. Equality saturation as a reasoning tool has constructive character in this sense. This is related to the situation in Datalog where negation may only be considered in the presence of a stratification condition.

This encoding allows for ordinary boolean algebraic manipulation of logical formula. One can also reflect the notion of e-graph equality itself into a term `eq(_,_)`.

```
A = B :- true = eq(A,B).
true = eq(A,B) :- A = B.
```

There exist equational formulations of logic with an internalized algebraic notion of equality for which it is suggested Egglog0 is suited [5, 6].

2.4 Uniqueness Quantification

Equality generating dependencies and tuple generating dependencies are two constraints from the theory of relational databases [1]. Equality generating dependencies can be directly modeled using native e-graph _=_ in the head of clauses. Tuple generating dependencies can be modeled using Skolemization. An axiom of the form

$$\forall x_1, \dots, x_n, \phi(x_1, \dots, x_n) \implies \exists y_1, \dots, y_m, \psi(x_1, \dots, x_n, y_1, \dots, y_m) \tag{1}$$

can be converted into an axiom where $y_1 \dots y_m$ are n-arity globally fresh function symbols.

```
psi(X1,...,Xn,y1(X1,...,Xn),...,ym(X1,...,Xn)) :-
    phi(X1, ... ,Xn)
```

Combining these two abilities allows one to express uniqueness quantification ∃!, which one finds for example in the definition of universal properties in category theory. Diagram chasing is considered to be laborious and yet fairly automatic activity in category theory. A large part of it consists of reasoning about commuting paths (equalities), and finding places to instantiate universal properties[10]. This search and instantiation can be encoded in Egglog0 clauses [14, 15].

2.5 Generalized Algebraic Theories

Algebraic theories are those described by equational axioms, for example the theory of groups. Generalized algebraic theories (GATs) extend these by introducing a limited version of dependent types [3, 9]. The types also have an equational theory and the equational axioms only hold between terms of the same type, therefore there may be typing obligations to discharge before one can use an equality. The typing requirements of GAT axioms naturally map to Horn clauses modulo equality. It is also possible to build an inefficient equational encoding using type tagging[2] or twist the formulation to use guards rather than multipatterns [17].

3 Related Work

This work draws on the ideas of Relational E-matching [13]

Egg-lite is an implementation of related functionality built on top of sql-lite [12]. It is the dual of this work, building on a system targeted to database queries by extending them with e-graph functionality.

Souffle[7] is a Datalog engine supporting algebraic datatypes, equivalence relations, and provenance production. These features make it possible to emulate Egglog0 [16], albeit seemingly at a performance loss for e-graph intensive problems [18].

4 Further Work

The most tantalizing of all are the places where we don't quite understand the analogs of Datalog techniques in equality saturation. Two prominent examples are the magic set transformation, which makes Datalog queries prune their bottom up search using the query, and semi-naive evaluation, which avoids a great deal of unnecessary recomputation [1]. In a different direction, one may also wish to extend the expressivity of the system as a theorem prover towards hereditary Harrop formula [8]. Finally, a important question is how to naturally extend the syntax, semantics, and implementation to internalize and express lattice analyses and extraction [11].

Acknowledgments

The author would like to thank Yihong Zhang, Remy Yisu Wang, Max Willsey, Zachary Tatlock, Alessandro Cheli, Cody Roux, James Fairbanks, and Evan Patterson for their helpful discussions.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases: The Logical Level* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[2] Jasmin Christian Blanchette, Sascha Böhme, Andrei Popescu, and Nicholas Smallbone. 2013. Encoding Monomorphic and Polymorphic Types. In *Tools and Algorithms for the Construction and Analysis of Systems*, Nir Piterman and Scott A. Smolka (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 493–507.

[3] John Cartmell. 1986. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic* 32 (1986), 209–243. https://doi.org/10.1016/0168-0072(86)90053-9

[4] S. Ceri, G. Gottlob, and L. Tanca. 1989. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering* 1, 1 (1989), 146–166. https://doi.org/10.1109/69.43410

[5] Edsger W. Dijkstra and Carel S. Schotten. 1990. *Predicate Calculus and Program Semantics.* Springer New York, New York, NY. https://doi.org/10.1007/978-1-4612-3228-5_1

[6] David Gries and Fred B. Schneider. 1993. *A Logical Approach to Discrete Math.* Springer New York, New York, NY. https://doi.org/10.1007/978-1-4757-3837-7_1

[7] Herbert Jordan, Bernhard Scholz, and Pavle Subotic. 2016. Soufflé: On Synthesis of Program Analyzers. In *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9780)*, Swarat Chaudhuri and Azadeh Farzan (Eds.). Springer, 422–430. https://doi.org/10.1007/978-3-319-41540-6_23

[8] Dale Miller and Gopalan Nadathur. 2012. *Programming with Higher-Order Logic.* Cambridge University Press. https://doi.org/10.1017/CBO9781139021326

[9] Evan Patterson, James Fairbanks, Andrew Baas, Kristopher Brown, Micah E Halter, Sophie Libkind, and Owen Lynch. 2022. Catlab.jl: A framework for applied category theory. https://doi.org/10.17605/OSF.IO/HMNFE

[10] Cody Roux. 2015. The Categorical Nearly Automatic Property Prover. https://github.com/codyroux/catnapp

[11] Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. 2021. Egg: Fast and Extensible Equality Saturation. *Proc. ACM Program. Lang.* 5, POPL, Article 23 (jan 2021), 29 pages. https://doi.org/10.1145/3434304

[12] Yihong Zhang. 2022. Your next e-graph framework looks like a Datalog. 6, PLDI (2022).

[13] Yihong Zhang, Yisu Remy Wang, Max Willsey, and Zachary Tatlock. 2022. Relational E-Matching. *Proc. ACM Program. Lang.* 6, POPL, Article 35 (jan 2022), 22 pages. https://doi.org/10.1145/3498696

[14] Philip Zucker. 2021. Egglog 2: Automatically Proving the Pullback of a Monic is Monic. https://www.philipzucker.com/egglog2-monic/

[15] Philip Zucker. 2021. Egglog Examples: Pullbacks, SKI, Lists, and Arithmetic. https://www.philipzucker.com/egglog-3/

[16] Philip Zucker. 2021. Naive E-graph Rewriting in Souffle Datalog. https://www.philipzucker.com/datalog-egraph-deux/

[17] Philip Zucker. 2021. Rewriting Monoidal Categories in the Browser with Egg. https://www.philipzucker.com/rust-category/

[18] Philip Zucker. 2022. A Questionable Idea: Hacking findParent into Souffle with User Defined Functors. https://www.philipzucker.com/souffle-functor-hack/