Datalog Implementation in JastAdd

Hampus Balldin
C13, Lund University, Sweden
dat12hba@student.lu.se

Abstract

Text of abstract ....

1 Introduction

Datalog is a syntactically simple declarative language that enables expression and evaluation of certain first-order logic propositions. From its' inception in the nineteen-eighties it received substantial interest from the academic community into the early nineteen-nineties[3]. The efforts primary drive were to enable knowledge based systems that allowed the generation of new facts based on rules stated using the logic programming paradigm. At the time, this had applications in both artificial intelligence as well as a complement to the traditional relational database querying systems such as SQL[2][1].

After a time of cooling interest, Datalog has emerged again as an attractive way to express complex inter-dependencies[3]. A notable example is from Program Analysis where systems such as Doop [4] make heavy use of Datalog to derive e.g. call-graph and points-to information, both of which typically have mutually recursive dependencies in languages using dynamic dispatch.

There are currently many (CITE) Datalog implementations.

1.1 Core Language

There are many flavors of the Datalog language but they all build on and possibly extend a common core.

A *program* P consists of a set of *Horn clauses* $H_1, \ldots H_n$. A horn clause has a *head* and a *body*. The head is a single *literal* and the body is a sequence of literals. A literal is identified by a *predicate symbol* and a sequence of *terms*. An example of a propositional rule (without terms) is shown below:

$$A :- B_1, B_2, \ldots B_m$$

Above we have a single horn clause (hereafter called a *rule*). It has head A body $B_1, B_2, \ldots B_m$. The semantics of the above rule is that if the conjunction of all the literals in the body hold in a given *model* then we conclude that A holds in that model.

A term is either *variable* or *constant*. A model is a complete and consistent instatiation of all variables with constant values where each constant is drawn from the *domain* of the model. Consider the following first-order example:

$$A(x, y, "C") :- B_1(x, "C"), B_2("C", y), B_3(x, y)$$

Assume that we have model domain Ω. Then the possible instantiations are $(x, y) \in \Omega \times \Omega$. A predicate that contain only constant terms is called *ground*. A ground predicate is true if it can be derived by the set of rules in the program, or if it is declared as being axiomatically true. We call a true ground predicate a *fact*. A fact $A(C_1, \ldots C_k)$ can be interpreted as the k-tuple $(C_1, \ldots, C_k)$ belonging to the relation A of *arity* k.

The most general goal then is to, for a given predicate symbol A, find all tuples that belong to A in any possible model given the Datalog rules and the model domain of objects. Consider again the first-order rule above. In the rule we have variables x, y and constant "C". The semantics of the rule is shown below with model domain Ω.

$$\frac{\forall c_1 \in \Omega.\ \forall c_2 \in \Omega.\ \ (c_1, "C") \in B_1, ("C", c_2) \in B_2, (c_1, c_2) \in B_3}{(c_1, c_2, "C") \in A}$$

1.2 Security Features and Time Complexity

1.3 Query Evaluation

1.4 Goal

A common front-end that allows cross-compilation into different Datalog implementations. A common front-end enables a convenient way to compare the performance of different Datalog evaluation methods.

1.5 JastAdd

JastAdd (CITE) is a meta-compilation system that enables the expression of arbitrary graphs on top of an abstract syntax tree (AST). Information is propagated in the AST through the use of so called Reference Attribute Grammars. JastAdd also supports aspects (CITE) which allow the weaving of methods and class fields from different source locations into a single generated class. This allows easy extension of the generated AST classes with additional properties.

2 Content

Replace the header of this section with something appropriate for your paper. Use around 1-4 sections for the content.

Acknowledgments

Text of acknowledgments ....

Course paper, EDAN70, Lund University, Sweden
November 14, 2018.

References

[1] Francois Bancilhon and Raghu Ramakrishnan. 1986. An Amateur's Introduction to Recursive Query Processing Strategies. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data (SIGMOD '86)*. ACM, New York, NY, USA, 16–52. https://doi.org/10.1145/16894.16859

[2] S. Ceri, G. Gottlob, and L. Tanca. 1989. What You Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Trans. on Knowl. and Data Eng.* 1, 1 (March 1989), 146–166. https://doi.org/10.1109/69.43410

[3] Todd J. Green, Shan Shan Huang, Boon Thau Loo, and Wenchao Zhou. 2013. Datalog and Recursive Query Processing. *Found. Trends databases* 5, 2 (Nov. 2013), 105–195. https://doi.org/10.1561/1900000017

[4] Yannis Smaragdakis and Martin Bravenboer. 2011. Using Datalog for Fast and Easy Program Analysis. In *Proceedings of the First International Conference on Datalog Reloaded (Datalog'10)*. Springer-Verlag, Berlin, Heidelberg, 245–251. https://doi.org/10.1007/978-3-642-24206-9_14