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 frameworks such as Doop [4] make 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 and body $B_1, B_2, \ldots B_m$. The intuitive meaning of the above rule is that if the conjunction of all the literals in the body are true then we conclude A.

Datalog deals not only with propositional rules, but allows a restricted range of first-order propositions where each literal is associated with a sequence of terms. A term is either *variable* or *constant*. A literal that contain only constant terms is called a *ground literal*. We further partition the predicates into extensional (EDB) and intensional (IDB). The EDB's are all predicates that are taken as input from an external database. The IDB's are the predicates that are not EDB and are intensionally defined through rules. The EDB's introduce *facts*, i.e. ground literals. One may additionally introduce facts by declaring a rule with a ground literal head that has an empty body.

- The set of all constants in all facts is called the *domain* and is denoted Ω.
- The set of all facts is called the *active database instance* and is denoted I.

There are three main semantic interpretations of Datalog: model-, fixpoint-, and proof-theoretic semantics. [3]. A brief overview is given below.

Model-theoretic Semantics

A *model* of a Datalog program P is a consistent (satisfying all rules of P) extension of the initial EDB facts. Each rule is interpreted as a universally quantified rule. For example, below is given a rule and its' corresponding semantic interpretation. In the example $B(c_1, "C")$ is written as $(c_1, "C") \in B$ to emphasize the practical correspondence between predicates and relations.

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

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

An inference algorithm attempts to find the *minimal model*, i.e. a model m of P such that for any other model m' of P, all facts of m are in m'. In practice this means that an inference algorithm should only add a fact if it is required by the semantics of a rule (even if adding the fact may lead to an extended model of P).

Fixpoint-theoretic Semantics

Begin with the set of all facts in the active database instance I^0. The set of new facts that can be derived (under model-theoretic semantics) using the rules of a program P and the existing facts in I^i is denoted Δ_i. We get the following inductive definition of I:

$$I^0 = \{\text{EDB Facts in } P\}$$
$$I^{i+1} = I^i \cup \Delta_i$$

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

It can be shown(CITE) that the minimal model is computed as I^n for n such that $I^n = I^{n+1}$. Since $I^i \subseteq I^{i+1}$ (monotonically increasing) and with the practical assumption of a finite domain, the fix-point algorithm is guaranteed to terminate.

Proof-theoretic Semantics

Consider a ground literal $A(C_1, \ldots, C_n)$. A query for the ground literal asks for a proof that $A(C_1, \ldots, C_n)$ is in the minimal model of a program P. A proof can be visualized as an *and-or tree* T. T has the proposition (ground literal) to prove as the root. At an OR-node, all possible rules are tested. If any of them succeed then the proposition has been proven. At an AND-node, all the children proposition need to be proven for the node to become true. An example is shown in figure X.

$$r_1 : A :- B, C$$
$$r_2 : A :- B, D$$

When traversing the tree the model is updated with new

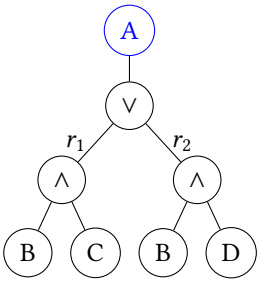

Figure 1. An and-or tree for rules $r_1 : A :- B, C$, $r_2 : A :- B, D$

facts that are needed to prove the root proposition. Those facts then become part of the extended model.

1.2 Security Features and Time Complexity

Above a fact was stated to be a ground literal that is true in a given model. An axiomatic fact can be declared as a rule with no body: $A(t_1, \ldots, t_n)$. Datalog disallows axiomatic facts that contain variables. This is implied by the following more general rule: all variables occurring in the head of a rule must also occur in the body of the rule. This is called the *range restriction property* [3].

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 ....

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