Efficient and Extensible Datalog
Aaron Bembenek\textsuperscript{1}, Stephen Chong\textsuperscript{1}, and Marco Gaboardi\textsuperscript{2}
\textsuperscript{1} Harvard University \textsuperscript{2} University of Dundee

Motivation and Initial Objectives
We were primarily motivated by the lack of a good, open-source, Java-based implementation of the logic programming language Datalog. We wanted to have such an implementation as the backend for other projects within Privacy Tools.

Accordingly, our primary goal for this project was to create a Java-based Datalog engine. Ideally, the engine would also:

1. Take advantage of multiple cores during query evaluation
2. Be easily extensible with new evaluation strategies or even new language features
3. Provide a clean (and easily extensible) Java interface so that it could efficiently be integrated within other JVM-based applications

As secondary goals, we also hoped to:

1. Create pedagogical material to accompany our implementation, such as a tutorial
2. Implement new language features, such as types or limited function symbols

Current Status
We have built an engine that seamlessly supports over half-a-dozen query evaluation strategies, including:

- Bottom-up techniques that derive all facts
- Top-down techniques that derive only the facts necessary to answer a particular query
- Program rewriting techniques that transform programs in response to a query so that bottom-up evaluation is as efficient as top-down evaluation

We have experimented with different concurrent evaluation techniques to improve scalability, both by adapting single-threaded algorithms and creating new algorithms more suitable for concurrent evaluation.

We have also implemented a language extension that allows for hypothetical reasoning known as DatalogHypothetical.

Finally, we have created a simple graphical user interface, which is suitable for use in an undergraduate logic programming language course.

- Developer can load, edit and save Datalog source code
- Datalog programs can be dynamically loaded and queried with a built-in interpreter to give the developer real-time feedback

Datalog Overview [1,3,6]

Grammar:

\begin{verbatim}
Grammar: PLFor plPredSym plAtom plConst plClause
\end{verbatim}

DatalogHypothetical [2,8]

Basics:

- Extends Datalog with the ability to reason hypothetically, i.e., what can we derive if we assume a given fact is in the input database?
- PSPACE-complete, so much more powerful than Datalog (P-complete) and not as powerful as Prolog (Turing-complete)

Grammar:

\begin{verbatim}
DatalogHypothetical extends the grammar of Datalog as follows:
\end{verbatim}

Example: Graph Transitive Closure

Suppose we want to compute graph transitive closure. We can represent a particular query so that bottom-up evaluation is as efficient as top-down evaluation

Example: Graduation Eligibility

Say we have a database of students, courses, and the courses students have taken. Using Datalog, we can compute facts about graduation eligibility. For instance, maybe a student is eligible to graduate after taking CS 152 and STAT 110

\texttt{grade(S)<student(S),taken(S,CS152),taken(S,STAT110).}

But what if we want to reason about who is one (or two) courses away from graduation?

DatalogHypothetical! to the rescue!

Results:

- User interface mirrors that of Prolog
- Results are shown in milliseconds

Potential Applications in Privacy Tools

Modeling Privacy Legislation for DataTags:

- DataTags broadly state permitted and disallowed actions on datasets, always giving “safe” answers, even when a particular action might be allowed on a particular dataset
- We hope to encode relevant portions of privacy legislation in a logic program to allow a fine-grained analysis
- Having our own implementation of Datalog will allow us to create new language features as necessary
- In particular, our DatalogHypothetical implementation might be helpful, as hypotheticals were found to be useful in formalizing the British Nationality Act [4]

Declarative Encoding of DataTags Questionnaires:

- Questionnaires currently encoded in a procedural manner
- Datalog has the potential to encode questionnaires declaratively, which would allow for easier extensibility and maintenance
- Open questions include how to extend Datalog with dynamic user interaction, and how to determine the best order to ask questions
- Having a home-built Datalog implementation will allow us to more easily explore these issues

References