Logic Programming and Databases

pukkamustard

BOB 2021 – February 26, 2021

Datalog

or: How I Learned to Stop Worrying and Build My Own Database

Variety of Data

- Relational data
- Graph
- Text
- Geospatial data

Variety of Data

- Relational data
- Graph
- Text
- Geospatial data

Not all data fits into one kind of a database.

Multi-Model Application

```
friends = knows(alice, depth=2)

friends_in_vicinity =
  filter_in_vicinity(friends, alice)

comments = get_comments(friends)

comments_about_logic =
  comments_full_text_search(comments, 'Logic')
```

Multi-Model Applications

- Combine data from different databases
- Application unifies views of databases

Multi-Model Applications

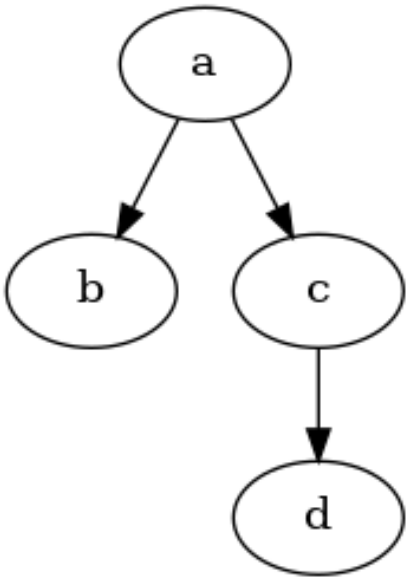
- Combine data from different databases
- Application unifies views of databases

Not entirely satisfactory.

- Complexity
- Performance
- Ergonomics

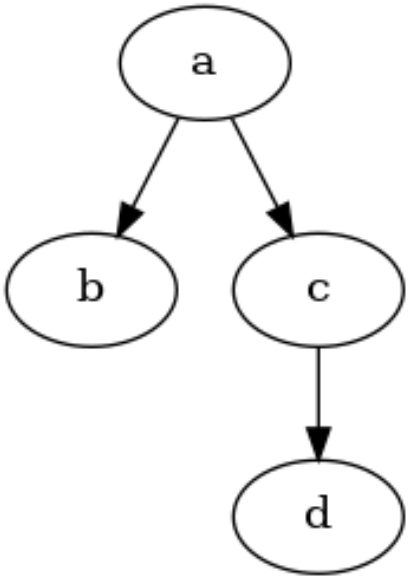
Outline

Graph




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

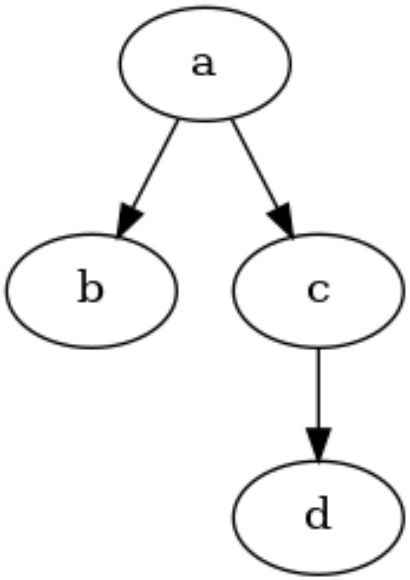
Graph




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

```
edge(a,b)?
true.
```

Graph

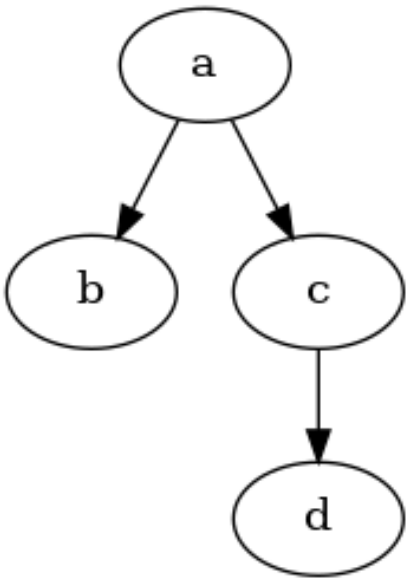



```
edge(a,b).
edge(a,c).
edge(c,d).
```

```
edge(a,b)?
true.
```

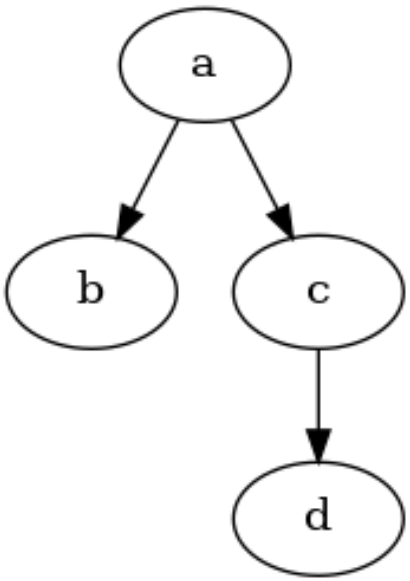
```
edge(a,d)?
false.
```

Graph




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

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

Graph




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

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


path(a,d)?
true.
```

Syntax

Term Constant value (a) or variable (X)

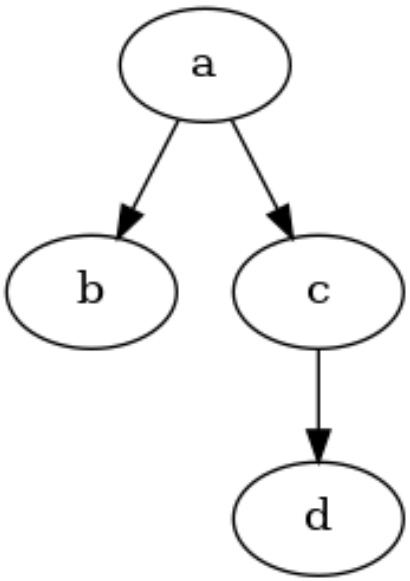
Literal Literal Predicate symbol and argument terms
$edge(a, b)$ or $edge(X, Y)$

Fact A literal
$edge(a, b)$.

Rule A head literal followed by body of literals
$path(X, Z) :- edge(X, Y), path(Y, Z)$.

Goal A literal
$path(a, d)$?

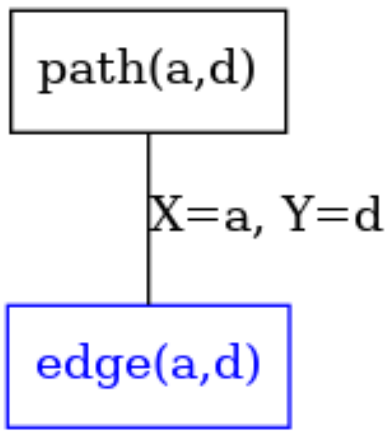
Syntax




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

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

path(a,d)?
```

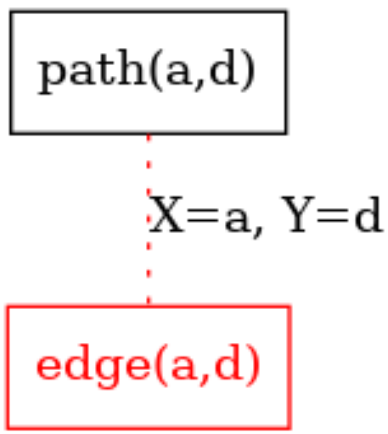
Resolution

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

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

path(a,d)?
```



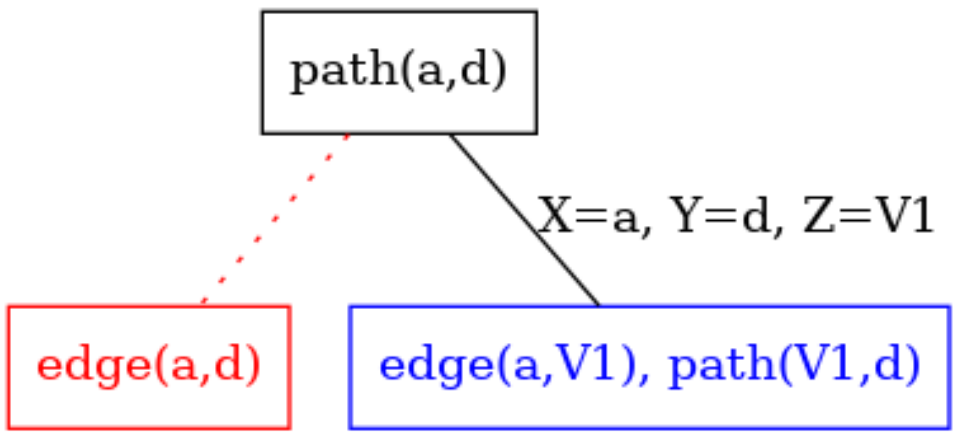
Resolution

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

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

path(a,d)?
```



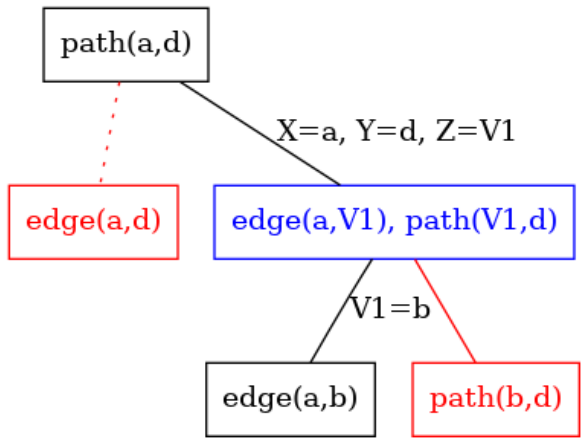
Resolution

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

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

path(a,d)?
```



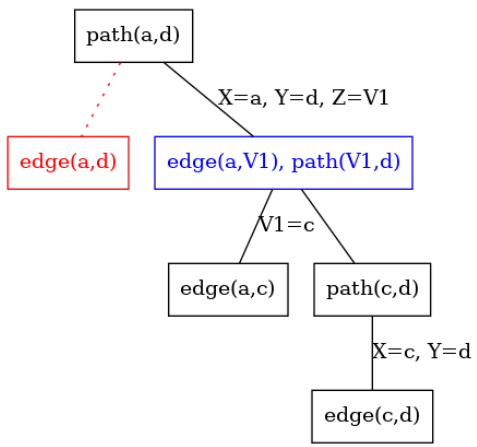
Resolution

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

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

path(a,d)?
```

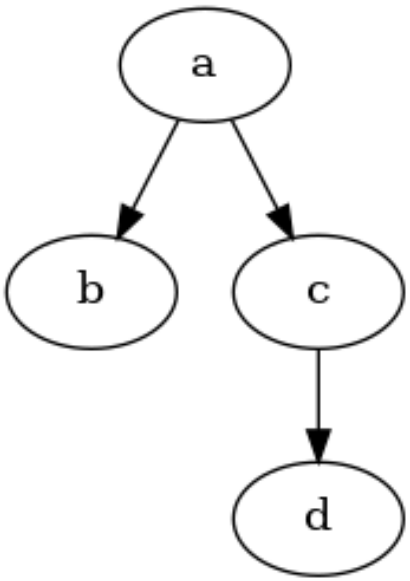


Resolution

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

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

path(a,d)?
true.
```

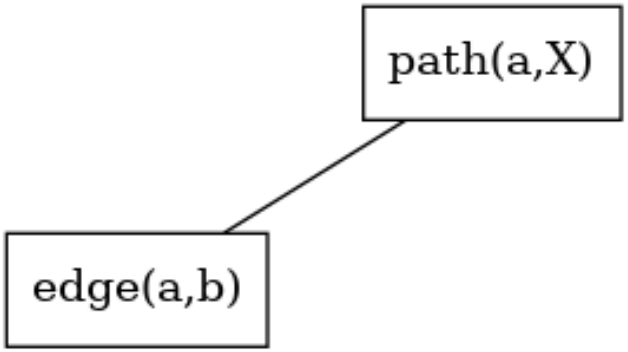


Multiple answers (Prolog)

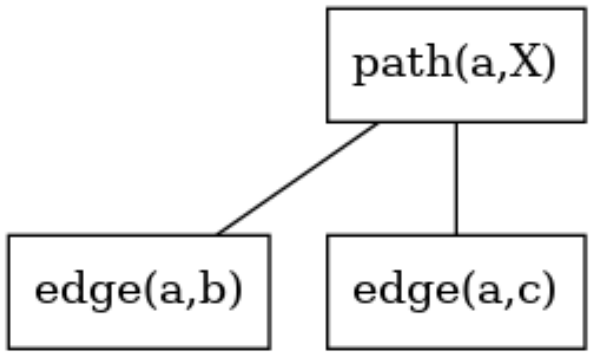



```
edge(a,b).
edge(a,c).
edge(c,d).

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

path(a,X)?
X = b ;
X = c ;
X = d ;
false.
```

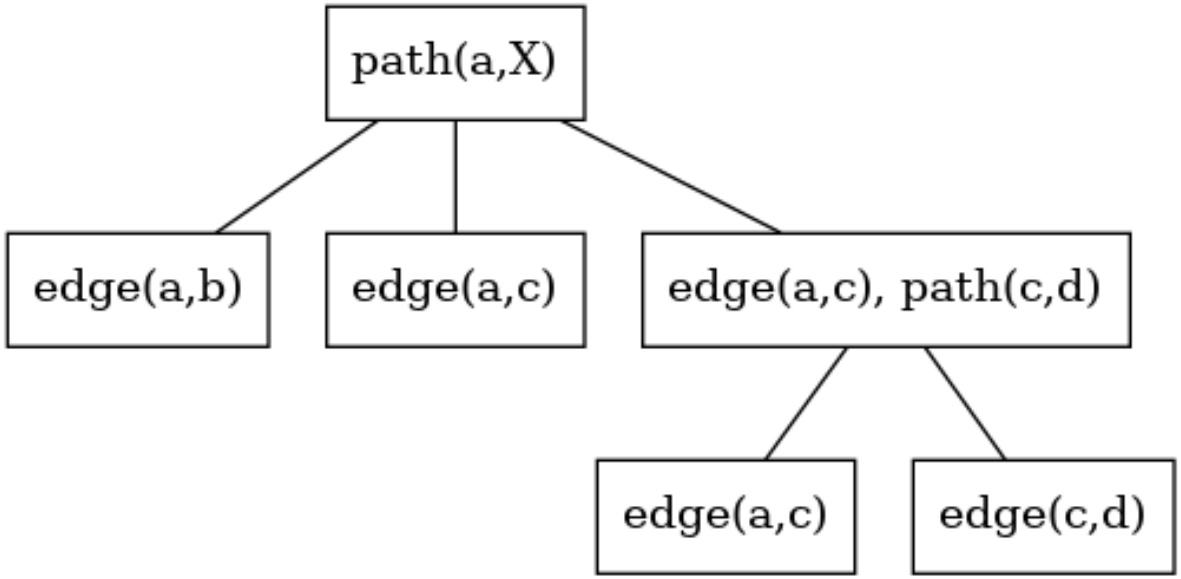
Multiple answers



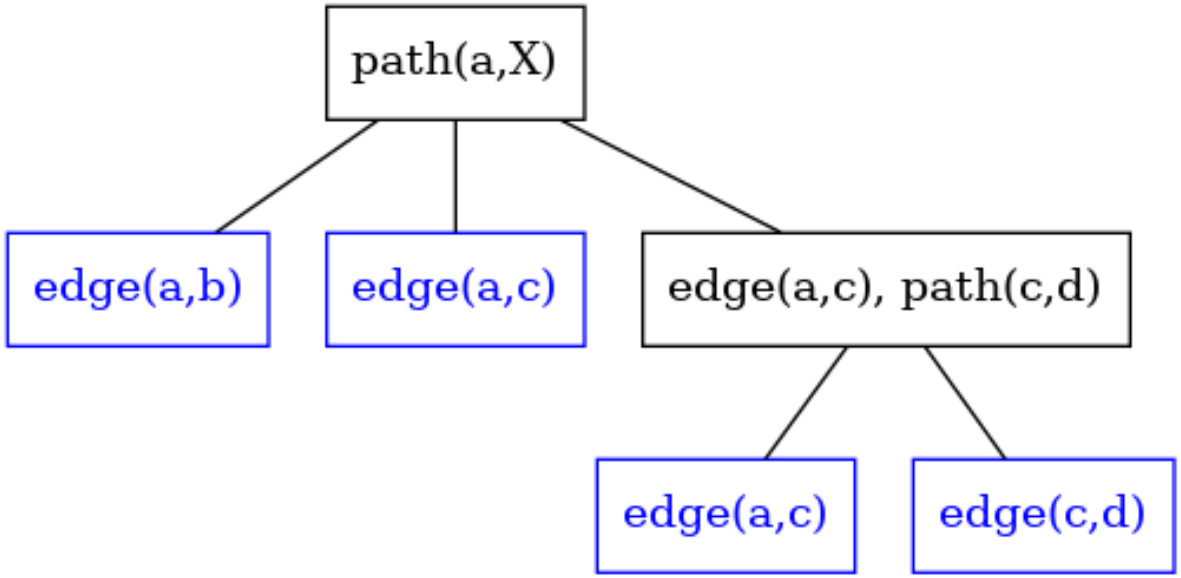
Multiple answers



Multiple answers



Facts from Database

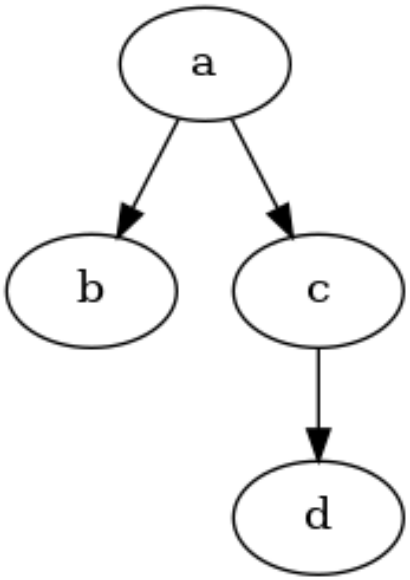


Not very efficient!

Relational Algebra

```
comment(alice, comment1).
comment(bob, comment2).
comment(alice, comment3).

likes(bob, comment1).
likes(charlie, comment2).
likes(charlie, comment1).

likes_author(Who, Author) :-
    likes(Who, Comment), comment(Author, Comment).
```

Relations




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

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

path(a,X)?
path(X,d)?
```

Order of Rules

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

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

path(a,d)?
```

Order of Rules

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

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

path(a,d)?
```

Infinite recursion!

General Logic Programming

- Simple
- Expressive
- Turing complete

General Logic Programming

- Simple
- Expressive
- Turing complete
- Termination not guaranteed
- Not efficient for querying databases
- Not declarative

General Logic Programming

- Simple
- Expressive
- Turing complete
- Termination not guaranteed
- Not efficient for querying databases
- Not declarative

Enter Datalog.

Syntax

Term Constant value (a) or variable (X)

Literal Literal Predicate symbol and argument terms
$edge(a, b)$ or $edge(X, Y)$

Fact A literal
$edge(a, b)$.

Rule A head literal followed by body of literals
$path(X, Z) :- edge(X, Y), path(Y, Z)$.

Goal A literal
$path(a, d)$?

Safety

Every fact only contains constants (ground)

`edge(a, b)`

`edge(a, X)`

Safety

Every fact only contains constants (ground)

```
edge(a,b)
```

```
edge(a,X)
```

Every variable appearing in the head of a rule must appear in the body.

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

```
p(X,Y) :- s(X), t(X)
```

Safety

Every fact only contains constants (ground)

```
edge(a,b)
```

```
edge(a,X)
```

Every variable appearing in the head of a rule must appear in the body.

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

```
p(X,Y) :- s(X), t(X)
```

Guarantees that solution is finite.

Database and Program

Extensional Database (EDB)

Ground Facts in a Database

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

Intensional Database (IDB)

Relations defined in the query/program

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

Datalog defines the mapping from EDB to IDB.

Logical Semantics

Rules

$$L_0 \text{:-} L_1, \ldots L_n$$

$$\forall X_1 \ldots X_m (L_1 \wedge \cdots \wedge L_n \implies L_0)$$

Program

A Datalog Program P is a set of formulas.

Interpretation

Set of ground literals using constants and predicate symbols appearing in EDB and IDB.

$$
\begin{aligned}
I_0 =& \{edge(a,b), edge(a,c), edge(c,d), \\
& path(a,b), path(a,c)\} \\
I_1 =& \{edge(a,b), edge(a,c), edge(c,d), \\
& path(a,b), path(a,c), path(c,d), path(a,d)\}
\end{aligned}
$$

Interpretation

Set of ground literals using constants and predicate symbols appearing in EDB and IDB.

$$
\begin{aligned}
I_0 =& \{edge(a,b), edge(a,c), edge(c,d), \\
& path(a,b), path(a,c)\} \\
I_1 =& \{edge(a,b), edge(a,c), edge(c,d), \\
& path(a,b), path(a,c), path(c,d), path(a,d)\}
\end{aligned}
$$

$$\forall X \forall Y \, (edge(X,Y) \implies path(X,Y))$$

$$\forall X \forall Y \forall Z \, (edge(X,Z) \wedge path(Z,Y) \implies path(X,Y))$$

$$I_0 \not\models P, I_1 \models P$$

Models

$$
\begin{aligned}
I_2 = \{ & edge(a, b), edge(a, c), edge(c, d), \\
& path(a, b), path(a, c), path(c, d), path(a, d) \\
& path(a, a), path(b, c)\}
\end{aligned}
$$

$$I_2 \models P$$

Models

$$I_2 = \{edge(a, b), edge(a, c), edge(c, d),$$
$$path(a, b), path(a, c), path(c, d), path(a, d)$$
$$path(a, a), path(b, c)\}$$

$$I_2 \models P$$

The intersection of two models is a model.

Models

$$\begin{aligned} I_2 = \{ & edge(a, b), edge(a, c), edge(c, d), \\ & path(a, b), path(a, c), path(c, d), path(a, d) \\ & path(a, a), path(b, c)\} \end{aligned}$$

$$I_2 \models P$$

The intersection of two models is a model.

The intersection of all models is unique.

Model-theoretic Semantics

The solution of the Datalog program P on input EDB E is the *minimal model* of P that contains E.

Model-theoretic Semantics

The solution of the Datalog program P on input EDB E is the *minimal model* of P that contains E.

- Proof-theoretic Semantics
- Fixed-point Semantics

Datalog and Relational Algebra

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

$$PATH \supseteq EDGE$$

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

$$PATH \supseteq \Pi_{X,Y}(EDGE \bowtie PATH)$$

Datalog and Relational Algebra

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

$$PATH \supseteq EDGE$$

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

$$PATH \supseteq \Pi_{X,Y}(EDGE \bowtie PATH)$$

$$PATH = EDGE \cup \Pi_{X,Y}(EDGE \bowtie PATH)$$

Evaluation

System of Relational Equations

$$I_0 = \text{RelationalExpression}_0(I_0, \ldots, I_n, E_0 \ldots E_m)$$
$$\vdots$$
$$I_n = \text{RelationalExpression}_n(I_0, \ldots, I_n, E_0 \ldots E_m)$$

IDB relations I_i and EDB relations E_j.

Iteratively solve system of equations (naive and semi-naive evaluation).

Goals

`path(a,d)?`

Bottom-up

Unnecessarily computes the entire relation for *path*.

Goals

`path(a,d)?`

Bottom-up

Unnecessarily computes the entire relation for *path*.

Top-down

Compute only relevant facts (Query-SubQuery and Magic Sets).

Working with Relations

Memory

Weight-balanced binary trees

Persistent

Ordered Key-Value Stores (e.g. LMDB) [1]

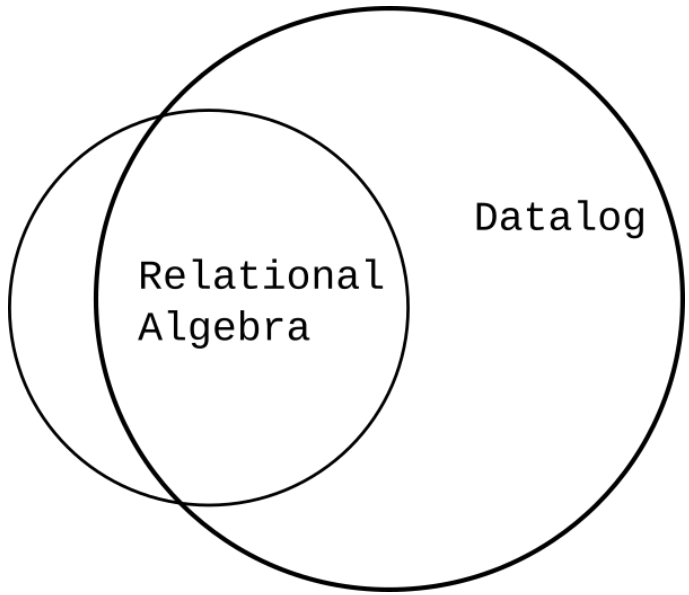
[1] See also SRFI-167 and SRFI-168

Recipe for Deduction

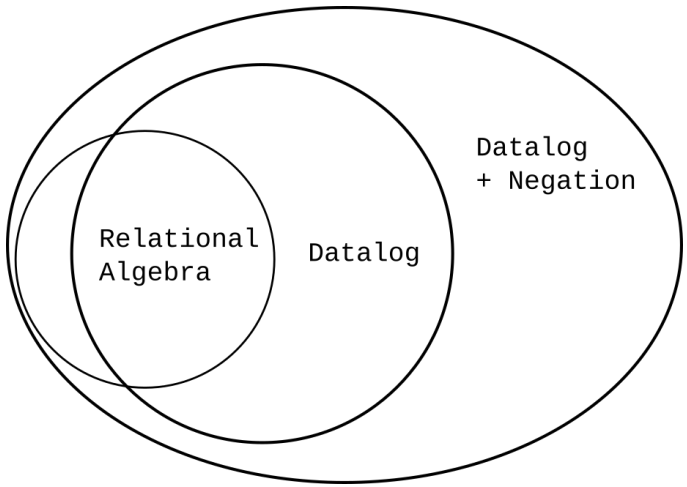
- High-performance Ordered Key-Value Store
- In-memory data structure for tuples
- Your favorite Datalog Evaluation algorithm
- A dash of Relational Algebra

Mix together and query with Datalog.

Datalog and Relational Algebra



Datalog and Relational Algebra



- Stratified Negation
- Monotonic Aggregation

Multi-Model Database

```
extended_friends(X,Y) :- friends(X,Y).
extended_friends(X,Y) :-
  friends(X,Z),
  friends(Z,Y).

talking_about_logic(X) :-

  extended_friends(alice, Friend),

  in_vicinity(alice,Friend),

  comment(Friend, Comment),

  full_text_search(Comment, 'Logic').
```

Logic Programming and Databases: Datalog

- Simple
- Expressive
- Efficient
- Declarative
- Guaranteed to terminate

Logic Programming and Databases: Datalog

- Simple
- Expressive
- Efficient
- Declarative
- Guaranteed to terminate
- Not Turing complete

Logic Programming and Databases: Datalog

- Simple
- Expressive
- Efficient
- Declarative
- Guaranteed to terminate
- Not Turing complete

or: How I Learned to Stop Worrying and Build My Own Database

Datalog for Big Data

- Yedalog: Exploring Knowledge at Scale[2]
- Distributed SociaLite: A Datalog-Based Language for Large-Scale Graph Analysis[3]

[2]Chin, Brian, et al. "Yedalog: Exploring knowledge at scale." 1st Summit on Advances in Programming Languages (SNAPL 2015). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[3]Seo, Jiwon, Stephen Guo, and Monica S. Lam. "Socialite: Datalog extensions for efficient social network analysis." 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013.

Datalog for Small Data

- Embedded devices
- Web browser
- In a Unikernel (e.g. MirageOS)

DREAM

https://dream.public.cat/

Keeping CALM[4]

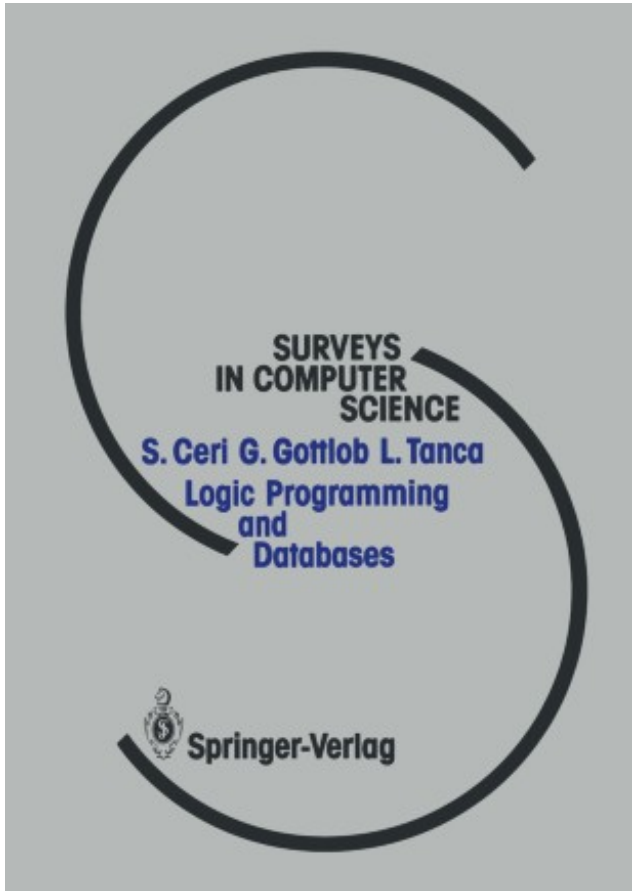
A program has an eventually consistent, coordination-free execution strategy if and only if it is expressible in Datalog.

[4] Hellerstein, Joseph M., and Peter Alvaro. "Keeping CALM: when distributed consistency is easy." arXiv preprint arXiv:1901.01930 (2019).

Implementations

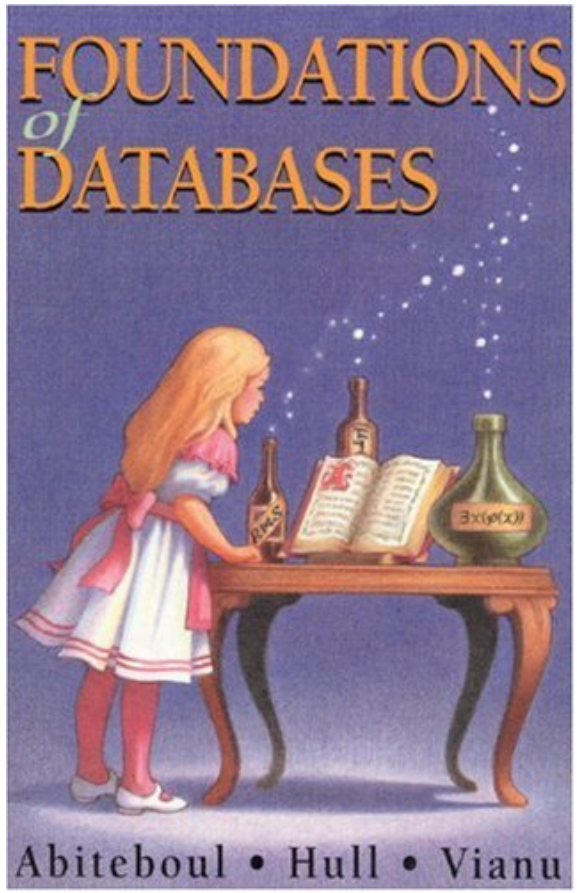
Soufflé	Full-featured and performant Datalog implementation (`https://souffle-lang.github.io`)
AbcDatalog	Simple Datalog implementation with a nice GUI interface (`https://abcdatalog.seas.harvard.edu/`)

Logic Programming and Databases



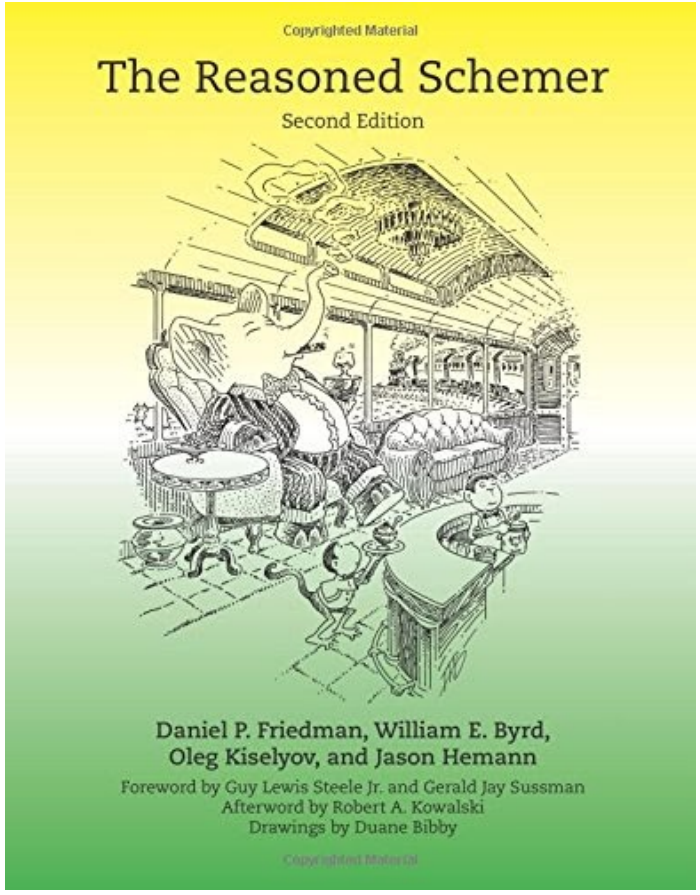
Stefano Ceri, Georg Gottlob, Letizia Tanca – Logic Programming and Databases (1990)

The Alice Book



Serge Abiteboul, Richard Hull and Victor Vianu – Foundations of Databases (1995)

miniKanren



P.Daniel P. Friedman, William E. Byrd, Oleg Kiselyov and Jason Hemann - The Reasoned Schemer (Second Edition)

openEngiadina

https://openengiadina.net

NGI ZERO

nlnet FOUNDATION

Thank you!

pukkamustard@posteo.net
https://inqlab.net