

# Datalog Revival

Serge Abiteboul

INRIA Saclay, Collège de France, ENS Cachan



# Datalog history

Started in 77: logic and database workshop

Simple idea: **add recursion to positive FO queries**

Blooming in the 80<sup>th</sup>

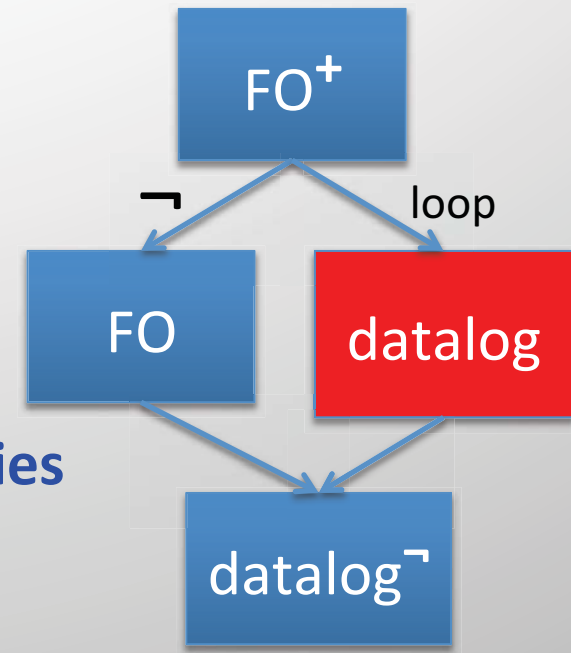
- Logic programming was hot

Industry was not interested:

- “No practical applications of recursive query theory ... have been found to date.” Hellerstein and Stonebraker (Readings in DB Systems)

Quasi dead except local resistance [e.g., A.,Gottlob]

Revival in this century



# Organization

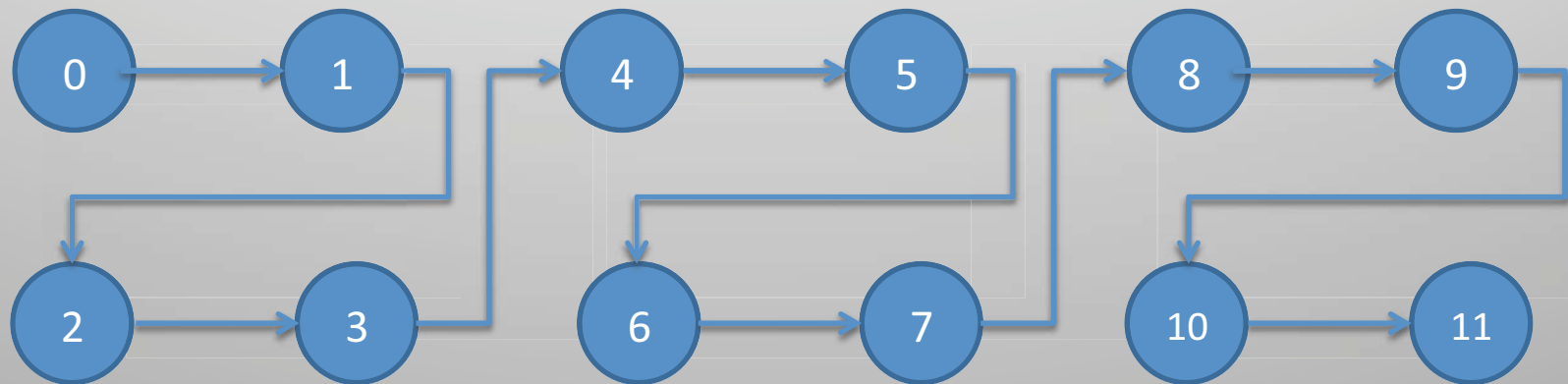
- Datalog
- Datalog evaluation
- Datalog with negation
- Datalog revival
- Conclusion

# Datalog

# Limitation of relational calculus

G a graph:  $G(0,1)$ ,  $G(1,2)$ ,  $G(2,3)$ , ...  $G(10,11)$

Is there a path from 0 to 11 in the graph?



k-path  $\exists x_1 \dots x_k ( G(0,x_1) \wedge G(x_1,x_2) \wedge \dots \wedge G(x_{k-1},x_k) \wedge G(x_k,11) )$

Path of unbounded length: **infinite** formula

$$\bigvee_{k=1 \text{ to } \infty} \text{k-path}$$

Term = constant or  
variable

Datalog **program** =  
set of datalog rules

## Datalog

$G(2,3)$

$T(x, y) \leftarrow G(x, z), T(z, y)$

fact

rule

datalog rule :  $R_1(u_1) \leftarrow R_2(u_2), \dots, R_n(u_n)$  for  $n \geq 1$

head

body

- Each  $u_i$  is a vector of **terms**
- **Safe**: each variable occurring in head must occur in body
- **Intentional** relation: occurs in the head
- **Extensional** relation: does not

# Datalog program

1.  $G(0,1), G(1,2), G(2,3), \dots G(10,11)$

2.  $T(x, y) \leftarrow G(x, y)$

3.  $T(x, y) \leftarrow G(x, z), T(z, y)$

4.  $Ok() \leftarrow T(0, 11)$



**edb**(P) = {G}

**idb**(P) = {T,Ok}

**program** P

# Datalog program

1.  $G(0,1), G(1,2), G(2,3), \dots G(10,11)$
2.  $T(x, y) \leftarrow G(x, y)$   $T(10, 11) \leftarrow G(10, 11)$
3.  $T(x, y) \leftarrow G(x, z), T(z, y)$   $T(0, 11) \leftarrow G(0,10), T(10,11)$
4.  $Ok() \leftarrow T(0, 11)$   $Ok() \leftarrow T(0, 11)$

Rule 2:  $v(x)=10 \ \& \ v(y) = 11$  ☞  $T(10,11)$

Rule 3:  $v(x)=9, v(z)=10 \ \& \ v(y)=11$  ☞  $T(9,11)$

...

Rule 3:  $v(x)=0, v(z)=1 \ \& \ v(y)=11$  ☞  $T(0,11)$

Rule 4:  $v(x)=0, v(y)=11$  ☞  $Ok()$

# Model semantics

View **P** as a first-order sentence  $\Sigma_P$  describing the answer

- Associate a formula to each rule

$$R_1(u_1) \leftarrow R_2(u_2), \dots, R_n(u_n) :$$

$$\forall x_1, \dots, x_m ( R_2(u_2) \wedge \dots \wedge R_n(u_n) \Rightarrow R_1(u_1) )$$

where  $x_1, \dots, x_m$  are the variables occurring in the rule

$$\mathbf{P} = \{r_1, \dots, r_n\}, \Sigma_P = r_1 \wedge \dots \wedge r_n$$

The **semantics** of **P** for a database **I**, denoted **P(I)**, is the **minimum model of  $\Sigma_P$  containing I**

Does it always exist?

How can it be computed?

## Example: Transitive closure

$G(0,1), G(1,2), G(2,3)$

$T(x,y) \leftarrow G(x,y)$

$T(x,y) \leftarrow G(x,z), T(z,y)$

| G   | P   |
|-----|-----|
| 0 1 | 0 1 |
| 1 2 | 1 2 |
|     | 0 2 |

Does not  
contain I

| G   | P   |
|-----|-----|
| 0 1 | 0 1 |
| 1 2 | 1 2 |
| 2 3 | 2 3 |
|     | 0 2 |
|     | 1 3 |

Not a model  
of the formula

| G   | P   |
|-----|-----|
| 0 1 | 0 1 |
| 1 2 | 1 2 |
| 2 3 | 2 3 |
|     | 0 2 |
|     | 1 3 |
|     | 0 3 |

Minimum model  
containing I

| G   | P   |
|-----|-----|
| 0 1 | 0 1 |
| 1 2 | 1 2 |
| 2 3 | 2 3 |
|     | 0 2 |
|     | 1 3 |
|     | 0 3 |
|     | 6 3 |

Model but not  
minimal

## Existence of $P(I)$

There exists at least one such model: the largest instance one can build with the constants occurring in  $I$  and  $P$  is a model of  $P$  that includes  $I$  –  $B(I,P)$

$P(I)$  always exists: it is the intersection of all models of  $P$  that include  $I$  over the constants occurring in  $I$  and  $P$

How can it be computed?

## Fixpoint semantics

A fact  $A$  is an *immediate consequence* for  $K$  and  $P$  if

1.  $A$  is an extensional fact in  $K$ , or
2. for some instantiation  $A \leftarrow A_1, \dots, A_n$  of a rule in  $P$ , each  $A_i$  is in  $K$

Immediate consequence operator:

$$T_p(K) = \{ \text{immediate consequences for } K \text{ and } P \}$$

Note:  $T_p$  is monotone

## Fixpoint semantics – continued

$P(I)$  is a fixpoint of  $T_p$  – That is:  $T_p(P(I)) \subseteq P(I)$

Indeed,  $P(I)$  is the least fixpoint of  $T_p$  containing  $I$

Yields a means of computing  $P(I)$

$$I \subseteq T_p(I) \subseteq T_p^2(I) \subseteq \dots \subseteq T_p^i(I) = T_p^{i+1}(I) = P(I) \subseteq B(I, P)$$

# Proof theory

- Proof technique: SLD resolution
- A fact  $A$  is in  $P(I)$  iff there exists a proof of  $A$

# Static analysis

## Hard

- Deciding **containment** ( $P \subseteq P'$ ) is undecidable
- Deciding **equivalence** is undecidable
- Deciding **boundedness** is undecidable
  - There exists  $k$  such that for any  $l$ , the fixpoint converges in less than  $k$  stages
- So, **optimization is hard**

# Datalog evaluation by example

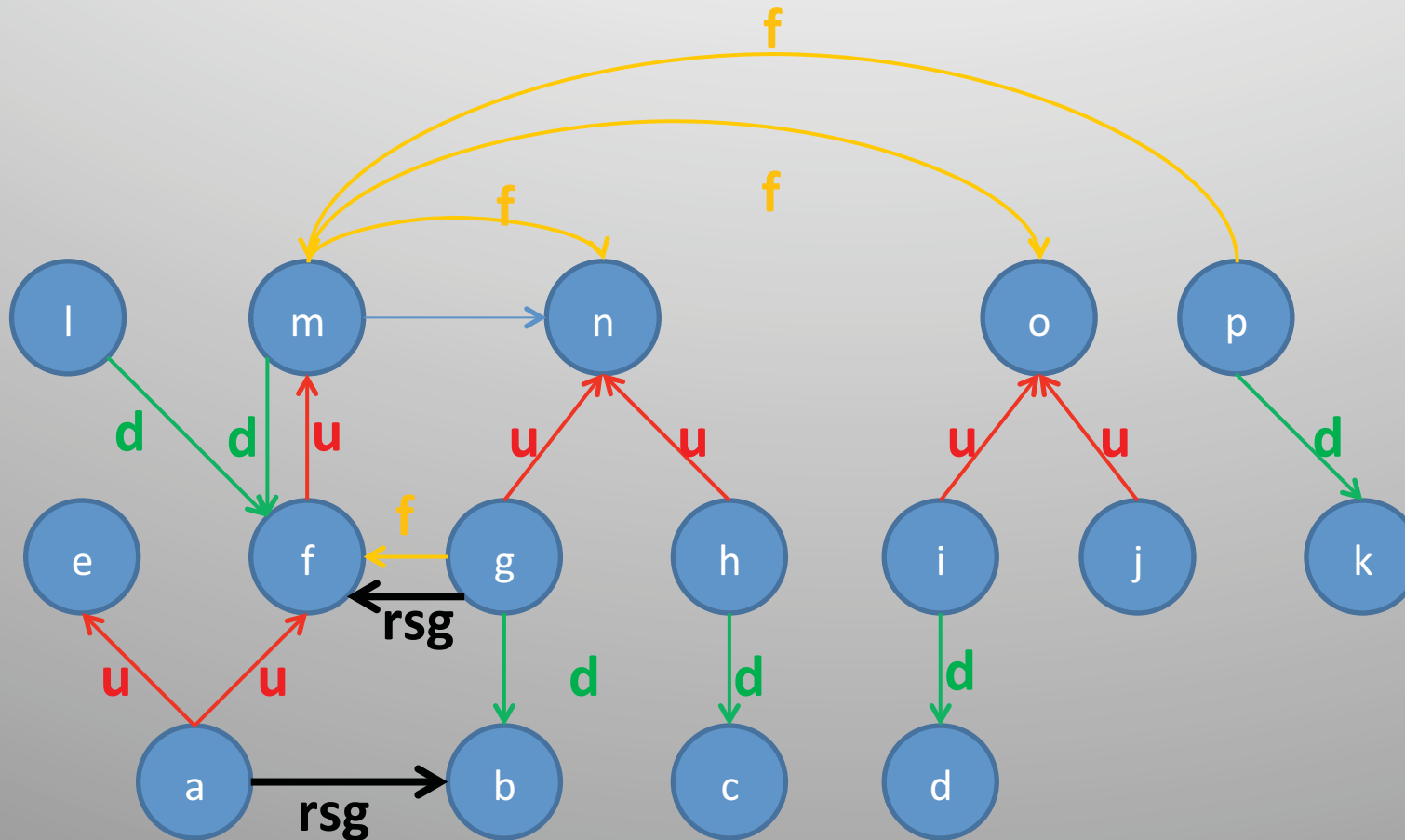
## More complicated example: Reverse same generation

| up |   | flat |   | down |   |
|----|---|------|---|------|---|
| a  | e | g    | f | l    | f |
| a  | f | m    | n | m    | f |
| f  | m | m    | o | g    | b |
| g  | n | p    | m | h    | c |
| h  | n |      |   | i    | d |
| i  | o |      |   | p    | k |
| j  | o |      |   |      |   |

$\text{rsg}(x,y) \leftarrow \text{flat}(x,y)$

$\text{rsg}(x,y) \leftarrow \text{up}(x,x1), \text{rsg}(y1,x1), \text{down}(y1,y)$

$$\text{rsg}(x,y) \leftarrow \text{flat}(x,y)$$

$$\text{rsg}(x,y) \leftarrow \text{up}(x,x1), \text{rsg}(y1,x1), \text{down}(y1,y)$$


|   |   |
|---|---|
| g | f |
| m | n |
| m | o |
| p | m |
| a | b |
| h | f |
| i | f |
| j | f |
| f | k |
| a | c |
| a | d |

# Naive algorithm

## Fixpoint

$$\text{rsg}_0 = \emptyset$$

$$\text{rsg}_{i+1} = \text{flat} \cup \text{rsg}_i \cup \pi_{16}(\sigma_{2=4}(\sigma_{3=5}(\text{up} \times \text{rsg}_i \times \text{down})))$$

## Program

$\text{rsg} := \emptyset ;$

repeat

$\text{rsg} := \text{flat} \cup \text{rsg} \cup \pi_{16}(\sigma_{2=4}(\sigma_{3=5}(\text{up} \times \text{rsg} \times \text{down})))$

until fixpoint

## Semi-naive

$$\Delta_1(x, y) \leftarrow \text{flat}(x, y)$$

$$\Delta_{i+1}(x, y) \leftarrow \text{up}(x, x1), \Delta_i(y1, x1), \text{down}(y1, y)$$

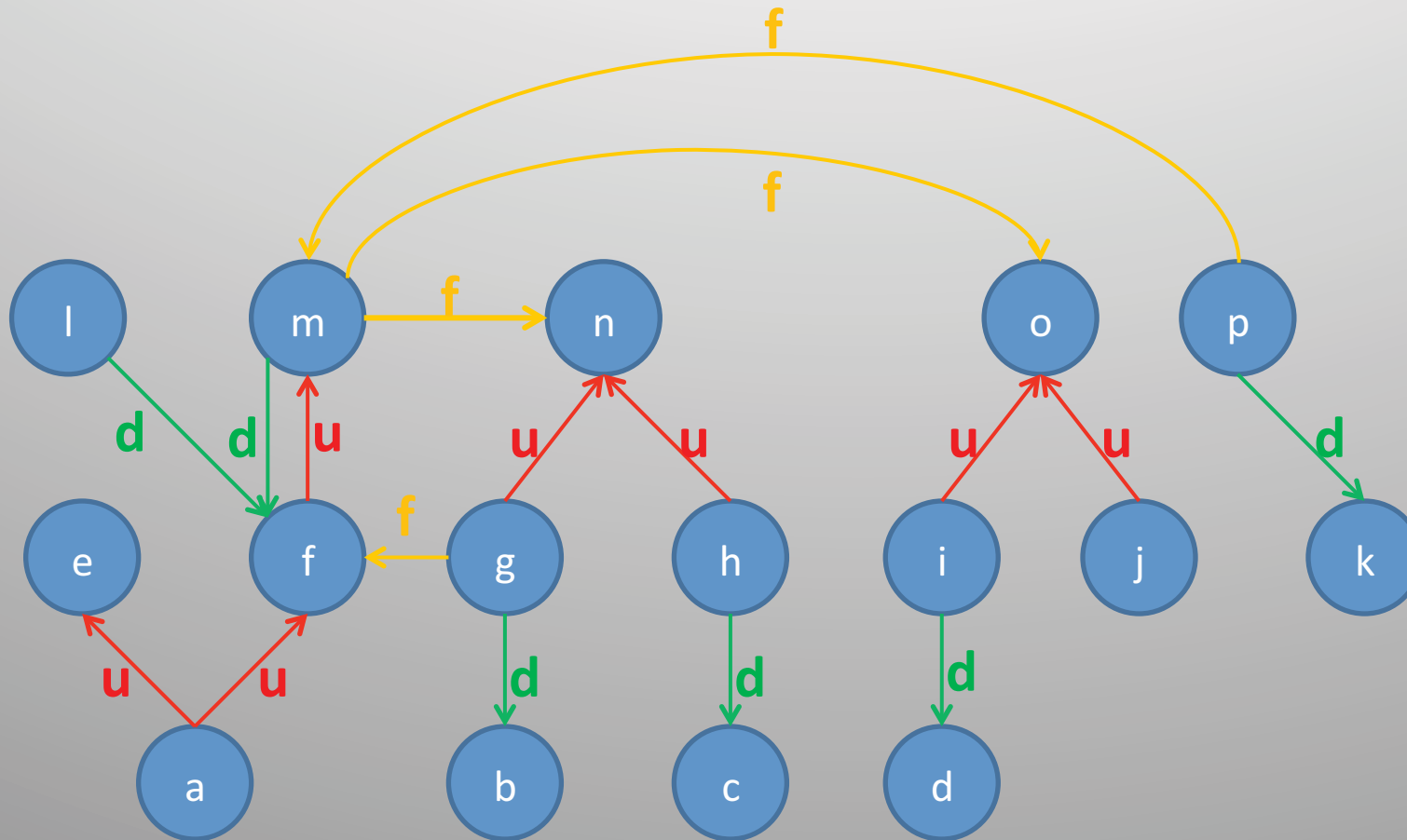
Compute  $\bigcup \Delta_i$

### Program

- Converges to the answer
- Not recursive & not a datalog program
- Still redundant – to avoid it:

$$\Delta_{i+1}(x, y) \leftarrow \text{up}(x, x1), \Delta_i(y1, x1), \text{down}(y1, y), \neg \Delta_i(x, y)$$

$$\text{rsg}(x,y) \leftarrow \text{flat}(x,y)$$

$$\text{rsg}(x,y) \leftarrow \text{up}(x,x1), \text{rsg}(y1,x1), \text{down}(y1,y)$$


|   |   |   |            |
|---|---|---|------------|
| g | f | } | $\Delta_1$ |
| m | n |   |            |
| m | o |   |            |
| p | m |   |            |
| a | b | } | $\Delta_2$ |
| h | f |   |            |
| i | f |   |            |
| j | f |   |            |
| f | k | } | $\Delta_3$ |
| a | c |   |            |
| a | d |   |            |
|   |   |   |            |

## Semi-naïve (end)

More complicated if the rules are not linear

$$T(x, y) \leftarrow G(x, y)$$

$$T(x, y) \leftarrow T(x, z), T(z, y)$$

- $\Delta_1(x, y) \leftarrow G(x, y)$
- $\text{anc}_1 := \Delta_1$
- $\text{temp}_{i+1}(x, y) \leftarrow \Delta_i(x, z), \text{anc}_i(z, y)$
- $\text{temp}_{i+1}(x, y) \leftarrow \text{anc}_i(x, z), \Delta_i(z, y)$
- $\Delta_{i+1} := \text{temp}_{i+1} - \text{anc}_i$
- $\text{anc}_{i+1} := \text{anc}_i \cup \Delta_{i+1}$

## And beyond

Start from a program and a query

$\text{rsg}(x,y) \leftarrow \text{flat}(x,y)$

$\text{rsg}(x,y) \leftarrow \text{up}(x,x1), \text{rsg}(y1,x1), \text{down}(y1,y)$

$\text{query}(y) \leftarrow \text{rsg}(a, y)$

Optimize to avoid deriving useless facts

Two competing techniques that are roughly equivalent

- Query-Sub-Query
- Magic Sets

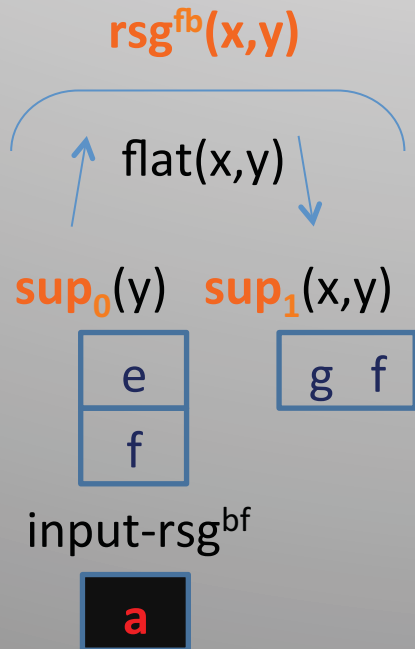
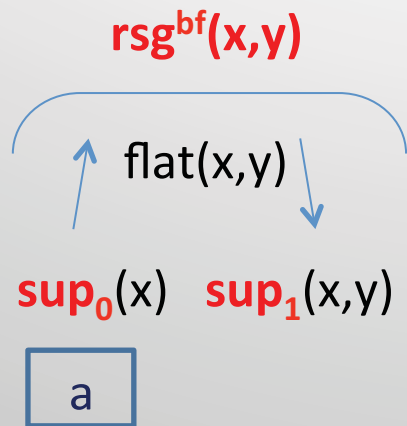
# Magic Set

$\text{rsg}^{\text{bf}}(x, y) \leftarrow \text{input\_rsg}^{\text{bf}}(x), \text{flat}(x, y)$   
 $\text{rsgfb}(x, y) \leftarrow \text{input\_rsg}^{\text{fb}}(y), \text{flat}(x, y)$   
 $\text{sup31}(x, x1) \leftarrow \text{input\_rsg}^{\text{bf}}(x), \text{up}(x, x1)$   
 $\text{sup32}(x, y1) \leftarrow \text{sup31}(x, x1), \text{rsg}^{\text{fb}}(y1, x1)$   
 $\text{rsg}^{\text{bf}}(x, y) \leftarrow \text{sup32}(x, y1), \text{down}(y1, y)$   
 $\text{sup41}(y, y1) \leftarrow \text{input\_rsg}^{\text{fb}}(y), \text{down}(y1, y)$   
 $\text{sup42}(y, x1) \leftarrow \text{sup41}(y, y1), \text{rsg}^{\text{bf}}(y1, x1)$   
 $\text{rsg}^{\text{fb}}(x, y) \leftarrow \text{sup42}(y, x1), \text{up}(x, x1)$   
 $\text{input\_rsg}^{\text{bf}}(x1) \leftarrow \text{sup31}(x, x1)$   
 $\text{input\_rsg}^{\text{fb}}(y1) \leftarrow \text{sup41}(y, y1)$

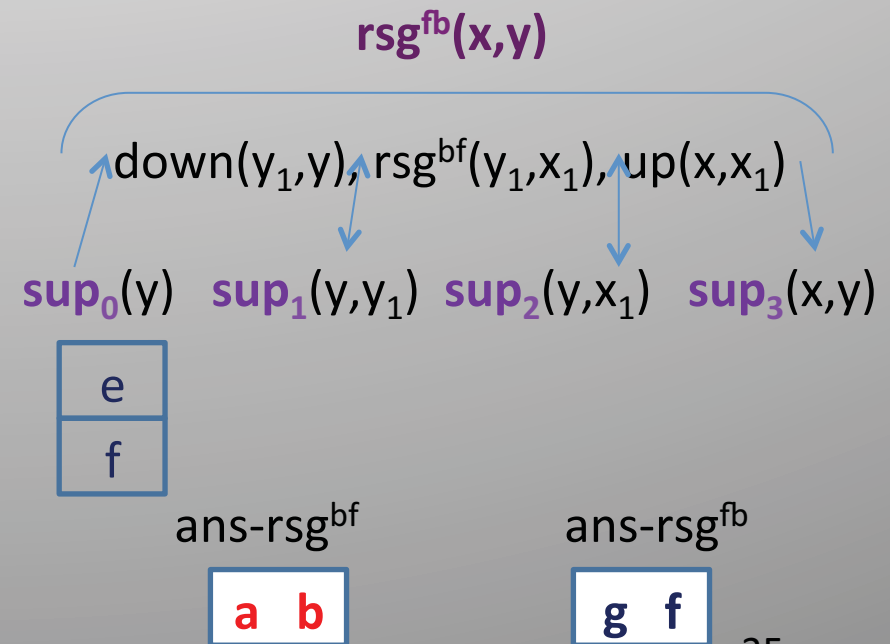
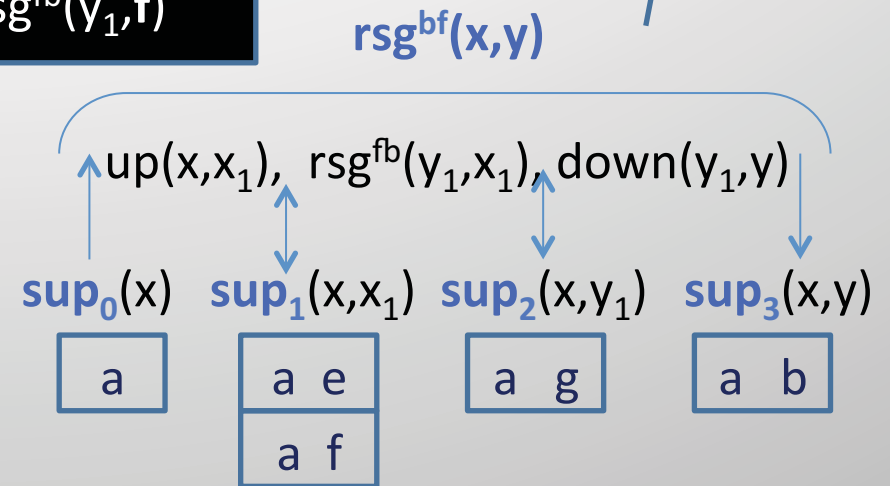
Seed  $\text{input\_rsg}^{\text{bf}}(a) \leftarrow$

Query  $\text{query}(y) \leftarrow \text{rsg}^{\text{bf}}(a, y)$

# QSQ at work



Subqueries  
 $rsg^{fb}(y_1,e)$   
 $rsg^{fb}(y_1,f)$



# SLD-resolution by example

← query(y)

query(y) ← rsg(a, y)

← rsg(a,y)

rsg(x,y) ← up(x,x1), rsg(y1,x1), down(y1,y)

← rup(a,x1), rsg(y1,x1), down(y1,y)

up(a,f)

← rsg(y1,f), down(y1,y)

rsg(x,y) ← flat(x,y)

← flat(y1,f), down(y1,y)

flat(g,f)

← down(g,y)

down(g,b)

←

y=b is an answer

## Datalog<sup>-</sup> by example

Accept negative literal in body

Complement of transitive closure

$\text{CompG}(x,y) \leftarrow \neg G(x,y)$

# More complicated

Some  $T_p$  are not monotone

Some  $T_p$  have no fixpoint containing  $I$

- $P_1 = \{p \leftarrow \neg p\}$
- $\emptyset \rightarrow \{p\} \rightarrow \emptyset \rightarrow \{p\} \rightarrow \dots$

Some  $T_p$  have several minimal fixpoints containing  $I$

- $P_2 = \{p \leftarrow \neg q, q \leftarrow \neg p\}$
- Two minimal fixpoints:  $\{p\}$  and  $\{q\}$ .

Some  $T_p$  have a least fixpoint but sequence diverges

- $P_3 = \{p \leftarrow \neg r ; r \leftarrow \neg p ; p \leftarrow \neg p, r\}$
- alternates between  $\emptyset$  and  $\{p, r\}$
- But  $\{p\}$  is a least fixpoint

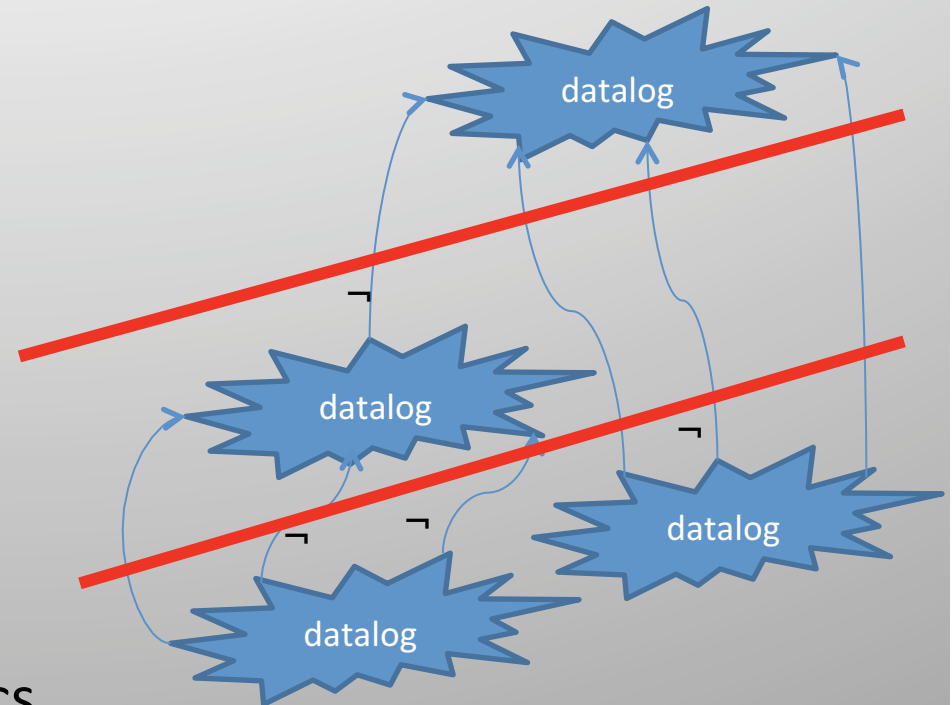
Model semantics

- Some programs have no model containing  $I$
- Some program have several minimal models containing

# First fix: stratification

Impose condition on the syntax

- Stratified programs



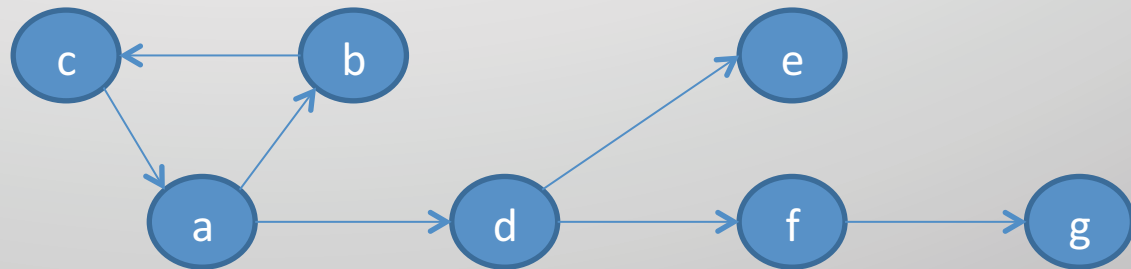
Consider more complex semantics

- Many such proposals
- Well-founded semantics based on 3-valued logic

# Well-founded by example: 2-player game

e, g are  
loosing

move graph:  
(relation **K**)



There is a pebble in a node

2 players alternate playing

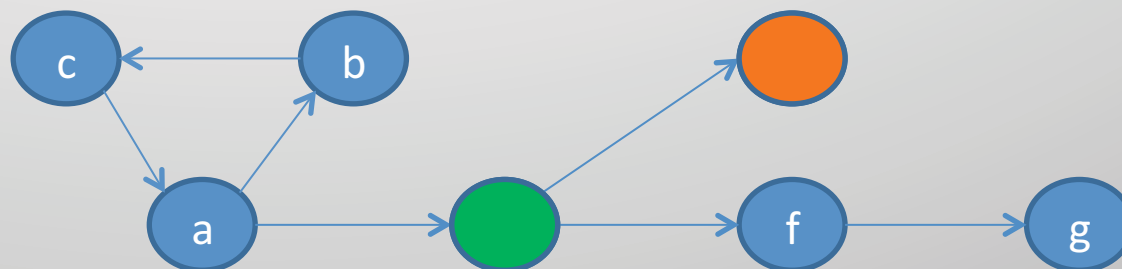
A player moves the pebble following an edge

A player who cannot move loses

d,f are  
winning

## Winning position

move graph:  
(relation  $K$ )

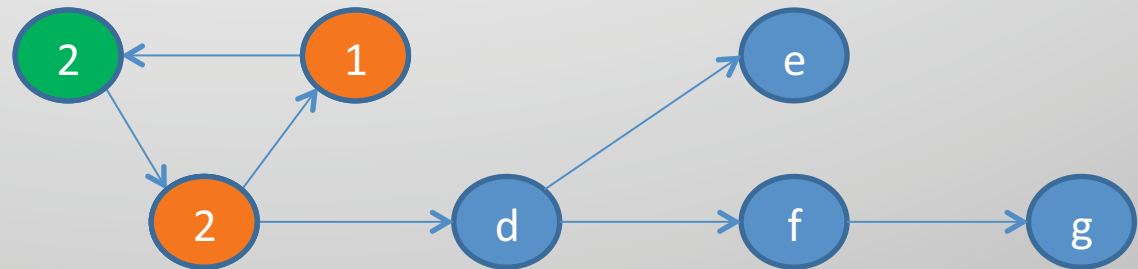


There is a pebble in a node  
2 players alternate playing  
A player moves the pebble following an edge  
A player who cannot move loses

a, b, c  
unknown

## No winner no loser

move graph:  
(relation **K**)



There is a pebble in a node  
2 players alternate playing  
A player moves the pebble following an edge  
A player who cannot move loses

# Program to specify the winning/loosing positions

$\text{win}(x) \leftarrow \text{move}(x, y), \neg \text{win}(y)$

Well-founded semantics: find the instance **J** that  
agrees with K on move and  
satisfies the formula corresponding to the rule

**Instance J**

**– three-valued instance**

$\text{win}(d), \text{win}(f)$

are true

$\text{win}(e), \text{win}(g)$

are false

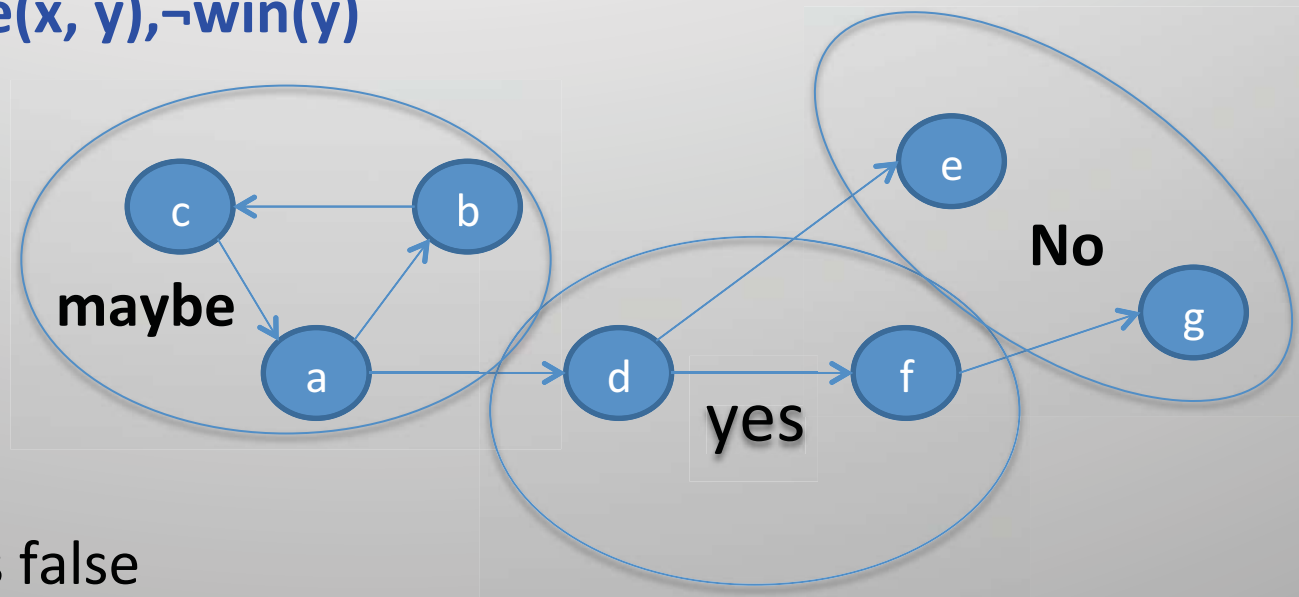
$\text{win}(a), \text{win}(b), \text{win}(c)$

are unknown

Fixpoint semantics based on 3-valued logic

# Fixpoint computation

- $\text{win}(x) \leftarrow \text{move}(x, y), \neg \text{win}(y)$



- $I_0$ : win is always false
- $I_1$ : win: a, b, c, d, f
- $I_2$ : win: d, f
- $I_3$ : win: a, b, c, d, f
- $I_4$ : win: d, f

# Complexity and expressivity

- Datalog and Datalog<sup>-</sup> evaluations are easy
- **Datalog  $\subset$  Ptime**
  - In the data
  - Inclusion in ptime: polynomial number of stages; each stage in ptime
  - Strict: Expresses only monotone queries;
  - But does not even express all PTIME monotone queries
- **Datalog<sup>-</sup> with well-founded semantics = fixpoint  $\subset$  Ptime**
  - In the data
  - On ordered databases, it is exactly PTIME

# Datalog revival

# Datalog revival

Datalog needs to be extended to be useful

Updates, value creation, nondeterminism

[e.g. A., Vianu]

Skolem [e.g. Gottlob]

Constraints [e.g. Revesz]

Time [e.g. Chomicki]

Distribution [e.g. *ActiveXML*]

Trees [e.g. *ActiveXML*]

Aggregations [e.g. Consens and Mendelzon]

Delegation [e.g. *Webdamlog*]

# Datalog revival: different domains

|                               |                        |                                                                                       |
|-------------------------------|------------------------|---------------------------------------------------------------------------------------|
| Declarative networking        | [e.g. Lou et al]       |                                                                                       |
| Data integration and exchange | [e.g. Clio, Orchestra] |                                                                                       |
| Program verification          | [e.g. Semmle]          |                                                                                       |
| Data extraction from the Web  | [e.g. Gottlob, Lixto]  |                                                                                       |
| Knowledge representation      | [e.g. Gottlob]         |                                                                                       |
| Artifact and workflows        | [e.g. ActiveXML]       |  |
| Web data management           | [e.g. Webdamlog]       |  |

# Declarative networking

## Traditional vs. declarative

Network state

Distributed database

Network protocol

Datalog program

Messages

Messages

## Series of languages/systems from Hellerstein groups in Berkeley

- Overlog, bloom, dedalus, bud...
- Performance: scalability

## Many systems have been developed

Internet routing

Overlay networks

Sensor networks

...

# Data integration

$\forall \text{Eid, Name, Addr}$

$( \text{employee}(\text{Eid, Name, Addr}) \Rightarrow$

$\exists \text{Ssn } ( \text{name}(\text{Ssn, Name}) \wedge \text{address}(\text{Ssn, Addr}) ) )$

Use “inverse” rules with Skolem

$\text{name}(\text{ssn}(\text{Name, Addr}), \text{Name})$

$\leftarrow \text{employee}(\text{X, Name, Addr})$

$\text{address}(\text{ssn}(\text{Name, Addr}), \text{Addr})$

$\leftarrow \text{employee}(\text{X, Name, Addr})$

Possibly infinite chase and issues with termination

# Program analysis

Analyze the possible runs of a program

Recursion

Lots of possible runs – lots of data

- Optimization techniques are essential
- Semi-naïve, Magic Sets, Typed-based optimization

# Data extraction

- Georg's talk next

# Conclusion

# Issues

Give precise semantics to the extensions

Some challenges for the Web

- Scaling to large volumes
- Datalog with distribution
- Datalog with uncertainty
- Datalog with inconsistencies

Berkeley's works

Webdamlog 



# Georg Gottlob,

- Professor at **Oxford** University & **TU Wien**
- Research: database, AI, logic and complexity
- Fellow of St John's and **Ste Anne's** College, Oxford
- Fellow: ACM, ECCAI, Royal Society
- Academy: Austrian, German, Europaea
- Program chair: IJCAI, PODS...
- Founder of Lixto, a company on web data extraction
- ERC Advanced Investigator's Grant (DIADEM)

