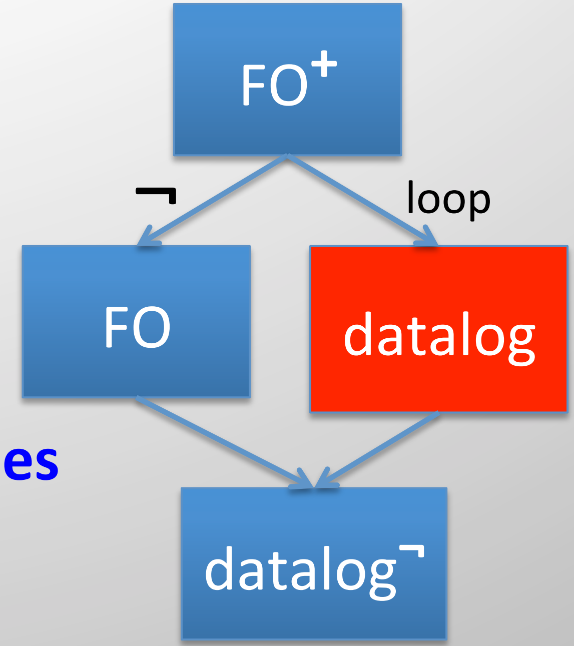
Datalog Revival

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 80th

- 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

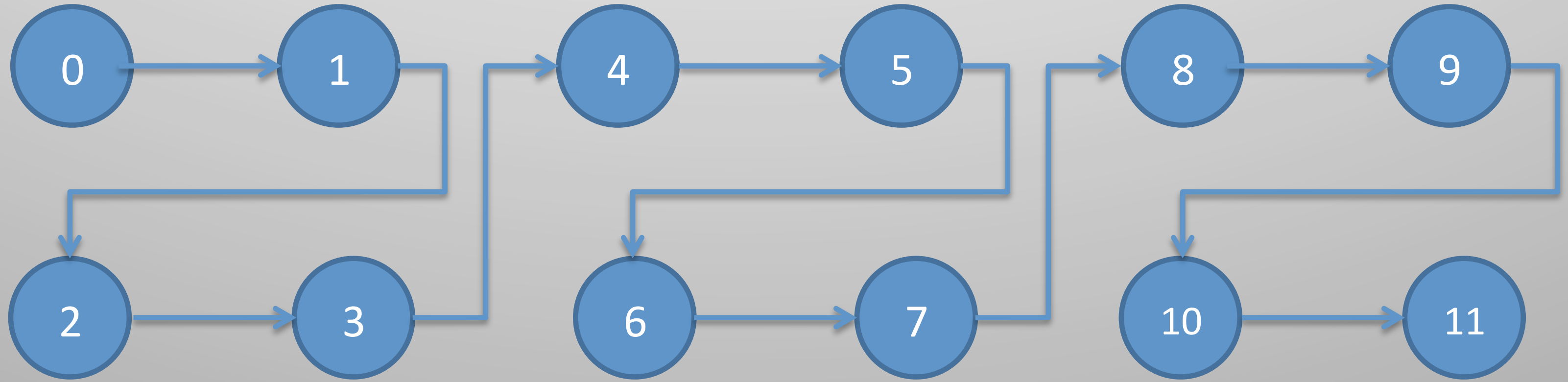
- Datalog
- Datalog evaluation
- Datalog with negation
- Datalog revival
- Conclusion

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) fact

T(x, y) ← G(x, z), T(z, y) rule

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

head: $R_1(u_1)$ — body: $R_2(u_2), \ldots, R_n(u_n)$

- 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), …   G(10,11)
2. T(x, y) ← G(x, y)
3. T(x, y) ← G(x, z), T(z, y)
4. Ok()   ← T(0, 11)

edb(P) = {G}

idb(P) = {T,Ok}

program P

Datalog program

1. G(0,1), G(1,2), G(2,3), … G(10,11)
2. T(x, y) ← G(x, y) **T(10, 11) ← G(10, 11)**
3. T(x, y) ← G(x, z), T(z, y) **T(9, 11) ← G(9,10), T(10,11)** **T(0, 11) ← G(0,1), T(1,11)**
4. Ok() ← T(0, 11) **Ok() ← 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 Σ_P describing the answer

- Associate a formula to each rule

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

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

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

P $= \{r_1, \ldots, r_n\}$, $\Sigma_P = r_1 \wedge \ldots \wedge r_n$

The **semantics** of **P** for a database **I**, denoted **P(I)**, is the **minimum model of Σ_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) ← G(x,y)

T(x,y) ← 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 $A \leftarrow A_1, \ldots, A_n$ of a rule in P, each A_i is in K

Immediate consequence operator:

$T_P(K)$ = { immediate consequences for K 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)

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

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 I, the fixpoint converges in less than k stages

- So, **optimization is hard**

Datalog evaluation by example

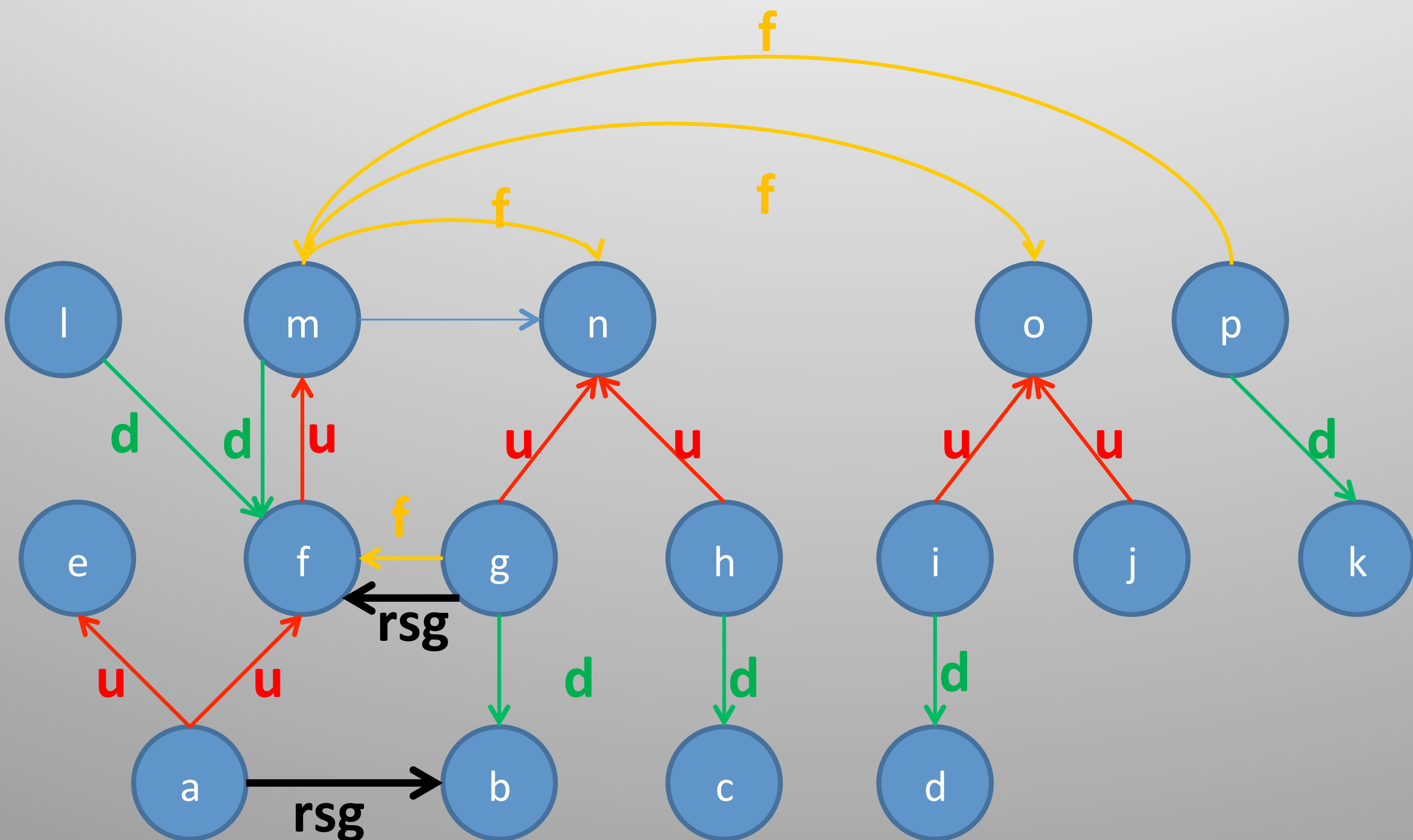
More complicated example: Reverse same generation

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				

rsg(x,y) ← flat(x,y)

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

rsg(x,y) ← up(x,x1),rsg(y1,x1),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

$rsg_0 = \emptyset$

$rsg_{i+1} = flat \cup rsg_i \cup$ **$\pi_{16}(\sigma_{2=4}(\sigma_{3=5}(up \times rsgi_i \times down)))$**

Program

rsg := $\emptyset$;

repeat

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

until fixpoint

Semi-naive

$\Delta_1(x, y) \leftarrow flat(x, y)$

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

Compute $\cup \Delta_I$

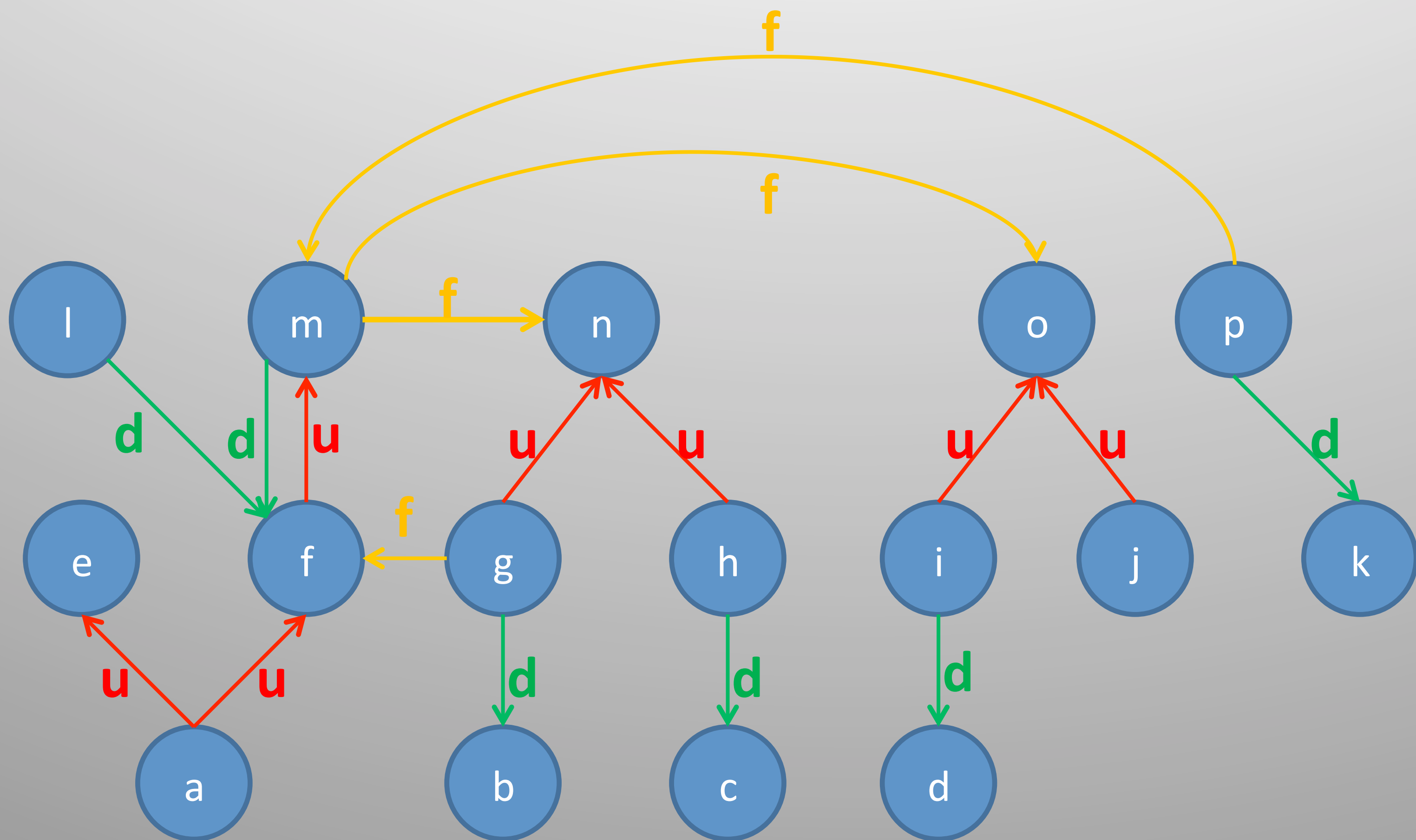
Program

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

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

rsg(x,y) ← flat(x,y)

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



g	f	
m	n	
m	o	Δ_1
p	m	
a	b	
h	f	
i	f	
j	f	Δ_2
f	k	
a	c	
a	d	Δ_3

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)

- $\Delta_1(x, y) \leftarrow G(x, y)$
- $anc_1 := \Delta_1$

- $temp_{i+1}(x, y) \leftarrow \Delta_i(x, z), anc_i(z, y)$
- $temp_{i+1}(x, y) \leftarrow anc_i(x, z), \Delta_i(z, y)$
- $\Delta_{i+1} := temp_{i+1} - anc_i$
- $anc_{i+1} := anc_i \cup \Delta_{i+1}$

And beyond

Start from a program and a query

rsg(x,y) ← flat(x,y)

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

query(y) ← rsg(a, y)

Optimize to avoid deriving useless facts

Two competing techniques that are roughly equivalent

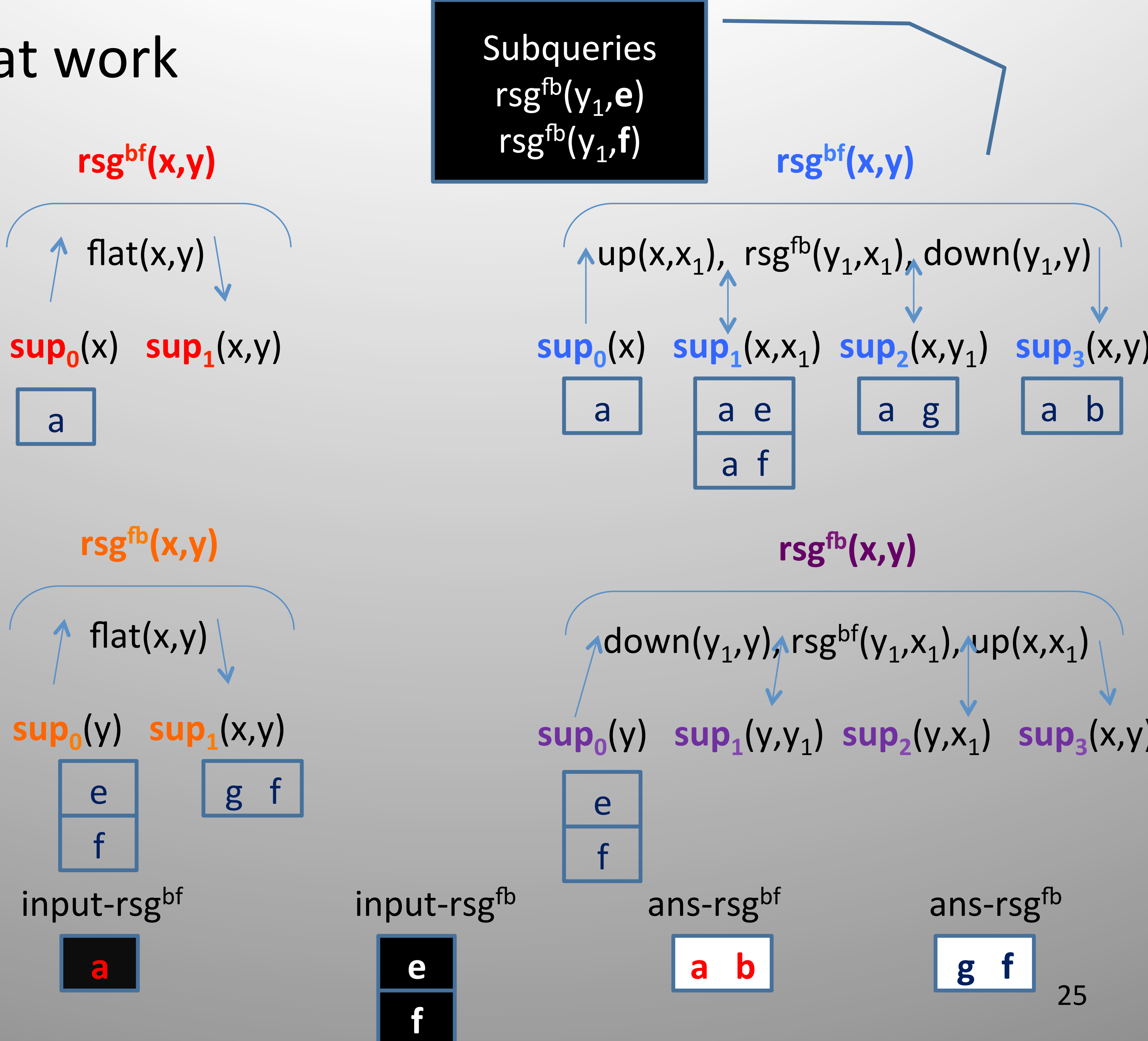
- Query-Sub-Query
- Magic Sets

Magic Set

rsg^{bf}(x, y) ←input_rsg^{bf}(x), flat(x, y)
rsgfb(x, y) ←input_rsg^{fb}(y), flat(x, y)
sup**31**(x, x**1**) ←input_rsg^{bf}(x), up(x, x**1**)
sup**32**(x, y**1**) ←sup**31**(x, x**1**), rsg^{fb}(y**1**, x**1**)
rsg^{bf}(x, y) ←sup**32**(x, y**1**), down(y**1**, y)
sup**41**(y, y**1**) ←input_rsg^{fb}(y), down(y**1**, y)
sup**42**(y, x**1**) ←sup**41**(y, y**1**), rsg^{bf}(y**1**, x**1**)
rsg^{fb}(x, y) ←sup**42**(y, x**1**), up(x, x**1**)
input_rsg^{bf}(x**1**) ←sup**31**(x, x**1**)
input_rsg^{fb}(y**1**)←sup**41**(y, y**1**)

Seed input_rsg^{bf}(a) ←
Query query(y) ←rsg^{bf}(a, y)

QSQ at work


Subqueries
rsgfb(y1,e)
rsgfb(y1,f)
rsgbf(x,y)
flat(x,y)
sup0(x) sup1(x,y)
a
rsgbf(x,y)
up(x,x1), rsgfb(y1,x1), down(y1,y)
sup0(x) sup1(x,x1) sup2(x,y1) sup3(x,y)
a
a e
a f
a g
a b
rsgfb(x,y)
flat(x,y)
sup0(y) sup1(x,y)
e
f
g f
rsgfb(x,y)
down(y1,y), rsgbf(y1,x1), up(x,x1)
sup0(y) sup1(y,y1) sup2(y,x1) sup3(x,y)
e
f
input-rsgbf
a
input-rsgfb
e
f
ans-rsgbf
a b
ans-rsgfb
g 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$^\neg$ by example

Accept negative literal in body

Complement of transitive closure

$CompG(x,y) \leftarrow \neg G(x,y)$

More complicated

Some T_P are not monotone

Some T_P have no fixpoint containing I

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

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

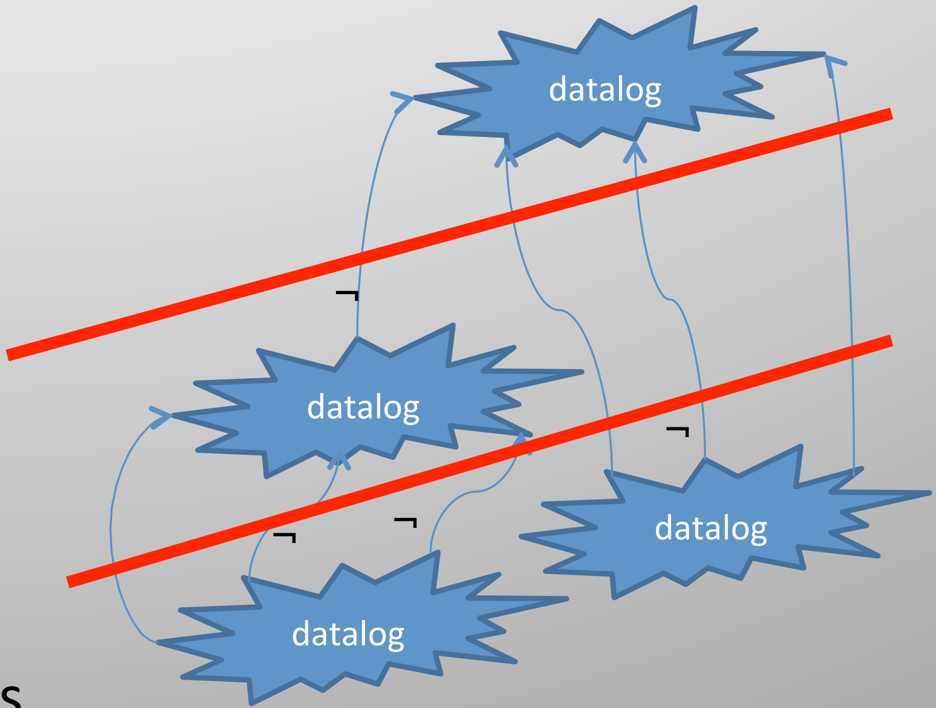
Model semantics

- 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

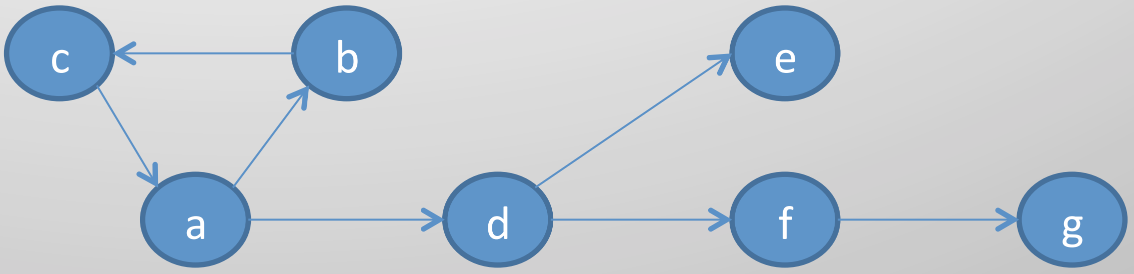
- Many such proposals
- Well-founded semantics based on 3-valued logic

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

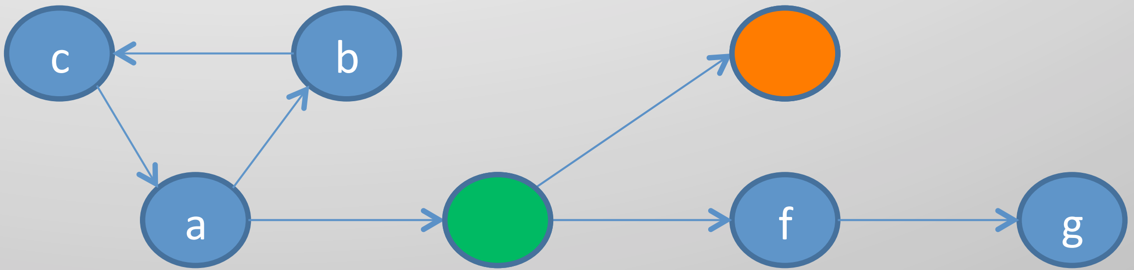
A player moves the pebble following an edge

A player who cannot move loses

Winning position

d,f are winning

move graph:
(relation K)



There is a pebble in a node

2 players alternate playing

A player moves the pebble following an edge

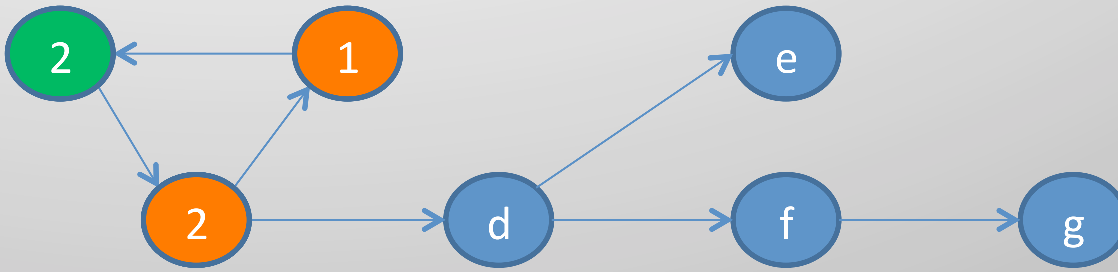
A player who cannot move loses

a, b, c unknown

No winner no looser

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

win(x) ← move(x, y),¬win(y)

Well-founded semantics: find the instance **J** that

agrees with K on move and

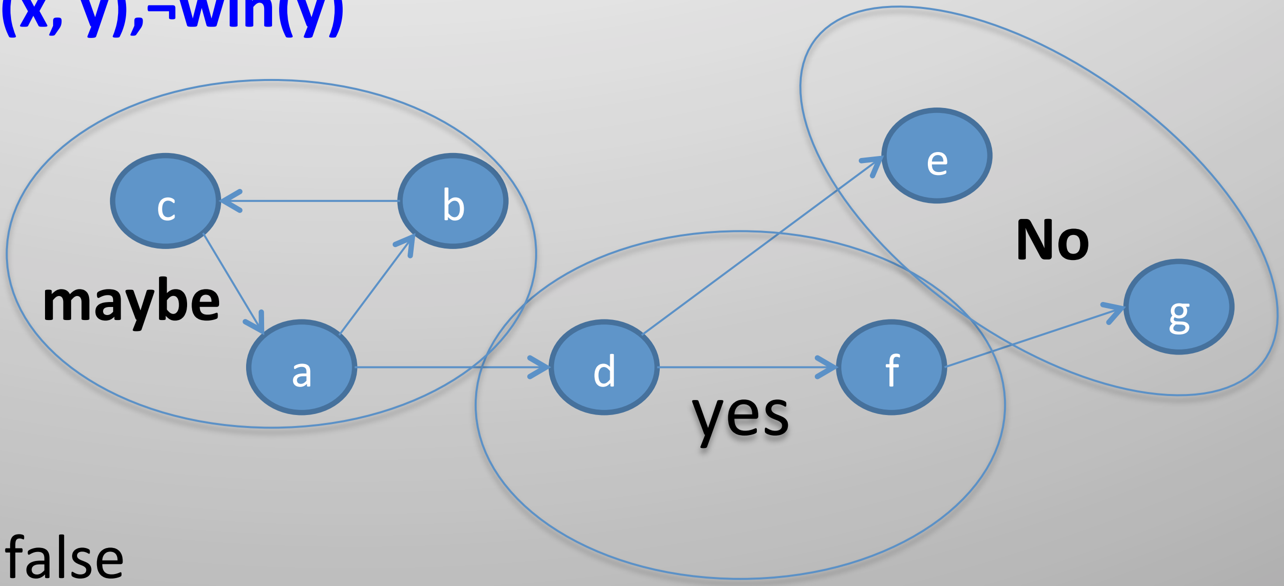
satisfies the formula corresponding to the rule

Instance J	**– three-valued instance**
win(d),win(f)	are true
win(e),win(g)	are false
win(a),win(b),win(c)	are unknown

Fixpoint semantics based on 3-valued logic

Fixpoint computation

- **win(x) ← move(x, y),¬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$^{\neg}$ 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$^{\neg}$ 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

Traditional	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

$\forall$ Eid, Name, Addr

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

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

Use "inverse" rules with Skolem

name(**ssn(Name, Addr**), Name) ← employee(X, Name, Addr)

address(ssn(Name, Addr), Addr) ← employee(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 ☛

inria informatiques mathématiques

COLLÈGE DE FRANCE
1530

ENS
CACHAN

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)