Capturing Factorized Provenance (FDB 2022)

Boris Glavic

Illinois Institute of Technology

Database Group

Joint work with:

Seokki Lee *(University of Cincinnati), Bertram Ludäscher (UIUC)*

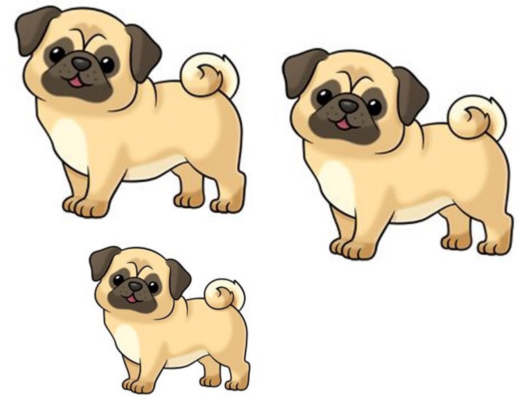
Outline

- **Overview & Motivation**
- Provenance Graph Model
- Capturing Provenance
- Factorization
- Experiments
- Conclusions & Future work

Debugging Datalog

- **Datalog has seen a resurgence in academia & industry**
  - Programming distributed systems
  - Complex data-intensive computations
  - Analytics & ML over query results
- **Urgent need for debugging Datalog programs**
  - Why did my program produce this unexpected result?
  - Why did my program not produce this expected result?
  - Which rules are responsible for deriving this result?
  - Why did this rule derivation fail?
  - Would deleting & inserting a tuple change the result?

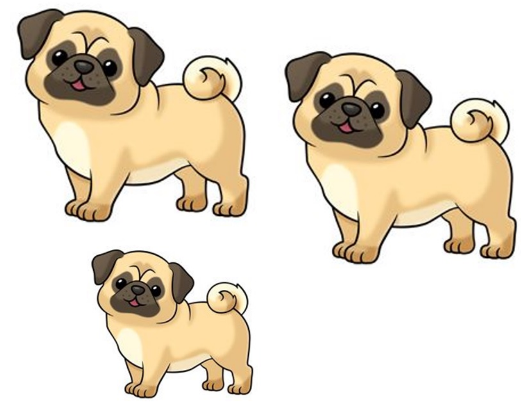
PUG + PUGS



Efficient capture *and* ***summarization*** *for* ***why and why-not*** *provenance through a* ***provenance model developed***

for queries with negation

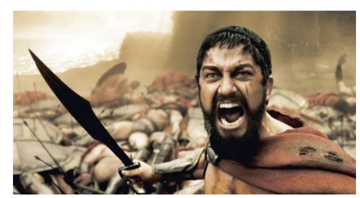
PUG + PUGS



Efficient capture** and **summarization** for **why and why-not** provenance through a **provenance model developed

for queries with negation

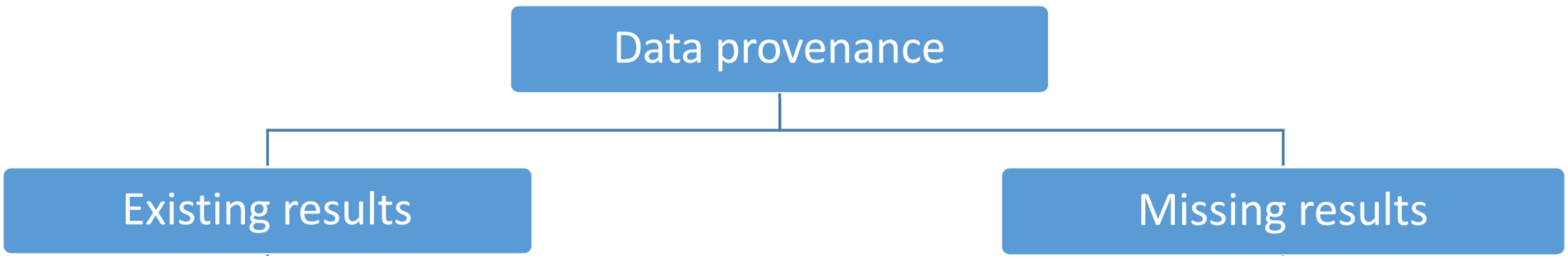
This is FDB!



PUG utilizes a flat-relational encoding of provenance graphs which essentially corresponds to a factorized representation of provenance

Schism between Why / Why-not

- How the rules of a program did derive / failed to derive an existing / missing output from the input data



Why

Why-not

Why/how Q(a) was derived?

Why Q(b) is **NOT** in the result?

R

A	B
b	a
a	b

S

B
b

$$Q(A) \text{ :- } R(A,B), S(B)$$

Q

A
a

Unifying Why / Why-not

- **Why** (provenance) and **why-not** (missing answers) have been mostly treated in isolation
- Why and why-not questions can be reduced to each other for a query language **L** if …
  - for any query **Q,** its complement $\mathbf{Q^C}$ is in **L**

$$t \notin Q(D) \equiv t \in Q^C(D)$$

```
Q(A) :- R(A,B)
Qᶜ(A):- R(A,B), adom(A), adom(B)
```

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Overview & Motivation
- **Provenance Graph Model**
- Capturing Provenance
- Factorization
- Experiments
- Conclusions & Future work

ILLINOIS INSTITUTE OF TECHNOLOGY

Requirements

- **Syntax-driven**
  - Provenance structure aligns with program structure
- **Compatible with well-established provenance models**
  - Provenance polynomials for positive queries [1]
  - Dual polynomials [2]
- **Build-in support for sharing common subexpressions**

[1] T. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In PODS, pages 31–40, 2007.

[2] E.Grädel and V.Tannen.Semiring provenance for first-order model checking. arXiv preprint arXiv:1712.01980, 2017.

Provenance Graph Model

ILLINOIS INSTITUTE OF TECHNOLOGY

R

A	B
b	a
a	b

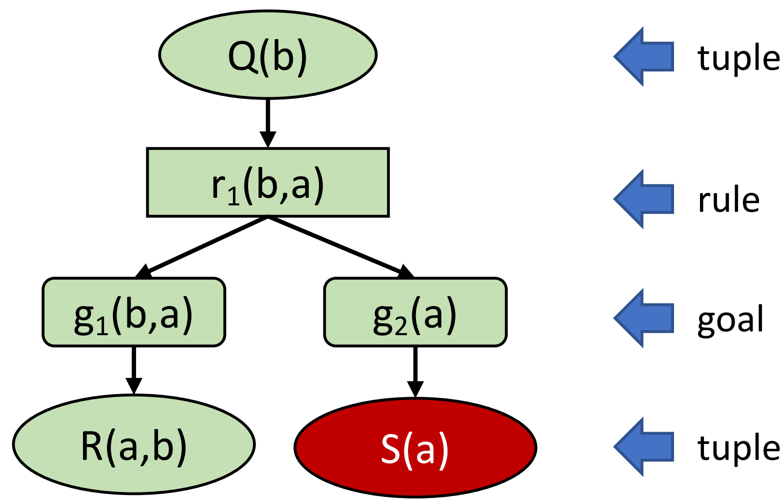
S

B
b

$$r_1: Q(A) \text{ :- } R(A,B), \neg S(B)$$

Q

A
b



Green: success / existence
Red: failure / absence

Provenance Graph Model

R

A	B
d	c
c	d

S

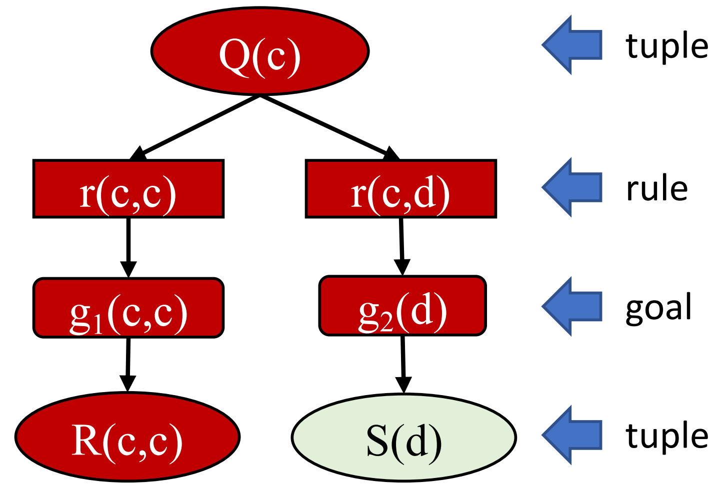
B
d

$$r: Q(A) \text{ :- } R(A,B),\ \neg\, S(B)$$

Q

A
d

Why Q(c) is **NOT** in the output?



Green: success / existence
Red: failure / absence

Provenance Graph Model

R

A	B
d	c
c	d

S

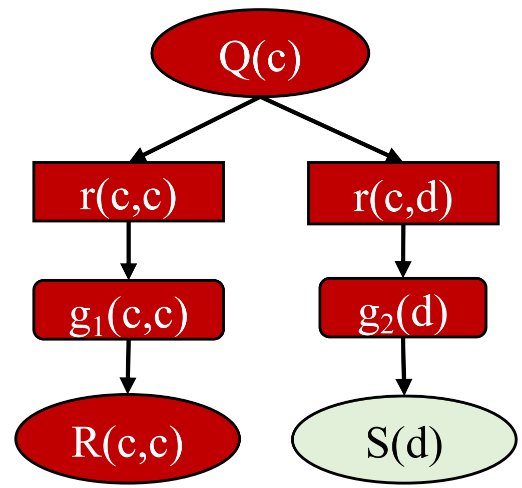
B
d

r: Q(A) :- R(A,B), ¬ S(B)

Q

A
d

Why Q(c) is **NOT** in the output?



Provenance Graph Model

R

A	B
d	c
c	d

S

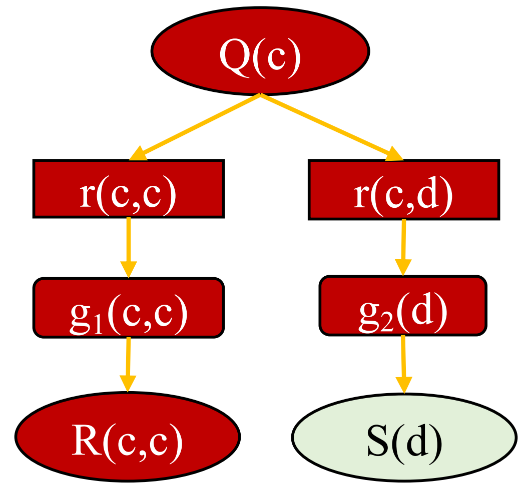
B
d

r: Q(A) :- R(A,B), ¬ S(B)

Q

A
d

Why Q(c) is **NOT** in the output?



Provenance Graph Model

R

A	B
d	c
c	d

S

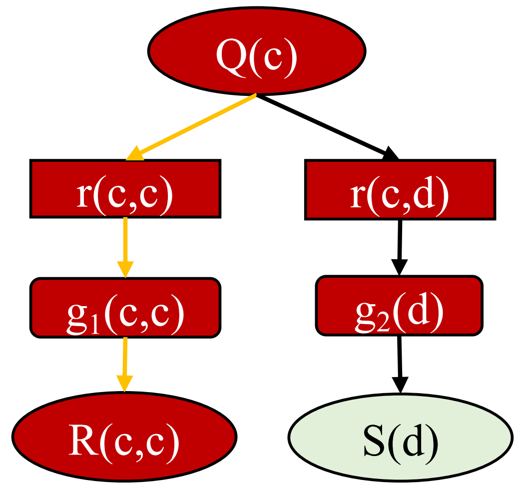
B
d

r: Q(c) :- R(c,c), ¬ S(c)

Q

A
d

Why Q(c) is **NOT** in the output?



Provenance Graph Model

R

A	B
d	c
c	d

S

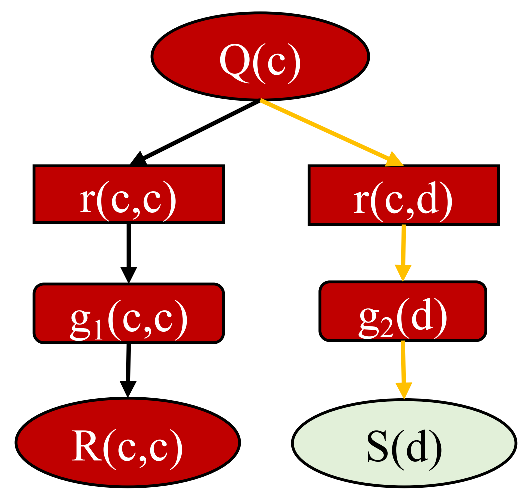
B
d

r: Q(c) :- R(c,d), ¬ S(d)

Q

A
d

Why Q(c) is **NOT** in the output?



Provenance Graph Model

R

A	B
d	c
c	d
e	d

S

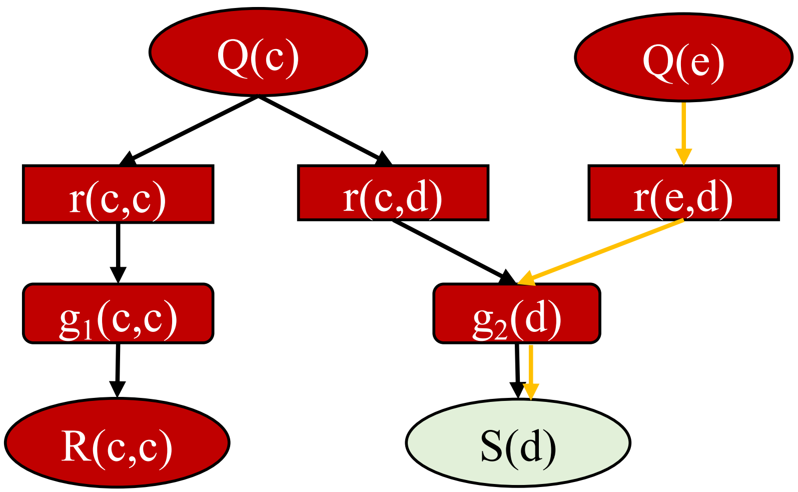
B
d

r: Q(e) :- R(e,d), ¬ S(d)

Q

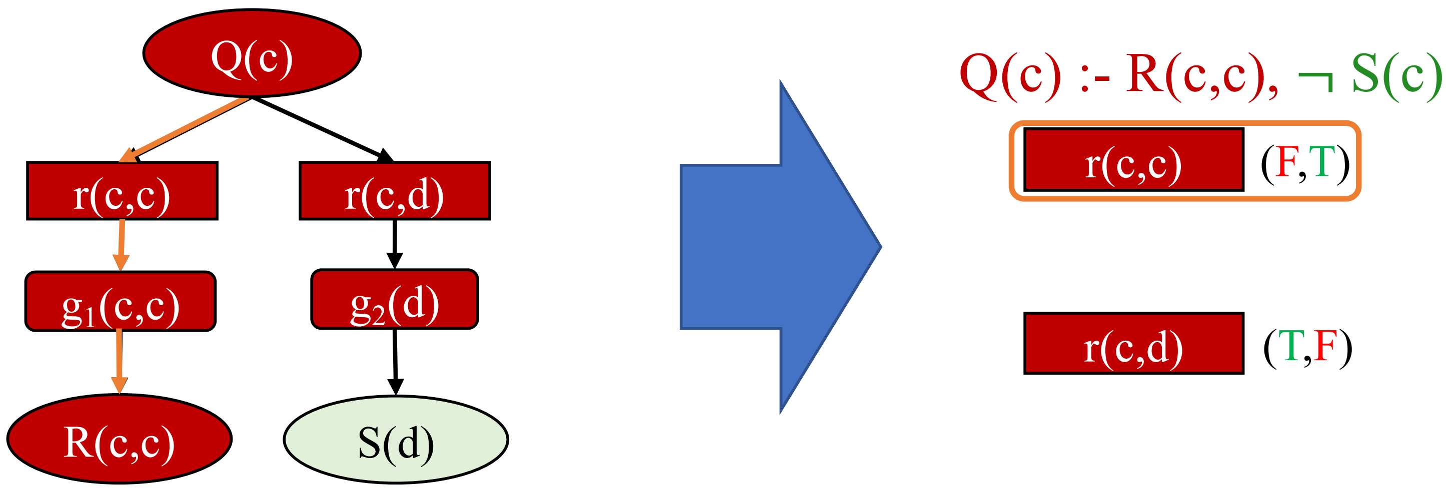
A
d

Why Q(c) is **NOT** in the output?



Provenance Graph Model

- Equivalent model (annotated rule derivations)

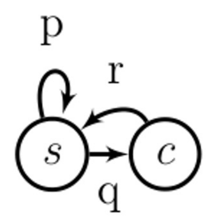


Extracting Provenance Polynomials

$$r_2 : \mathtt{Q}_{\mathtt{3hop}}(X, Y) :\!-\, \mathtt{T}(X, A), \mathtt{T}(A, B), \mathtt{T}(B, Y)$$

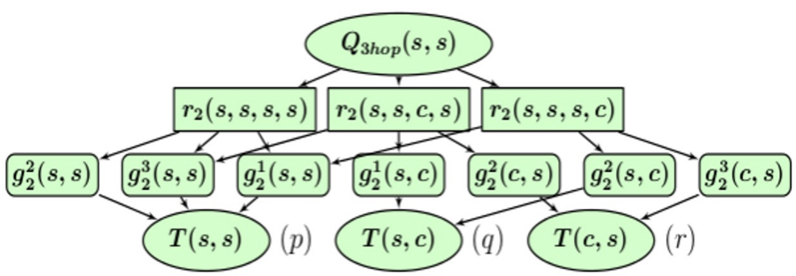
Relation **Train**

fromCity	toCity	$\mathbb{N}[X]$
seattle	seattle	p
seattle	chicago	q
chicago	seattle	r

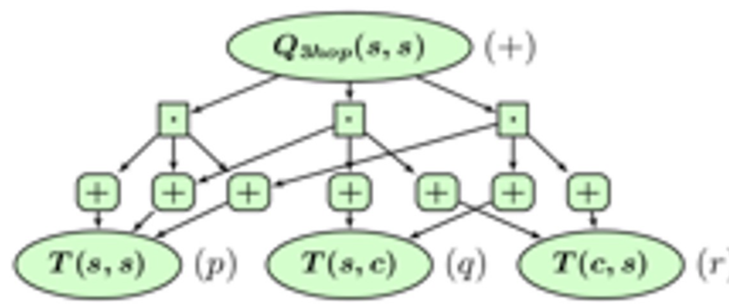


Result of query $\mathbf{Q_{3hop}}$

X	Y	$\mathbb{N}[X]$
seattle	seattle	$p^3 + 2pqr$



$$r_2(g_1(p) \cdot g_2(p) \cdot g_3(p)) + r_2(g_1(q) \cdot g_2(r) \cdot g_3(p)) + r_2(g_1(p) \cdot g_2(q) \cdot g_3(r))$$

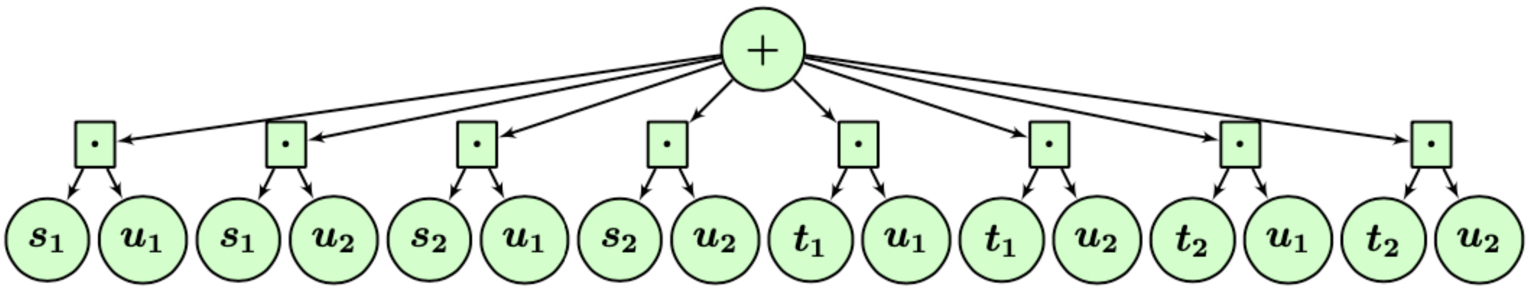


$$p^3 + 2 \cdot (p \cdot q \cdot r)$$

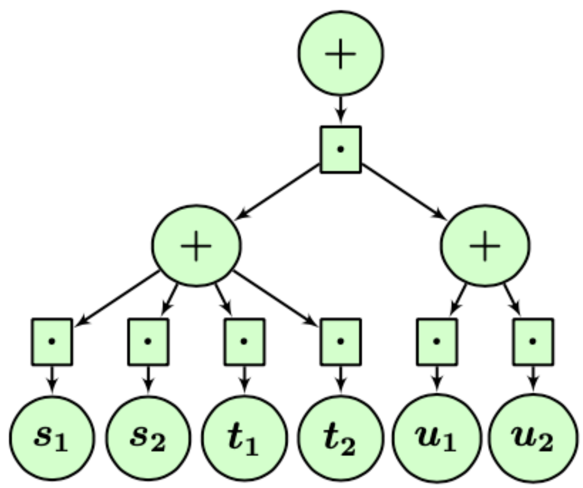
$$[[p, p, p], [p, q, r], [p, q, r]]$$

Factorized Provenance Graphs

ILLINOIS INSTITUTE OF TECHNOLOGY



$$s_1 \cdot u_1 + s_1 \cdot u_2 + s_2 \cdot u_1 + s_2 \cdot u_2 + t_1 \cdot u_1 + t_1 \cdot u_2 + t_2 \cdot u_1 + t_2 \cdot u_2$$



$$(s_1 + s_2 + t_1 + t_2) \cdot (u_1 + u_2)$$

Outline

- Overview & Motivation
- Provenance Graph Model
- **Capturing Provenance**
- Factorization
- Experiments
- Conclusions & Future work

Computing Provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

Problem definition

input: (missing) answers of interest

output: relevant subgraph for all answers of interest

Instrumentation of the input Datalog program

- Use **firing rules** to capture rule derivations
- **Goal-oriented** approach
  - Limit the provenance based on the user question
  - Efficient bottom-up evaluation using relational engines
  - Compiling rewritten program into SQL

Overview

Rewriting steps

- **Unifying** the program with specification of outputs of interest
- **Statically annotate** program to indicate interest in success / failure
- Creating **firing rules**
- Checking **connectivity with joins**
- **Computing** the **edge** relation of the provenance graph

Example query and provenance question

$$r_1 : \texttt{Q}(X,Y) :- \texttt{Train}(X,Z), \texttt{Train}(Z,Y), \neg\texttt{Train}(X,Y)$$

```
WHY Q(n,s)?
```

Computing Provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

Rewriting steps

- **Unifying** the program with specification of outputs of interest
- **Statically annotate** program to indicate interest in success / failure
- Creating **firing rules**
- Checking **connectivity with joins**
- **Computing** the **edge** relation of the provenance graph

Computing Explanations

Step 1: Unifying the program with the question

- **Limit computation** by binding variables to constants
- Propagating constants in the question throughout the program

Example

- Binding variables: X = n (New York) and Y = s (Seattle)

$$r_1 : \mathtt{Q}(X,Y) :- \mathtt{Train}(X,Z), \mathtt{Train}(Z,Y), \neg\mathtt{Train}(X,Y)$$

+

WHY Q(n,s)

$$r_1^{(\underline{X=n},\underline{Y=s})} : \mathtt{Q}(\underline{n},\underline{s}) :- \mathtt{T}(\underline{n},Z), \mathtt{T}(Z,\underline{s}), \neg\,\mathtt{T}(\underline{n},\underline{s})$$

Computing Explanations

Step 2: Annotating rule with success / failure

- **To determine the state of nodes**
- The question provides information about success / failure

True = Success / Exist   False = Failed / Not Exist

- Propagate annotations throughout the unified program

Example

▷ **WHY** Q(n,s) implies only successful goals need to be captured

$$r_1^{(X=n,Y=s)} : \mathtt{Q}(n,s) :- \mathtt{T}(n,Z), \mathtt{T}(Z,s), \neg\, \mathtt{T}(n,s)$$

+

WHY Q(n,s)

⇩

$$r_1^{(X=n,Y=s),T} : \mathtt{Q}(n,s)^T :- \mathtt{T}(n,Z)^T, \mathtt{T}(Z,s)^T, \neg\, \mathtt{T}(n,s)^T$$

Computing Explanations

Step 3: Create firing rules

- **Capture successful / failed derivations** for the variable binding

Example

- Create new head by adding the variable Z and use firing versions
- Invert the annotation **T** for the goal ¬ T(n,s) in the firing version

$$r_1^{(X=n,Y=s),T} : \boxed{\mathtt{Q}(n,s)^T} :- \underline{\mathtt{T}(n,Z)^T}, \mathtt{T}(Z,s)^T, \neg\,\mathtt{T}(n,s)^T \quad \longrightarrow$$

$$\begin{aligned}
\mathrm{F}_{\mathtt{Q},\mathtt{T}}(n,s) &:- \mathrm{F}_{\mathtt{r_1},\mathtt{T}}(n,s,Z)\\
\boxed{\mathrm{F}_{\mathtt{r_1},\mathtt{T}}(n,s,Z)} &:- \underline{\mathrm{F}_{\mathtt{T},\mathtt{T}}(n,Z)}, \mathrm{F}_{\mathtt{T},\mathtt{T}}(Z,s), \mathrm{F}_{\mathtt{T},\mathtt{F}}(n,s)\\
\mathrm{F}_{\mathtt{T},\mathtt{T}}(n,Z) &:- \mathtt{T}(n,Z)\\
\mathrm{F}_{\mathtt{T},\mathtt{T}}(Z,s) &:- \mathtt{T}(Z,s)
\end{aligned}$$

Computing Explanations

Step 4: Connectivity joins

- Firing rules are **not sufficient to determine** which **subgraphs** of the provenance explain the outputs of interest
- Filter derivations by checking whether connectivity
- Check connectivity from the question node one hop at a time

Example

- No guarantees for the nodes in the red box
- Tuple node `T(n,c)` is only connected iff `T(c,s)` exists

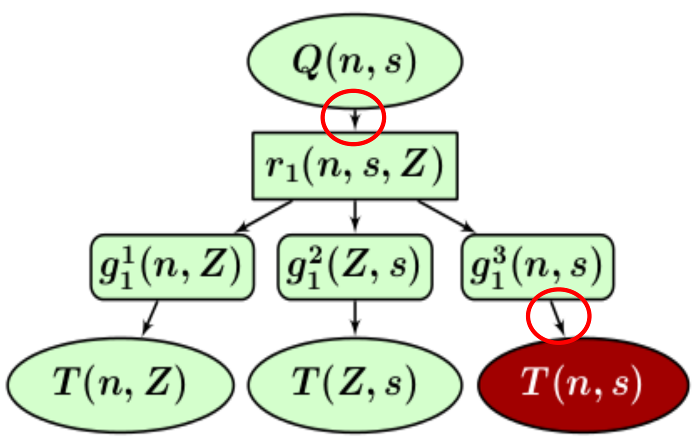
$$
\begin{aligned}
\mathtt{F}_{\mathtt{Q},\mathtt{T}}(n,s) &:- \mathtt{F}_{\mathtt{r}_1,\mathtt{T}}(n,s,Z)\\
\mathtt{F}_{\mathtt{r}_1,\mathtt{T}}(n,s,Z) &:- \mathtt{F}_{\mathtt{T},\mathtt{T}}(n,Z), \mathtt{F}_{\mathtt{T},\mathtt{T}}(Z,s), \mathtt{F}_{\mathtt{T},\mathtt{F}}(n,s)\\
\boxed{\mathtt{F}_{\mathtt{T},\mathtt{T}}(n,Z)} &:- \mathtt{T}(n,Z)\\
\mathtt{F}_{\mathtt{T},\mathtt{T}}(Z,s) &:- \mathtt{T}(Z,s)
\end{aligned}
$$

$$
\begin{aligned}
\mathtt{F}_{\mathtt{Q},\mathtt{T}}(n,s) &:- \mathtt{F}_{\mathtt{r}_1,\mathtt{T}}(n,s,Z)\\
\mathtt{F}_{\mathtt{r}_1,\mathtt{T}}(n,s,Z) &:- \mathtt{F}_{\mathtt{T},\mathtt{T}}(n,Z), \mathtt{F}_{\mathtt{T},\mathtt{T}}(Z,s), \mathtt{F}_{\mathtt{T},\mathtt{F}}(n,s)\\
\boxed{\mathtt{FC}_{\mathtt{r}_2,\mathtt{r}_1^1,\mathtt{T}}(n,Z) :- \mathtt{T}(n,Z), \mathtt{F}_{\mathtt{r}_1,\mathtt{T}}(n,s,Z)}\\
\mathtt{FC}_{\mathtt{r}_2,\mathtt{r}_1^2,\mathtt{T}}(Z,s) &:- \mathtt{T}(Z,s), \mathtt{F}_{\mathtt{r}_1,\mathtt{T}}(n,s,Z)\\
\mathtt{F}_{\mathtt{T},\mathtt{F}}(n,s) &:- \neg\,\mathtt{T}(n,s)
\end{aligned}
$$

Computing Explanations

Step 5: Computing provenance subgraph edge relation

- Create edges for the provenance graph (explanation)
- Generate rules for the edge relation based on the rule binding information
- Use node identifier $f^{T}_{r_1}(n, s, Z)$
- Type of the node, assignments to constants, success/failure state
- Each rule corresponds to a pattern in the graph



- Provenance graph structure -

$$\boxed{\mathtt{edge}(f^{T}_{Q}(n, s), f^{T}_{r_1}(n, s, Z)) :- \mathbf{F}_{r_1,T}(n, s, Z)}$$

$$\mathtt{edge}(f^{T}_{r_1}(n, s, Z), f^{T}_{g_1^1}(n, Z)) :- \mathbf{F}_{r_1,T}(n, s, Z)$$

$$\mathtt{edge}(f^{T}_{g_1^1}(n, Z), f^{T}_{T}(n, Z)) :- \mathbf{F}_{r_1,T}(n, s, Z)$$

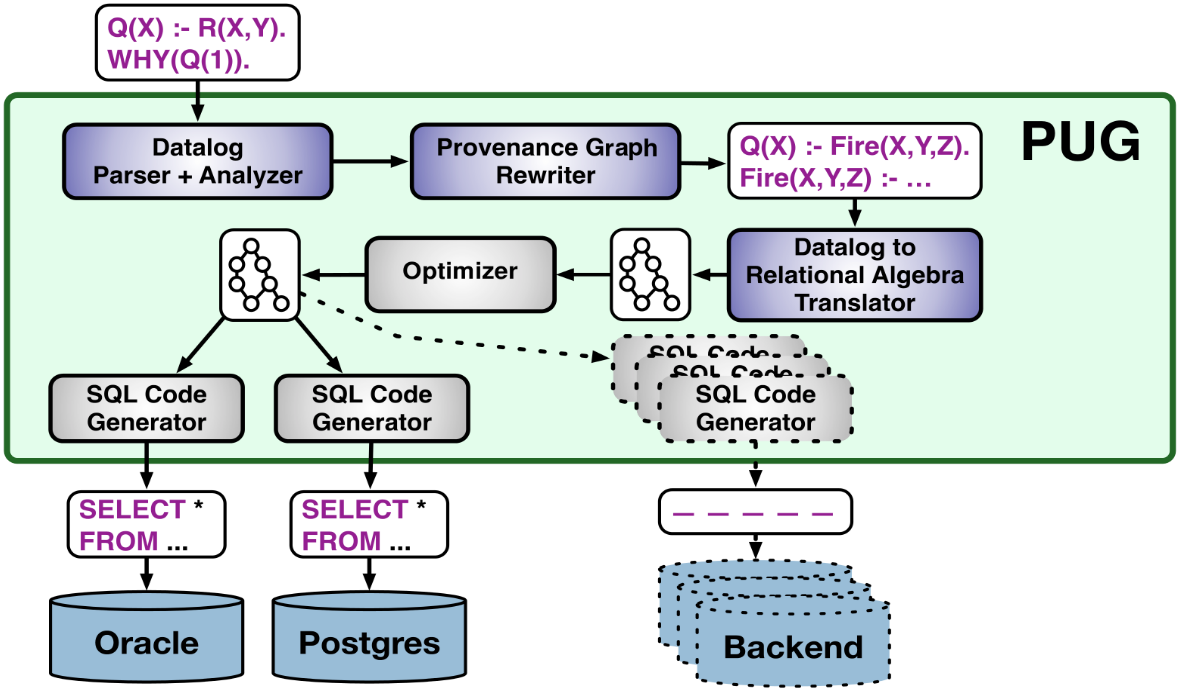
$$\boxed{\mathtt{edge}(f^{T}_{g_1^3}(n, s), f^{F}_{T}(n, s)) :- \mathbf{F}_{r_1,T}(n, s, Z)}$$

Example (partial) rules deriving the edge relation

Implementation

PUG (**P**rovenance **U**nification through **G**raphs) architecture

- Extension of **GProM** supporting Datalog provenance
- GProM is a SQL+X – to – SQL optimizing compiler
- Relational algebra as IR



Outline

- Overview & Motivation
- Provenance Graph Model
- Capturing Provenance
- **Factorization**
- Experiments
- Conclusions & Future work

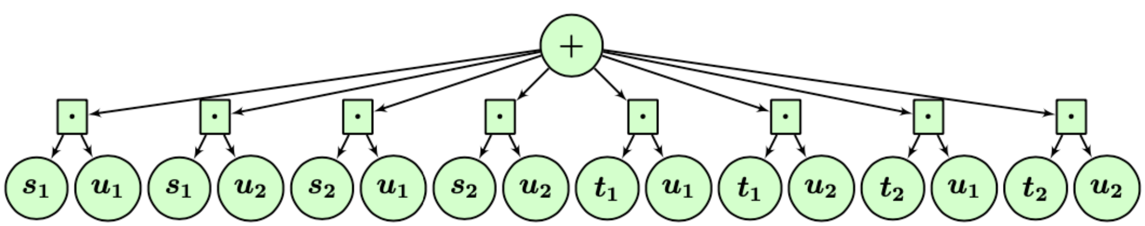
Recap

Our provenance graphs can encode factorized polynomials

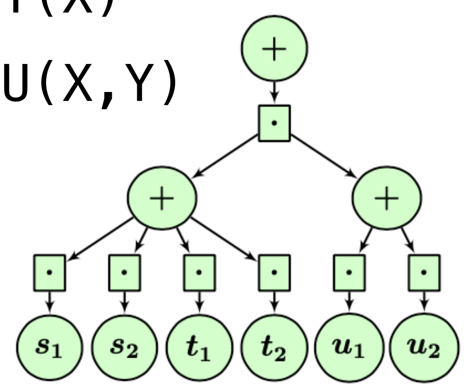
- What factorization we get is determined by the structure of the program

```
Q() :- S(X), U(X,Y)
Q() :- T(X), U(X,Y)
```

```
Q() :- Q2(X), Q1(X)
Q1(X) :- S(X)
Q1(X) :- T(X)
Q2(X) :- U(X,Y)
```



$s_1 \cdot u_1 + s_1 \cdot u_2 + s_2 \cdot u_1 + s_2 \cdot u_2 + t_1 \cdot u_1 + t_1 \cdot u_2 + t_2 \cdot u_1 + t_2 \cdot u_2$



$(s_1 + s_2 + t_1 + t_2) \cdot (u_1 + u_2)$

Utilizing work on factorization

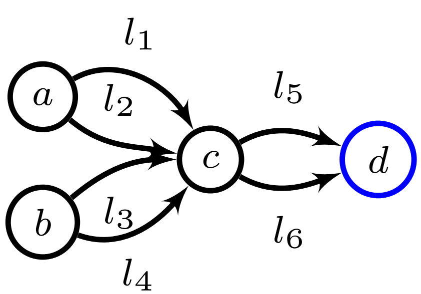
ILLINOIS INSTITUTE OF TECHNOLOGY

Approach

- Determine worst-case optimal d-tree factorization
- Rewrite input query to produce this factorization
- Apply capture rewriting

D. Olteanu and J. Závodny`. Size bounds for factorised repre- sentations of query results. ACM Transactions on Database Systems (TODS), 40(1):2, 2015.

$$r_3 : \mathtt{Q}_{\mathtt{2hop}}(X) :- \mathtt{H}(Y, L_1, Z), \mathtt{H}(Z, L_2, X)$$



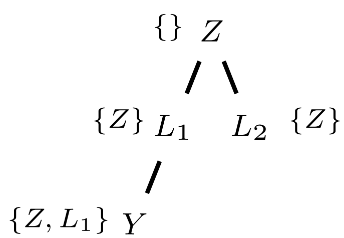
Approach

- Determine worst-case optimal d-tree factorization
- Rewrite input query to produce this factorization
- Apply capture rewriting

$r_5 : \mathtt{Q}_{\mathtt{2hop}}() :- \mathtt{Q}_{\mathtt{L}_1}(Z), \mathtt{Q}_{\mathtt{L}_2}(Z)$

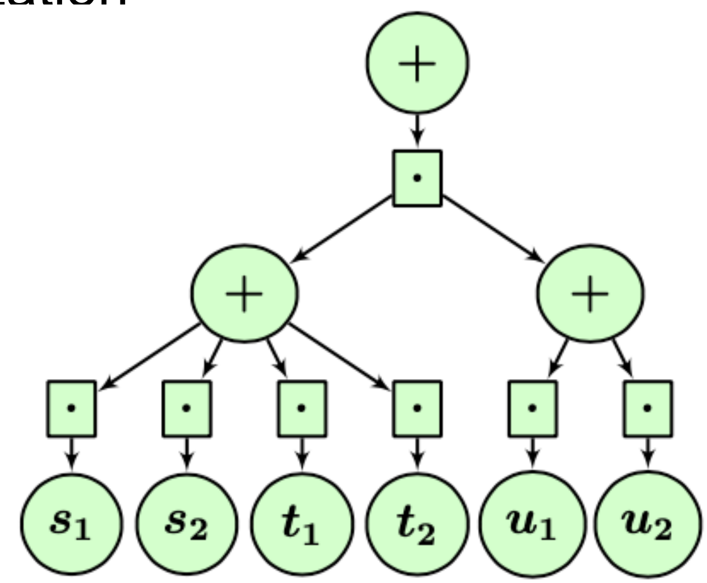
$r_{5'} : \mathtt{Q}_{\mathtt{L}_1}(Z) :- \mathtt{H}(Y, L_1, Z)$

$r_{5''} : \mathtt{Q}_{\mathtt{L}_2}(Z) :- \mathtt{H}(Z, L_2, d)$



Relation $\mathtt{H}$

S	L	E	
a	l_1	c	s_1
a	l_2	c	s_2
b	l_3	c	t_1
b	l_4	c	t_2
c	l_5	d	u_1
c	l_6	d	u_2



$$(s_1 + s_2 + t_1 + t_2) \cdot (u_1 + u_2)$$

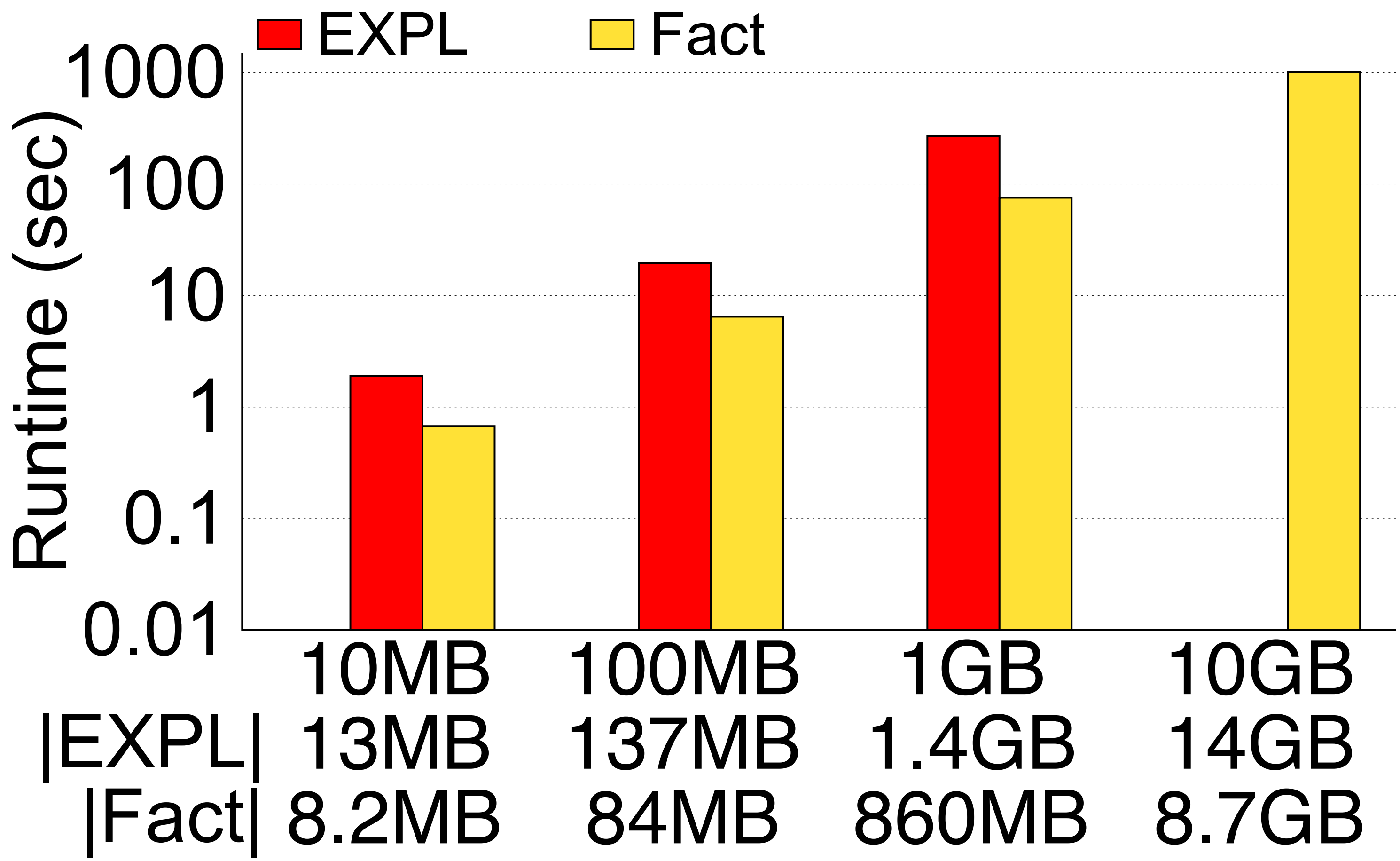
Outline

- Overview & Motivation
- Provenance Graph Model
- Capturing Provenance
- Factorization
- **Experiments**
- Conclusions & Future work

Experiments

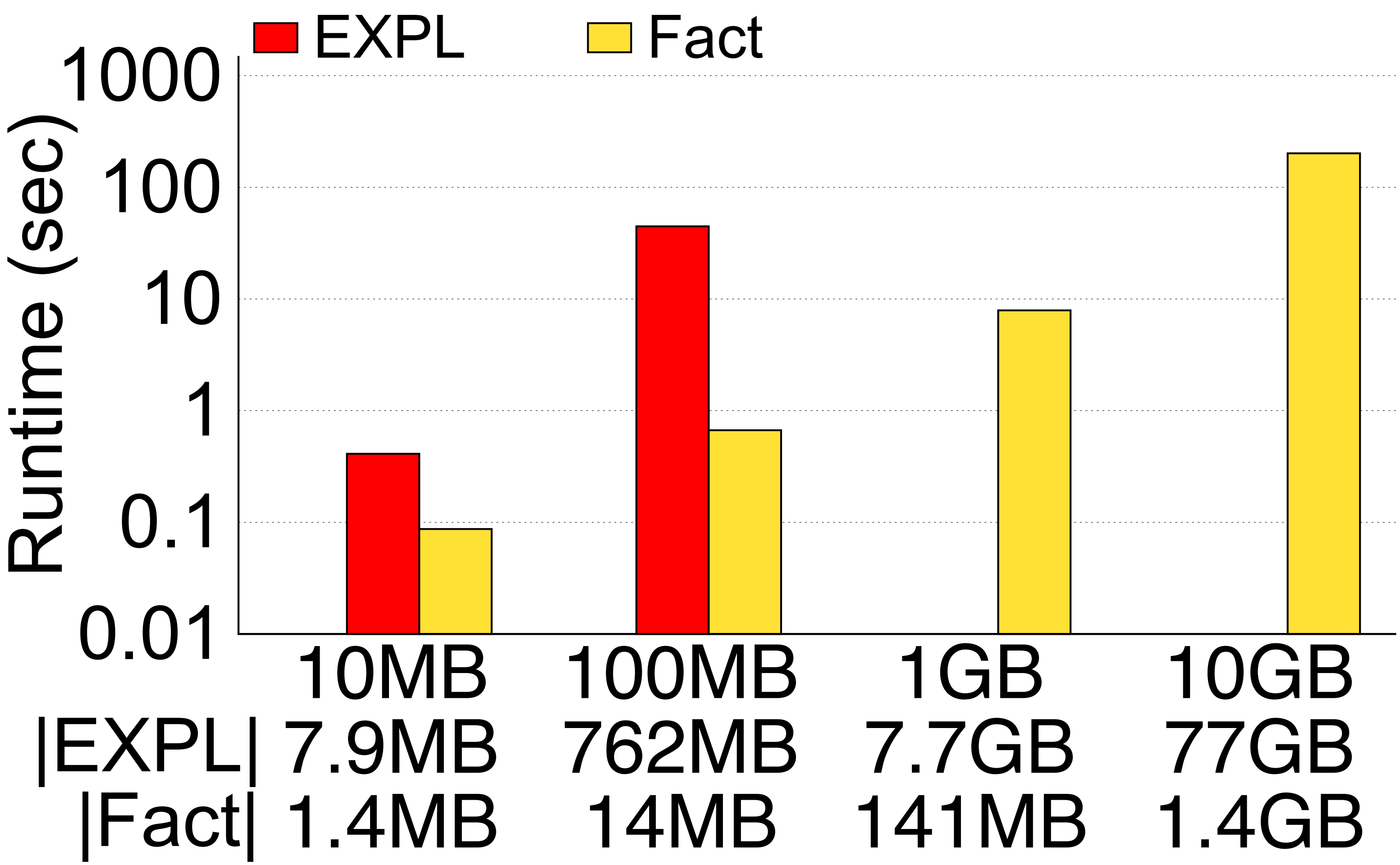
- TPC-H

```
ordDisc(X,Y) :- CUSTOMER(A,X,B,C,D,E,F,G),
                ORDERS(H,A,I,J,K,L,M,O,P),
                LINEITEM(H,Q,R,S,T,U,V,Y,W,Z,A′,B′,C′,D′,E′,F′)
```



Experiments

- TPC-H `suppCust(N):-SUPPLIER(A,B,C,N,D,E,F), CUSTOMER(G,H,I,N,J,K,L,M)`



Outline

- Overview & Motivation
- Provenance Graph Model
- Capturing Provenance
- Factorization
- Experiments
- **Conclusions & Future work**

Conclusions

- **Provenance graph model**
  - Provenance structure aligns with program structure
  - Compatible with well-established provenance models
  - Provenance polynomials for positive queries + dual polynomials
  - Build-in support for sharing common subexpressions
  - Flat relational encoding as edge relation
- **Capturing Provenance**
  - Incorporate user’s provenance interest into the capture query
  - Filter successful / failed assignments upfront (static analysis)
  - Output is a query returning the edge relation of the graph

Conclusions

- **Factorization**
  - Program structure determines factorization
- **Approximate Summarization of Why-not Provenance**
  - Use patterns to summarize provenance
  - Use sampling to generate such summaries for very large why-not provenance graphs

Future work

- **Expressiveness**
  - Support aggregation and recursion
- **Efficiency**
  - Leverage factorized DB techniques for aggregates?
  - Factorizing missing answers (complement representations)?
- **Going beyond SQL / Datalog as the target language**
  - What specialized algorithms & data structures would be beneficial?

Questions?

ILLINOIS INSTITUTE OF TECHNOLOGY

DBGroup

Seokki Lee

Assistant Professor (University of Cincinnati)

CEAS - Elect Eng & Computer Science

[Contact Information]

Office: Old Chemistry Building 613B

Email: [lee5sk] at [ucmail] o [uc] o [edu]

Phone: 513-556-5795

Homepage: https://researchdirectory.uc.edu/p/lee5sk

Grouppage: https://sites.google.com/view/ucdbg

PUG: https://github.com/IITDBGroup/PUG

Summarizing Provenance

Motivation (Example)

ILLINOIS INSTITUTE OF TECHNOLOGY

- Airbnb (bed and breakfast)

Listing (input)

Id	Name	Ptype	Rtype	NGroup	Neighbor
8403	central place	apt	shared	queen anne	east
9211	plum	apt	entire	ballard	adams
2445	cozy homebase	house	private	queen anne	west
8575	near SpaceNeedle	apt	shared	queen anne	lower
4947	seattle couch	condo	shared	downtown	first hill
2332	modern view	house	entire	queen anne	west

Availability (input)

Id	Date	Price
9211	2016-11-09	130
2445	2016-11-09	45
2332	2016-11-09	350
4947	2016-11-10	40

r_1: AL(N,R) :- L(I, N, T, R,queen anne, E), A(I, 2016-11-09, P)

" What are avaliable listings and the room types in Queen Anne on Nov 9th, 2016? "

AvailableListings (output)

Name	Rtype
cozy homebase	private
modern view	entire

Why no shared room exists?

Motivation (Example)

Why-not provenance
(instance-based)

All derivations

Why-not [Huang et al. VLDB 2008]
Artemis [Herschel et al. VLDB 2009]

Not scalable

Single derivation

Y! [Wu et al. SIGCOMM 2014]

Not comprehensive

Motivation (Example)

ILLINOIS INSTITUTE OF TECHNOLOGY

Listing (input)

Id	Name	Ptype	Rtype	NGroup	Neighbor
8403	central place	apt	shared	queen anne	east
9211	plum	apt	entire	ballard	adams
2445	cozy homebase	house	private	queen anne	west
8575	near SpaceNeedle	apt	shared	queen anne	lower
4947	seattle couch	condo	shared	downtown	first hill
2332	modern view	house	entire	queen anne	west

Availability (input)

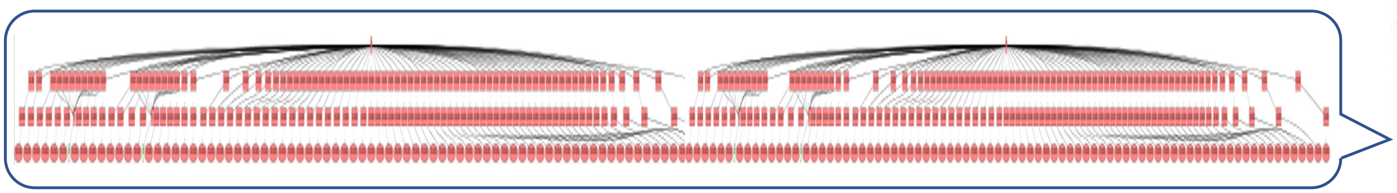
Id	Date	Price
9211	2016-11-09	130
2445	2016-11-09	45
2332	2016-11-09	350
4947	2016-11-10	40

Why no shared room exists?

r_1: AL(N,R) :- L(I, N, T, R, queen anne, E), A(I, 2016-11-09, P)

Attribute	Id	Name	Ptype	Rtype	NGroup	Neighbor	Date	Price
#Distinct Values	6	6	3	3	3	5	2	4

= 2160



~$15 \cdot 10^{20}$ derivations over full dataset (~ 1.4M)

Motivation (Example)

Listing (input)

Id	Name	Ptype	Rtype	NGroup	Neighbor
8403	central place	apt	shared	queen anne	east

Availability (input)

Id	Date	Price

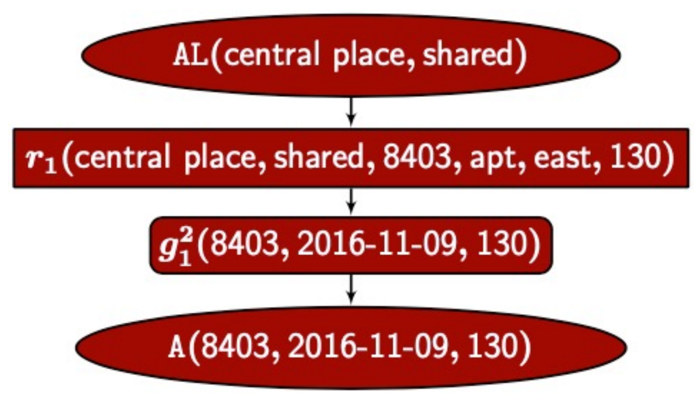
r_1: AL(N,R) :- L(I, N, T, R,queen anne, E), A(I, 2016-11-09, P)

Available**L**istings (output)

Name	Rtype
cozy homebase	private
modern view	entire

Why no shared room exists?

The listing ‘central place’ has a shared room which is not available at $130



Motivation (Example)

Listing (input)

Id	Name	Ptype	Rtype	NGroup	Neighbor
8403	central place	apt	shared	queen anne	east
9211	plum	apt	entire	ballard	adams
2445	cozy homebase	house	private	queen anne	west
8575	near SpaceNeedle	apt	shared	queen anne	lower
4947	seattle couch	condo	shared	downtown	first hill
2332	modern view	house	entire	queen anne	west

Availability (input)

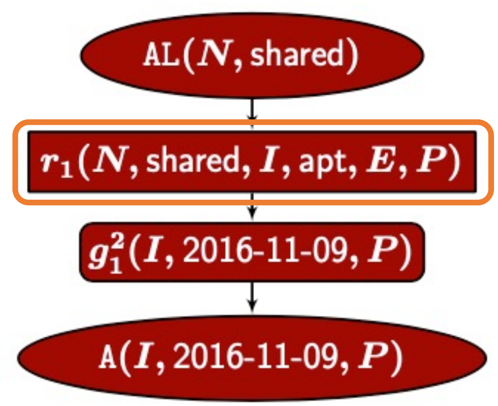
Id	Date	Price
9211	2016-11-09	130
2445	2016-11-09	45
2332	2016-11-09	350
4947	2016-11-10	40

r_1: AL(N,R) :- L(I, N, T, R,queen anne, E), A(I, 2016-11-09, P)

AvailableListings (output)

Name	Rtype
cozy homebase	private
modern view	entire

" All shared rooms of apartments in Queen Anne are not available at any price on Nov 9th, 2016 "



Summarizing Provenance

- Goals
  - **Concise** (small size of explanations)
  - **Complete** (covering all provenance)
  - **Informative** (providing new insights)

- Challenge
  - Fullfilling all 3 elements at the same time
  - Computing summaries using full why-not provenance

Summarizing Provenance

- Computing top-*k* summaries using patterns
  - Concise explanations
  - Meaningful (semantically)

- Integrating sampling into provenance capture process
  - Unbiased
  - Computing representative patterns
  - Calculating close enough approximate completeness of patterns

Summarizing Provenance

- What are patterns?

r_1(N, shared, I, apt, E, P) (T,F)

Summarizing Provenance

- What are patterns?

r_1(N, shared, I, apt, E, P) (T,F)

r_1(central place, shared, 8403, apt, east, 130) (T,F)

r_1(near SpaceNeedle, shared, 8575, apt, lower, 40) (T,F)

r_1(central place, shared, 8403, apt, east, 40) (T,F)

r_1(near SpaceNeedle, shared, 8575, apt, lower, 350) (T,F)

r_1(central place, shared, 8403, apt, east, 350) (T,F)

r_1(central place, shared, 8403, apt, east, 45) (T,F)

.....

Summarizing Provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

- What are patterns?

r_1(N, shared, I, apt, E, P) (T,F)

r_1(central place, shared, 8403, apt, east, 130) (T,F)

r_1(near SpaceNeedle, shared, 8575, apt, lower, 40) (T,F)

r_1(central place, shared, 8403, apt, east, 40) (T,F)

r_1(near SpaceNeedle, shared, 8575, apt, lower, 350) (T,F)

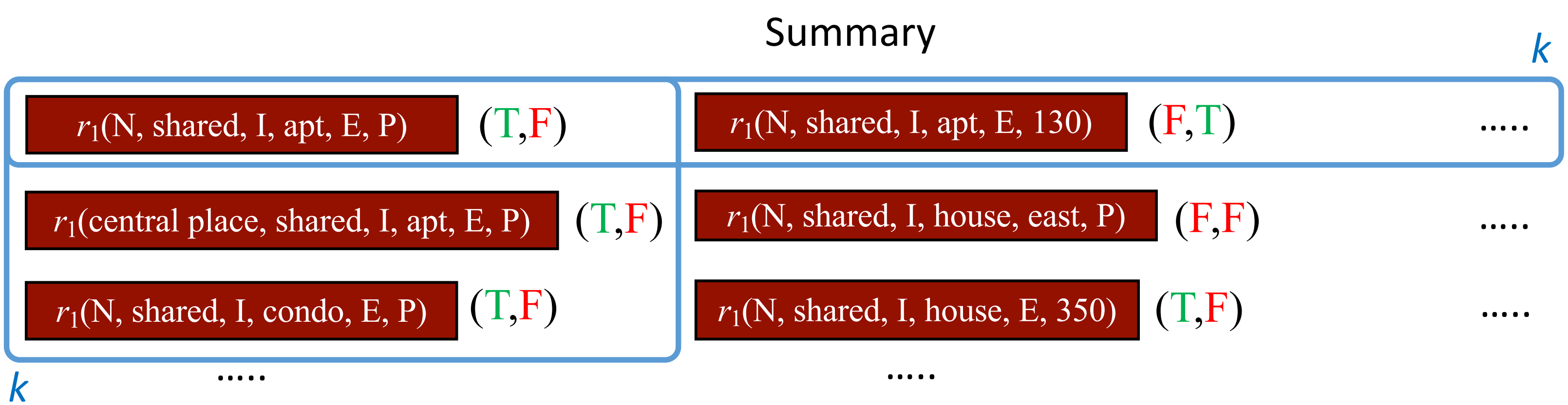
r_1(central place, shared, 8403, apt, east, 350) (T,F)

r_1(central place, shared, 8403, apt, east, 45) (T,F)

.....

Summarizing Provenance

- What are provenance summaries?



Summarizing Provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

- Quality metrics
  - Completeness (cp): fraction of provenance covered by a pattern

r_1(central place, shared, 8403, apt, east, 130) (T,F) $\preccurlyeq$ r_1(N, shared, I, apt, E, P) (T,F)

Summarizing Provenance

- Quality metrics
  - Completeness (cp): fraction of provenance covered by a pattern
  - Informativeness (info): degree of new information from a pattern

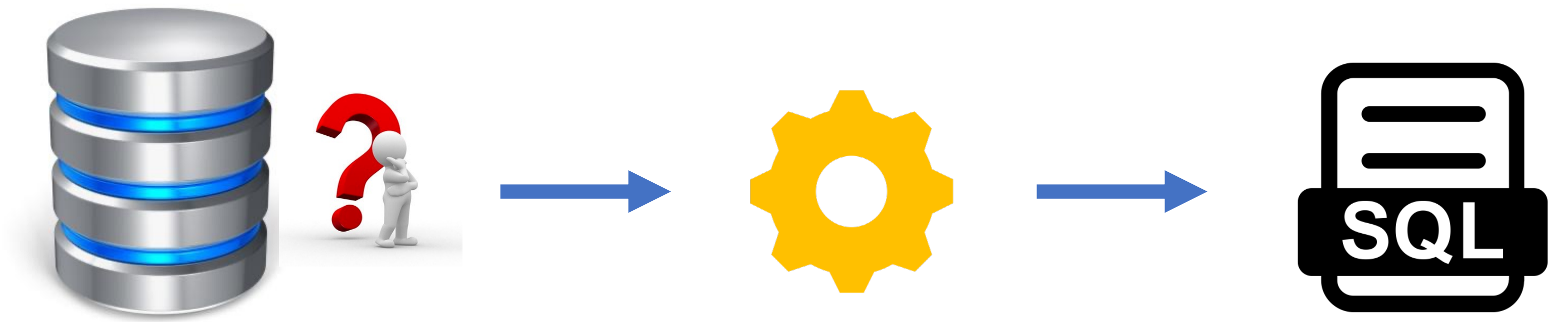
Why no shared room exists?

r_1(N, shared, I, apt, E, P) (T,F) > r_1(N, shared, I, R, E, P) (T,F)

Summarizing Provenance

- How to compute summaries
  - Heuristic using a sample of why-not provenance

Size of the summary (k)
Q(A) :- R(A,B), ¬ S(B)



Input

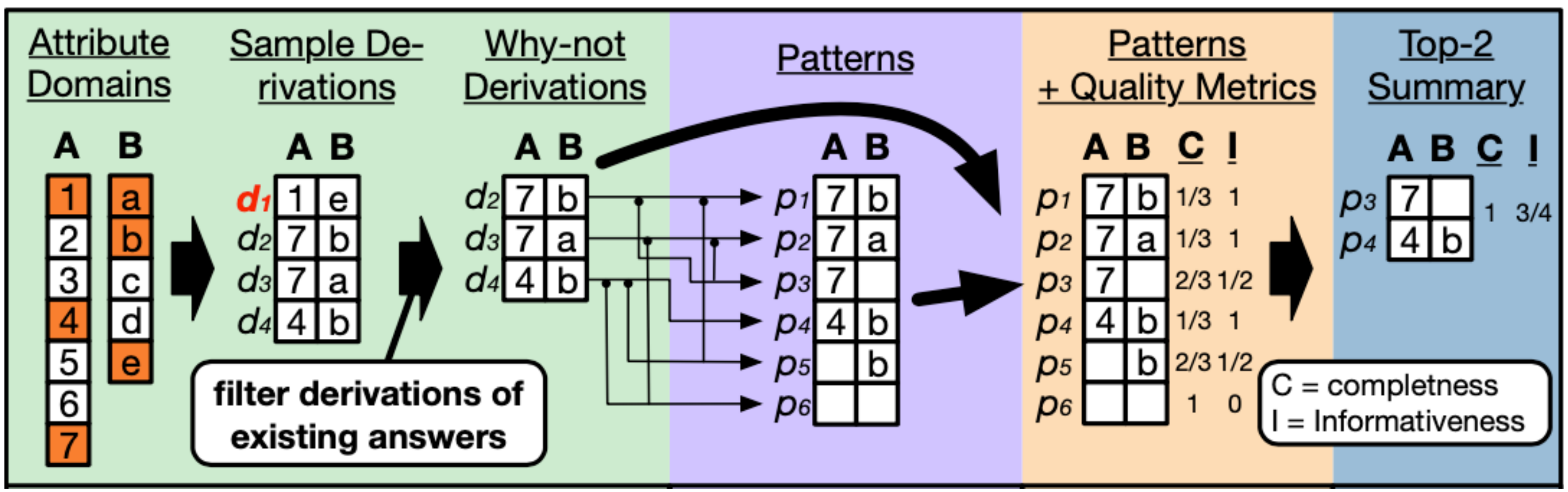
Summarization process

Provenance summary

Summarizing Provenance

- How to compute summaries

Summarization process



Sampling Why-not Provenance | Generating Patterns | Measuring Quality | Selecting top-*k*

Summarizing Provenance

- Sampling why-not provenance
  - Generating a sample that is equivalent to uniform random sample
    - Without computing full why-not provenance
  - Batch sampling: generating a query that returns an unbiased sample

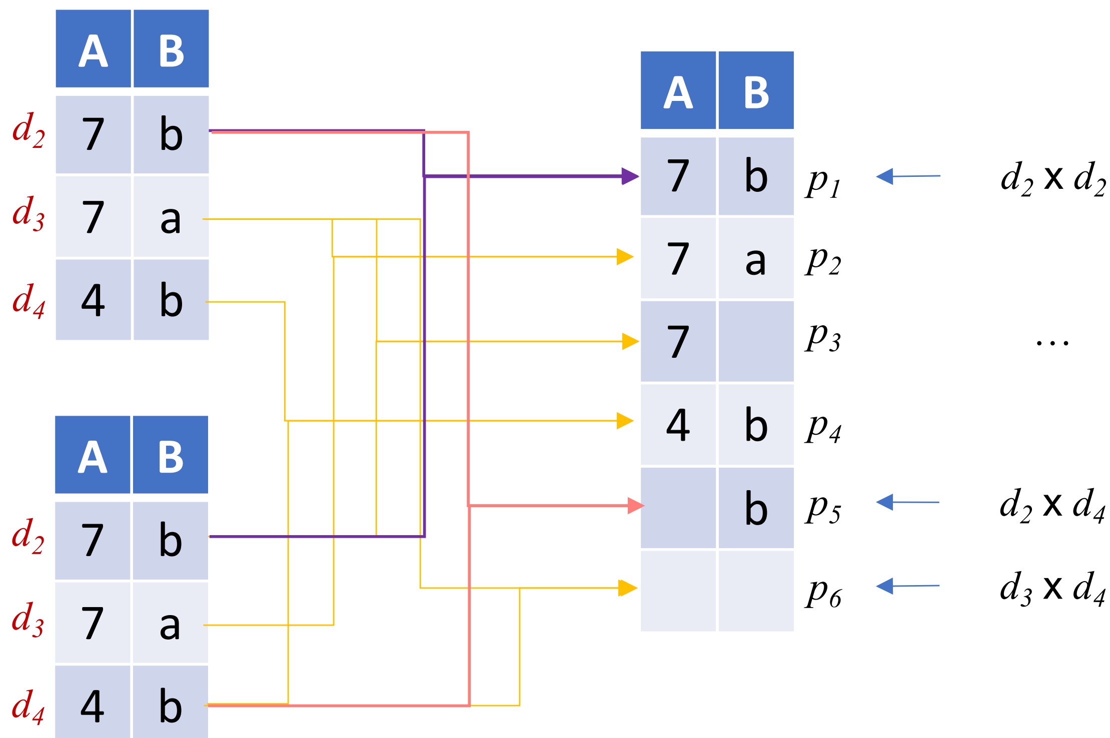
A	B
1	a
2	b
3	c
4	d
5	e
6	
7	

	A	B
d_1	1	e
d_2	7	b
d_3	7	a
d_4	4	b

	A	B
d_2	7	b
d_3	7	a
d_4	4	b

Summarizing Provenance

- Generating patterns
  - LCA (Lowest Common Ancestor)



Interpretable and Informative Explanations of Outcomes [Gebaly et al. PVLDB 2014]

Summarizing Provenance

- Measuring quality and selecting top-k patterns

	A	B
d_2	7	b
d_3	7	a
d_4	4	b

	A	B	C	I
p_1	7	b	1/3	1
p_2	7	a	1/3	1
p_3	7		2/3	1/2
p_4	4	b	1/3	1
p_5		b	2/3	1/2
p_6			1	0

S1 (disjointness)

	A	B
p_3	7	
p_4	4	b

C	I
$1 \leq S1 \leq 1$	3/4

S2 (generalization)

	A	B
p_1	7	b
p_5		b

C	I
$2/3 \leq S1 \leq 2/3$	3/4

C: completeness
I: Informativeness

Experiments

ILLINOIS INSTITUTE OF TECHNOLOGY

Experiments

- Performance of computing summaries
- Quality of summaries
- Comparison with other approaches

- Datasets
  - 4 real-world datasets
  - TPCH

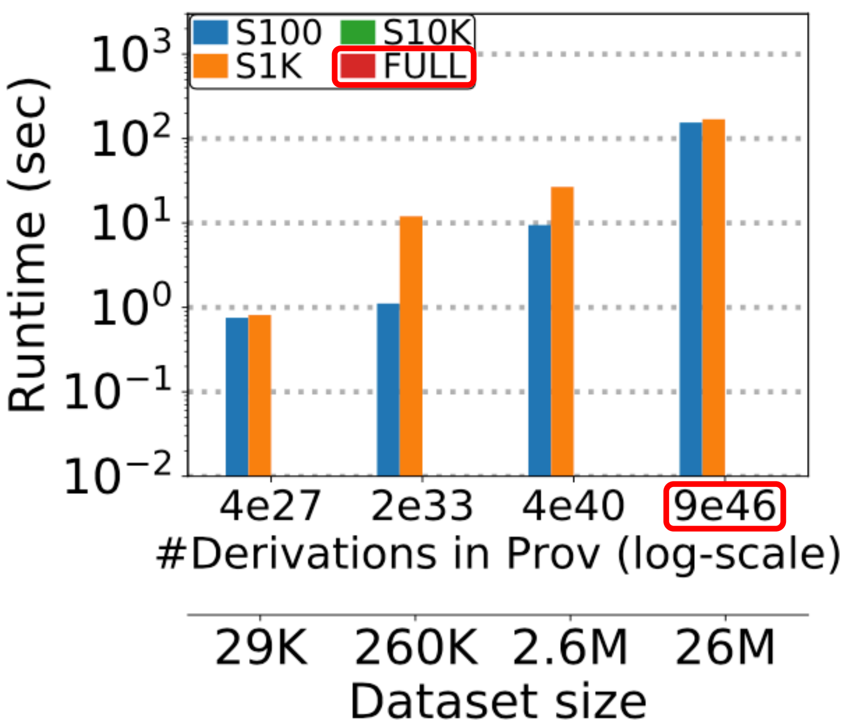
- Queries
  - Single rule through multiple rules
  - Negation and comparisons

Approximate summaries for why and why-not provenance (extended version) [Lee et al. https://arxiv.org/abs/2002.00084 2020]

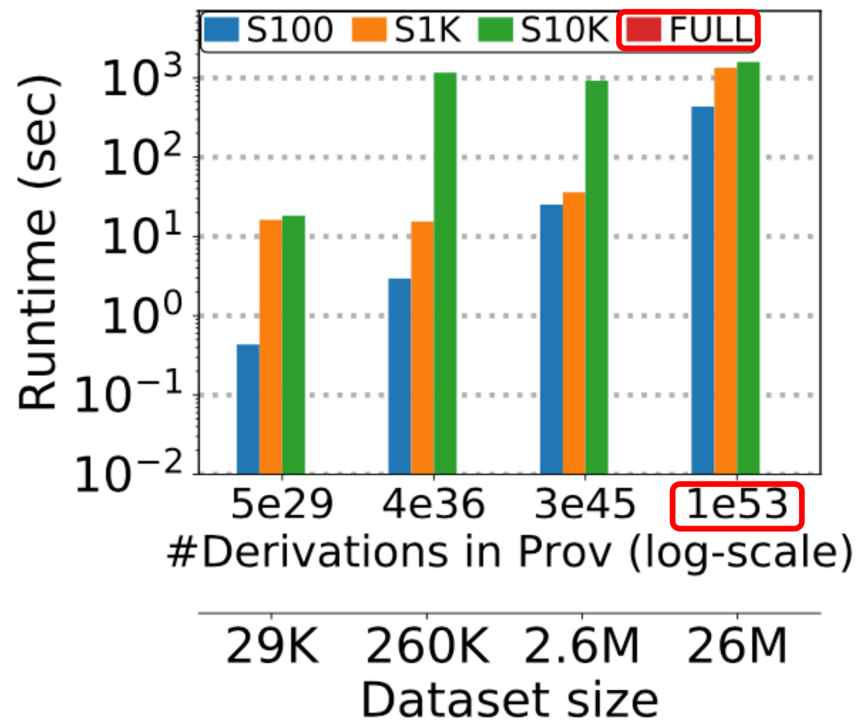
Experiments

- Computing summaries for ~10^{50} derivations

12 variables + 6 goals + negation



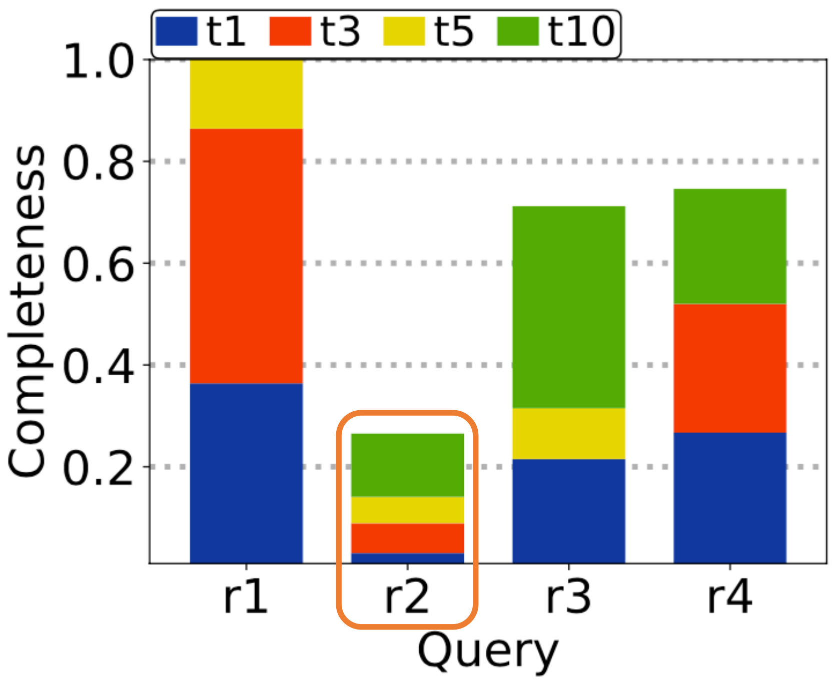
13 variables + 4 goals + multiple rules



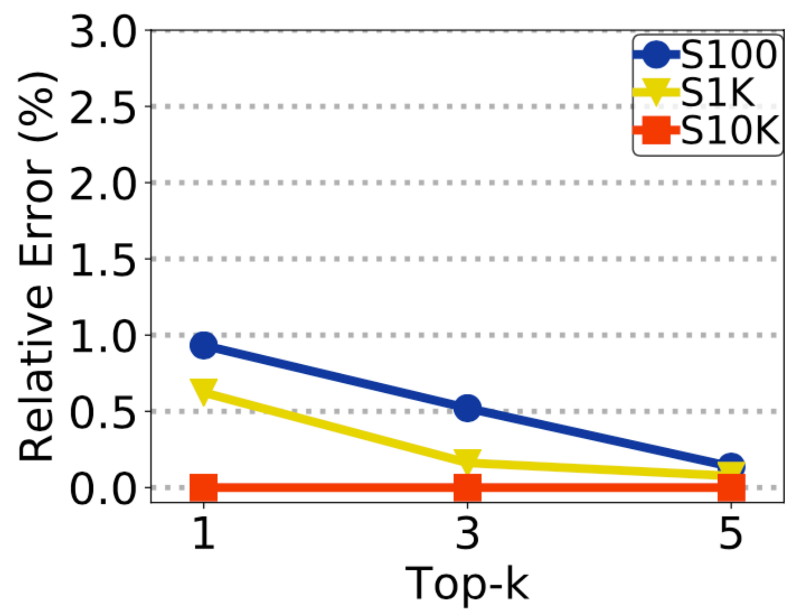
Experiments

ILLINOIS INSTITUTE OF TECHNOLOGY

- Generating high-quality summaries



[Completeness comparison]



[Quality metric error caused by sampling]