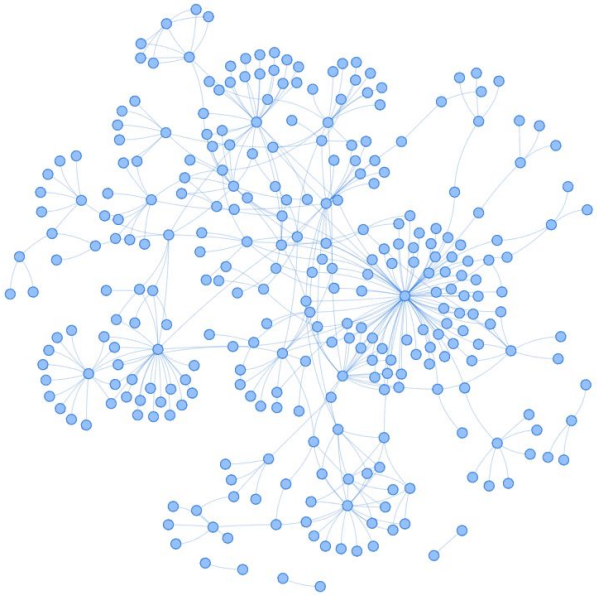
Explainable AI: Rule-aware Datalog Fact Explanation Using Group-SAT Solver

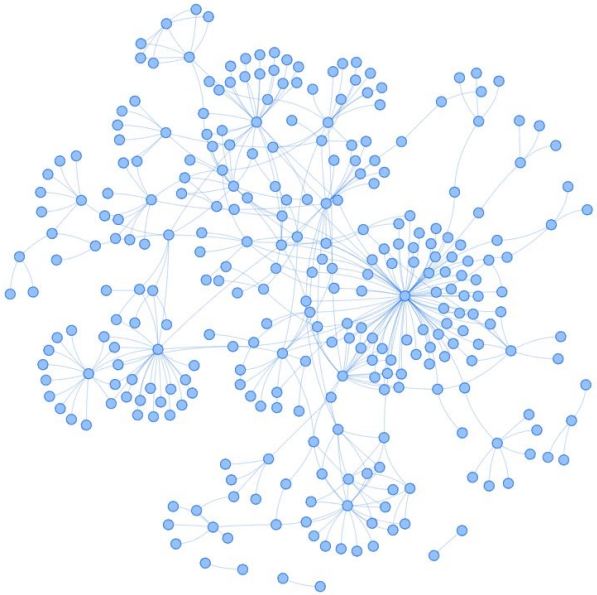
Akira Charoensit, David Carral, Pierre Bisquert, Federico Ulliana
Inria, LIRMM, Univ Montpellier, CNRS, France
IATE, Univ Montpellier, INRAE, Institut Agro, Montpellier, France

Introduction: making sense of data



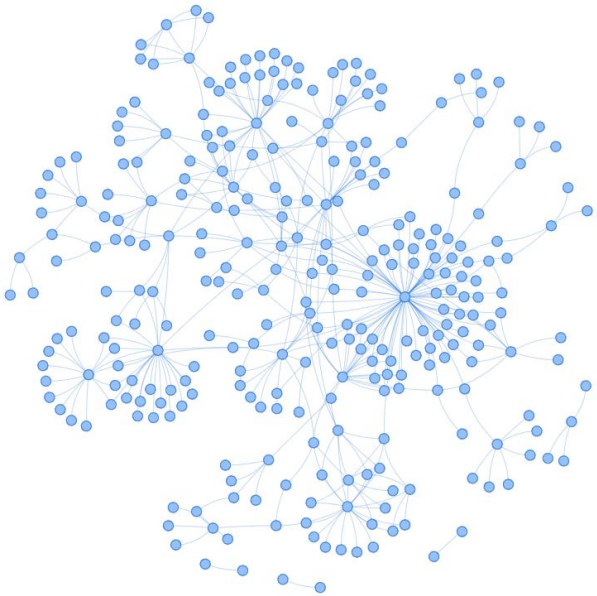
?

Introduction: making sense of data



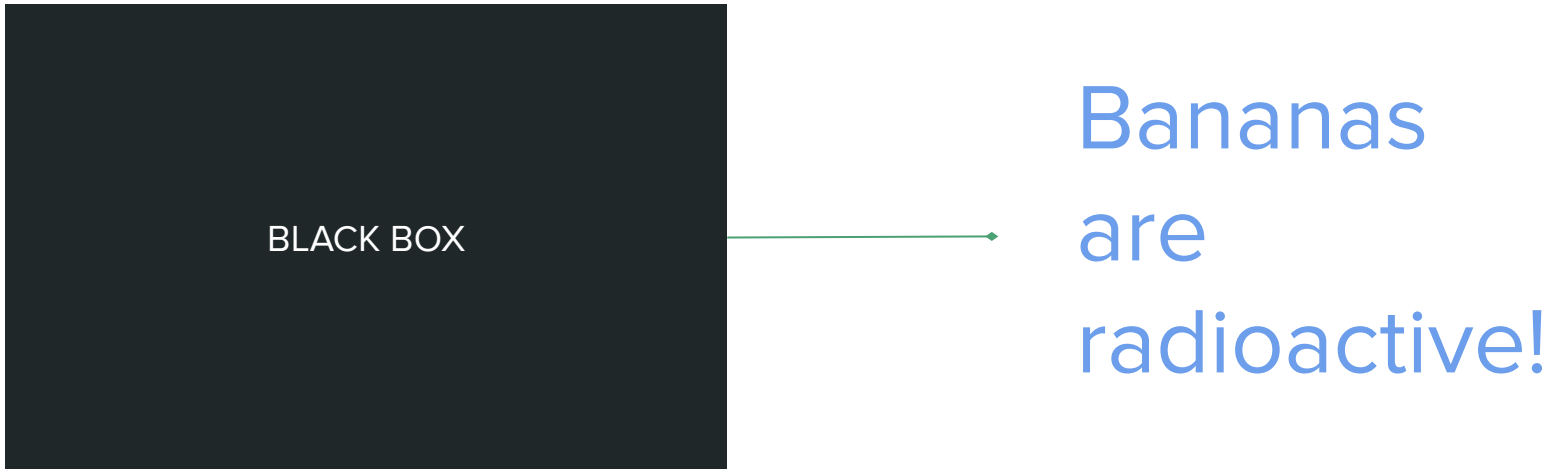
? radioactive

Introduction: making sense of data

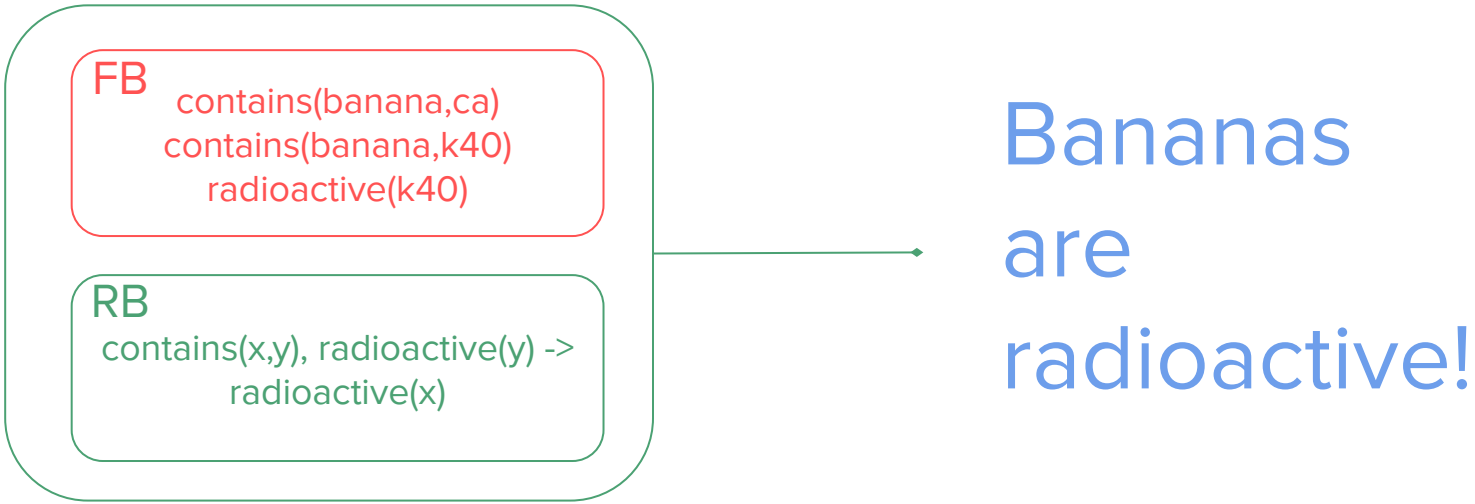


Bananas are radioactive!

Introduction: making sense of data



Introduction: making sense of data



Research Questions

1. **What is a useful form of explanations?**

2. **What is an efficient way to compute explanations?**

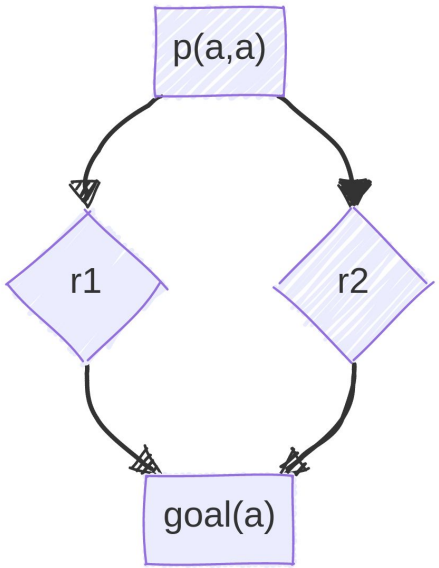
What is a ‘good’ explanation: fact-support vs KB-support

FB

p(a,a)

RB

r1: p(x,x) -> goal(x)
r2: p(x,y) -> goal(x)



Explaining goal(a)

2 possible forms of explanations:

1. Fact-support (why-provenance)

1 explanation: p(a,a)

2. KB-support

2 explanations:

p(a,a), r1
p(a,a), r2

Research Questions

1. **What is a useful form of explanations?**

   -> An explanation of a fact *p* with regard to KB K is a ***minimal*** subset of K that entails *p*.

2. **What is an efficient way to compute KB-support explanations?**

Research Questions

1. **What is a useful form of explanations?**

   -> An explanation of a fact *p* is a ***minimal*** subset of KB that entails *p*.

2. **What is an efficient way to compute KB-support explanations?**

   -> Hint: Explanations resemble MUS in SAT formula

Computing explanations: SAT Problem

Propositional formula:

$$p \wedge (p \rightarrow q) \wedge (\text{NOT } q) \wedge s \wedge (q \vee \text{NOT } s) \wedge t$$

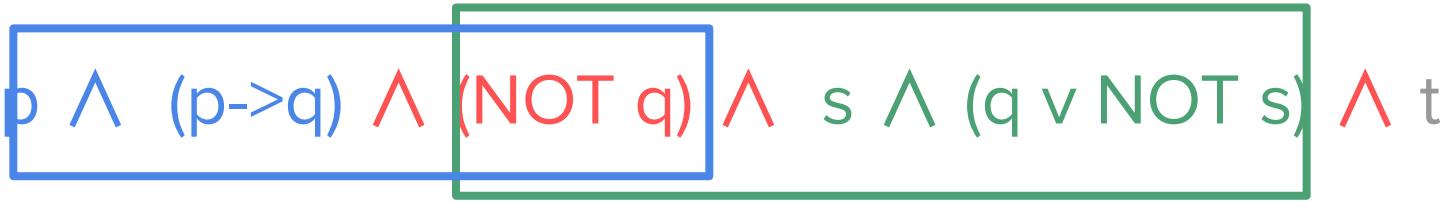
Computing explanations: MUS

MUS: minimal unsatisfiable subsets of clauses

$p \wedge (p \rightarrow q) \wedge (\neg q) \wedge s \wedge (q \vee \neg s) \wedge t$

Computing explanations: MUS

MUS: minimal unsatisfiable subsets of clauses



Research Questions

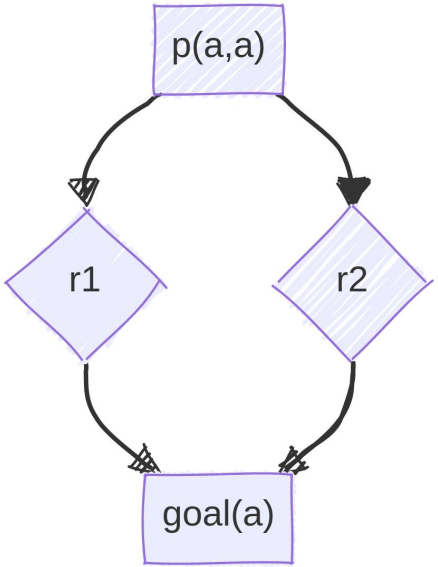
1. **What is a useful form of explanations?**

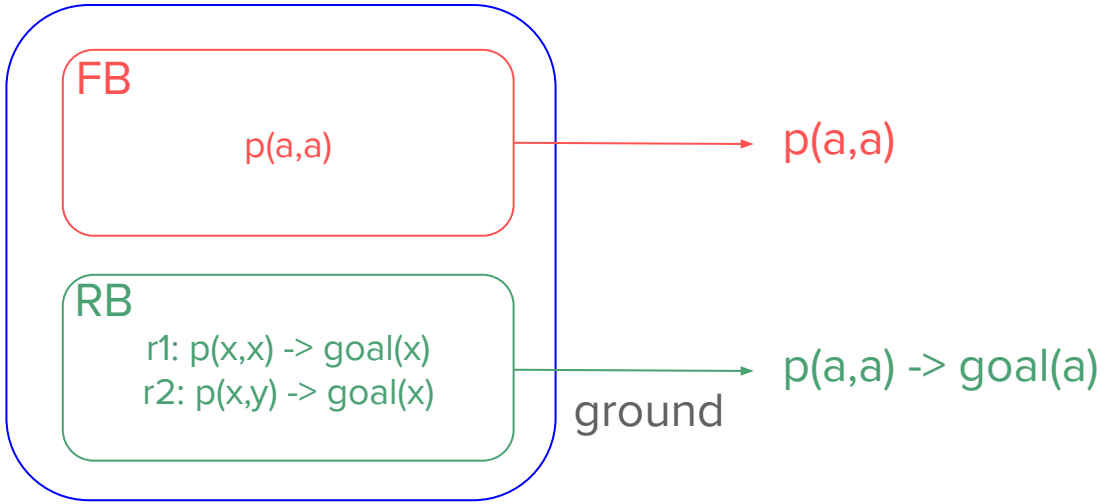
   -> An explanation of a fact *p* is a ***minimal*** subset of KB that entails *p*.

2. **What is an efficient way to compute explanations?**

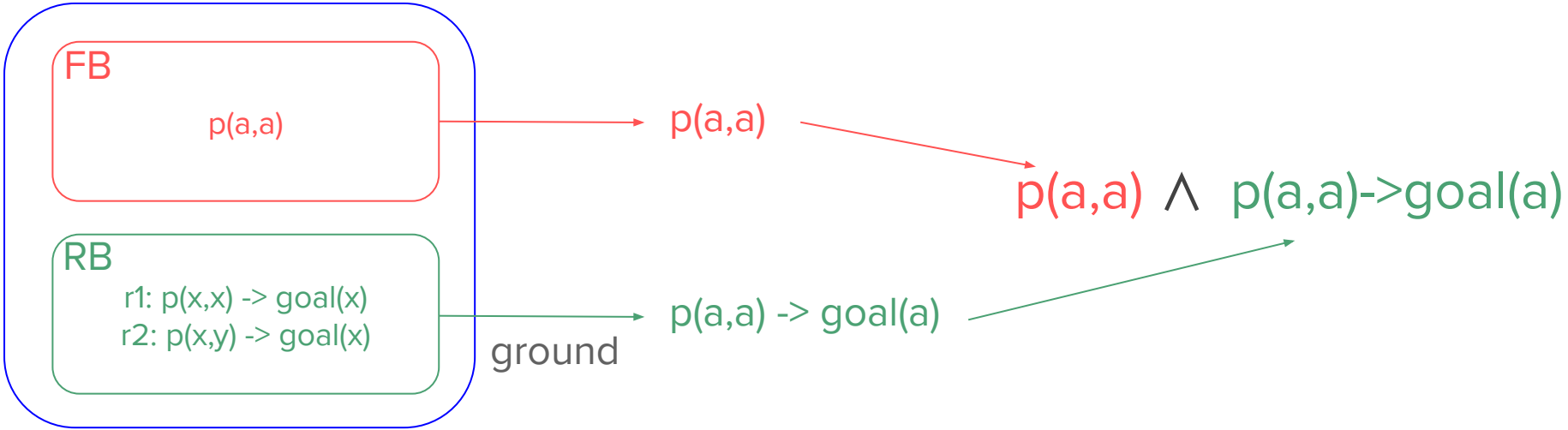
   -> Hint: Explanations resemble MUS in SAT formula

Building a SAT formula from a KB



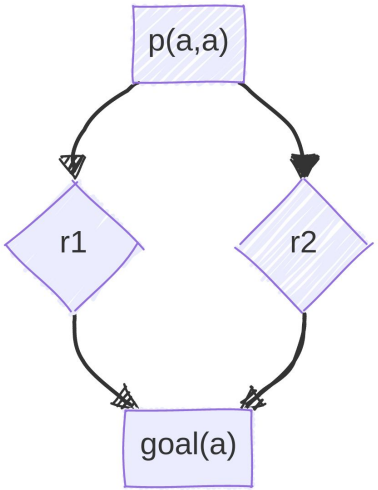


Building a SAT formula from a KB



Finally: reduction to MUS

r1: p(x,**x**) -> goal(x)
r2: p(x,**y**) -> goal(x)

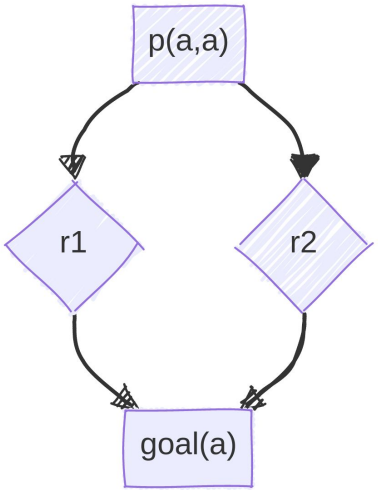


Explaining goal(a)

NOT goal(a)
∧ p(a,a)
∧ p(a,a)->goal(a)

Computing explanations: reduction to MUS

r1: p(x,x) -> goal(x)
r2: p(x,y) -> goal(x)



Example 1

Encoding

NOT goal(a)
p(a,a)
p(a,a)->goal(a)

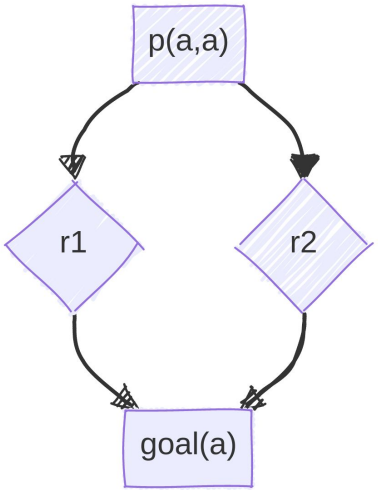
MUS Solver

MUS

NOT goal(a)
p(a,a)
p(a,a)->goal(a)

Computing explanations: reduction to (standard) MUS

r1: p(x,x) -> goal(x)
r2: p(x,y) -> goal(x)



Example 1

Encoding

NOT goal(a)
p(a,a)
p(a,a)->goal(a)

MUS Solver

MUS

NOT goal(a)
p(a,a)
p(a,a)->goal(a)

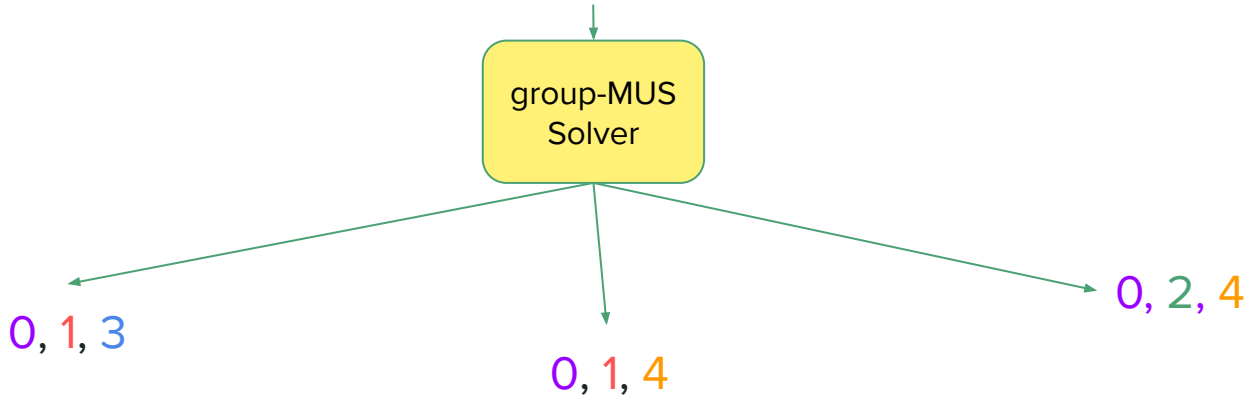
Issue with standard MUS:
1 MUS for 2 explanations

2, 4, 5 Computing explanations: group MUS

Group MUS: minimal unsatisfiable subsets of *groups of* clauses

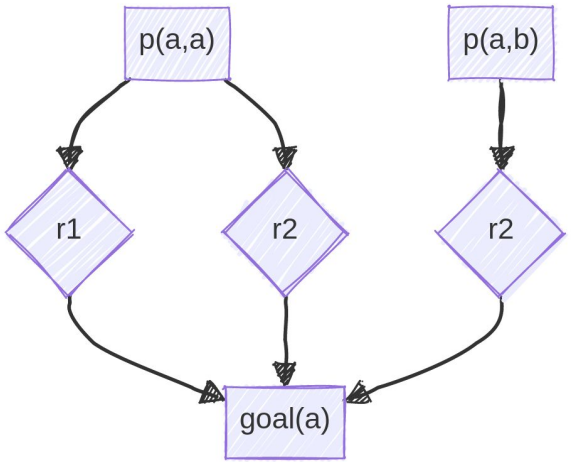
0 1 2 3 4

[NOT g(a)] ∧ [p(a,a)] ∧ [p(a,b)] ∧ [p(a,a)->g(a)] ∧ [p(a,a)->g(a),p(a,b)->g(a)]



Computing explanations: reduction to **group**-MUS

r1: p(x,**x**) -> goal(x)
r2: p(x,**y**) -> goal(x)



Example 1 (modified)

0 NOT goal(a)

1 p(a,a)

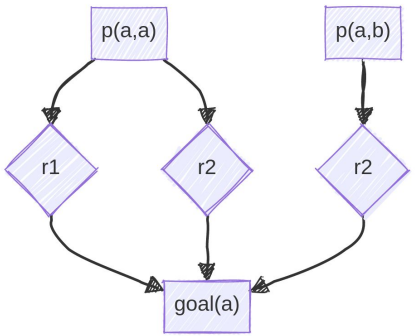
2 p(a,b)

3 p(a,a)->goal(a) (group for r1)

4 p(a,a)->goal(a)
p(a,b)->goal(a) (group for r2)

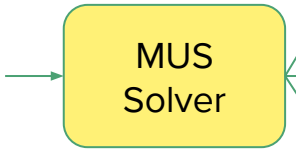
Computing explanations: reduction to **group**-MUS

r1: p(x,**x**) -> goal(x)
r2: p(x,**y**) -> goal(x)



Encoded KB

0 NOT goal(a)
1 p(a,a)
2 p(a,b)
3 p(a,a)->goal(a)
4 p(a,a)->goal(a)
4 p(a,b)->goal(a)



Group-MUS Explanation

0, 1, 3 => FB: p(a,a) RB: r1

0, 1, 4 => FB: p(a,a) RB: r2

0, 2, 4 => FB: p(a,b) RB: r2

Group MUS:
3 MUS for 3 explanations

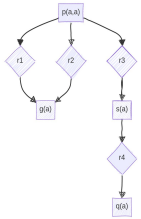
Filtering: MUS solving is expensive

Computing explanation via MUS can be expensive

1. Complexity of group-MUS enumeration
2. KB can be very large

Filtering: MUS solving is expensive

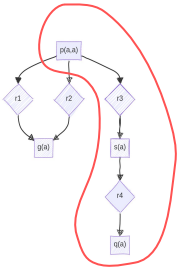
Goal: Reduce the size of the encoded formula as much as possible while preserving soundness & completeness -> Find the relevant subset of the KB



2 steps:

1. Static step
   - entailment graph building

2. Dynamic step
   - fact relevance tracing



Static Step: Entailment Graph Building

Desiderata: a data structure to trace the lineage of an atom

Solution: use a rule-based encoding to build this data structure

Static Step: Entailment Graph Building

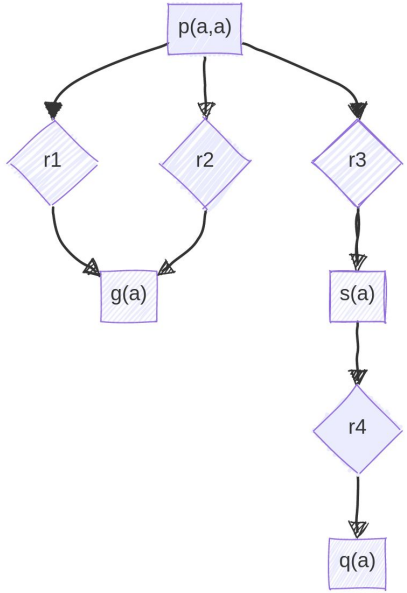
Desiderata: a data structure to trace the lineage of an atom

FB

p(a,a)

RB

r1: p(x,x) -> g(x)
r2: p(x,y) -> g(x)
r3: p(x,x) -> s(x)
r4: s(x) -> q(x)



.dlgp graph

```
#FB
E_r1(f_p(a,a),f_g(a))
E_r2(f_p(a,a),f_g(a))
E_r3(f_p(a,a),f_s(a))
E_r4(f_s(a),f_q(a))

#RB
[REL(Y) ^ E_r1(X1,Y) ->
REL(X1) ^ REL_E_r1(X1,Y),
...
REL(Y) ^ E_r4(X1,Y) ->
REL(X1) ^ REL_E_r4(X1,Y)]
```

Solution: use a rule-based encoding to build this data structure

Dynamic Step: Fact Tracing

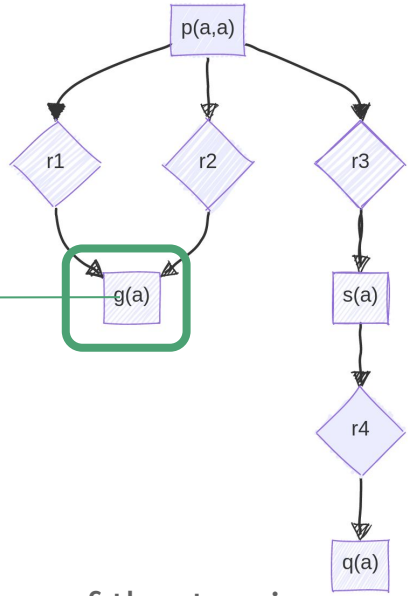
Desiderata: trace the FB-ancestors of the query atom on the graph

Solution: rule-based encoding of the tracing

Dynamic Step: Fact Tracing

Desiderata: trace the FB-ancestors of the query atom on the graph

REL(f_g(**a**))



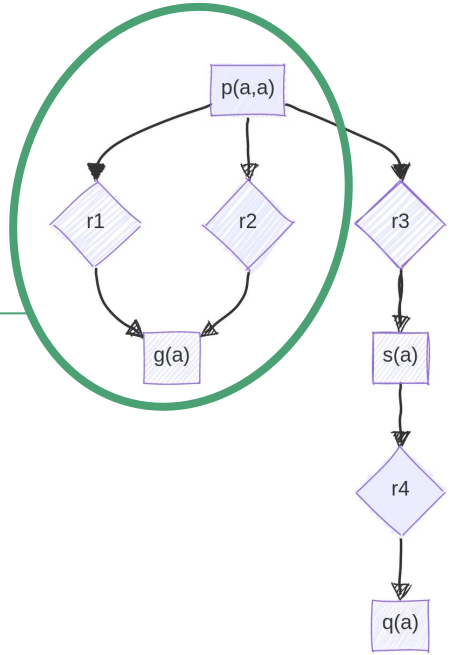

```
#FB
E_r1(f_p(a,a),f_g(a))
E_r2(f_p(a,a),f_g(a))
E_r3(f_p(a,a),f_s(a))
E_r4(f_s(a),f_q(a))
REL(f_g(a))

#RB
[REL(Y) ^ E_r1(X1,Y) ->
REL(X1) ^ REL_E_r1(X1,Y),
...
REL(Y) ^ E_r4(X1,Y) ->
REL(X1) ^ REL_E_r4(X1,Y)]
```

Solution: rule-based encoding of the tracing

Dynamic Step: Fact Tracing

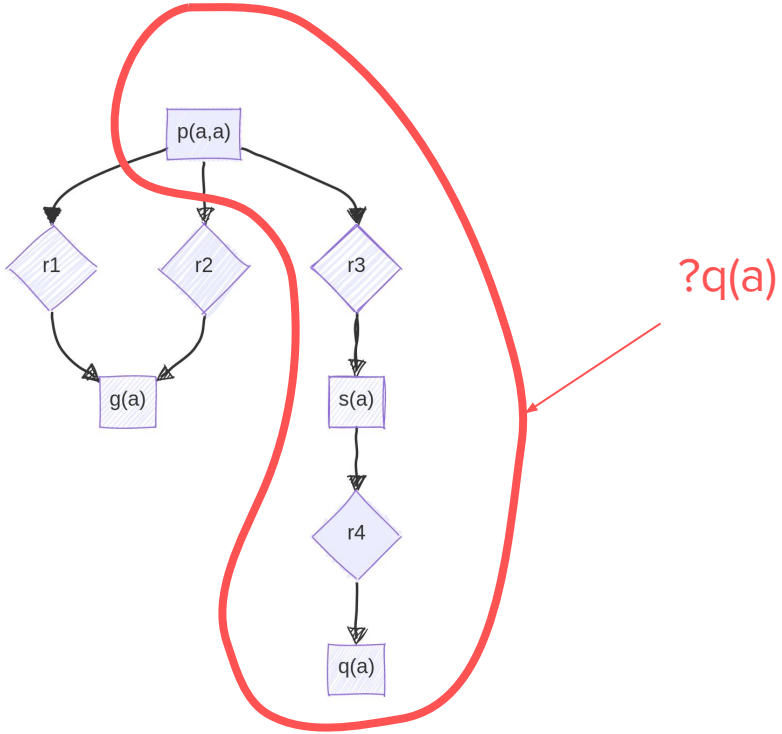
?g(a)




```
#FB
E r1(f_p(a,a),f_g(a))
E r2(f_p(a,a),f_g(a))
E r3(f_p(a,a),f_s(a))
E r4(f_s(a),f_q(a))
REL(f_g(a))

#RB
[REL(Y) ^ E r1(X1,Y) ->
REL(X1) ^ REL E r1(X1,Y),
...
REL(Y) ^ E r4(X1,Y) ->
REL(X1) ^ REL E r4(X1,Y)]
```

Dynamic Step: Fact Tracing



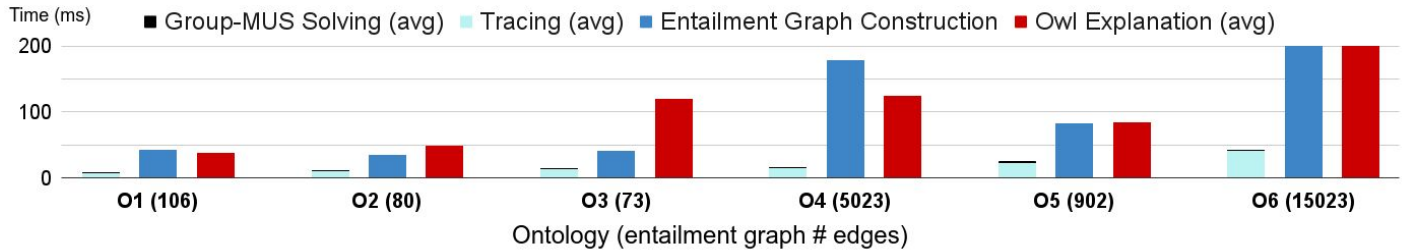

```
#FB
E_r1(f_p(a,a),f_g(a))
E_r2(f_p(a,a),f_g(a))
E_r3(f_p(a,a),f_s(a))
E_r4(f_s(a),f_q(a))
REL(f_q(a))

#RB
[REL(Y) ^ E_r1(X1,Y) ->
REL(X1) ^ REL_E_r1(X1,Y),
…
REL(Y) ^ E_r4(X1,Y) ->
REL(X1) ^ REL_E_r4(X1,Y)]
```

Experiment

Preprocessing:

- 24 ontologies from MOWL corpus translated to .dlgp (via Carral et al.)
- Choose 5 facts from deepest reasoning level
- Run against OWL Explanation (owl api) tool



Conclusion: our approach is generally more competitive as the number of queries grows (and sometimes even for one query!)

Current Status & Future plans

- Formal/proof development
- Implementation/experiment
- Submission to RuleML24

2 future directions:

1. Extension
2. Optimisation & comparison with other approaches

Training

Past:

- RuleML Summer School (2023) - 34h
- French course - 30h

Ongoing:

- Research Integrity - 15h
- French course - 30h

Reference

Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases. Add. Wesley (1995)

Baget, J.F., Bisquert, P., Leclère, M., Mugnier, M.L., Pérution-Kihli, G., Tornil, F., Ulliana, F.: InteGraal: a Tool for Data-Integration and Reasoning on Heterogeneous and Federated Sources. In: BDA (2023)

Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. J. ACM (1984)

Bourgaux, C., Bourhis, P., Peterfreund, L., Thomazo, M.: Revisiting semiring provenance for datalog. In: KR (2022)

Calautti, M., Livshits, E., Pieris, A., Schneider, M.: Computing the why-provenance for datalog queries via SAT solvers. In: AAAI. pp. 10459–10466 (2024)

Carral, D., Zalewski, J., Hitzler, P.: An efficient algorithm for reasoning over OWL EL ontologies with nominal schemas. J. Log. Comput 33(1), 136–162 (2022)

Elhalawati, A., Krötzsch, M., Mennicke, S.: An existential rule framework for computing why-provenance on-demand. In: RuleML+RR. pp. 146–163 (2022)

Horridge, M., Bechhofer, S.: The OWL Api. Semantic Web (2011)

Liffiton, M.H., Previti, A., Malik, A., Marques-Silva, J.: Fast, flexible mus enumeration. Constraints 21(2), 223–250 (2016)

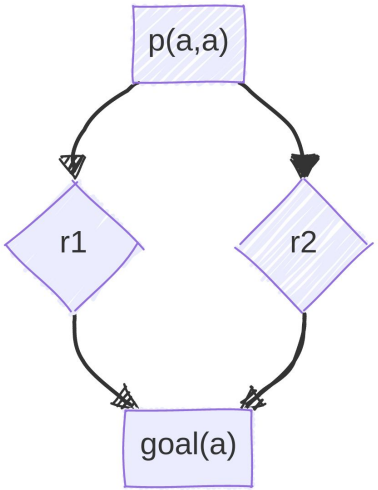
Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. Journal of Automated Reasoning 40, 1–33 (2008)

Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: IJCAI (2018)

Zhapa-Camacho, F., Kulmanov, M., Hoehndorf, R.: mOWL: Python library for machine learning with biomedical ontologies. Bioinformatics (2022)

Computing explanations: reduction to **group**-MUS

r1: p(x,x) -> goal(x)
r2: p(x,**y**) -> goal(x)



Example 1

{~goal(a)}
^{p(a,a)}_f1
^{p(a,a)->goal(a)}_r1
^{p(a,a)->goal(a)}_r2

MUS Solver

{~goal(a)}
^{p(a,a)}_f1
^{p(a,a)->goal(a)}_r1

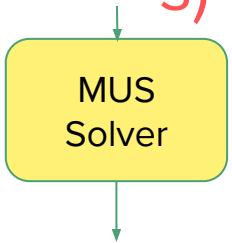
{~goal(a)}
^{p(a,a)}_f1
^{p(a,a)->goal(a)}_r2

Group MUS:
2 MUS for 2 explanations

Computing explanations: MUS

MUS: minimal unsatisfiable subsets of clauses

p ∧ (p->q) ∧ (NOT q)(NOT q) ∧ s ∧ (q v NOT s) ∧ t



MUS1: (NOT q) ∧ p ∧ (p->q)

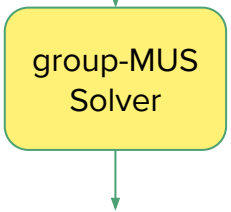
MUS2: (NOT q) ∧ s ∧ (s->q)

Computing explanations: group MUS

Group MUS: minimal unsatisfiable subsets of *groups of* clauses

1 2 3 4 5 6

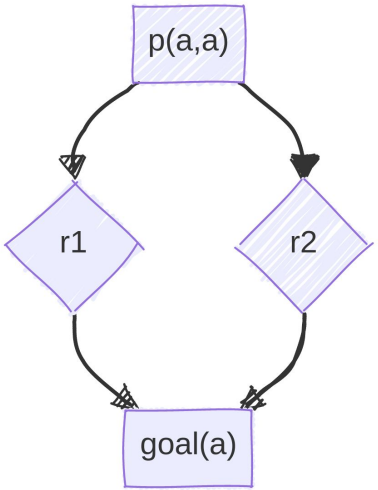
[p(a)] ∧ [p(a)] ∧ [p(b)] ∧ [p(a),p(b)] ∧ [p(a) ∧ p(b)->q(a,b)] ∧ [NOT q(a,b)]



1, 3, 5, 6
2, 3, 5, 6
4, 5, 6

Computing explanations: reduction to **group**-MUS

r1: p(x,**x**) -> goal(x)
r2: p(x,**y**) -> goal(x)



Example 1

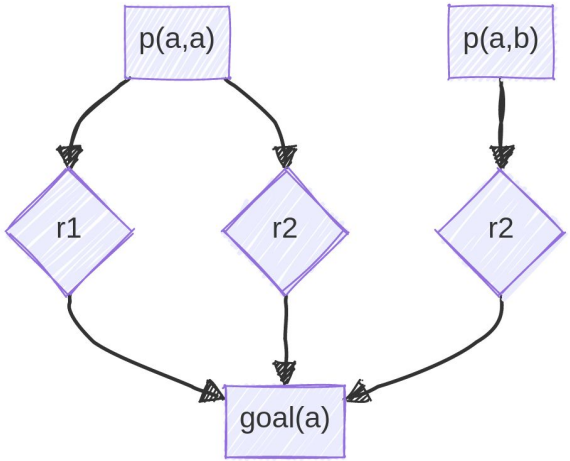
g1 : p(a,a)->goal(a) (group for r1)

g2 : p(a,a)->goal(a) (group for r2)

g3 : p(a,a)

Computing explanations: reduction to **group**-MUS

r1: p(x,x) -> goal(x)
r2: p(x,**y**) -> goal(x)



Example 1
(modified)

g0 : NOT goal(a)

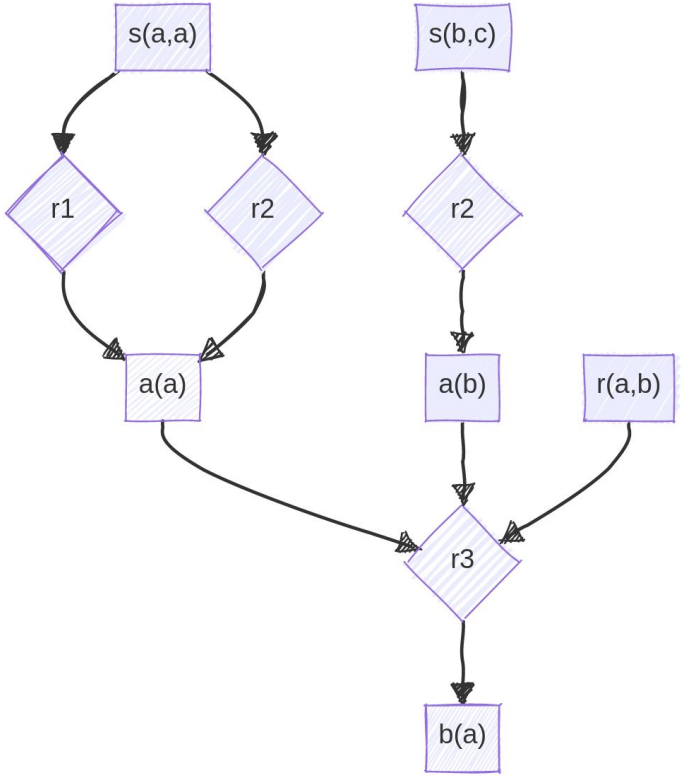
p(a,a)

p(a,b)

p(a,a)->goal(a) (group for r1)

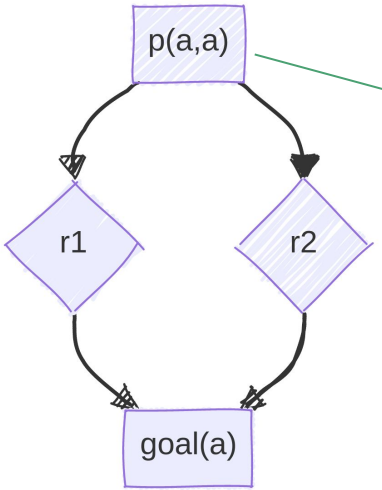
p(a,a)->goal(a)
p(a,b)->goal(a) (group for r2)

Computing explanations: group MUS



Computing explanations: reduction to MUS

r1: p(x,**x**) -> goal(x)
r2: p(x,**y**) -> goal(x)



How do we reduce a KB to SAT formula?

p(a,a)

grounding

p(a,a)->goal(a)

Computing explanations: reduction to MUS

r1: p(x,**x**) -> goal(x)
r2: p(x,**y**) -> goal(x)

How do we reduce a KB to SAT formula?

tains(banana,ca)
ains(banana,k40)
dioactive(k40)

(x,y), radioactive(y) ->
adioactive(x)

grounding

p(a,a)

p(a,a)->goal(a)