

EFTHIMIA TSAMOURA¹, DAVID CARRAL², ENRICO
MALIZIA³, AND JACOPO URBANI⁴

1. SAMSUNG AI RESEARCH

2. LIRMM, INRIA, UNIVERSITY OF MONTPELLIER, CNRS

3. UNIVERSITY OF BOLOGNA

4. VRIJE UNIVERSITEIT AMSTERDAM

Definition: Rules

A *rule* is a first-order formula of the form

$$\forall \vec{X} \forall \vec{Y} \bigwedge_{i=1}^n P_i(\vec{X}_i, \vec{Y}_i) \rightarrow \exists \vec{Z}. P(\vec{Y}, \vec{Z})$$

where P is an IDB predicate, $\vec{Y} \subseteq \vec{X}$, and $\vec{X}_i \subseteq \vec{X}$ for all $1 \leq i \leq n$

W.l.o.g. Assumptions

- Rules feature exactly one atom in the head.
- All rules are EDB or IDB:
 - ▶ IDB rule: all predicates in the body are IDB.
 - ▶ EDB rule: all predicates in the body are EDB.
- We assume that existentially quantified variables do not reoccur across different rules.

Definition: Rule Application

Consider a rule

$$\rho = \bigwedge_{i=1}^n P_i(\vec{X}_i, \vec{Y}_i) \rightarrow \exists \vec{Z}. P(\vec{Y}, \vec{Z}),$$

a fact set \mathcal{F} , and a homomorphism $h : \bigwedge_{i=1}^n P_i(\vec{X}_i, \vec{Y}_i) \rightarrow \mathcal{F}$.

- We define $Appl(\mathcal{F}, \rho, h) = \mathcal{F} \cup \{h_s(P(\vec{Y}, \vec{Z}))\}$ where h_s is the safe extension of h .
- We define $Appl(\mathcal{F}, \rho)$ as the set of facts that includes $Appl(\mathcal{F}, \rho, h)$ for all $h : \bigwedge_{i=1}^n P_i(\vec{X}_i, \vec{Y}_i) \rightarrow \mathcal{F}$.

EXECUTION GRAPHS: DEFINITION

Definition: Execution Graphs

An *execution graph* for a rule set \mathcal{R} is an acyclic digraph $G = (V, E, r)$ such that V is a finite set of vertices, E is a finite set of *edges* of the form $v \rightarrow_i u$ with $v, u \in V$ and $i \geq 1$, and the following hold:

- The function r maps every vertex in V to some rule in \mathcal{R} .
- If $r(v)$ is an EDB rule for some $v \in V$, then v is a root in G .
- Consider some $v \in V$ with $r(v)$ an IDB rule of the form $\alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \exists \vec{y}. \beta$. Then, for every $1 \leq i \leq n$, there is exactly one vertex $u \in V$ with $u \rightarrow_i v \in E$.

Remark

Consider an execution graph G for a rule set \mathcal{R} .

- For a database \mathcal{D} , we define the fact set $G(\mathcal{D})$.
- The execution graph G is a trigger graph for \mathcal{R} if, for all BCQs γ ,

$$\mathcal{R} \cup \mathcal{D} \models \gamma \iff G(\mathcal{D}) \models \gamma.$$

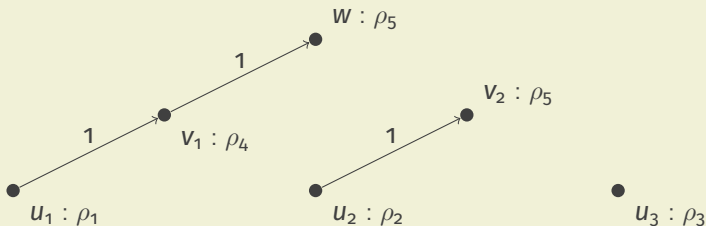
EXECUTION GRAPHS: AN EXAMPLE

Example

Consider the database $\mathcal{D} = \{a(t), b(t), c(t)\}$ and the rule set \mathcal{R} :

$$\begin{array}{lll} \rho_1 = a(x) \rightarrow A(x) & \rho_2 = b(x) \rightarrow B(x) & \rho_3 = c(x) \rightarrow C(x) \\ \rho_4 = A(x) \rightarrow B(x) & \rho_5 = B(x) \rightarrow C(x) & \rho_6 = A(x) \rightarrow C(x) \end{array}$$

Then, the following structure is an execution graph for \mathcal{R} :



EVALUATING EXECUTION GRAPHS: DEFINITION

Definition

Consider an execution graph $G = (V, E, r)$ for a rule set \mathcal{R} , and a database \mathcal{D} . Then, for a vertex $v \in V$, we inductively define the fact set $v(\mathcal{D})$ as follows:

- Case 1: if v is a root in G , then $v(\mathcal{D}) = \text{Appl}(\mathcal{D}, r(v))$.
- Case 2: $r(v)$ is an IDB rule of the form $\alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \exists \vec{y}. \beta$ and, for every $1 \leq i \leq n$, there is exactly one vertex $u_i \in V$ with $u_i \rightarrow_i v \in E$. Then, $v(\mathcal{D})$ is the set that contains $h_s(\beta)$ for every homomorphism h such that $h(\alpha_i) \subseteq u_i(\mathcal{D})$ for all $1 \leq i \leq n$.

We define $G(\mathcal{D}) = \bigcup_{v \in V} v(\mathcal{D}) \cup \mathcal{D}$.

Remark

Consider a rule set \mathcal{R} , a database \mathcal{D} , and an execution graph G for \mathcal{R} . Then, we can compute $G(\mathcal{D})$ if G is finite.

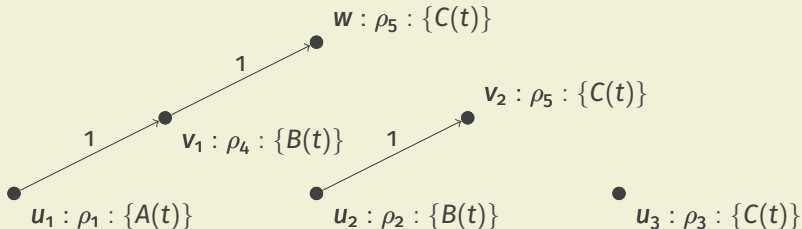
EVALUATING EXECUTION GRAPHS: EXAMPLE

Example

Consider the database $\mathcal{D} = \{a(t), b(t), c(t)\}$ and the rule set \mathcal{R} :

$$\begin{array}{lll} \rho_1 = a(x) \rightarrow A(x) & \rho_2 = b(x) \rightarrow B(x) & \rho_3 = c(x) \rightarrow C(x) \\ \rho_4 = A(x) \rightarrow B(x) & \rho_5 = B(x) \rightarrow C(x) & \rho_6 = A(x) \rightarrow C(x) \end{array}$$

Then, the following structure is an execution graph for \mathcal{R} :



To obtain $G(\mathcal{D})$, we inductively compute $v(\mathcal{D})$ for each vertex v .

TRIGGER GRAPHS: DEFINITION

Definition: Trigger Graphs

An execution graph G for a rule set \mathcal{R} is a *trigger graph* for \mathcal{R} if, for all databases \mathcal{D} and all queries γ , we have that $\mathcal{R} \cup \mathcal{D} \models \gamma$ iff $G(\mathcal{D}) \models \gamma$.

Example

Consider the rule set \mathcal{R} :

$$\rho_1 = a(x) \rightarrow A(x)$$

$$\rho_2 = b(x) \rightarrow B(x)$$

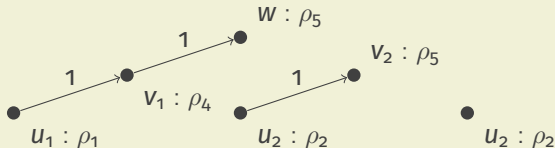
$$\rho_3 = c(x) \rightarrow C(x)$$

$$\rho_4 = A(x) \rightarrow B(x)$$

$$\rho_5 = B(x) \rightarrow C(x)$$

$$\rho_6 = A(x) \rightarrow B(x)$$

Then, the following structure is a trigger graph for \mathcal{R} :



TRIGGER GRAPHS: DEFINITION

Definition: Trigger Graphs

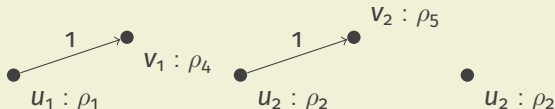
An execution graph G for a rule set \mathcal{R} is a *trigger graph* for \mathcal{R} if, for all databases \mathcal{D} and all queries γ , we have that $\mathcal{R} \cup \mathcal{D} \models \gamma$ iff $G(\mathcal{D}) \models \gamma$.

Example

Consider the rule set \mathcal{R} :

$$\begin{array}{lll} \rho_1 = a(x) \rightarrow A(x) & \rho_2 = b(x) \rightarrow B(x) & \rho_3 = c(x) \rightarrow C(x) \\ \rho_4 = A(x) \rightarrow B(x) & \rho_5 = B(x) \rightarrow C(x) & \rho_6 = A(x) \rightarrow B(x) \end{array}$$

The following structure G is NOT a trigger graph for \mathcal{R} :



For instance, $\mathcal{R} \cup \{a(t)\} \models C(t)$ whilst $G(\{a(t)\}) \not\models C(t)$.

TRIGGER GRAPHS: OUTPUT

Remark

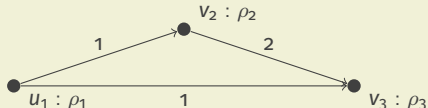
The output of a trigger graph G for a rule set \mathcal{R} on an input database \mathcal{D} may not be a model for $\mathcal{R} \cup \mathcal{D}$.

Example

Consider following rule set \mathcal{R} :

$$\rho_1 = r(x) \rightarrow \exists y.R(x, y) \quad \rho_2 = R(x, y) \rightarrow \exists z.R(y, z) \quad \rho_3 = R(x, y) \wedge R(y, z) \rightarrow R(y, x)$$

Then, the following structure G is a trigger graph for \mathcal{R} :



The set $G(r(a)) = \{r(a), R(a, n), R(n, m), R(n, a)\}$ is not a model for $\mathcal{R} \cup \{r(a)\}$.

Proposition

Given a trigger graph G for a rule set \mathcal{R} and a database \mathcal{D} , the core of $G(\mathcal{D})$ is a finite universal model for $\mathcal{R} \cup \mathcal{D}$.

RULE SETS THAT ADMIT TRIGGER GRAPHS

Remark

Not all rule sets admit (finite) trigger graphs. For instance, neither $\{R(x, y) \rightarrow \exists z. R(y, z)\}$ nor $\{R(x, y) \wedge R(y, z) \rightarrow R(x, z)\}$ admit such graphs!

Theorem

A rule set admits a trigger graph iff it is bounded.

Proof Sketch:

- \Rightarrow If a rule set \mathcal{R} admits a trigger graph G of depth k , then \mathcal{R} is bounded by k .
- \Leftarrow If a rule set \mathcal{R} is bound by k , then we can construct a “maximal” trigger graph of depth k that derives all of the facts produced by the chase up to the k -th step.

Corollary

A rule set admits a trigger graph iff it is FUS and it terminates with respect to the restricted chase.

TRIGGER GRAPHS FOR LINEAR RULE SETS

Algorithm 1 $\text{tglinear}(\mathcal{R})$

```
1: Let  $G$  be an empty EG
2: for each  $\varphi \in \mathcal{H}(\mathcal{R})$  do
3:    $\Gamma = (V, E, r)$  is an empty EG;  $\mu$  is the empty mapping
4:   for each  $\varphi_1 \rightarrow_{\rho} \varphi_2 \in \text{chaseGraph}(\mathcal{R}, \{\varphi\})$  do
5:     add a fresh node  $u$  to  $V$  with  $r(u) := \rho$ 
6:      $\mu(u) := \varphi_1 \rightarrow_{\rho} \varphi_2$ 
7:   for each  $v, u \in V$  do
8:     if  $\mu(v) = \varphi_1 \rightarrow_{\rho} \varphi_2$  and  $\mu(u) = \varphi_2 \rightarrow_{\rho'} \varphi_3$  then
9:       add  $v \rightarrow_1 u$  to  $E$ 
10:   $G := G \cup \Gamma$ 
11: return  $G$ 
```

- $\mathcal{H}(\mathcal{R})$ is a maximal set of facts formed over the predicates in \mathcal{R} , where no $\varphi_1 \in \mathcal{H}(\mathcal{R})$ is pattern isomorphic to another $\varphi_2 \in \mathcal{H}(\mathcal{R})$.
- Let $\text{chaseGraph}(\mathcal{R}, \mathcal{D})$ as the acyclic graph having as nodes the facts in the chase of $(\mathcal{R}, \mathcal{D})$ and having an edge from φ_1 to φ_2 labeled with rule $\rho \in \mathcal{R}$ if φ_2 is obtained from φ_1 by executing ρ .

MINIMIZING TRIGGER GRAPHS FOR LINEAR RULE SETS

Consider a linear rule set \mathcal{R} , a trigger graph $G = (V, E, r)$ for \mathcal{R} , and some $u, v \in V$.

Definition

- For a database \mathcal{D} , a homomorphism from $u(\mathcal{D})$ into $v(\mathcal{D})$ is *preserving* if it is the identity over the set of nulls in any fact set associated with an ancestor of u .
- The vertex $v \in V$ is dominated by another vertex $u \in V$ if there is a preserving homomorphism from $v(\{\varphi\})$ to $u(\{\varphi\})$ for all $\varphi \in \mathcal{H}(P)$.

Lemma

There is a *preserving homomorphism* from $u(\mathcal{D})$ into $v(\mathcal{D})$ for all databases \mathcal{D} iff there is a preserving homomorphism from $u(\{\varphi\})$ into $v(\{\varphi\})$ for all $\varphi \in \mathcal{H}(P)$.

Lemma

Assume that v is dominated by u and u is not a successor of v . Then, the following transformation produces a trigger graph for \mathcal{R} :

- Add an edge $u \rightarrow_1 w$ for every edge of the form $v \rightarrow_1 w \in E$.
- Remove the vertex v and all edges where this vertex occurs from G .

The above process can be exhaustively repeated to obtain a minimal trigger graph for \mathcal{R} .

TRIGGER GRAPHS: PRACTICAL CONTRIBUTION

Empirical Claim

Trigger graphs can be used to develop a very efficient implementation of the chase algorithm.

Some (cherry-picked) results that support our claim:

	VLog	RDFox	GLog
LUBM	170s	115s	16s
DBpedia	41s	198s	19s
Claros	431s	2373s	122s

Remarks

- The above theories only contain Datalog rules.
- When using trigger graphs, we only remove duplicates at the end.

TRIGGER GRAPHS: FUTURE WORK

Remark

We can only compute finite trigger graphs for bounded rule sets. For instance, rule sets as simple as $\{R(x, y) \rightarrow \exists z.R(y, z)\}$ nor $\{R(x, y) \wedge R(y, z) \rightarrow R(x, z)\}$ do not admit trigger graphs!

One possible solution is to redefine the notion of *trigger graphs*:

Redefinition

An (possibly cyclic) execution graph G for a rule set \mathcal{R} is a *trigger graph* if $\mathcal{R} \cup \mathcal{D} \models \gamma$ iff $G(\mathcal{D}) \models \varphi$ for all databases \mathcal{D} and facts φ .

Hypotheses

- All FUS rule sets admit acyclic trigger graphs.
- Every DL/BTS rule set admits a trigger graph \mathcal{G} in which all vertices occurring in a cycle are labelled with Datalog rules.

“DL = Description Logics” and “BTS = Bounded Treewidth Set”

TRIGGER GRAPHS: FUTURE WORK

The use of cyclic trigger graphs can allow us to implement well-known optimisations:

Example: Computing Transitivity

Consider the rule set

$$\mathcal{R} = \{\rho_1 = r(x, y) \rightarrow R(x, y), \\ \rho_2 = R(x, y) \wedge R(y, z) \rightarrow R(x, y)\}.$$

Then, $G = (V, E, r)$ is a trigger graph for \mathcal{R} .

$$\begin{aligned} V &= \{v_1, v_2\} \\ E &= \{(v_1, v_1, v_2), (v_1, v_2, v_2)\} \\ r &= \{v_1 \mapsto \rho_1, v_2 \mapsto \rho_2\} \end{aligned}$$

THANK YOU FOR YOUR ATTENTION!