Capturing Homomorphism-Closed Decidable Queries with Existential Rules

Camille Bourgaux — CNRS, DI ENS
David Carral — Inria, LIRMM
Markus Krötzsch — TU Dresden
Sebastian Rudolph — TU Dresden
Michaël Thomazo — Inria, DI ENS

1. Motivation

- Database: finite relational structures over a countably infinite set of nulls
- Abstract query: finite set of databases closed under isomorphism
- Existential rules: $\forall \vec{x}.\Big(\beta[\vec{x}] \rightarrow \exists \vec{y}.\eta[\vec{x},\vec{y}]\Big)$ with $\beta[\vec{x}]$, $\eta[\vec{x},\vec{y}]$ conjunctions of atoms

Existential rules capture the class of homomorphism-closed queries that are recursively enumerable

Can we characterize an existential rules fragment that can express every decidable homomorphism-closed query?

One way to ensure decidability: chase termination

- chase: repetitive, forward-chaining rule application, starting from the database
- several chase variants
- universal models

2. Results

Standard-chase-terminating existential rules capture the class of all decidable homomorphism-closed queries.
Implies that

- standard-chase-terminating and core-chase-terminating existential rule queries are equally expressive
- no decidable enhancement that preserves homomorphism-closedness can be strictly more expressive

Membership in this fragment is *not semi-decidable*, but this is *unavoidable* (via a diagonalisation argument).

3. Overview of the Construction

Let $\mathfrak{Q}$ be a homomorphism-closed query over signature $\mathcal{S}$, and $M = \langle Q, \Gamma, \delta\rangle$ be a Turing machine that decides $\mathfrak{Q}$. We construct a set of standard-chase-terminating existential rules Σ such that $\mathcal{D} \in \mathfrak{Q}$ iff $\langle \Sigma, \mathcal{D}\rangle \models \texttt{Goal}$.

Construction in three steps:

- Capturing $\mathfrak{Q}$ with disjunctive rules (details in 4)
  - disjunctive existential rules: $\forall \vec{x}.\Big(\beta[\vec{x}] \rightarrow \bigvee_{i=1}^{k} \exists \vec{y}_i.\eta_i[\vec{x}_i,\vec{y}_i]\Big)$
  - guess completion of $\mathcal{D}$ (linear order + extensions of predicates and their complements) with disjunctive rules
  - simulate the run of M for the initial configuration corresponding to the completion with existential rules
- Ensuring chase termination with the emergency brake technique
  - our rules may lead to infinite chase trees
  - refine and generalise the "emergency brake" technique of Krötzsch, Marx, Rudolph (ICDT 2019)
  - general rule set transformation: given rule set $\mathcal{R}$ and fresh nullary predicate $\texttt{Halt}$
    - $\mathit{brake}(\mathcal{R}, \texttt{Halt})$
      - add a "brake" null that will be made "real" only when $\texttt{Halt}$ is derived
      - stop the chase when "brake" becomes "real"
    - add rules that "pull the brake" by deriving $\texttt{Halt}$
- Removing disjunctions: express disjunctive Datalog with existential rules
  - our rules can be split into disjunctive Datalog and existential rules, s.t. the disjunctive part can be chased first
  - given such a rule set $\mathcal{R}$, we define an existential rule set Σ such that $\langle \mathcal{R}, \mathcal{D}\rangle \models \texttt{Goal}$ iff $\langle \Sigma, \mathcal{D}\rangle \models \texttt{Goal}$
  - adapt a technique for modeling sets with existential rules from Krötzsch, Marx, Rudolph (ICDT 2019)
    - build all possible worlds corresponding to the choices made by disjunctive rules
    - simulate the application of the non-disjunctive rules in each world
    - aggregate results from all worlds
  - the transformation preserves chase termination

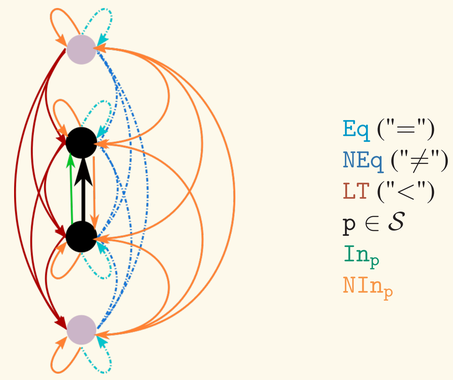
4. Focus : Capturing $\mathfrak{Q}$ with Disjunctive Rules

Construct rule sets $\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_3 \subseteq \mathcal{R}_4 \subseteq \mathcal{R}_5$ such that for every database $\mathcal{D}$ over $\mathcal{S}$, there is a universal model set $\mathfrak{M}$ of $\mathcal{R}_5$ and $\mathcal{D}$ such that $\mathcal{D} \in \mathfrak{Q}$ iff $\texttt{Goal} \in \mathcal{I}$ for every $\mathcal{I} \in \mathfrak{M}$.

- Applying $\mathcal{R}_1$ to $\mathcal{D}$
  - adds two new nulls (labelled $\texttt{First}$ and $\texttt{Last}$)
  - disjunctively "guesses" all first-to-last linear orders over nulls
    - cave: distinct elements may be "misclassified" as equal
  - disjunctively "guesses" extensions (and their complements) of all database predicates
    - correctly records all relation tuples present in $\mathcal{D}$
    - but: some guesses may contain "false positives"

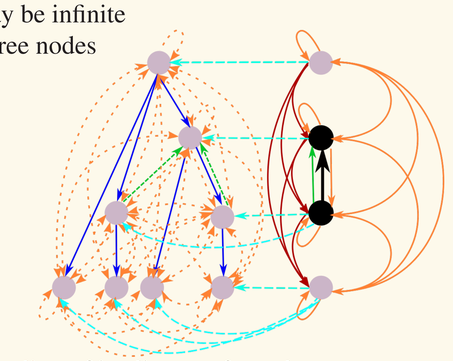
Minimal models of $\mathcal{D}$ and $\mathcal{R}_1$ represent all such guessed "completions" of $\mathcal{D}$.

$$\rightarrow \exists y.\texttt{First}(y) \wedge \texttt{DbDom}(y) \quad (1)$$
$$\rightarrow \exists z.\texttt{Last}(z) \wedge \texttt{DbDom}(z) \quad (2)$$
$$\texttt{p}(\vec{x}) \rightarrow \texttt{In}_\texttt{p}(\vec{x}) \wedge \textstyle\bigwedge_{x\in\vec{x}} \texttt{DbDom}(x) \quad (3)$$
$$\texttt{DbDom}(x) \rightarrow \texttt{Eq}(x,x) \quad (4)$$
$$\texttt{Eq}(x,y) \rightarrow \texttt{Eq}(y,x) \quad (5)$$
$$\texttt{NEq}(x,y) \rightarrow \texttt{NEq}(y,x) \quad (6)$$
$$\texttt{R}(\vec{x}) \wedge \texttt{Eq}(x_i,y) \rightarrow \texttt{R}(\vec{x}_{x_i\mapsto y}) \quad (7)$$
$$\texttt{DbDom}(x) \wedge \texttt{DbDom}(y) \rightarrow \texttt{Eq}(x,y) \vee \texttt{NEq}(x,y) \quad (8)$$
$$\texttt{LT}(x,y) \wedge \texttt{LT}(y,z) \rightarrow \texttt{LT}(x,z) \quad (9)$$
$$\texttt{First}(x) \wedge \texttt{NEq}(x,y) \rightarrow \texttt{LT}(x,y) \quad (10)$$
$$\texttt{NEq}(x,y) \wedge \texttt{Last}(y) \rightarrow \texttt{LT}(x,y) \quad (11)$$
$$\texttt{NEq}(x,y) \rightarrow \texttt{LT}(x,y) \vee \texttt{LT}(y,x) \quad (12)$$
$$\textstyle\bigwedge_{x\in\vec{x}} \texttt{DbDom}(x) \rightarrow \texttt{In}_\texttt{p}(\vec{x}) \vee \texttt{NIn}_\texttt{p}(\vec{x}) \quad (13)$$



- Applying $\mathcal{R}_2 \setminus \mathcal{R}_1$ to a minimal model of $\mathcal{D}$ and $\mathcal{R}_1$
  - builds a tree structure where each path represents a sequence of nulls that respects the linear order $\texttt{LT}$
    - may skip some nulls
    - but one path is complete: successor relation corresponding to $\texttt{LT}$
    - cave: if disjunctive guessing of $\texttt{LT}$ led to a cycle, chase may be infinite
  - replicates relations $\texttt{In}_\texttt{p}$ and $\texttt{NIn}_\texttt{p}$ as $\texttt{In}'_\texttt{p}$ and $\texttt{NIn}'_\texttt{p}$ between the tree nodes

$$\texttt{First}(x) \rightarrow \exists u.\texttt{Root}(u) \wedge \texttt{Rep}(x,u) \quad (14)$$
$$\texttt{Rep}(x,v) \wedge \texttt{LT}(x,z) \rightarrow \exists w.\texttt{Chi}(v,w) \wedge \texttt{Rep}(z,w) \quad (15)$$
$$\texttt{Last}(x) \wedge \texttt{Rep}(x,u) \rightarrow \texttt{Leaf}(u) \quad (16)$$
$$\texttt{Rep}(x,u) \wedge \texttt{Eq}(x,y) \rightarrow \texttt{Rep}(y,u) \quad (17)$$
$$\texttt{In}_\texttt{p}(\vec{x}) \wedge \textstyle\bigwedge_{i=1}^{|\vec{x}|} \texttt{Rep}(x_i,u_i) \rightarrow \texttt{In}'_\texttt{p}(\vec{u}) \quad (18)$$
$$\texttt{NIn}_\texttt{p}(\vec{x}) \wedge \textstyle\bigwedge_{i=1}^{|\vec{x}|} \texttt{Rep}(x_i,u_i) \rightarrow \texttt{NIn}'_\texttt{p}(\vec{u}) \quad (19)$$



- Applying $\mathcal{R}_3 \setminus \mathcal{R}_2$ associates each node in the tree with a binary encoding of its distance from the root
- Applying $\mathcal{R}_4 \setminus \mathcal{R}_3$ encodes initial Turing machine configurations corresponding to some branch in the tree
  - serializes $\texttt{In}'_\texttt{p}$ facts between nulls on the branch using ordering and binary encoding
- Applying $\mathcal{R}_5 \setminus \mathcal{R}_4$ simulates the run of the deterministic Turing machine M on each initial tape