Rewriting Consistent Answers on Annotated Data and Semiring Circuits

Jonni Virtema
joint work with Phokion G. Kolaitis and Nina Pardal

University of Sheffield, UK

Dagstuhl workshop 250801: Semirings in Databases, Automata, and Logic
February 20th, 2025

Funded by
Deutsche Forschungsgemeinschaft
German Research Foundation

Consistent answers

- Let $q(\vec{x}) \in \mathrm{FO}$ be a query and $\sigma \subseteq \mathrm{FO}$ be a set of integrity constraints.
- A relational database D is consistent, if $\mathrm{D} \models \sigma$, and inconsistent otherwise.
- A repair of an inconsistent database D is a consistent database D' such that there is no other consistent database D'' such that $\mathrm{D} \leq \mathrm{D}'' < \mathrm{D}'$.
- Consistent answers $\mathrm{CA}(\mathrm{D}, q)$ of q are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

Example

Let $\mathrm{D} = \{R(a,a), R(a,b), R(c,d)\}$, $\sigma = \{\forall xyz \left(R(x,y) \wedge R(x,z) \rightarrow y = z\right)\}$, and $q = R(x,y)$.
Then $\mathrm{D}_1 = \{R(a,a), R(c,d)\}$ and $\mathrm{D}_2 = \{R(a,b), R(c,d)\}$ are subset repairs of D.
$\mathrm{CA}(\mathrm{D}, q) = q(\mathrm{D}_1) \cap q(\mathrm{D}_2) = \{(a,a),(c,d)\} \cap \{(a,b),(c,d)\} = \{(c,d)\}$.

Consistent answers

- Let $q(\vec{x}) \in \mathrm{FO}$ be a query and $\sigma \subseteq \mathrm{FO}$ be a set of integrity constraints.
- A relational database D is consistent, if $\mathrm{D} \models \sigma$, and inconsistent otherwise.
- A repair of an inconsistent database D is a consistent database D' such that there is no other consistent database D'' such that $\mathrm{D} \leq \mathrm{D}'' < \mathrm{D}'$.
- Consistent answers $\mathrm{CA}(\mathrm{D}, q)$ of q are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

Example

Let $\mathrm{D} = \{R(a,a), R(a,b), R(c,d)\}$, $\sigma = \{\forall xyz\,(R(x,y) \wedge R(x,z) \rightarrow y = z)\}$, and $q = R(x,y)$.
Then $\mathrm{D}_1 = \{R(a,a), R(c,d)\}$ and $\mathrm{D}_2 = \{R(a,b), R(c,d)\}$ are subset repairs of D.
$\mathrm{CA}(\mathrm{D}, q) = q(\mathrm{D}_1) \cap q(\mathrm{D}_2) = \{(a,a),(c,d)\} \cap \{(a,b),(c,d)\} = \{(c,d)\}$.

Consistent answers

- Let $q(\vec{x}) \in \mathrm{FO}$ be a query and $\sigma \subseteq \mathrm{FO}$ be a set of integrity constraints.
- A relational database D is consistent, if $\mathrm{D} \models \sigma$, and inconsistent otherwise.
- A repair of an inconsistent database D is a consistent database D' such that there is no other consistent database D'' such that $\mathrm{D} \leq \mathrm{D}'' < \mathrm{D}'$.
- Consistent answers $\mathrm{CA}(\mathrm{D}, q)$ of q are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

Example

Let $\mathrm{D} = \{R(a, a), R(a, b), R(c, d)\}$, $\sigma = \{\forall xyz \left(R(x, y) \wedge R(x, z) \rightarrow y = z\right)\}$, and $q = R(x, y)$.
Then $\mathrm{D}_1 = \{R(a, a), R(c, d)\}$ and $\mathrm{D}_2 = \{R(a, b), R(c, d)\}$ are subset repairs of D.
$\mathrm{CA}(\mathrm{D}, q) = q(\mathrm{D}_1) \cap q(\mathrm{D}_2) = \{(a, a), (c, d)\} \cap \{(a, b), (c, d)\} = \{(c, d)\}$.

Rewriting consistent answers and Boolean circuits

A query q' is a rewriting of the CA's of q, if $q'(\mathrm{D}) = \mathrm{CA}(\mathrm{D}, q)$, for every database D.

Theorem ([Koutris and Wijsen, 2017])

Let q be a self-join free conjunctive query with one key constraint per relation. The consistent answers of q are a) FO-rewritable, or b) computable in PTIME but not FO-rewritable, or c) coNP-complete.

Data complexity of first-order logic is DLOGTIME-uniform AC^0 (i.e., constant depth polynomial size Boolean circuits) [Barrington et al., 1990].

Goal: Obtain similar trichotomy for semiring-annotated data.

Rewriting consistent answers and Boolean circuits

A query q' is a rewriting of the CA's of q, if $q'(\mathrm{D}) = \mathrm{CA}(\mathrm{D}, q)$, for every database D.

Theorem ([Koutris and Wijsen, 2017])

Let q be a self-join free conjunctive query with one key constraint per relation. The consistent answers of q are a) FO-rewritable, or b) computable in PTIME but not FO-rewritable, or c) coNP-complete.

Data complexity of first-order logic is DLOGTIME-uniform AC^0 (i.e., constant depth polynomial size Boolean circuits) [Barrington et al., 1990].

Goal: Obtain similar trichotomy for semiring-annotated data.

Rewriting consistent answers and Boolean circuits

A query q' is a rewriting of the CA's of q, if $q'(\mathrm{D}) = \mathrm{CA}(\mathrm{D}, q)$, for every database D.

Theorem ([Koutris and Wijsen, 2017])

Let q be a self-join free conjunctive query with one key constraint per relation. The consistent answers of q are a) FO*-rewritable, or b) computable in PTIME but not* FO*-rewritable, or c) coNP-complete.*

Data complexity of first-order logic is DLOGTIME-uniform AC^0 (i.e., constant depth polynomial size Boolean circuits) [Barrington et al., 1990].

Goal: Obtain similar trichotomy for semiring-annotated data.

Consistent answers on semiring annotated data

Let $K = (K, +, \times, 0, 1)$ be a positive semiring and A a set.

- A K-relation is a function $f : A^n \to K$.
- A support of f is $\{\vec{a} \mid f(\vec{a}) \neq 0\}$.
- A K-database is a finite collection of K-relations (with finite supports).

Consider semiring semantics of FO given earlier by Val and Erich:

- The answer of a query $q(\vec{x})$ on a K-database D is the K-relation $q(\mathrm{D})$.
- Consistent answers $\mathrm{CA}(\mathrm{D}, q)$ of q are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

We need to define what returned by all repairs of D means!

Consistent answers on semiring annotated data

Let $K = (K, +, \times, 0, 1)$ be a positive semiring and A a set.

- A K-relation is a function $f : A^n \to K$.
- A support of f is $\{\vec{a} \mid f(\vec{a}) \neq 0\}$.
- A K-database is a finite collection of K-relations (with finite supports).

Consider semiring semantics of FO given earlier by Val and Erich:

- The answer of a query $q(\vec{x})$ on a K-database D is the K-relation $q(\mathrm{D})$.
- Consistent answers $\mathrm{CA}(\mathrm{D}, q)$ of q are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

We need to define what returned by all repairs of D means!

Repairs of K databases

Recall: A repair of an inconsistent database D is a consistent database D' such that there is no other consistent database D'' such that $\mathrm{D} \leq \mathrm{D}'' < \mathrm{D}'$.

Definition: A K-database D satisfies a 0-ary query q, if $q(\mathrm{D}) \neq 0$.

To compare K-databases, we stipulate that K is a naturally ordered positive semiring.

- For K-relations, define $R \leq S$ if and only if $R(\vec{a}) \leq S(\vec{a})$, for every suitable $\vec{a}$.
- Annotation aware generalisations of subset and superset repairs arise.
- For key constraints, this definition coincides with set-based subset repairs.

Example

If $\mathrm{D} = \{R(\underline{a}, a) = 3, R(\underline{a}, b) = 2, R(\underline{c}, d) = 4\}$ and _ indicates the key attributes, then $\mathrm{D}_1 = \{R(\underline{a}, a) = 3, R(\underline{c}, d) = 4\}$ and $\mathrm{D}_2 = \{R(\underline{a}, b) = 2, R(\underline{c}, d) = 4\}$ are the (subset) repairs of D.

Repairs of K databases

Recall: A repair of an inconsistent database D is a consistent database D' such that there is no other consistent database D'' such that $\mathrm{D} \leq \mathrm{D}'' < \mathrm{D}'$.

Definition: A K-database D satisfies a 0-ary query q, if $q(\mathrm{D}) \neq 0$.

To compare K-databases, we stipulate that K is a naturally ordered positive semiring.

- For K-relations, define $R \leq S$ if and only if $R(\vec{a}) \leq S(\vec{a})$, for every suitable $\vec{a}$.
- Annotation aware generalisations of subset and superset repairs arise.
- For key constraints, this definition coincides with set-based subset repairs.

Example

If $\mathrm{D} = \{R(\underline{a}, a) = 3, R(\underline{a}, b) = 2, R(\underline{c}, d) = 4\}$ and _ indicates the key attributes, then $\mathrm{D}_1 = \{R(\underline{a}, a) = 3, R(\underline{c}, d) = 4\}$ and $\mathrm{D}_2 = \{R(\underline{a}, b) = 2, R(\underline{c}, d) = 4\}$ are the (subset) repairs of D.

Repairs of K databases

Recall: A repair of an inconsistent database D is a consistent database D' such that there is no other consistent database D'' such that $\mathrm{D} \leq \mathrm{D}'' < \mathrm{D}'$.

Definition: A K-database D satisfies a 0-ary query q, if $q(\mathrm{D}) \neq 0$.

To compare K-databases, we stipulate that K is a naturally ordered positive semiring.

- For K-relations, define $R \leq S$ if and only if $R(\vec{a}) \leq S(\vec{a})$, for every suitable $\vec{a}$.
- Annotation aware generalisations of subset and superset repairs arise.
- For key constraints, this definition coincides with set-based subset repairs.

Example

If $\mathrm{D} = \{R(\underline{a}, a) = 3, R(\underline{a}, b) = 2, R(\underline{c}, d) = 4\}$ and _ indicates the key attributes, then $\mathrm{D}_1 = \{R(\underline{a}, a) = 3, R(\underline{c}, d) = 4\}$ and $\mathrm{D}_2 = \{R(\underline{a}, b) = 2, R(\underline{c}, d) = 4\}$ are the (subset) repairs of D.

Consistent answers in semiring semantics

Recall: Consistent answers $\mathrm{CA}(\mathrm{D}, \varphi)$ of are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

In the ordered semiring setting, we replace the intersection by taking the minimum:

$$\mathrm{mCA}(\mathrm{D}, \alpha, q) := \min_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}', \alpha).$$

(cf. [Feng et al., 2019], for bounding CAs from below and above.)

We add assignment α to the syntax, so that the value is an element of a semiring.

Example

If $\mathrm{D}_1 = \{R(\underline{a}, a) = 3, R(\underline{c}, a) = 4\}$ and $\mathrm{D}_2 = \{R(\underline{a}, b) = 2, R(\underline{c}, a) = 4\}$ are repairs of D, and $q = \exists x R(x, y)$, then $\mathrm{mCA}(\mathrm{D}, y \mapsto a, q) = \min\{7, 4\} = 4$ and $\mathrm{mCA}(\mathrm{D}, y \mapsto b, q) = \min\{0, 2\} = 0$.

Consistent answers in semiring semantics

Recall: Consistent answers $\mathrm{CA}(\mathrm{D},\varphi)$ of are those that are returned by all repairs of D

$$\mathrm{CA}(\mathrm{D},q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

In the ordered semiring setting, we replace the intersection by taking the minimum:

$$\mathrm{mCA}(\mathrm{D},\alpha,q) := \min_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}',\alpha).$$

(cf. [Feng et al., 2019], for bounding CAs from below and above.)

We add assignment α to the syntax, so that the value is an element of a semiring.

Example

If $\mathrm{D}_1 = \{R(\underline{a},a)=3, R(\underline{c},a)=4\}$ and $\mathrm{D}_2 = \{R(\underline{a},b)=2, R(\underline{c},a)=4\}$ are repairs of D, and $q = \exists x R(x,y)$, then $\mathrm{mCA}(\mathrm{D}, y\mapsto a, q) = \min\{7,4\} = 4$ and $\mathrm{mCA}(\mathrm{D}, y\mapsto b, q) = \min\{0,2\} = 0$.

Reminder of the goal: trichotomy theorem for semiring-annotated data

Definition (Recall)

The consistent answers $\mathrm{CA}(q)$ of q is FO-rewritable, if there exists a $\varphi \in \mathrm{FO}$ such that

$$\mathrm{CA}(D, q) = \varphi(D),$$

for every database D.

Theorem ([Koutris and Wijsen, 2017])

Let q be a self-join free conjunctive query with one key constraint per relation. The the consistent answers $\mathrm{CA}(q)$ is first-order rewritable, or it is polynomial-time computable but it is not first-order rewritable, or it is coNP-complete.

Data complexity of first-order logic is DLOGTIME-uniform AC^0 (i.e., constant depth polynomial size Boolean circuits).

Logic for rewriting $\mathrm{mCA}(\mathrm{D}, \alpha, q)$

Ingredients for rewriting a conjunctive query $q_{\mathrm{path}} = \exists x \exists y \exists z (R(x; y) \wedge S(y; z))$:

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

[Fuxman and Miller, 2007] rewriting: $\exists x \exists z' (R(x, z') \wedge \forall z (R(x, z) \rightarrow \exists y S(z, y)))$

Semiring setting: Similar rewriting requires care; a) universal quantifier, b) implication.

$$\mathrm{mCA}(\mathrm{D}, \alpha, q) := \min_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}', \alpha).$$

a) We wish to take a minimum value over all repairs, not to multiply values.
b) In semiring setting implication (read: negation) is problematic to define.
c) Rewriting should retain some benefits of FO-rewriting (e.g., complexity-wise).

Logic for rewriting $\mathrm{mCA}(\mathrm{D}, \alpha, q)$

Ingredients for rewriting a conjunctive query $q_{\mathrm{path}} = \exists x \exists y \exists z (R(x; y) \wedge S(y; z))$:

$$\mathrm{CA}(\mathrm{D}, q) := \bigcap_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}').$$

[Fuxman and Miller, 2007] rewriting: $\exists x \exists z' (R(x, z') \wedge \forall z (R(x, z) \rightarrow \exists y S(z, y)))$

Semiring setting: Similar rewriting requires care; a) universal quantifier, b) implication.

$$\mathrm{mCA}(\mathrm{D}, \alpha, q) := \min_{\mathrm{D}' \text{ is repair of } \mathrm{D}} q(\mathrm{D}', \alpha).$$

a) We wish to take a minimum value over all repairs, not to multiply values.
b) In semiring setting implication (read: negation) is problematic to define.
c) Rewriting should retain some benefits of FO-rewriting (e.g., complexity-wise).

Logic for rewritings: FO(∇_G)

In semiring semantics, for $q_{\text{path}} = \exists x \exists y \exists z (R(x; y) \wedge S(y; z))$

$$\mathrm{mCA}(\mathrm{D}, q_{\text{path}}) = \min_{\mathrm{D}' \in \mathrm{Rep}(\mathrm{D})} \sum_{a,b,c \in D'} R^{\mathrm{D}'}(a, b) \times S^{\mathrm{D}'}(b, c)$$

$$= \sum_{a \in D} \min_{b \in D : R^{\mathrm{D}}(a,b) \neq 0} (R^{\mathrm{D}}(a, b) \times \min_{c \in D : S^{\mathrm{D}}(b,c) \neq 0} S^{\mathrm{D}}(b, c))$$

This can be rewritten as

$$\exists x \nabla_{R(x,y)} y. (R(x, y) \times \nabla_{S(y,z)} z. S(y, z)).$$

if we interpret the quantifier ∇_G as sort of a minimisation over a guard G.

$\nabla_G x$ is not a satisfactory quantifier; we show how to express it without a guard!

Logic for rewritings: FO(∇_G)

In semiring semantics, for $q_{\text{path}} = \exists x \exists y \exists z (R(x;y) \wedge S(y;z))$

$$\mathrm{mCA}(\mathrm{D}, q_{\text{path}}) = \min_{\mathrm{D}' \in \mathrm{Rep}(\mathrm{D})} \sum_{a,b,c \in D'} R^{\mathrm{D}'}(a,b) \times S^{\mathrm{D}'}(b,c)$$

$$= \sum_{a \in D} \min_{b \in D : R^{\mathrm{D}}(a,b) \neq 0} (R^{\mathrm{D}}(a,b) \times \min_{c \in D : S^{\mathrm{D}}(b,c) \neq 0} S^{\mathrm{D}}(b,c))$$

This can be rewritten as

$$\exists x \nabla_{R(x,y)} y. (R(x,y) \times \nabla_{S(y,z)} z. S(y,z)).$$

if we interpret the quantifier ∇_G as sort of a minimisation over a guard G.

$\nabla_G\, x$ is not a satisfactory quantifier; we show how to express it without a guard!

Logic for rewritings: $\mathcal{L}_K$

The syntax of $\mathcal{L}_K$, for naturally ordered positive semiring K, is:

$$\varphi \coloneqq R(\vec{x}) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \nabla x \varphi(x) \mid \overline{\mathrm{Supp}}(\varphi).$$

Semantics is the semiring semantics for FO: $\vee$ is addition, $\wedge$ is multiplication, $\exists x$ is aggregate sum, $R(\alpha(\vec{x}))$ is the annotation given by the K-relation R, and $x = y$ is the Boolean truth value of the identity.

Quantifier ∇ corresponds to minimisation and $\overline{\mathrm{Supp}}$ is a weak negation.

$$\nabla x\, \varphi(x)(\mathrm{D}, \alpha) = \min_{a \in D} \varphi(\mathrm{D}, \alpha(a/x)) \qquad \overline{\mathrm{Supp}}(\varphi)(\mathrm{D}, \alpha) = \begin{cases} 1 & \text{if } \varphi(\mathrm{D}, \alpha) = 0 \\ 0 & \text{otherwise,} \end{cases}$$

$\varphi(\mathrm{D}, \alpha)$ is the value of the formula φ in structure D and assignment α.

Logic for rewritings: $\mathcal{L}_K$

The syntax of $\mathcal{L}_K$, for naturally ordered positive semiring K, is:

$$\varphi \coloneqq R(\vec{x}) \,|\, x = y \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi \,|\, \exists x\, \varphi \,|\, \nabla x \varphi(x) \,|\, \overline{\mathrm{Supp}}(\varphi).$$

Semantics is the semiring semantics for FO: $\vee$ is addition, $\wedge$ is multiplication, $\exists x$ is aggregate sum, $R(\alpha(\vec{x}))$ is the annotation given by the K-relation R, and $x = y$ is the Boolean truth value of the identity.

Quantifier ∇ corresponds to minimisation and $\overline{\mathrm{Supp}}$ is a weak negation.

$$\nabla x\, \varphi(x)(\mathrm{D}, \alpha) = \min_{a \in D} \varphi(\mathrm{D}, \alpha(a/x)) \qquad \overline{\mathrm{Supp}}(\varphi)(\mathrm{D}, \alpha) = \begin{cases} 1 & \text{if } \varphi(\mathrm{D}, \alpha) = 0 \\ 0 & \text{otherwise,} \end{cases}$$

$\varphi(\mathrm{D}, \alpha)$ is the value of the formula φ in structure D and assignment α.

Expressing guarded minimisation without guards

The formula $\nabla x\, \varphi(x)$ computes the minimum value of $\varphi(a/x)$, where a ranges over the active domain of the database. When we want a to range over the support of some definable predicate we use the following shorthand

$$\nabla_G z.\, \varphi(\vec{y}, z) := \nabla z. \Big(\big(\overline{\mathrm{Supp}}(G(\vec{y}, z)) \wedge \exists z \varphi(\vec{y}, z) \wedge \chi\big) \vee \big(\varphi(\vec{y}, z) \wedge \chi\big) \Big),$$

where $G, \varphi \in \mathcal{L}_K$ and $\chi := \mathrm{Supp}(\exists z G(\vec{y}, z))$.

Above, χ makes the evaluation to 0, if the guard is "empty".

Expressing guarded minimisation without guards

The formula $\nabla x\, \varphi(x)$ computes the minimum value of $\varphi(a/x)$, where a ranges over the active domain of the database. When we want a to range over the support of some non-empty definable predicate we use the following shorthand

$$\nabla_G z.\, \varphi(\vec{y}, z) := \nabla z. \Big(\big(\overline{\mathrm{Supp}}(G(\vec{y}, z)) \wedge \exists z \varphi(\vec{y}, z)\big) \vee \varphi(\vec{y}, z) \Big),$$

Expressing guarded minimisation without guards

The formula $\nabla x\, \varphi(x)$ computes the minimum value of $\varphi(a/x)$, where a ranges over the active domain of the database. When we want a to range over the support of some non-empty definable predicate we use the following shorthand

$$\nabla_G z.\, \varphi(\vec{y}, z) := \nabla z. \Big(\big(\overline{\mathrm{Supp}}(G(\vec{y}, z)) \wedge \exists z \varphi(\vec{y}, z)\big) \vee \varphi(\vec{y}, z) \Big),$$

Proposition

If G and φ are $\mathcal{L}_K$-formulae, D is a K-database, and α is an assignment, we have that

$$\nabla_G x. \varphi(x)(D, \alpha) = \min_{a \in D : G(D, \alpha(a/x)) \neq 0} \varphi(D, \alpha(a/x)).$$

Rewritability theorem

Theorem ([Koutris and Wijsen, 2017])

Let q be a self-join free conjunctive query and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{CA}(q, \Sigma)$ is FO-rewritable.

Theorem

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

The rewriting of $\mathrm{mCA}_K(q)$ is defined recursively starting from an un-attacked atom R:

$$\exists \vec{y}_{\vec{x}} \nabla_{R(\vec{y};\vec{z})} \vec{z}_{\vec{x}}.\, \mathrm{mCA}_K(q[\vec{y}_{\vec{x}}, \vec{z}_{\vec{x}}] \setminus R(\vec{y}; \vec{z})).$$

Rewritability theorem

Theorem ([Koutris and Wijsen, 2017])

Let q be a self-join free conjunctive query and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{CA}(q, \Sigma)$ is FO-rewritable.

Theorem

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

The rewriting of $\mathrm{mCA}_K(q)$ is defined recursively starting from an un-attacked atom R:

$$\exists \vec{y}_{\vec{x}} \nabla_{R(\vec{y};\vec{z})} \vec{z}_{\vec{x}}.\, \mathrm{mCA}_K(q[\vec{y}_{\vec{x}}, \vec{z}_{\vec{x}}] \setminus R(\vec{y}; \vec{z})).$$

Why is rewriting in $\mathcal{L}_K$ a nice thing to have?

Data complexity of FO is DLOGTIME-uniform AC^0. How about $\mathcal{L}_K$?

The correct model of computation to relate $\mathcal{L}_K$ is a variant of arithmetic AC^0 with gates corresponding to semiring operations!

- The model needs to be able to take semiring values as input.
- It needs to have gates for evaluating $\mathcal{L}_K$-formulae compositionally:
  - $+$ -gate for disjunction (fan-in 2),
  - $\times$ -gate for conjunction (fan-in 2),
  - $+$ -gate for existential quantifier (unbounded fan-in),
  - *min* -gate for ∇ (unbounded fan-in),
  - $\overline{\mathrm{Supp}}$-gate (fan-in 1).

Why is rewriting in $\mathcal{L}_K$ a nice thing to have?

Data complexity of FO is DLOGTIME-uniform AC^0. How about $\mathcal{L}_K$?

The correct model of computation to relate $\mathcal{L}_K$ is a variant of arithmetic AC^0 with gates corresponding to semiring operations!

- The model needs to be able to take semiring values as input.
- It needs to have gates for evaluating $\mathcal{L}_K$-formulae compositionally:
  - $+$ -gate for disjunction (fan-in 2),
  - $\times$ -gate for conjunction (fan-in 2),
  - $+$ -gate for existential quantifier (unbounded fan-in),
  - *min* -gate for ∇ (unbounded fan-in),
  - $\overline{\mathrm{Supp}}$-gate (fan-in 1).

K-circuits

Definition

Let K be a naturally ordered positive semiring.
A *K-circuit* is a finite simple directed acyclic graph of labeled nodes (i.e., gates) s.t.

- there are gates labeled *input*, with indegree 0,
- there are gates labeled *constant*, with indegree 0 and labeled with a $c \in K$,
- there are gates labeled *addition*, *multiplication*, *min*, and $\overline{\mathrm{Supp}}$,
- exactly one gate of outdegree 0 is additionally labeled *output*,

Addition, multiplication, and min gates have arbitrary in-degree.
Depth of a circuit is the length of the longest path from an input to an output gate.
Size of a circuit is the number of gates in it.

K-circuits

Definition

Let K be a naturally ordered positive semiring.

A *K-circuit* is a finite simple directed acyclic graph of labeled nodes (i.e., gates) s.t.

- there are gates labeled *input*, with indegree 0,
- there are gates labeled *constant*, with indegree 0 and labeled with a $c \in K$,
- there are gates labeled *addition*, *multiplication*, *min*, and $\overline{\mathrm{Supp}}$,
- exactly one gate of outdegree 0 is additionally labeled *output*,

Addition, multiplication, and min gates have arbitrary in-degree.

Depth of a circuit is the length of the longest path from an input to an output gate.

Size of a circuit is the number of gates in it.

K-circuits

Definition

Let K be a naturally ordered positive semiring.
A K*-circuit* is a finite simple directed acyclic graph of labeled nodes (i.e., gates) s.t.

- there are gates labeled *input*, with indegree 0,
- there are gates labeled *constant*, with indegree 0 and labeled with a $c \in K$,
- there are gates labeled *addition*, *multiplication*, *min*, and $\overline{\mathrm{Supp}}$,
- exactly one gate of outdegree 0 is additionally labeled *output*,
- input gates are ordered.

Addition, multiplication, and min gates have arbitrary in-degree.
Depth of a circuit is the length of the longest path from an input to an output gate.
Size of a circuit is the number of gates in it.

A circuit computes functions of type $K^n \to K$.

Circuit families

- A K-circuit C_n computes a function $f_n \colon K^n \to K$, for some $n \in \mathbb{N}$.
- A family of K-circuits $(C_n)_{n\in\mathbb{N}}$ computes a function $f_{\mathcal{C}} \colon K^* \to K$.
- To consider $(C_n)_{n\in\mathbb{N}}$ as an algorithm, $n \mapsto C_n$ should be computable.
- DLOGTIME-uniform $\mathrm{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$
  - $(C_n)_{n\in\mathbb{N}}$ is a family of constant depth polynomial size (in n) circuits,
  - indegree of $\times$-gates is 2,
  - there is a DLOGTIME algorithm that describes C_n, given n.

Fact

DLOGTIME-*uniform* $\mathrm{AC}^0_B(+, \times_2, \min, \overline{\mathrm{Supp}})$ *is* DLOGTIME-*uniform* AC^0.

Proposition

Data complexity of $\mathcal{L}_K$ *is in* DLOGTIME-*uniform* $\mathrm{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$.

Circuit families

- A K-circuit C_n computes a function $f_n \colon K^n \to K$, for some $n \in \mathbb{N}$.
- A family of K-circuits $(C_n)_{n\in\mathbb{N}}$ computes a function $f_{\mathcal{C}} \colon K^* \to K$.
- To consider $(C_n)_{n\in\mathbb{N}}$ as an algorithm, $n \mapsto C_n$ should be computable.
- DLOGTIME-uniform $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$
  - $(C_n)_{n\in\mathbb{N}}$ is a family of constant depth polynomial size (in n) circuits,
  - indegree of $\times$-gates is 2,
  - there is a DLOGTIME algorithm that describes C_n, given n.

Fact

DLOGTIME-*uniform* $\mathsf{AC}^0_B(+, \times_2, \min, \overline{\mathrm{Supp}})$ *is* DLOGTIME-*uniform* AC^0.

Proposition

Data complexity of $\mathcal{L}_K$ *is in* DLOGTIME-*uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$.

Circuit families

- A K-circuit C_n computes a function $f_n \colon K^n \to K$, for some $n \in \mathbb{N}$.
- A family of K-circuits $(C_n)_{n\in\mathbb{N}}$ computes a function $f_{\mathcal{C}} \colon K^* \to K$.
- To consider $(C_n)_{n\in\mathbb{N}}$ as an algorithm, $n \mapsto C_n$ should be computable.
- DLOGTIME-uniform $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$
  - $(C_n)_{n\in\mathbb{N}}$ is a family of constant depth polynomial size (in n) circuits,
  - indegree of $\times$-gates is 2,
  - there is a DLOGTIME algorithm that describes C_n, given n.

Fact

DLOGTIME-*uniform* $\mathsf{AC}^0_B(+, \times_2, \min, \overline{\mathrm{Supp}})$ *is* DLOGTIME-*uniform* AC^0.

Proposition

Data complexity of $\mathcal{L}_K$ *is in* DLOGTIME-*uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$.

Circuit families

- A K-circuit C_n computes a function $f_n \colon K^n \to K$, for some $n \in \mathbb{N}$.
- A family of K-circuits $(C_n)_{n\in\mathbb{N}}$ computes a function $f_{\mathcal{C}} \colon K^* \to K$.
- To consider $(C_n)_{n\in\mathbb{N}}$ as an algorithm, $n \mapsto C_n$ should be computable.
- DLOGTIME-uniform $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$
  - $(C_n)_{n\in\mathbb{N}}$ is a family of constant depth polynomial size (in n) circuits,
  - indegree of $\times$-gates is 2,
  - there is a DLOGTIME algorithm that describes C_n, given n.

Fact

DLOGTIME-*uniform* $\mathsf{AC}^0_B(+, \times_2, \min, \overline{\mathrm{Supp}})$ *is* DLOGTIME-*uniform* AC^0.

Proposition

Data complexity of $\mathcal{L}_K$ *is in* DLOGTIME-*uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$.

Why are $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$-circuits nice?

- $\mathsf{AC}^0_k(+, \times_2)$ circuit families compute polynomial functions of constant degree.
- $\mathsf{AC}^0_K(+, \times_2, \min)$-circuits add polynomial many min comparisons between values.
- Addition of $\overline{\mathrm{Supp}}$ gates adds polynomial many comparisons between values and 0.
- Assuming $a \leq b$ comparisons between semiring values can be checked effectively, $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$ families compute in a strong sense polynomial functions.

Theorem (Recap)

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

Data complexity of $\mathcal{L}_K$ is in DLOGTIME-*uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$*, which is computationally nice.*

Why are $AC^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$-circuits nice?

- $AC^0_k(+, \times_2)$ circuit families compute polynomial functions of constant degree.
- $AC^0_K(+, \times_2, \min)$-circuits add polynomial many min comparisons between values.
- Addition of $\overline{\mathrm{Supp}}$ gates adds polynomial many comparisons between values and 0.
- Assuming $a \leq b$ comparisons between semiring values can be checked effectively, $AC^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$ families compute in a strong sense polynomial functions.

Theorem (Recap)

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

Data complexity of $\mathcal{L}_K$ is in DLOGTIME-uniform $AC^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$, which is computationally nice.

Why are $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$-circuits nice?

- $\mathsf{AC}^0_k(+, \times_2)$ circuit families compute polynomial functions of constant degree.
- $\mathsf{AC}^0_K(+, \times_2, \min)$-circuits add polynomial many min comparisons between values.
- Addition of $\overline{\mathrm{Supp}}$ gates adds polynomial many comparisons between values and 0.
- Assuming $a \leq b$ comparisons between semiring values can be checked effectively, $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$ families compute in a strong sense polynomial functions.

Theorem (Recap)

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

Data complexity of $\mathcal{L}_K$ is in DLOGTIME-*uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$*, which is computationally nice.*

Why are $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$-circuits nice?

- $\mathsf{AC}^0_k(+, \times_2)$ circuit families compute polynomial functions of constant degree.
- $\mathsf{AC}^0_K(+, \times_2, \min)$-circuits add polynomial many min comparisons between values.
- Addition of $\overline{\mathrm{Supp}}$ gates adds polynomial many comparisons between values and 0.
- Assuming $a \leq b$ comparisons between semiring values can be checked effectively, $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$ families compute in a strong sense polynomial functions.

Theorem (Recap)

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

Data complexity of $\mathcal{L}_K$ is in DLOGTIME*-uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$*, which is computationally nice.*

Why are $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$-circuits nice?

- $\mathsf{AC}^0_k(+, \times_2)$ circuit families compute polynomial functions of constant degree.
- $\mathsf{AC}^0_K(+, \times_2, \min)$-circuits add polynomial many min comparisons between values.
- Addition of $\overline{\mathrm{Supp}}$ gates adds polynomial many comparisons between values and 0.
- Assuming $a \leq b$ comparisons between semiring values can be checked effectively, $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$ families compute in a strong sense polynomial functions.

Theorem (Recap)

Let K be a naturally ordered positive semiring, q be a self-join free conjunctive query, and Σ a set of key constraints, one for each relation in q. The attack graph of q is acyclic if and only if $\mathrm{mCA}_K(q, \Sigma)$ is $\mathcal{L}_K$-rewritable.

Data complexity of $\mathcal{L}_K$ is in DLOGTIME*-uniform* $\mathsf{AC}^0_K(+, \times_2, \min, \overline{\mathrm{Supp}})$*, which is computationally nice.*

Descriptive complexity theory

Logic and Computation Through the Lens of Semirings (by Helsinki + Hannover, '25)

T. Barlag, N. Fröhlich, T. Hankala, M. Hannula, M. Hirvonen, V. Holzapfel, J.Kontinen, A. Meier, L. Strieker.

- For positive commutative semirings K and the BSS-model of computation:
  - Data complexity of FO_{K} is in P_{K}.
  - Model checking for FO_{K} is in $\mathsf{PSPACE}_{\mathrm{K}}$.
- For positive commutative semirings K and ordered structures:
  - $\mathrm{FO}_{\mathrm{K}}(\mathrm{Arb}_{\mathrm{K}}) = \mathrm{FAC}^0_{\mathrm{K}}$ (non-uniform).

Barlag, T., Fröhlich, N., Hankala, T., Hannula, M., Hirvonen, M., Holzapfel, V., Kontinen, J., Meier, A., and Strieker, L. (2025).
Logic and computation through the lens of semirings.
CoRR, abs/2502.12939.

Barrington, D. A. M., Immerman, N., and Straubing, H. (1990).
On uniformity within nc^1.
J. Comput. Syst. Sci., 41(3):274–306.

Feng, S., Huber, A., Glavic, B., and Kennedy, O. (2019).
Uncertainty annotated databases - A lightweight approach for approximating certain answers.
In *SIGMOD Conference*, pages 1313–1330. ACM.

Fuxman, A. and Miller, R. J. (2007).
First-order query rewriting for inconsistent databases.
J. Comput. Syst. Sci., 73(4):610–635.

Kolaitis, P. G., Pardal, N., and Virtema, J. (2024).
Rewriting consistent answers on annotated data.
CoRR, abs/2412.11661.

Koutris, P. and Wijsen, J. (2017).
Consistent query answering for self-join-free conjunctive queries under primary key constraints.
ACM Trans. Database Syst., 42(2):9:1–9:45.