First-Order Definable Counting-Only Queries

Jelle Hellings[1]

Marc Gyssens[1] Dirk Van Gucht[2] Yuqing Wu[3]

[1] Hasselt University
[2] Indiana University
[3] Pomona College

UHASSELT
KNOWLEDGE IN ACTION

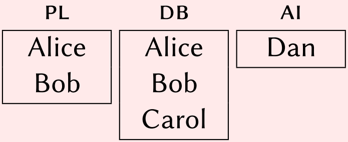
Bag-of-sets datasets and queries

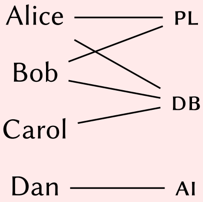
- 'Return students who take at least 2 courses.'

$$\{\langle x\rangle \mid \mathsf{count}(x) \geq 2\}$$

- 'Return pairs of students who take the same number of courses.'

$$\{\langle x, y\rangle \mid (x \neq y) \wedge \mathsf{count}(x) = \mathsf{count}(y)\}$$





Formalization: the bag-of-sets data model

Definition

A *structure* $\mathbf{S}$ over domain $\mathcal{D}$ of objects is a pair $\mathbf{S} = (\mathbf{N}, \gamma)$, with:

- $\mathbf{N}$ a finite set of *set names*;
- $\gamma \subset \mathcal{D} \times \mathbf{N}$ a finite *set-membership* relation.

Let $\mathbf{S} = (\mathbf{N}, \gamma)$, $\text{N} \in \mathbf{N}$ a set name, and $\text{A} \subset \mathcal{D}$ a finite set of objects:

- $\text{objects}(\text{N}; \mathbf{S}) = \{\text{o} \mid (\text{o}, \text{N}) \in \gamma\}$.
- The *cover* is defined by

$$\text{cover}(\text{A}; \mathbf{S}) = \{\text{N} \mid (\text{N} \in \mathbf{N}) \wedge (\text{A} \subseteq \text{objects}(\text{N}; \mathbf{S}))\}.$$

- The *support* is defined by

$$[\![\text{count}(\text{A})]\!]_{\mathbf{S}} = |\text{cover}(\text{A}; \mathbf{S})|.$$

Counting-only queries

- 'Return pairs of distinct students which take a common course.'

$$\{\langle x, y\rangle \mid (x \neq y) \wedge \mathsf{count}(x, y) \geq 1\}$$

- 'Return pairs of distinct students which take the same courses.'

$$\{\langle x, y\rangle \mid (x \neq y) \wedge \mathsf{count}(x, y) = \mathsf{count}(x) = \mathsf{count}(y)\}$$

PL	DB	AI	VR
Alice	Alice	Bob	
Bob	Carol	Carol	

PL	DB	AI	VR
Alice	Alice	Bob	Carol
Bob			
Carol			

$$\begin{aligned}
\mathsf{count}() &= 4;\\
\mathsf{count}(A) &= 2;\\
\mathsf{count}(B) &= 2;\\
\mathsf{count}(C) &= 2;\\
\mathsf{count}(A, B) &= 1;\\
\mathsf{count}(A, C) &= 1;\\
\mathsf{count}(B, C) &= 1.
\end{aligned}$$

Formalization: k-counting-only queries

Definition

Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be two structures.

- $\mathbf{S}_1$ and $\mathbf{S}_2$ are *exactly-k-counting-equivalent* if, for every set of objects A with $|\mathrm{A}| = k$,

$$[\![\mathrm{count}(\mathrm{A})]\!]_{\mathbf{S}_1} = [\![\mathrm{count}(\mathrm{A})]\!]_{\mathbf{S}_2}.$$

- $\mathbf{S}_1$ and $\mathbf{S}_2$ are *k-counting-equivalent* if they are exactly-j-counting-equivalent for every j, $0 \leq j \leq k$.

Definition

Let $\mathtt{Q}$ be a query.

- $\mathtt{Q}$ is *k-counting-only* if $[\![\mathtt{Q}]\!]_{\mathbf{S}_1} = [\![\mathtt{Q}]\!]_{\mathbf{S}_2}$ for every pair of k-counting-equivalent structures $\mathbf{S}_1$ and $\mathbf{S}_2$.
- $\mathtt{Q}$ is *counting-only* if it is k-counting-only for some k.

Proving that a query is not 2-counting-only

'Does there exist a course taken by 3 students?'

$$\{\langle\rangle \mid \exists x \exists y \exists z\, ((x \neq y \wedge x \neq z \wedge y \neq z) \wedge \mathsf{count}(x, y, z) \geq 1)\}$$

This query is clearly 3-counting-only

Is this query 2-counting-only?

It can distinguish between 2-counting-equivalent structures!

PL	DB	AI	VR
Alice	Alice	Bob	
Bob	Carol	Carol	

PL	DB	AI	VR
Alice	Alice	Bob	Carol
Bob			
Carol			

Proving that a query is 2-counting-only

'Does there exists a student who takes courses that are taken by a pair of other students?'

$$\{\langle\rangle \mid \exists x \exists y_1 \exists y_2\ (x \neq y_1) \wedge (x \neq y_2) \wedge (y_1 \neq y_2) \wedge \\ \mathrm{count}(y_1) = \mathrm{count}(x, y_1) \wedge \mathrm{count}(y_2) = \mathrm{count}(x, y_2) \wedge \\ \mathrm{count}(x) = \mathrm{count}(x, y_1) + \mathrm{count}(x, y_2) - \mathrm{count}(x, y_1, y_2)\}.$$

This query is clearly 3-counting-only.

Is this query 2-counting-only?

It is equivalent to the 2-counting-only query

$$\{\langle\rangle \mid \exists x \exists y_1 \exists y_2\ (x \neq y_1) \wedge (x \neq y_2) \wedge (y_1 \neq y_2) \wedge \\ \mathrm{count}(y_1) = \mathrm{count}(x, y_1) \wedge \mathrm{count}(y_2) = \mathrm{count}(x, y_2) \wedge \\ \mathrm{count}(x) = \mathrm{count}(x, y_1) + \mathrm{count}(x, y_2) - \mathrm{count}(y_1, y_2)\}.$$

First-order logic and counting-only queries

SyCALC: first-order logic on bag-of-sets

Definition

SyCALC formulae are defined by the grammar

$$e := \Gamma(x, X) \mid x = y \mid X = Y \mid e \vee e \mid \neg e \mid \exists x\ e \mid \exists X\ e.$$

A *SyCALC query* is a formula without free set name variables.

Example

- 'Return students who take at least 2 courses.'

$$\{\langle x\rangle \mid \mathrm{count}(x) \geq 2\} \quad \longrightarrow \quad \{\langle x\rangle \mid \exists Y \exists Z\ ((Y \neq Z) \wedge \Gamma(x, Y) \wedge \Gamma(x, Z))\}$$

- 'Return pairs of distinct students which take a common course.'

$$\{\langle x, y\rangle \mid (x \neq y) \wedge \mathrm{count}(x, y) \geq 1\} \quad \longrightarrow \quad \{\langle x, y\rangle \mid (x \neq y) \wedge \exists C\ (\Gamma(x, C) \wedge \Gamma(y, C))\}$$

k-SyCALC: k-counting-only SyCALC

Definition

k-SyCALC, $k \geq 0$, denotes the *k-counting-only SyCALC queries*.

Proposition

Let $\mathbf{Q}_1$ and $\mathbf{Q}_2$ be k-SyCALC queries:

- $\mathbf{Q}_1 \lor \mathbf{Q}_2$ *is in k-SyCALC.*
- $\neg\mathbf{Q}_1$ *is in k-SyCALC.*
- $\exists x\ \mathbf{Q}_1$ *is in k-SyCALC.*

Hence: k-SyCALC is closed under disjunction, negation, and object quantification.

Not all counting-only queries are in SyCALC

- ‘Return pairs of students who take the same number of courses.’

  $$\{\langle x, y\rangle \mid (x \neq y) \wedge \mathrm{count}(x) = \mathrm{count}(y)\}.$$

  This query is 1-counting-only, but not first-order definable.
- ‘Are there an even number of courses?’

  $$\{\langle\rangle \mid \exists k \;\; \mathrm{count}() = 2k\}.$$

  This query is 0-counting-only, but not first-order definable.

Not all SyCALC queries are counting-only

- 'Does there exist a course followed by all students?'

$$\{\langle\rangle \mid \exists C \forall x \exists Y\, (\Gamma(x, Y) \implies \Gamma(x, C))\}$$

- 'Does there exist a course followed by no students?'

$$\{\langle\rangle \mid \exists C \forall x\, (\neg\Gamma(x, C))\}$$

How to prove this?

Find k-counting equivalent structures distinguished by these queries (for every k).

Constructing k-counting equivalent structures

Example: $k = 3$.

s_1
A
B
C
D

Constructing k-counting equivalent structures

Example: $k = 3$.

S_1
A
B
C
D

T_1	T_2	T_3	T_4
A	A	A	B
B	B	C	C
C	D	D	D

Constructing k-counting equivalent structures

Example: $k = 3$.

S_1	S_2	S_3	S_4	S_5	S_6	S_7
A	A	A	A	B	B	C
B	B	C	D	C	D	D
C						
D						

T_1	T_2	T_3	T_4
A	A	A	B
B	B	C	C
C	D	D	D

Constructing k-counting equivalent structures

Example: $k = 3$.

S_1	S_2	S_3	S_4	S_5	S_6	S_7
A	A	A	A	B	B	C
B	B	C	D	C	D	D
C						
D						

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
A	A	A	B	A	B	C	D
B	B	C	C				
C	D	D	D				

Constructing k-counting equivalent structures

Example: $k = 3$.

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
A	A	A	A	B	B	C	
B	B	C	D	C	D	D	
C							
D							

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
A	A	A	B	A	B	C	D
B	B	C	C				
C	D	D	D				

Constructing k-counting equivalent structures

Example: $k = 3$.

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
A	A	A	A	B	B	C	
B	B	C	D	C	D	D	
C							
D							

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
A	A	A	B	A	B	C	D
B	B	C	C				
C	D	D	D				

Structures are 3-counting equivalent, but not 4-counting equivalent.

A counting-only fragment of SyCALC

A powerful observation

Many queries can be expressed using simple *generalized count* terms.

Example

'Does there exists a student who takes courses that are taken by a pair of other students?'

$$\begin{aligned}\{\langle\rangle \mid \exists x \exists y_1 \exists y_2\ &(x \neq y_1) \wedge (x \neq y_2) \wedge (y_1 \neq y_2) \wedge \\ &\mathsf{count}(y_1) = \mathsf{count}(x, y_1) \wedge \mathsf{count}(y_2) = \mathsf{count}(x, y_2) \wedge \\ &\mathsf{count}(x) = \mathsf{count}(x, y_1) + \mathsf{count}(x, y_2) - \mathsf{count}(x, y_1, y_2)\}.\end{aligned}$$

This query is equivalent to

$$\begin{aligned}\{\langle\rangle \mid \exists x \exists y_1 \exists y_2\ &(x \neq y_1) \wedge (x \neq y_2) \wedge (y_1 \neq y_2) \wedge \\ &\mathsf{gcount}(x; y_1, y_2) = 0 \wedge \mathsf{gcount}(y_1; x) = 0 \wedge \mathsf{gcount}(y_2; x) = 0\}.\end{aligned}$$

QuineCALC and SimpleCALC

Terms $\mathrm{count}(\mathcal{S}) \geq c$ or $\mathrm{gcount}(\mathcal{X}; \mathcal{Y}) \geq c$ are simple to express:

$$\mathrm{gcount}(\mathcal{X}; \mathcal{Y}) \geq c = \exists Z_1 \dots \exists Z_c \Big(\bigwedge_{1 \leq i < j \leq c} (Z_i \neq Z_j) \wedge \bigwedge_{x \in \mathcal{X}} \big(\Gamma(x, Z_1) \wedge \cdots \wedge \Gamma(x, Z_c)\big) \wedge \bigwedge_{y \in \mathcal{Y}} \big(\neg\Gamma(y, Z_1) \wedge \cdots \wedge \neg\Gamma(y, Z_c)\big)\Big).$$

QuineCALC and SimpleCALC

Terms $\mathrm{count}(\mathcal{S}) \geq c$ or $\mathrm{gcount}(\mathcal{X}; \mathcal{Y}) \geq c$ are simple to express:

$$\mathrm{gcount}(\mathcal{X}; \mathcal{Y}) \geq c = \exists Z_1 \ldots \exists Z_c$$
$$\Big(\bigwedge_{1 \leq i < j \leq c} (Z_i \neq Z_j) \wedge \bigwedge_{x \in \mathcal{X}} \big(\Gamma(x, Z_1) \wedge \cdots \wedge \Gamma(x, Z_c)\big) \wedge$$
$$\bigwedge_{y \in \mathcal{Y}} \big(\neg\Gamma(y, Z_1) \wedge \cdots \wedge \neg\Gamma(y, Z_c)\big)\Big).$$

Definition

- QuineCALC-k consist of all SyCALC queries that do not use object quantification and with at most k free object variables.
- SimpleCALC-k consists of all queries that are built from QuineCALC-k queries using disjunction, negation, and object quantification.

Every SimpleCALC-k query is in k-SyCALC

Proposition

Every SimpleCALC-k query is k-counting-only.

- SimpleCALC-k are built from QuineCALC-k queries using disjunction, negation, and object quantification.
- Closure results of k-SyCALC apply to SimpleCALC-k.

Hence: prove that QuineCALC-k queries are k-counting-only!

Every QuineCALC-k query is in k-SyCALC

Definition

Let $\mathbf{S} = (\mathbf{N}, \gamma)$ be a structure. If A is a set of objects, then $\mathbf{S}|_{\mathrm{A}}$ denotes the structure $(\mathbf{N}, \gamma \cap (\mathrm{A} \times \mathbf{N}))$.

Lemma

Let $\mathbf{S} = (\mathbf{N}, \gamma)$ be a structure and $\mathtt{Q}(x_1, \ldots, x_k)$ be a QuineCALC-k query. We have

$$\langle \mathrm{o}_1, \ldots, \mathrm{o}_k \rangle \in [\![\mathtt{Q}]\!]_{\mathbf{S}} \iff \langle \mathrm{o}_1, \ldots, \mathrm{o}_k \rangle \in [\![\mathtt{Q}]\!]_{\mathbf{S}|_{\{\mathrm{o}_1, \ldots, \mathrm{o}_k\}}}.$$

Every QuineCALC-k query is in k-SyCALC

Definition

Let $\mathbf{S} = (\mathbf{N}, \gamma)$ be a structure. If A is a set of objects, then $\mathbf{S}|_{\mathrm{A}}$ denotes the structure $(\mathbf{N}, \gamma \cap (\mathrm{A} \times \mathbf{N}))$.

Lemma

Let $\mathbf{S} = (\mathbf{N}, \gamma)$ be a structure and $\mathtt{Q}(x_1, \ldots, x_k)$ be a QuineCALC-k query. We have

$$\langle o_1, \ldots, o_k \rangle \in [\![\mathtt{Q}]\!]_{\mathbf{S}} \iff \langle o_1, \ldots, o_k \rangle \in [\![\mathtt{Q}]\!]_{\mathbf{S}|_{\{o_1, \ldots, o_k\}}}.$$

Lemma

Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be two k-counting-equivalent structures. If A is a set of objects with $|\mathrm{A}| \leq k$, then $\mathbf{S}_1|_{\mathrm{A}}$ is isomorphic to $\mathbf{S}_2|_{\mathrm{A}}$.

A counting-only hierarchy

Theorem

Let $k \geq 0$. There are

1. *QuineCALC-(k+1) queries and*
2. *Boolean SimpleCALC-(k+1) queries*

that are not k-counting-only.

Proof.

1. 'Return $k+1$ objects that occur together.'

$$\{\langle x_1, \dots, x_{k+1}\rangle \mid \mathsf{count}(x_1, \dots, x_{k+1}) \geq 1)\} = \{\langle x_1, \dots, x_{k+1}\rangle \mid \exists X \left(\bigwedge_{1 \leq i \leq k+1} \Gamma(x_i, X)\right)\}$$

2. 'Does there exist a set with $k+1$ objects?'

$$\exists x_1 \dots x_{k+1} \left(\left(\bigwedge_{1 \leq i < j \leq k+1} (x_i \neq x_j)\right) \wedge \mathsf{count}(x_1, \dots, x_{k+1}) \geq 1\right).$$

Limits to SimpleCALC-k

Proposition

Let $\mathbf{Q}(x_1, \ldots, x_m)$ *be a SimpleCALC-k query. There exists an n such that*

$$\langle \mathrm{o}_1, \ldots, \mathrm{o}_k \rangle \in [\![\mathbf{Q}]\!]_{\mathbf{S}} \iff \langle \mathrm{o}_1, \ldots, \mathrm{o}_k \rangle \in [\![\mathbf{Q}]\!]_{\mathbf{S}|_{\mathrm{A}}}$$

for some A *with* $|\mathrm{A}| \leq n$ *and* $\{\mathrm{o}_1, \ldots, \mathrm{o}_k\} \subseteq \mathrm{A}$.

Example

'Return students that take courses with another student.'

$$\mathbf{Q} = \{\langle x \rangle \mid \exists y \exists C\ (\Gamma(x, C) \land \Gamma(y, C))\}$$

Not all 2-SyCALC queries are in SimpleCALC

'Is each course followed by a unique student?'

$$\texttt{set-ids} = \{\langle\rangle \mid \forall C \exists x\, (\Gamma(x, C) \land \neg\exists Y\, ((X \neq Y) \land \Gamma(x, Y)))\}$$

Proposition

Query `set-ids` *is* 2*-counting-only, but not* 1*-counting-only.*

Not all 2-SyCALC queries are in SimpleCALC

'Is each course followed by a unique student?'

$$\texttt{set-ids} = \{\langle\rangle \mid \forall C \exists x\, (\Gamma(x, C) \land \neg\exists Y\, ((X \neq Y) \land \Gamma(x, Y)))\}$$

Proposition

Query `set-ids` *is* 2*-counting-only, but not* 1*-counting-only.*

Proof.

Assume we have n set names. The query `set-ids` is equivalent to

$$\{\langle\rangle \mid \exists x_1 \ldots \exists x_n\, (\bigwedge_{1 \leq i \leq n} \mathsf{count}(x_j) = 1) \land (\bigwedge_{1 \leq i < j \leq n} \mathsf{gcount}(x_i; x_j) = 0)\}.$$

Not all 2-SyCALC queries are in SimpleCALC

'Is each course followed by a unique student?'

$$\texttt{set-ids} = \{\langle\rangle \mid \forall C \exists x\, (\Gamma(x, C) \wedge \neg \exists Y\, ((X \neq Y) \wedge \Gamma(x, Y)))\}$$

Proposition

Query `set-ids` *is* 2*-counting-only, but not* 1*-counting-only.*

Proof.

Assume we have n set names. The query `set-ids` is equivalent to

$$\{\langle\rangle \mid \exists x_1 \ldots \exists x_n\, (\bigwedge_{1 \leq i \leq n} \mathsf{count}(x_j) = 1) \wedge (\bigwedge_{1 \leq i < j \leq n} \mathsf{gcount}(x_i; x_j) = 0)\}.$$

Not 1-counting only:

S_1	S_2
A	
B	

T_1	T_2
A	B

Decision problems and counting-only queries

Classical decision problems

- Satisfiability.
- Validity.
- Query containment.
- Query equivalence.

Classical decision problems

- Satisfiability.
- Validity.
- Query containment.
- Query equivalence.

Satisfiability and first-order logic

Satisfiability of first-order logic is *not decidable*.

Satisfiability is decidable for FO when *restricted* to

- unary predicates (monadic first-order logic);
- two variables (FO^2);
- formulae of the form
  - $\exists \cdots \exists \forall \exists \cdots \exists$ (the Ackermann class);
  - $\exists \cdots \exists \forall \forall \exists \cdots \exists$ (the Gödel class);
  - $\exists \cdots \exists \forall \cdots \forall$ (the Schönfinkel-Bernays class).

What if we restrict FO to counting-only queries?

Satisfiability of 2-SyCALC is undecidable 1/3

FO is undecidable when querying undirected unlabeled graphs without self-loops.

Satisfiability of 2-SyCALC is undecidable 1/3

FO is undecidable when querying undirected unlabeled graphs without self-loops.

Encode graphs as structures

Encode graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ as the structure $\mathsf{enc}(\mathbf{G}) = (\mathbf{V}, \gamma)$ with

$$\gamma = \{(\{m, n\}, m), (\{m, n\}, n) \mid (m, n) \in \mathbf{E}\} \cup \{(\{n\}, n) \mid n \in \mathbf{V}\}.$$

$\mathsf{enc}(\mathbf{G})$ always satisfies the SyCALC query:

$$\begin{aligned}\texttt{enc-graph} = \texttt{set-ids} \land \\ &\forall e \exists M \exists N\, (((M \neq N) \land \Gamma(e, M) \land \Gamma(e, N)) \Rightarrow \\ &\qquad \forall X\, ((M \neq X) \land (N \neq X) \Rightarrow \neg\Gamma(e, X))).\end{aligned}$$

Satisfiability of 2-SyCALC is undecidable 2/3

FO is undecidable when querying undirected unlabeled graphs without self-loops.

Encode FO queries as SyCALC queries

Encode Boolean FO query φ by $\mathsf{enc}(\varphi) = \texttt{enc-graph} \wedge \tau(\varphi)$ with

$$\begin{aligned}
\tau(M = N) &\equiv M = N;\\
\tau(\mathbf{E}(M, N)) &\equiv (M \neq N) \wedge \exists e\, (\Gamma(e, M) \wedge \Gamma(e, N));\\
\tau(e_1 \vee e_2) &\equiv \tau(e_1) \vee \tau(e_2);\\
\tau(\neg e) &\equiv \neg\tau(e);\\
\tau(\exists N\, e) &\equiv \exists N\, \tau(e).
\end{aligned}$$

Lemma

Let $\mathbf{G}$ *be a graph and let* φ *be a Boolean FO query. Then,* $[\![\varphi]\!]_{\mathbf{G}} = [\![\mathsf{enc}(\varphi)]\!]_{\mathsf{enc}(\mathbf{G})}$.

Satisfiability of 2-SyCALC is undecidable 3/3

FO is undecidable when querying undirected unlabeled graphs without self-loops.

Proposition

If φ is a Boolean FO query, then $\mathsf{enc}(\varphi)$ *is a* 2*-SyCALC query.*

Proof.

If $\mathbf{S}_1$ and $\mathbf{S}_2$ are structures that are 2-counting-equivalent, and $[\![\texttt{set-ids}]\!]_{\mathbf{S}_1} = [\![\texttt{set-ids}]\!]_{\mathbf{S}_2} = \texttt{true}$, then $\mathbf{S}_1$ and $\mathbf{S}_2$ are isomorphic.

Lemma

Let φ be a Boolean FO query. If there exists a structure $\mathbf{S}$ satisfying $\mathsf{enc}(\varphi)$*, then we can construct from $\mathbf{S}$ a graph satisfying φ.*

Satisfiability of 2-SyCALC is undecidable 3/3

FO is undecidable when querying undirected unlabeled graphs without self-loops.

Proposition

If φ is a Boolean FO query, then $\mathsf{enc}(\varphi)$ *is a* 2*-SyCALC query.*

Proof.

If $\mathbf{S}_1$ and $\mathbf{S}_2$ are structures that are 2-counting-equivalent, and $[\![\texttt{set-ids}]\!]_{\mathbf{S}_1} = [\![\texttt{set-ids}]\!]_{\mathbf{S}_2} = \texttt{true}$, then $\mathbf{S}_1$ and $\mathbf{S}_2$ are isomorphic.

Lemma

Let φ be a Boolean FO query. If there exists a structure $\mathbf{S}$ satisfying $\mathsf{enc}(\varphi)$*, then we can construct from $\mathbf{S}$ a graph satisfying φ.*

Theorem

The satisfiability problem is undecidable for 2*-SyCALC queries.*

What about weaker counting-only languages?

- 1-SyCALC.
- SimpleCALC.
- (QuineCALC).

What about weaker counting-only languages?

- 1-SyCALC.
- SimpleCALC.
- (QuineCALC).

These classes have *the finite model property*.

Satisfiability of 1-SyCALC is decidable 1/2

Definition

Let $k, d \geq 0$. Structures $\mathbf{S}_1 = (\mathbf{N}_1, \gamma_1)$ and $\mathbf{S}_2 = (\mathbf{N}_2, \gamma_2)$ are *d-partial k-counting-equivalent* if, for every pair of sets of objects I and E with $|\mathsf{I} \cup \mathsf{E}| \leq k$, either

1. $[\![\mathsf{gcount}(\mathsf{I}; \mathsf{E})]\!]_{\mathbf{S}_1} = [\![\mathsf{gcount}(\mathsf{I}; \mathsf{E})]\!]_{\mathbf{S}_2} \leq d$; or
2. $d < [\![\mathsf{gcount}(\mathsf{I}; \mathsf{E})]\!]_{\mathbf{S}_i} < |\mathbf{N}_i| - d$, $i \in \{1, 2\}$,
3. $|\mathbf{N}_1| - [\![\mathsf{gcount}(\mathsf{I}; \mathsf{E})]\!]_{\mathbf{S}_1} = |\mathbf{N}_2| - [\![\mathsf{gcount}(\mathsf{I}; \mathsf{E})]\!]_{\mathbf{S}_2} \leq d$.

Lemma

Let $\mathtt{Q}$ be a SyCALC query with set name quantifier depth d and let $\mathbf{S}_1$ and $\mathbf{S}_2$ be d-partial k-counting-equivalent structures with $k = |\mathsf{adom}(\mathbf{S}_1)| = |\mathsf{adom}(\mathbf{S}_2)|$. Then $[\![\mathtt{Q}]\!]_{\mathbf{S}_1} = [\![\mathtt{Q}]\!]_{\mathbf{S}_2}$.

Satisfiability of 1-SyCALC is decidable 2/2

Proposition

Let $d \geq 0$, and let $\mathbf{S} = (\mathsf{N}, \gamma)$ be a structure. There exists a structure $\mathbf{S}' = (\mathsf{N}', \gamma')$ with $|\mathsf{N}'| \leq 2d + 1$ such that $\mathbf{S}$ and $\mathbf{S}'$ are d-partial 1-counting-equivalent structures.

Proposition

Let $\mathtt{Q}$ be a 1-SyCALC query with set name quantifier depth d and object quantifier depth r, and let $\mathbf{S} = (\mathsf{N}, \gamma)$ be a structure. Then, $[\![e]\!]_{\mathbf{S}} \neq \emptyset$ if and only if there exists a structure $\mathbf{S}' = (\mathsf{N}', \gamma')$ with $|\mathsf{N}'| \leq 2d + 1$, $|\mathrm{adom}(\mathbf{S}')| \leq r(2d + 1)$, and $[\![e]\!]_{\mathbf{S}'} \neq \emptyset$.

Satisfiability of 1-SyCALC is decidable 2/2

Proposition

Let $d \geq 0$, and let $\mathbf{S} = (\mathsf{N}, \gamma)$ be a structure. There exists a structure $\mathbf{S}' = (\mathsf{N}', \gamma')$ with $|\mathsf{N}'| \leq 2d + 1$ such that $\mathbf{S}$ and $\mathbf{S}'$ are d-partial 1-counting-equivalent structures.

Proposition

Let $\mathtt{Q}$ be a 1-SyCALC query with set name quantifier depth d and object quantifier depth r, and let $\mathbf{S} = (\mathsf{N}, \gamma)$ be a structure. Then, $[\![e]\!]_{\mathbf{S}} \neq \emptyset$ if and only if there exists a structure $\mathbf{S}' = (\mathsf{N}', \gamma')$ with $|\mathsf{N}'| \leq 2d + 1$, $|\mathrm{adom}(\mathbf{S}')| \leq r(2d + 1)$, and $[\![e]\!]_{\mathbf{S}'} \neq \emptyset$.

Theorem

The satisfiability problem is decidable for 1-SyCALC queries.

Satisfiability of SimpleCALC is decidable

Proposition (Reminder)

Let $\mathbf{Q}(x_1, \ldots, x_m)$ *be a SimpleCALC-k query. There exists an n such that*

$$\langle o_1, \ldots, o_k \rangle \in [\![\mathbf{Q}]\!]_{\mathbf{S}} \iff \langle o_1, \ldots, o_k \rangle \in [\![\mathbf{Q}]\!]_{\mathbf{S}|_{\mathrm{A}}}$$

for some A *with* $|\mathrm{A}| \leq n$ *and* $\{o_1, \ldots, o_k\} \subseteq \mathrm{A}$.

Proposition

Let $\mathbf{S} = (\mathbf{N}, \gamma)$ *be a structure with* $|\mathrm{adom}(\mathbf{S})| = z$, *and let* $d \geq 0$. *There exists a structure* $\mathbf{S}' = (\mathbf{N}', \gamma')$ *with* $|\mathbf{N}'| \leq (d+1) \cdot 2^z$ *such that* $\mathbf{S}$ *and* $\mathbf{S}'$ *are* d*-partial* z*-counting-equivalent structures.*

Satisfiability of SimpleCALC is decidable

Proposition (Reminder)

Let $\mathbf{Q}(x_1, \ldots, x_m)$ *be a SimpleCALC-k query. There exists an n such that*

$$\langle o_1, \ldots, o_k \rangle \in [\![\mathbf{Q}]\!]_{\mathbf{S}} \iff \langle o_1, \ldots, o_k \rangle \in [\![\mathbf{Q}]\!]_{\mathbf{S}|_{\mathrm{A}}}$$

for some A *with* $|\mathrm{A}| \leq n$ *and* $\{o_1, \ldots, o_k\} \subseteq \mathrm{A}$.

Proposition

Let $\mathbf{S} = (\mathbf{N}, \gamma)$ *be a structure with* $|\mathrm{adom}(\mathbf{S})| = z$, *and let* $d \geq 0$. *There exists a structure* $\mathbf{S}' = (\mathbf{N}', \gamma')$ *with* $|\mathbf{N}'| \leq (d+1) \cdot 2^z$ *such that* $\mathbf{S}$ *and* $\mathbf{S}'$ *are d-partial z-counting-equivalent structures.*

Theorem

Satisfiability is decidable for SimpleCALC queries, and is NEXPTIME-hard for SimpleCALC-k queries, $k \geq 2$.

Conclusion and discussion

An overview of our results

First-order definable queries (SyCALC)			
QuineCALC	SimpleCALC	c.-o. SyCALC	c.-o. queries
⋮	⋮	⋮	⋮
QuineCALC-3	SimpleCALC-3	3-SyCALC	3-counting-only
QuineCALC-2	SimpleCALC-2	2-SyCALC	2-counting-only
QuineCALC-1	SimpleCALC-1	1-SyCALC	1-counting-only
QuineCALC-0 $\equiv$ SimpleCALC-0 $\equiv$ 0-SyCALC			0-counting-only

An overview of our results

First-order definable queries (SyCALC)			
QuineCALC	SimpleCALC	c.-o. SyCALC	c.-o. queries
$\vdots$	$\vdots$	$\vdots$	$\vdots$
QuineCALC-3	SimpleCALC-3	3-SyCALC	3-counting-only
QuineCALC-2	SimpleCALC-2	2-SyCALC	2-counting-only
QuineCALC-1	SimpleCALC-1	1-SyCALC	1-counting-only
QuineCALC-0 $\equiv$ SimpleCALC-0 $\equiv$ 0-SyCALC			0-counting-only

An overview of our results

First-order definable queries (SyCALC)			
QuineCALC	SimpleCALC	c.-o. SyCALC	c.-o. queries
⋮	⋮	⋮	⋮
QuineCALC-3	SimpleCALC-3	3-SyCALC	3-counting-only
QuineCALC-2	SimpleCALC-2	2-SyCALC	2-counting-only
QuineCALC-1	SimpleCALC-1	1-SyCALC	1-counting-only
QuineCALC-0 $\equiv$ SimpleCALC-0 $\equiv$ 0-SyCALC			0-counting-only

Future work: generalize bag-of-sets

- 'Return students that take courses offered by two departments.'

$$\{\langle x\rangle \mid \exists C \exists D_1 \exists D_2\, (\mathtt{SC}(x, C) \wedge \mathtt{CD}(C, D_1) \wedge \mathtt{CD}(C, D_2))\}$$

- 'Return student-department pairs in which the student only takes courses offered by that department.'

$$\{\langle x, y\rangle \mid |\{z \mid \mathtt{SC}(x, z) \wedge \mathtt{DC}(y, z)\}| = |\{z \mid \mathtt{SC}(x, z)\}|\}$$

Future work

- Are SimpleCALC-k and '$\text{gcount}(\mathcal{X}; \mathcal{Y}) \geq c$'-queries equivalent? E.g. 'Return pairs of distinct students which take the same courses.'

$$\{\langle x, y\rangle \mid (x \neq y) \wedge \text{gcount}(x; y) = \text{gcount}(y; x) = 0\}$$

- Decision problems: is a SimpleCALC-k query l-counting-only? E.g. 'Are there at least two students taking courses?'

$$\{\langle\rangle \mid \exists x \exists y \exists X \exists Y\ ((x \neq y) \wedge \Gamma(x, X) \wedge \Gamma(y, Y))\}$$

- Possibilities for query optimization?