Descriptive Complexity: Complexity of First-Order Logic

Stavros Potsakis

INTER-INSTITUTIONAL GRADUATE PROGRAMM
"Algorithms, Logic and Discrete Mathematics"

Table of Contents

Data, Expression and Combined Complexity

Let us first consider the complexity of the model-checking problem: that is, given a sentence Φ in a logic $\mathcal{L}$ and a structure $\mathfrak{A}$, does $\mathfrak{A}$ satisfy Φ?

There are two parameters of this question: the sentence Φ, and the structure $\mathfrak{A}$.

Data, Expression and Combined Complexity

Suppose we have a structure $\mathfrak{A} \in$STRUCT[σ]. Let A=$\{a_1, a_2, \ldots, a_n\}$.

We choose an order of the universe, say, $a_1 < a_2 < \ldots. < a_n$.

Data, Expression and Combined Complexity

Suppose we have a structure $\mathfrak{A} \in$STRUCT[σ]. Let A=$\{a_1, a_2, \ldots, a_n\}$.

We choose an order of the universe, say, $a_1 < a_2 < \ldots. < a_n$.

The encoding of a k-ary relation $R^{\mathfrak{A}}$ will be as follows:

The jth bit of enc($R^{\mathfrak{A}}$) is 1 if $\vec{a_j} \in R^{\mathfrak{A}}$, and 0 if $\vec{a_j} \notin R^{\mathfrak{A}}$.

Data, Expression and Combined Complexity

Suppose we have a structure $\mathfrak{A} \in$STRUCT[σ]. Let A=$\{a_1, a_2, \ldots, a_n\}$.

We choose an order of the universe, say, $a_1 < a_2 < \ldots . < a_n$.

The encoding of a k-ary relation $R^{\mathfrak{A}}$ will be as follows:

The jth bit of enc($R^{\mathfrak{A}}$) is 1 if $\vec{a_j} \in R^{\mathfrak{A}}$, and 0 if $\vec{a_j} \notin R^{\mathfrak{A}}$.

If $\sigma=\{R_1, R_2, \ldots, R_p\}$ then the encoding of a structure is the concatenation of 0^n1 and all the enc($R_i^{\mathfrak{A}}$)'s:

$$enc(\mathfrak{A}) = 0^n 1 enc(R_1^{\mathfrak{A}}) \cdots enc(R_p^{\mathfrak{A}})$$

Data, Expression and Combined Complexity

Definition

Let $\mathcal{K}$ be a complexity class, and $\mathcal{L}$ a logic. We say that

- the data complexity of $\mathcal{L}$ is $\mathcal{K}$ if for every sentence Φ of $\mathcal{L}$, the language
$$\{\mathrm{enc}(\mathfrak{A}) \mid \mathfrak{A} \models \Phi\}$$
belongs to $\mathcal{K}$;
- the expression complexity of $\mathcal{L}$ is $\mathcal{K}$ if for every finite structure $\mathfrak{A}$, the language
$$\{\mathit{enc}(\Phi) \mid \mathfrak{A} \models \Phi\}$$
belongs to $\mathcal{K}$; and
- the combined complexity of $\mathcal{L}$ is $\mathcal{K}$ if the language
$$\{(\mathrm{enc}(\mathfrak{A}), \mathrm{enc}(\Phi)) \mid \mathfrak{A} \models \Phi\}$$
belongs to $\mathcal{K}$.

Data, Expression and Combined Complexity

The notion of data complexity is most often in the database context.

The notion of expression and combined complexity are often used in verification and model-checking.

Data, Expression and Combined Complexity

The notion of data complexity is most often in the database context.

The notion of expression and combined complexity are often used in verification and model-checking.

We shall see that for most logics of interest, all hardness results for the combined complexity will be shown on very simple structures, thereby giving us matching bounds for expression and combined complexity.

Circuits and FO Queries

Definition

A Boolean circuit with n inputs $x_1,\ldots,x_n$ is a tuple

$$C = (V,E,\lambda,o)$$

where

1. (V,E) is a directed graph with the set of nodes V (which we call gates) and the set of edges E.
2. λ is a function from V to $\{x_1,\ldots,x_n\} \cup \{\wedge, \vee, \neg\}$ such that
   - $\lambda(u) \in \{x_1,\ldots,x_n\}$ implies that u has in-degree 0;
   - $\lambda(u) = \neg$ implies that u has in-degree 1.
3. $o \in V$.

The in-degree of a node is called its fan-in. The size of C is the number of nodes in V; the depth of C is the length of the longest path from a node of in-degree 0 to o.

Circuits and FO Queries

A circuit C computes a Boolean function with n inputs x_1,..,x_n as follows. Suppose we are given values of x_1,...,x_n. We compute the values associated with each node of in-degree 0:

- for a node x_i, it is the value of x_i
- for a node labeled $\vee$ it is false
- for a node labeled $\wedge$ it is true

Circuits and FO Queries

A circuit C computes a Boolean function with n inputs $x_1,..,x_n$ as follows. Suppose we are given values of $x_1,...,x_n$. We compute the values associated with each node of in-degree 0:

- for a node x_i, it is the value of x_i
- for a node labeled $\vee$ it is false
- for a node labeled $\wedge$ it is true

Next we compute the value of each node by induction: if we have a node u with incoming edges from $u_1,..,u_l$ and we know that their values are $a_1,..,a_l$ then the value of u is:

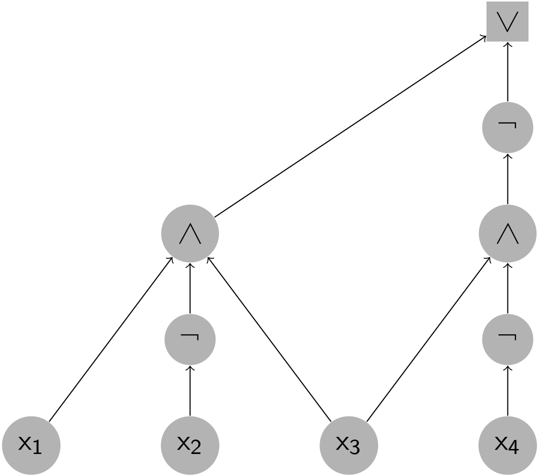
- $a_1 \vee ... \vee a_l$ if $\lambda(u) = \vee$;
- $a_1 \wedge ... \wedge a_l$ if $\lambda(u) = \wedge$;
- $\neg a_1$ if $\lambda(u) = \neg$(in this case we know that $l=1$)

Circuits and FO Queries

An example of a circuit computing the Boolean function $(x_1 \wedge \neg x_2 \wedge x_3) \vee \neg(x_3 \wedge \neg x_4)$:

Circuits and FO Queries

An example of a circuit computing the Boolean function $(x_1 \wedge \neg x_2 \wedge x_3) \vee \neg(x_3 \wedge \neg x_4)$:



Circuits and FO Queries

Definition

A family of circuits is a sequence $\mathbf{C}=(C_n)_{n\geq 0}$ where each C_n is a circuit with n inputs. It accepts the language $L(\mathbf{C})\subseteq \{0,1\}^*$ defined as follows. Let s be a string of length n. It can be viewed as a Boolean vector $\vec{x_s}$ such that the ith component of $\vec{x_s}$ is the ith symbol in s. Then $s\in L(\mathbf{C})$ iff C_n outputs 1 on $\vec{x_s}$.

Circuits and FO Queries

Definition

A family of circuits is a sequence $\mathbf{C}=(C_n)_{n\geq 0}$ where each C_n is a circuit with n inputs. It accepts the language $L(\mathbf{C})\subseteq \{0,1\}^*$ defined as follows. Let s be a string of length n. It can be viewed as a Boolean vector $\vec{x_s}$ such that the ith component of $\vec{x_s}$ is the ith symbol in s. Then $s\in L(\mathbf{C})$ iff C_n outputs 1 on $\vec{x_s}$.

The class of the languages accepted by polynomial-sized constant-depth families of circuits is called nonuniform AC^0.

Circuits and FO Queries

Definition

A family of circuits is a sequence $\mathbf{C}=(\mathrm{C}_n)_{n\geq 0}$ where each C_n is a circuit with n inputs. It accepts the language $\mathrm{L}(\mathbf{C})\subseteq \{0,1\}^*$ defined as follows. Let s be a string of length n. It can be viewed as a Boolean vector $\vec{x_s}$ such that the ith component of $\vec{x_s}$ is the ith symbol in s. Then $s\in\mathrm{L}(\mathbf{C})$ iff C_n outputs 1 on $\vec{x_s}$.

The class of the languages accepted by polynomial-sized constant-depth families of circuits is called nonuniform AC^0.

Example

The language that consists of strings containing at least two ones is in nonuniform AC^0: each circuit C_n, $n > 1$, has $\wedge$-gates for every pair of inputs x_i and x_j, and then the outputs of those $\wedge$-gates form an input for one $\vee$-gate.

Circuits and FO Queries

A class of structures $\mathcal{C} \subseteq$STRUCT[σ] is in nonuniform AC^0 if so is the language $\{enc(\mathfrak{A}) \mid \mathfrak{A} \in C\}$.

Circuits and FO Queries

A class of structures $\mathcal{C} \subseteq$STRUCT[σ] is in nonuniform AC^0 if so is the language $\{$enc($\mathfrak{A}$) $\mid \mathfrak{A} \in$C $\}$.

The class EVEN of structures of the empty vocabulary: that is , $\{\langle A, \emptyset\rangle \mid |A| \;\; mod 2 \;\; = \;\; 0\}$ belongs to nonuniform AC^0 and is not FO-definable. The encoding of such a structure with $|A|$=n is simply 0^n1; hence C_k returns true for odd k and false for even k.

Circuits and FO Queries

Let $\mathcal{P}$ be a collection, finite or infinite, numerical predicates;that is, subset of $\mathbb{N}^k$.For example, $+$ considered as a ternary predicate $\{$(i,j,l)$|$i$+$j$=$l$\}$.

Circuits and FO Queries

Let $\mathcal{P}$ be a collection, finite or infinite, numerical predicates;that is, subset of $\mathbb{N}^k$.For example, $+$ considered as a ternary predicate $\{$(i,j,l)$|$i$+$j$=$l$\}$.

For $\mathcal{P}$ including the linear order, we define FO($\mathcal{P}$) an an extension of FO with atomic formulas of the form P(x_1,...,x_k), for a k-ary P$\in \mathcal{P}$.

Circuits and FO Queries

Let $\mathcal{P}$ be a collection, finite or infinite, numerical predicates;that is, subset of $\mathbb{N}^k$.For example, $+$ considered as a ternary predicate $\{$(i,j,l)|i+j=l$\}$.

For $\mathcal{P}$ including the linear order, we define FO($\mathcal{P}$) an an extension of FO with atomic formulas of the form P(x_1,...,x_k), for a k-ary P$\in \mathcal{P}$.

Suppose $\mathfrak{A}$ is a σ-structure, ant its universe is ordered by $<$ as $a_0 <$....$<a_{n-1}$. Then $\mathfrak{A} \models$P(a_{i_1},....,a_{i_k}) iff the tuple of numbers (i_1,...,i_k) belongs to P.

Circuits and FO Queries

Let $\mathcal{P}$ be a collection, finite or infinite, numerical predicates;that is, subset of $\mathbb{N}^k$.For example, $+$ considered as a ternary predicate $\{$(i,j,l)|i+j=l$\}$.

For $\mathcal{P}$ including the linear order, we define FO($\mathcal{P}$) an an extension of FO with atomic formulas of the form P(x_1,...,x_k), for a k-ary P$\in \mathcal{P}$.

Suppose $\mathfrak{A}$ is a σ-structure, ant its universe is ordered by $<$ as $a_0 <....< a_{n-1}$. Then $\mathfrak{A} \models$P(a_{i_1},....,a_{i_k}) iff the tuple of numbers (i_1,...,i_k) belongs to P.

For example, let $P_2 \subseteq \mathbb{N}$ consist of the even numbers. Then the query EVEN is expressed as an FO($\{<$,$P_2\}$) sentence as follows:

$$\forall x(\forall y(y \leq x) \rightarrow P_2(x))$$

.

Circuits and FO Queries

Theorem

Let $\mathcal{C}$ be a class of structures definable by an FO(All) sentence. Then $\mathcal{C}$ is in nonuniform AC^0. That is,

$$FO(All) \subseteq nonuniform \;\; AC^0$$

Furthermore, for every FO(All) sentence Φ, there is a family of circuits of depth $\mathrm{O}(\|\Phi\|)$ accepting $\{\mathfrak{A} \mid \mathfrak{A} \models \Phi\}$.

Circuits and FO Queries

Proof.

- If k is not of the form $||\mathfrak{A}||$ for some structure $\mathfrak{A}$, then C_k always returns false.
- Assume that k is the size of the encodings of structures $\mathfrak{A}$ with n-elements universe.
- We replace in Φ each quantifier $\exists x \varphi(x, \vec{y})$ or $\forall x \varphi(x, \vec{y})$ with $\bigvee_{c=0}^{n-1} \varphi(c, \vec{y})$ and $\bigwedge_{c=0}^{n-1} \varphi(c, \vec{y})$ respectively and we have Φ'.

Circuits and FO Queries

Proof.

- If k is not of the form $||\mathfrak{A}||$ for some structure $\mathfrak{A}$, then C_k always returns false.
- Assume that k is the size of the encodings of structures $\mathfrak{A}$ with n-elements universe.
- We replace in Φ each quantifier $\exists x \varphi(x, \vec{y})$ or $\forall x \varphi(x, \vec{y})$ with $\bigvee_{c=0}^{n-1} \varphi(c, \vec{y})$ and $\bigwedge_{c=0}^{n-1} \varphi(c, \vec{y})$ respectively and we have Φ'.
- Note that Φ' is a Boolean combination of formulas of type form
  - $P(i_1, \ldots, i_k)$, where P is a numerical predicate and
  - $R(i_1, \ldots, i_k)$, where R is a m-ary symbol in σ.

Circuits and FO Queries

Proof.

- If k is not of the form $||\mathfrak{A}||$ for some structure $\mathfrak{A}$, then C_k always returns false.
- Assume that k is the size of the encodings of structures $\mathfrak{A}$ with n-elements universe.
- We replace in Φ each quantifier $\exists x \varphi(x, \vec{y})$ or $\forall x \varphi(x, \vec{y})$ with $\bigvee_{c=0}^{n-1} \varphi(c, \vec{y})$ and $\bigwedge_{c=0}^{n-1} \varphi(c, \vec{y})$ respectively and we have $\Phi^{'}$.
- Note that $\Phi^{'}$ is a Boolean combination of formulas of type form
  - $P(i_1, \ldots, i_k)$, where P is a numerical predicate and
  - $R(i_1, \ldots, i_k)$, where R is a m-ary symbol in σ.
- The depth of the resulting circuit is bounded by the number of the connectives in $\Phi^{'}$ and hence depends only on Φ, and not on k and the size of the circuit is polynomial in k.

□

Circuits and FO Queries

Corollary

The data complexity of FO(ALL) is nonuniform AC^0.

Circuits and FO Queries

Corollary

The data complexity of FO(ALL) is nonuniform AC^0.

Given an FO formula φ, its width is the maximum number of free variables is a subformula of φ.

Proposition

Let Φ be an FO sentence in vocabulary σ, and let $\mathfrak{A} \in$STRUCT$[\sigma]$. If the width of Φ is k, then checking whether $\mathfrak{A} \models \Phi$ can be done in time

$$O(\|\Phi\| \times \|\mathfrak{A}\|^k).$$

Circuits and FO Queries

Proof.

- Let $\varphi_1,..,\varphi_m$ enumerate all the subformulae of Φ; we know that they contain at most k free variables.

Circuits and FO Queries

Proof.

- Let $\varphi_1,..,\varphi_m$ enumerate all the subformulae of Φ; we know that they contain at most k free variables.
- We know inductively construct $\varphi_i(\mathfrak{A})$. If φ_i has k_i free variables, then $\varphi_i(\mathfrak{A}) \subseteq A^{k_i}$. If φ_i is:
  - an atomic formula
  - $\neg\varphi_j(\mathfrak{A})$
  - $\varphi_j \bigwedge \varphi_l$
  - $\varphi_i(\vec{x}) = \exists z \varphi_j(z, \vec{x})$

Circuits and FO Queries

Proof.

- Let φ_1,..,φ_m enumerate all the subformulae of Φ; we know that they contain at most k free variables.
- We know inductively construct $\varphi_i(\mathfrak{A})$. If φ_i has k_i free variables, then $\varphi_i(\mathfrak{A}) \subseteq A^{k_i}$. If φ_i is:
  - an atomic formula
  - $\neg\varphi_j(\mathfrak{A})$
  - $\varphi_j \wedge \varphi_l$
  - $\varphi_i(\vec{x}) = \exists z \varphi_j(z, \vec{x})$
- It is easy to see that the above algorithm can be implemented in time $O(||\Phi|| \times ||\mathfrak{A}||^k)$, since none of the formulae φ_i has more than k free variables.

$\square$

Combined Complexity of FO

Theorem

The combined complexity of FO is PSPACE-complete.

Combined Complexity of FO

Theorem

The combined complexity of FO is PSPACE-complete.

Proof.

The membership in PSPACE follows from the evaluation method used in the proof of a previous proposition. To show hardness, recall the problem QBF:

Input : A formula $\Phi = Q_1 x_1 \ldots Q_n x_n a(x_1, \ldots, x_n)$, where: each Q_i is either $\exists$ or $\forall$, and a is a proposition formula in $x_1, \ldots, x_n$.

Question : If all x_i's range $\{\text{true}, \text{false}\}$, is Φ true?

Combined Complexity FO

Given a formula $\Phi = Q_1 x_1 \ldots Q_n x_n a(x_1, \ldots, x_n)$, we construct a structure $\mathfrak{A}$ whose vocabulary includes one unary relation U as follows: $A = \{0,1\}$, and $U^{\mathfrak{A}} = \{1\}$. Then modify a by changing each occurrence of x_i to $U(x_i)$, and each occurrence of $\neg x_i$ to $\neg U(x_i)$.

Examples

If $a(x_1,x_2,x_3) = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$, then a^U is $(U(x_1) \wedge U(x_2)) \vee (\neg U(x_1) \wedge U(x_3))$.

Combined Complexity FO

Given a formula $\Phi = Q_1 x_1 \ldots Q_n x_n a(x_1,\ldots,x_n)$, we construct a structure $\mathfrak{A}$ whose vocabulary includes one unary relation U as follows: $A=\{0,1\}$, and $U^{\mathfrak{A}}=\{1\}$. Then modify a by changing each occurrence of x_i to $U(x_i)$, and each occurrence of $\neg x_i$ to $\neg U(x_i)$.

Examples

If $a(x_1,x_2,x_3)=(x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$, then a^U is $(U(x_1) \wedge U(x_2)) \vee (\neg U(x_1) \wedge U(x_3))$.

Then Φ is true $\Leftrightarrow \mathfrak{A} \models Q_1 x_1 \ldots Q_n x_n a^U(x_1,\ldots,x_n)$. □

Combined Complexity FO

Given a formula $\Phi = Q_1x_1 \ldots Q_nx_na(x_1,\ldots,x_n)$, we construct a structure $\mathfrak{A}$ whose vocabulary includes one unary relation U as follows: $A=\{0,1\}$, and $U^{\mathfrak{A}}=\{1\}$. Then modify a by changing each occurrence of x_i to $U(x_i)$, and each occurrence of $\neg x_i$ to $\neg U(x_i)$.

Examples

If $a(x_1,x_2,x_3)=(x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$, then a^U is $(U(x_1) \wedge U(x_2)) \vee (\neg U(x_1) \wedge U(x_3))$.

Then Φ is true $\Leftrightarrow \mathfrak{A} \models Q_1x_1 \ldots Q_nx_na^U(x_1,\ldots,x_n)$. □

Corollary

The expression complexity of FO is PSPACE-complete.

Parametric Complexity and Locality

We already know that the checking whether $\mathfrak{A} \models \Phi$ can be done in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|^k)$, where k is the width of Φ.

Parametric Complexity and Locality

We already know that the checking whether $\mathfrak{A} \models \Phi$ can be done in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|^k)$, where k is the width of Φ.

Parametric Complexity is a complexity where the standard input of a problem is split into the input part and the parameter part, and one looks for fixed-parameter tractable problems that admit algorithms with running time $O(g(\pi) \cdot n^k)$ for a fixed k.

Parametric Complexity and Locality

Definition

We say that the model-checking problem for $\mathcal{L}$ on $\mathcal{C}$($\mathcal{C}$ is a class of structures) is FPT, if there is a constant p and a function g:$\mathbb{N} \longrightarrow \mathbb{N}$ such that for every $\mathfrak{A} \in \mathcal{C}$ and every $\mathcal{L}$-sentence Φ, checking whether $\mathfrak{A} \models \Phi$ can be done in time

$$g(||\Phi||) \cdot ||\mathfrak{A}||^p.$$

Remark

For p=1 the model-checking problem can be done in time

$$g(||\Phi||) \cdot ||\mathfrak{A}||,$$

and is called fixed parameter linear.

Parametric Complexity and Locality

Definition(Threshold equivalence)

Given two structures $\mathfrak{A}$,$\mathfrak{B}$ in a relational vocabulary, we write $\mathfrak{A} \rightleftarrows^{thr}_{d,m} \mathfrak{B}$ if for every isomorphism type τ of a d-neighborhood of a point either

- both $\mathfrak{A}$ and $\mathfrak{B}$ have the same number of points that d-realize τ, or
- both $\mathfrak{A}$ and $\mathfrak{B}$ have at least m points that d-realize τ.

Theorem 1

For each k,l>0, there exist d,m > 0 such that for $\mathfrak{A},\mathfrak{B} \in$STRUCT$_l[\sigma]$,

$$\mathfrak{A} \rightleftarrows^{thr}_{d,m} \mathfrak{B} \quad \textit{implies} \quad \mathfrak{A} \equiv_k \mathfrak{B}$$

Parametric Complexity and Locality

Theorem 2

Fix l>0. Then the model-checking problemfor FO on $\mathrm{STRUCT}_l[\sigma]$ is fixed-parameter linear.

Parametric Complexity and Locality

Theorem 2

Fix l>0. Then the model-checking problemfor FO on $\mathrm{STRUCT}_l[\sigma]$ is fixed-parameter linear.

Proof.

- Given l and Φ, we can find numbers d and m such that for every $\mathfrak{A},\mathfrak{B} \in \mathrm{STRUCT}_l[\sigma]$, its is the case that $\mathfrak{A} \rightleftarrows^{\mathfrak{A}}_{d,m} \mathfrak{B}$ implies that $\mathfrak{A}$ and $\mathfrak{B}$ agree on Φ.
- We assume that $\tau_1,\ldots,\tau_M$ enumerate isomorphism types of all the structures of the form $N^{\mathfrak{A}}_d(a)$ for $\mathfrak{A} \in \mathrm{STRUCT}_l[\sigma]$.

Parametric Complexity and Locality

Theorem 2

Fix l>0. Then the model-checking problemfor FO on $\mathrm{STRUCT}_l[\sigma]$ is fixed-parameter linear.

Proof.

- Given l and Φ, we can find numbers d and m such that for every $\mathfrak{A},\mathfrak{B} \in \mathrm{STRUCT}_l[\sigma]$, its is the case that $\mathfrak{A} \rightleftarrows^{\mathfrak{A}}_{d,m} \mathfrak{B}$ implies that $\mathfrak{A}$ and $\mathfrak{B}$ agree on Φ.
- We assume that $\tau_1,\ldots,\tau_M$ enumerate isomorphism types of all the structures of the form $N_d^{\mathfrak{A}}(a)$ for $\mathfrak{A} \in \mathrm{STRUCT}_l[\sigma]$.
- Let $n_i(\mathfrak{A})=|\{ a \mid N_d^{\mathfrak{A}}(a) \text{ of type } \tau_i \}|$. With each structures $\mathfrak{A}$, we now associate an M-tuple $\vec{t}(\mathfrak{A})=(t_1,..,t_M)$ such that

$$t_i = \begin{cases} n_i(\mathfrak{A}) & , if \;\; n_i(\mathfrak{A}) \leq m, \\ * & , otherwise \end{cases}$$

Parametric Complexity and Locality

- Let T be the set of all M-tuples whose elements come from $\{1,2...,m\} \cup \{*\}$, so each $\vec{t}(\mathfrak{A})$ is a member of T.
- From Theorem 1, $\vec{t}(\mathfrak{A}) = \vec{t}(\mathfrak{B})$ implies that $\mathfrak{A}$ and $\mathfrak{B}$ agree on Φ.

Parametric Complexity and Locality

- Let T be the set of all M-tuples whose elements come from $\{1,2\ldots,m\} \cup \{*\}$, so each $\vec{t}(\mathfrak{A})$ is a member of T.
- From Theorem 1, $\vec{t}(\mathfrak{A})=\vec{t}(\mathfrak{B})$ implies that $\mathfrak{A}$ and $\mathfrak{B}$ agree on Φ.
- Let T_0 be the set of $\vec{t} \in T$ such that for some structure $\mathfrak{A} \in STRUCT_l[\sigma]$, we have $\mathfrak{A} \models \Phi$ and $\vec{t}(\mathfrak{A})=\vec{t}$.
- We compute, for a given structure $\mathfrak{A}$, the tuple $\vec{t}(\mathfrak{A})$ and then check if $\vec{t} \in T_0$.

Parametric Complexity and Locality

- Let T be the set of all M-tuples whose elements come from $\{1,2\ldots,m\} \cup \{*\}$, so each $\vec{t}(\mathfrak{A})$ is a member of T.
- From Theorem 1, $\vec{t}(\mathfrak{A})=\vec{t}(\mathfrak{B})$ implies that $\mathfrak{A}$ and $\mathfrak{B}$ agree on Φ.
- Let T_0 be the set of $\vec{t} \in T$ such that for some structure $\mathfrak{A} \in \mathrm{STRUCT}_l[\sigma]$, we have $\mathfrak{A} \models \Phi$ and $\vec{t}(\mathfrak{A})=\vec{t}$.
- We compute, for a given structure $\mathfrak{A}$, the tuple $\vec{t}(\mathfrak{A})$ and then check if $\vec{t} \in T_0$.
- The computation of T_0 depends entirely on Φ and l, but not on $\mathfrak{A}$; hence the resulting algorithm has linear running time.

$\square$

Parametric Complexity and Locality

Theorem

If $\mathcal{C}$ is a minor-closed class of graphs which does not include all the graphs, then model-checking for FO on $\mathcal{C}$ is fixed parameter tractable.

Corollary

Model-checking for FO on the class of planar graphs is fixed parameter tractable.

Conjunctive Queries

Definition

A first order formula $\varphi(\vec{x})$ over a relational vocabulary σ is called a conjunctive query if it is built from atomic formulae using only conjunction $\wedge$ and existential quantification $\exists$.

Every conjunctive query can be expressed as:

$$\varphi(\vec{x}) = \exists \vec{y} \wedge_{i=1}^{k} \mathsf{a}_i(\vec{x}, \vec{y}).$$

Conjunctive Queries

Definition

A first order formula $\varphi(\vec{x})$ over a relational vocabulary σ is called a conjunctive query if it is built from atomic formulae using only conjunction $\wedge$ and existential quantification $\exists$.

Every conjunctive query can be expressed as:

$$\varphi(\vec{x}) = \exists \vec{y} \wedge_{i=1}^{k} a_i(\vec{x}, \vec{y}).$$

Example

If there is a path of length k+1 between x and x' in a graph E, one can write as a conjunctive query as follows:

$$\exists y_1, ..., y_k R(x, y_1) \wedge R(y_1, y_2) \wedge ... \wedge R(y_k, x')$$

Conjunctive Queries

Suppose we have a formula $\varphi(x_1,\ldots,x_m)$ over vocabulary σ, then for each $\mathfrak{A} \in$STRUCT$[\sigma]$ this formula defines an m-ary relation $\varphi(\mathfrak{A})=\{ \vec{a} \mid \mathfrak{A} \models \varphi(\vec{a})\}$.

Conjunctive Queries

Suppose we have a formula $\varphi(x_1,\ldots,x_m)$ over vocabulary σ, then for each $\mathfrak{A} \in$STRUCT[σ] this formula defines an m-ary relation $\varphi(\mathfrak{A})=\{\ \vec{a} \mid \mathfrak{A} \models \varphi(\vec{a})\}$.

The join of R and S is defined as

$$R \bowtie S = \{t : X \cup Y \rightarrow A \mid t|_X \in R, t|_Y \in S\}$$

where R is viewed as a set of mappings t: $X \rightarrow A$ and S is viewed as a set of mappings t: $Y \rightarrow A$.

Suppose that R is $\varphi(\mathfrak{A})$ and S is $\psi(\mathfrak{A})$ then $R \bowtie S=[\varphi \wedge \psi](\mathfrak{A})$.

Conjunctive Queries

Suppose we have a formula $\varphi(x_1,\ldots,x_m)$ over vocabulary σ, then for each $\mathfrak{A} \in$STRUCT[σ] this formula defines an m-ary relation $\varphi(\mathfrak{A})=\{\ \vec{a} \mid \mathfrak{A} \models \varphi(\vec{a})\}$.

The join of R and S is defined as

$$R \bowtie S = \{t : X \cup Y \to A \mid t|_X \in R, t|_Y \in S\}$$

where R is viewed as a set of mappings t: $X \to A$ and S is viewed as a set of mappings t: $Y \to A$.

Suppose that R is $\varphi(\mathfrak{A})$ and S is $\psi(\mathfrak{A})$ then $R \bowtie S = [\varphi \wedge \psi](\mathfrak{A})$.

The projection of R on $Y(\subseteq X)$ is defined as:

$$\pi_Y = \{\ t\colon\ Y \to A \mid \exists t' \in R : t'|_Y = t\ \}.$$

If R is $\varphi(\mathfrak{A})$, where φ has free variables $(\vec{x}, \vec{y})$, then $\pi_{\vec{y}}(R) = [\exists \vec{x} \varphi(\vec{x}, \vec{y})](\mathfrak{A})$.

Conjunctive Queries

Suppose we have a formula $\varphi(x_1,...,x_m)$ over vocabulary σ, then for each $\mathfrak{A} \in$STRUCT$[\sigma]$ this formula defines an m-ary relation $\varphi(\mathfrak{A})=\{ \vec{a} \mid \mathfrak{A} \models \varphi(\vec{a})\}$.

The join of R and S is defined as

$$R \bowtie S = \{t : X \cup Y \to A \mid t|_X \in R, t|_Y \in S\}$$

where R is viewed as a set of mappings t: $X \to A$ and S is viewed as a set of mappings t: $Y \to A$.

Suppose that R is $\varphi(\mathfrak{A})$ and S is $\psi(\mathfrak{A})$ then $R \bowtie S=[\varphi \wedge \psi](\mathfrak{A})$.

The projection of R on $Y(\subseteq X)$ is defined as:

$$\pi_Y = \{ t\colon Y \to A \mid \exists t' \in R : t'|_Y = t \}.$$

If R is $\varphi(\mathfrak{A})$, where φ has free variables $(\vec{x},\vec{y})$, then $\pi_{\vec{y}}(R)=[\exists \vec{x} \varphi(\vec{x},\vec{y})](\mathfrak{A})$.

Suppose we have a conjunctive query

$$\varphi(\vec{y}) \equiv \exists \vec{x} \, (a_1(\vec{u_1}) \wedge .... \wedge a_n(\vec{u_n})).$$

Then for any structure $\mathfrak{A}$

$$\varphi(\mathfrak{A}) \equiv \pi_{\vec{y}} \, (a_1(\mathfrak{A}) \bowtie .... \bowtie a_n(\mathfrak{A})).$$

Conjunctive Queries

Theorem

The combined and expression complexity of conjunctive queries are NP-complete(even for Boolean conjunctive queries).

Conjunctive Queries

Theorem

The combined and expression complexity of conjunctive queries are NP-complete(even for Boolean conjunctive queries).

Proof.

The combined complexity is NP: for the query $\varphi(\vec{x}) = \exists\vec{y}\bigwedge_{i=1}^{k} \mathrm{a}_i(\vec{x},\vec{y})$ and a tuple $\vec{a}$, to check if $\varphi(\vec{a})$ holds, one has to guess a tuple $\vec{b}$ and then check in polynomial time if $\bigwedge_{i=1}^{k} \mathrm{a}_i(\vec{a},\vec{b})$ holds.

Conjunctive Queries

Theorem

The combined and expression complexity of conjunctive queries are NP-complete(even for Boolean conjunctive queries).

Proof.

The combined complexity is NP: for the query $\varphi(\vec{x}) = \exists\vec{y}\wedge_{i=1}^{k} a_i(\vec{x},\vec{y})$ and a tuple $\vec{a}$, to check if $\varphi(\vec{a})$ holds, one has to guess a tuple $\vec{b}$ and then check in polynomial time if $\wedge_{i=1}^{k} a_i(\vec{a},\vec{b})$ holds.

For completeness,we use reduction from 3-colorability

- Define a structure $\mathfrak{A}=\langle\{0,1,2\},N\rangle$, where N is the binary inequality relation $N=\{(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)\}$

Conjunctive Queries

Theorem

The combined and expression complexity of conjunctive queries are NP-complete(even for Boolean conjunctive queries).

Proof.
The combined complexity is NP: for the query $\varphi(\vec{x}) = \exists \vec{y} \bigwedge_{i=1}^{k} \mathrm{a}_i(\vec{x}, \vec{y})$ and a tuple $\vec{a}$, to check if $\varphi(\vec{a})$ holds, one has to guess a tuple $\vec{b}$ and then check in polynomial time if $\bigwedge_{i=1}^{k} \mathrm{a}_i(\vec{a}, \vec{b})$ holds.
For completeness,we use reduction from 3-colorability

- Define a structure $\mathfrak{A}=\langle\{0,1,2\},\mathrm{N}\rangle$, where N is the binary inequality relation $\mathrm{N}=\{(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)\}$
- Let $\mathrm{G}=(\mathrm{U},\mathrm{E})$ be a graph, where $\mathrm{U}=\{\mathrm{a}_1,\ldots,\mathrm{a}_n\}$ and $\mathrm{E}\subseteq\mathrm{U}\times\mathrm{U}$

Conjunctive Queries

Theorem

The combined and expression complexity of conjunctive queries are NP-complete(even for Boolean conjunctive queries).

Proof.

The combined complexity is NP: for the query $\varphi(\vec{x}) = \exists\vec{y}\bigwedge_{i=1}^{k} \mathrm{a}_i(\vec{x},\vec{y})$ and a tuple $\vec{a}$, to check if $\varphi(\vec{a})$ holds, one has to guess a tuple $\vec{b}$ and then check in polynomial time if $\bigwedge_{i=1}^{k} \mathrm{a}_i(\vec{a},\vec{b})$ holds.

For completeness,we use reduction from 3-colorability

- Define a structure $\mathfrak{A}=\langle\{0,1,2\},\mathrm{N}\rangle$, where N is the binary inequality relation $\mathrm{N}=\{(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)\}$
- Let $\mathrm{G}=(\mathrm{U},\mathrm{E})$ be a graph, where $\mathrm{U}=\{\mathrm{a}_1,\ldots,\mathrm{a}_n\}$ and $\mathrm{E}\subseteq\mathrm{U}\times\mathrm{U}$
- Define the Boolean conjunctive query Φ

$$\exists x_1 \ldots \exists x_n \bigwedge_{(a_i,a_j)\in E} N(x_i, x_j)$$

Conjunctive Queries

Theorem

The combined and expression complexity of conjunctive queries are NP-complete(even for Boolean conjunctive queries).

Proof.

The combined complexity is NP: for the query $\varphi(\vec{x}) = \exists\vec{y} \bigwedge_{i=1}^{k} a_i(\vec{x}, \vec{y})$ and a tuple $\vec{a}$, to check if $\varphi(\vec{a})$ holds, one has to guess a tuple $\vec{b}$ and then check in polynomial time if $\bigwedge_{i=1}^{k} a_i(\vec{a}, \vec{b})$ holds.

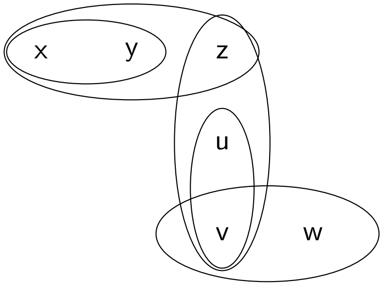
For completeness,we use reduction from 3-colorability

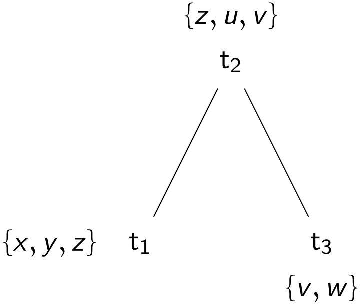
- Define a structure $\mathfrak{A}=\langle\{0,1,2\},N\rangle$, where N is the binary inequality relation $N=\{(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)\}$
- Let $G=(U,E)$ be a graph, where $U=\{a_1,\ldots,a_n\}$ and $E\subseteq U\times U$
- Define the Boolean conjunctive query Φ
$$\exists x_1 \ldots \exists x_n \bigwedge_{(a_i,a_j)\in E} N(x_i, x_j)$$
- $\mathfrak{A} \models \Phi$ iff G is 3-colorable. □

Conjunctive Queries

A tree decomposition of a hyper-graph $\mathcal{H}$ is a tree $\mathcal{T}$ together with a set $B_t \subseteq U$ for each node t of $\mathcal{T}$ such that the two following condition hold:

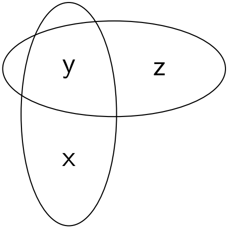
1. For every $a \in U$, the set $\{ t \mid a \in B_t \}$ is a subtree of $\mathcal{T}$.
2. Every hyper-edge of $\mathcal{H}$ is contained in one of the B_t's.





Conjunctive Queries

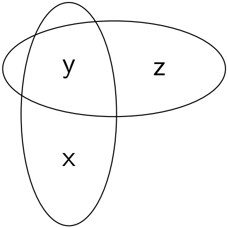
$\Phi \equiv \exists x \exists y \exists z R(x,y) \wedge R(y,z)$



Φ is acyclic

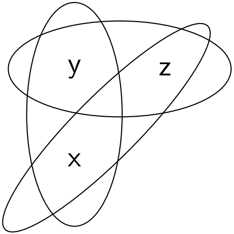
Conjunctive Queries

$\Phi \equiv \exists x \exists y \exists z R(x,y) \wedge R(y,z)$



Φ is acyclic

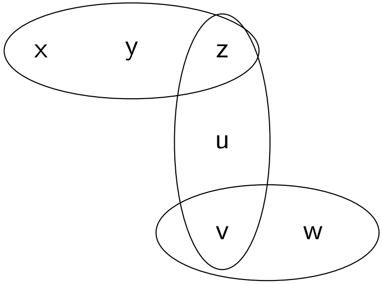
$\Phi \equiv \exists x \exists y \exists z R(x,y) \wedge R(y,z) \wedge$
$\wedge R(x,z)$



Φ is not acyclic

Conjunctive Queries

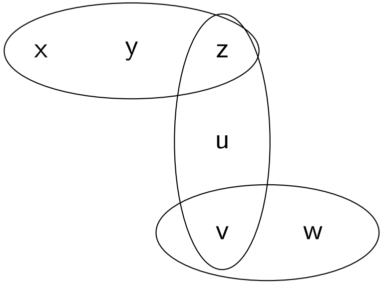
$\Phi \equiv \exists x \exists y \exists z \exists u \exists v$
R(x,y,z)$\wedge$R(z,u,v)$\wedge$
$\wedge$S(u,z)$\wedge$S(x,y)$\wedge$S(v,w)



Φ is acyclic

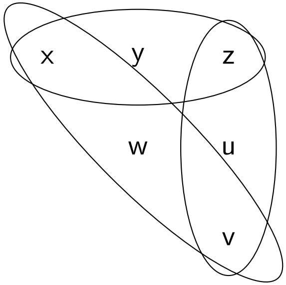
Conjunctive Queries

$\Phi \equiv \exists x \exists y \exists z \exists u \exists v$
R(x,y,z)∧R(z,u,v)∧
∧S(u,z)∧S(x,y)∧S(v,w)



Φ is acyclic

$\Phi \equiv \exists x \exists y \exists z \exists u \exists v$
R(x,y,z)∧R(z,u,v)∧R(x,v,w)



Φ is not acyclic

Conjunctive Queries

Theorem

Let Φ be a Boolean acyclic conjunctive query over σ-structure, and let $\mathfrak{A} \in$STRUCT[σ]. Then checking whether $\mathfrak{A} \models \Phi$ can be done in time $O(||\Phi|| \cdot ||\mathfrak{A}||)$.

Conjunctive Queries

Theorem

Let Φ be a Boolean acyclic conjunctive query over σ-structure, and let $\mathfrak{A} \in$STRUCT$[\sigma]$. Then checking whether $\mathfrak{A} \models \Phi$ can be done in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|)$.

Proof.

- Let $\Phi \equiv \exists x_1 \ldots x_m \bigwedge_{i=1}^{n} a_i(\vec{u_i})$

Conjunctive Queries

Theorem

Let Φ be a Boolean acyclic conjunctive query over σ-structure, and let $\mathfrak{A} \in$STRUCT$[\sigma]$. Then checking whether $\mathfrak{A} \models \Phi$ can be done in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|)$.

Proof.

- Let $\Phi \equiv \exists x_1 ... x_m \bigwedge_{i=1}^{n} a_i(\vec{u_i})$
- A decomposition $(\mathcal{T}, (B_t)_{t \in \mathcal{T}})$ for $\mathcal{H}(\Phi)$ can compute in time $O(\|\Phi\|)$.
- We define

$$R_t \quad = \quad \bowtie_{i \in [1,n], v_i = t} a_i(\mathfrak{A}) \quad (1)$$

- Our goal is to compute the join of all R_t's,since

$$\mathfrak{A} \models \Phi \Leftrightarrow \bowtie_{t \in \mathcal{T}} R_t \neq \emptyset \ (2)$$

Conjunctive Queries

- We define

$$R_t \quad = \quad \bowtie_{i\in[1,n],v_i=t} a_i(\mathfrak{A}) \;\; (1)$$

Conjunctive Queries

- We define
$$R_t \quad = \quad \bowtie_{i\in[1,n],v_i=t} a_i(\mathfrak{A}) \ \ (1)$$

- Our goal is to compute the join of all R_t's, since
$$\mathfrak{A} \models \Phi \Leftrightarrow \bowtie_{t\in\mathcal{T}} \mathrm{R}_t \neq \emptyset \ (2)$$

Conjunctive Queries

- We define

$$R_t \;=\; \bowtie_{i\in[1,n],v_i=t} a_i(\mathfrak{A}) \;\; (1)$$

- Our goal is to compute the join of all R_t's, since

$$\mathfrak{A} \models \Phi \Leftrightarrow \bowtie_{t\in\mathcal{T}} \mathrm{R}_t \neq \emptyset \; (2)$$

- We can see that $\mathrm{R}_t \subseteq \mathrm{a}_{i_t}(\mathfrak{A})$ and we conclude that the entire family R_t,t$\in \mathcal{T}$, can be computed in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|)$.

Conjunctive Queries

- We define
$$R_t \;=\; \bowtie_{i\in[1,n], v_i=t} a_i(\mathfrak{A}) \;\; (1)$$
- Our goal is to compute the join of all R_t's, since
$$\mathfrak{A} \models \Phi \Leftrightarrow \bowtie_{t\in\mathcal{T}} R_t \neq \emptyset \; (2)$$
- We can see that $R_t \subseteq a_{i_t}(\mathfrak{A})$ and we conclude that the entire family R_t, $t \in \mathcal{T}$, can be computed in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|)$.
- We define $P_t \;=\; \bowtie_{u\succeq t} R_u$:
  - $P_t = R_t$, if t is a leaf of T
  - $P_t = R_t \bowtie (\bowtie_{1\leq i\leq l} P_{t_i})$, where $t_1,..,t_l$ are children of t.

Conjunctive Queries

- We define

$$R_t \quad = \quad \bowtie_{i\in[1,n],v_i=t} a_i(\mathfrak{A}) \ \ (1)$$

- Our goal is to compute the join of all R_t's, since

$$\mathfrak{A} \models \Phi \Leftrightarrow \bowtie_{t\in\mathcal{T}} R_t \neq \emptyset \ (2)$$

- We can see that $R_t \subseteq a_{i_t}(\mathfrak{A})$ and we conclude that the entire family R_t,t$\in \mathcal{T}$, can be computed in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|)$.
- We define $P_t \quad = \quad \bowtie_{u\succeq t} R_u$:
  - $P_t = R_t$, if t is a leaf of T
  - $P_t = R_t \bowtie (\bowtie_{1\leq i\leq l} P_{t_i})$, where $t_1,..,t_l$ are children of t.
- We compute $P_r = \bowtie_t R_t$ in time $O(\|\mathcal{T}\| \cdot max_t\|R_t\|) = O(\|\Phi\| \cdot \|\mathfrak{A}\|)$, which together with (2) implies that $\mathfrak{A} \models \Phi$ can be done in time $O(\|\Phi\| \cdot \|\mathfrak{A}\|)$. □

Bibliography

- Leonid Libkin, "Elements of Finite Model theory" [2012]

Figure: Leonid Libkin