Descriptive Complexity: Preliminaries from Logic

Stathis Zachos, Petros Potikas, Ioannis Kokkinis and Aggeliki Chalki

ALMA
INTER-INSTITUTIONAL GRADUATE PROGRAM "ALGORITHMS, LOGIC AND DISCRETE MATHEMATICS"

Overview

Overview

Interaction

Instructors:

- Stathis Zachos: zachos@cs.ntua.gr
- Petros Potikas: ppotik@cs.ntua.gr
- Ioannis Kokkinis: ykokkinis@gmail.com
- Aggeliki Chalki: achalki@corelab.ntua.gr

Lectures:

Tuesday 11:00-15:00
Classroom 1.1.31, old ECE building, NTUA

Assignments and Evaluation

- Presentation
- Final Exam

References

- Immerman, Neil. Descriptive complexity. (https://people.cs.umass.edu/~immerman/book/corrections.html)
- Libkin, Leonid. Elements of finite model theory.

Overview

Edge Existence (The Imperative Way)

Does $G = \langle V, E \rangle$ have an edge?

Edge Existence (The Imperative Way)

Does $G = \langle V, E \rangle$ have an edge?

The imperative way:

```
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        if (E[i,j] == 1) then
            printf("G has an edge!\n");
```

Edge Existence (The Declarative Way)

Does $G = \langle V, E \rangle$ have and edge?

Edge Existence (The Declarative Way)

Does $G = \langle V, E \rangle$ have and edge?

G is actually a structure of a first-order (FO) language with only one binary relation symbol, E.

Edge Existence (The Declarative Way)

Does $G = \langle V, E \rangle$ have and edge?

G is actually a structure of a first-order (FO) language with only one binary relation symbol, E.

The declarative way:

$$G \models \exists x \exists y E(x, y).$$

Vertex Cover

Does $G = \langle V, E \rangle$ have a vertex cover of size k?

$$G \models (\exists W \subseteq V)\Big[|W| \leq k \wedge (\forall x, y \in V)\big[E(x,y) \rightarrow (x \in W \vee y \in W)\big]\Big]$$

Vertex Cover

Does $G = \langle V, E \rangle$ have a vertex cover of size k?

$$G \models (\exists W \subseteq V)\Big[|W| \leq k \wedge (\forall x, y \in V)\big[E(x,y) \rightarrow (x \in W \vee y \in W)\big]\Big]$$

Be careful: We quantified over sets.

Descriptive Complexity

Descriptive Complexity

The computational complexity of a problem can be understood as the richness of the language needed to specify the problem.

"Edge Existence" is easier than "Has a Vertex Cover of size k" since the formula $\exists x \exists y E(x, y)$ is FO whereas the formula

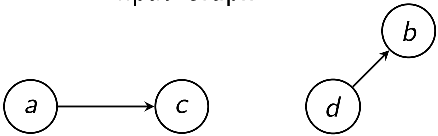
$$G \models (\exists W \subseteq V)\Big[|W| \leq k \wedge (\forall x, y \in V)\big[E(x, y) \rightarrow (x \in W \vee y \in W)\big]\Big]$$

is SO.

Application to Databases

All database management languages, like SQL are extensions of *FO*-logic.

Input Graph



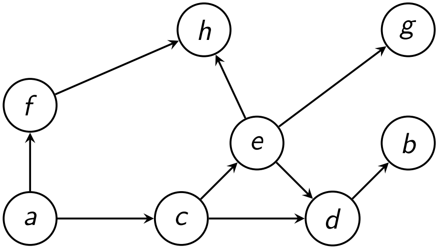
Database DB

E	
a	c
d	b

Edge existence query:

```
EXISTS (
  SELECT *
  FROM E
)
```

Input Representation



$$G = \langle V, E, a, b \rangle$$

$$G \models \phi?$$

$$V = \{a, b, \ldots\}, E = \{(a, c), (d, b), \ldots\}$$

Database DB

E	
a	c
d	b
a	f
⋮	

Query Q

does DB satisfy Q?

Overview

Computational Complexity Measures

Engineers: space and time are natural resources.

Mathematicians: space and time depend on the model . . .

Theorem (Fagin, 1974)

NP = set of problems describable in existential second-order logic ($\exists SO$).

Complexity as Expressibility

The most important complexity classes and questions have elegant descriptive analogues.

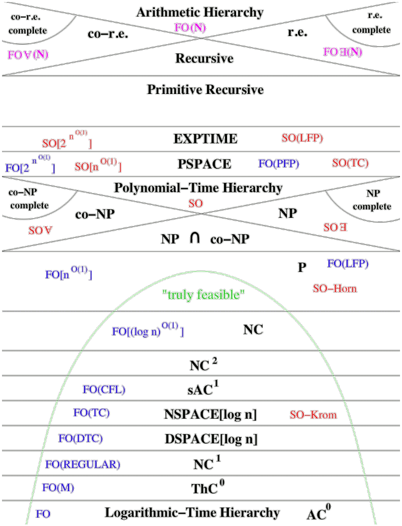
P is the set of problems describable in *FO* plus inductive definitions.

$\vdots$

P = NP iff every problem describable in *SO* is already expressible in *FO* plus inductive definitions.

$\vdots$

The Complexity World (From a Logician's View)



The Case for Finite Models

All objects (programs, inputs, databases, . . .) handled by computers are finite.

So, the structures we are going to study will always be finite.

Most of the techniques/results we knew do not apply any more . . .

No Proof Theory for Finite Models!

In the case of *FO*-logic, when infinite and finite models are allowed

Validity: R.E.-complete and axiomatizable (Gödel, 1930)

Satisfiability: non-R.E

but when only finite models are allowed

Validity: non-R.E, thus non-axiomatizable!

Satisfiability: R.E.-complete (Trakhtenbrot, 1950)

Overview

Vocabularies

A vocabulary τ contains:

Logical Symbols

variables: $x, y, z, \ldots$

equality: $\approx$

logical connectives: $\wedge, \neg, \forall$

Non-Logical Symbols

constants: $c, r, \ldots$

relational symbols: $P, R, Q, \ldots$

We write $\tau = \langle P^{a_1}, R^{a_2}, Q^{a_3}, \ldots, c, r, \ldots \rangle$.

Terms and Formulas

Vocabulary $\tau = \langle P^{a_1}, R^{a_2}, Q^{a_3}, \ldots, c, r, \ldots \rangle$

Definition (Terms over τ)

Every variable and constant is a term.

Definition (Formulas over τ)

- If $t_1, \ldots, t_n$ are terms then $t_1 \approx t_2$ and $P(t_1, \ldots, t_n)$ are formulas.
- if ϕ, ψ are formulas and x is a variable then $(\neg\phi)$, $(\phi \wedge \psi)$ and $\forall x \psi$ are formulas.

Finite Relational Structures

Let $\tau = \langle P^{a_1}, R^{a_2}, Q^{a_3}, \ldots, c, r, \ldots \rangle$.

A structure over τ looks like:

$$\mathcal{A} = \{|\mathcal{A}|, P^{\mathcal{A}}, R^{\mathcal{A}}, Q^{\mathcal{A}}, \ldots, c^{\mathcal{A}}, r^{\mathcal{A}}, \ldots\}.$$

$\mathrm{STRUCT}(\tau) = \{\mathcal{B} \mid \mathcal{B} \text{ is a finite structure over } \tau\}$.

Free Variables and Interpretations

x is bounded in ϕ if x occurs under the scope of a quantifier.

x is free in ϕ if x is not bounded in ϕ.

For a structure $\mathcal{A}$, an interpretation is a partial function $i : VARS \rightharpoonup |\mathcal{A}|$, e.g.:

$$x \overset{i}{\mapsto} m$$

$$y \overset{i}{\mapsto} n$$

$$c \overset{i}{\mapsto} c^{\mathcal{A}}$$

where c is a constant and $m, n, c^{\mathcal{A}} \in |\mathcal{A}|$.

Satisfiability

Definition (Truth by Tarski)

Let $\mathcal{A} \in \text{STRUCT}(\tau)$, let ϕ be a formula and $i : VARS \rightharpoonup |\mathcal{A}|$ an interpretation. Then $\mathcal{A}, i \models \phi$ if

- $\phi = t_1 \approx t_2$ and $i(t_1) = i(t_2)$.
- $\phi = P(t_1, \ldots, t_n)$ and $(i(t_1), \ldots, i(t_n)) \in P^{\mathcal{A}}$
- $\phi = \neg\psi$ and $\mathcal{A}, i \not\models \psi$
- $\phi = \psi \wedge \chi$ and $A, i \models \psi$ and $A, i \models \chi$
- $\phi = \forall x \psi$ and for all $a \in |\mathcal{A}|$, $A, i[x/a] \models \psi$

Strings as Relational Structures

A string with 5 characters can be seen as a relational structure:

Position	4	3	2	1	0
String	0	1	0	0	1

$$\text{Vocabulary } \langle S^1, \leq^2 \rangle$$

$$\mathcal{A} = \langle |\mathcal{A}|, S^{\mathcal{A}}, \leq^{\mathcal{A}} \rangle$$

$$|\mathcal{A}| = \{0, 1, 2, 3, 4\}$$

$$S^{\mathcal{A}} = \{0, 3\}$$

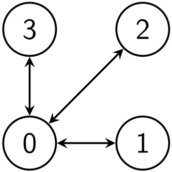
$$\leq^{\mathcal{A}} = \{(0,1), (0,2), \ldots\}$$

Example:

$$\phi = \exists u, v \Big[\neg S(u) \land \neg S(v) \land \neg \exists w (v < w < u) \Big].$$

It holds that $\mathcal{A} \models \phi$

Graphs as Relational Structures



$$\text{Vocabulary } \tau = \langle E^2 \rangle$$

$$\mathcal{G} = \langle V, E \rangle,\ V = \{0, 1, 2, 3\}, E = \{(0,1), (1,0), \ldots\}$$

$$\psi = (\forall x, y)\Big[\neg E(x,x) \wedge \big(E(x,y) \leftrightarrow E(y,x)\big)\Big]$$

$$\mathcal{G} \models \psi$$

Adding n-bit Numbers

Let $\tau_{ab} = \langle \leq^2, A^1, B^1 \rangle$.

Let $\mathcal{A} = \langle |\mathcal{A}|, A, B \rangle$ where $|\mathcal{A}| = \{0, 1, \ldots, n-1\}$.

Relations A and B practically represent two n-bit numbers.

Position	4	3	2	1	0
1st Number	0	1	1	1	0
2nd Number	0	1	0	0	1

$A = \{1, 2, 3\}$

$B = \{3, 0\}$

Addition is FO-expressible in τ_{ab} (see the carry look-ahead algorithm on whiteboard).

Ordering and Arithmetic

Position	$n-1$	$n-2$	$\ldots$	1	0
Input	s_{n-1}	s_{n-2}	$\ldots$	s_1	s_0

We need $\log n$ bits to code the input.

Our input structures will always contain $\leq$ (i.e. a total order on the input domain), $0, 1, \max$ and:

SUC(i, j) $j = i + 1$

BIT(i, j): the j-th bith of element i is 1

PLUS(i, j, k): $i + j = k$

TIMES(i, j, k): $i \times j = k$

BIT$(i, 0)$ holds iff i is odd

Definability (1/2)

Many of the previous predicates and constants are *FO*-definable from each other:

- $x = 0 \iff \forall y (x \leq y)$
- $x = 1 \iff x > 0 \land \forall y \neg (0 < y < x)$
- $\mathrm{SUC}(i, j) \iff j > i \land \forall z \neg (i < z < j)$
- We've seen that PLUS is definable from BIT

Definability (2/2)

$$
\begin{aligned}
u = \lfloor x/y \rfloor &\iff (u \cdot y) \leq x \land (\exists v < y)\big[x \approx u \cdot y + v\big] \\
u = x \bmod y &\iff \exists v\big[v \approx \lfloor x/y \rfloor \land (u + y \cdot v \approx x)\big] \\
x \mid y &\iff x \bmod y \approx 0 \\
\mathrm{PRIME}(x) &\iff x > 1 \land (\forall y, z)\Big[(x \approx y \cdot z) \to (y \approx 1 \lor z \approx 1)\Big] \\
\mathrm{pow}_2(x) &\iff (\forall y)\Big[(y \mid x \land \mathrm{PRIME}(y)) \to (y = 2)\Big] \\
\mathrm{BIT}'(i, j) &\iff \lfloor x/y \rfloor \bmod 2 = 1
\end{aligned}
$$

The Bit-Sum Lemma

Lemma

Let $BSUM(x, y)$ be true iff y is equal to the number of ones in the binary representation of x. BSUM is FO-expressible using only BIT *and ordering.*

Proof.

See lemma 7.2 from *David A. Mix Barrington, Neil Immerman, and Howard Straubing. "On uniformity within NC^1" Journal of Computer and System Sciences 41.3 (1990): 274-306.* □

FO (BIT) $=FO$ (PLUS, TIMES)

Using the previous results we can prove that FO (BIT) $=FO$ (PLUS, TIMES).

For more informationm see:

- Immerman, Section 1.2.1
- Libkin, Section 6.4

Isomorphism

Let $\mathcal{A}, \mathcal{B} \in \mathrm{STRUCT}(\tau)$. Then $\mathcal{A}$ is isomorphic to $\mathcal{B}$ iff there is a bijection $f : |\mathcal{A}| \to |\mathcal{B}|$ such that:

- $P^{\mathcal{A}}(a_1, \ldots, a_n) \iff P^{\mathcal{B}}(f(a_1), \ldots, f(a_n))$
- $f(c^{\mathcal{A}}) = c^{\mathcal{B}}$.

Overview

First Order Queries

Input Schema: σ

Output Schema: $\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle$

A k-ary *FO*-query is a mapping $I : c\mathrm{STRUCT}(\sigma) \to \mathrm{STRUCT}(\tau)$, that consists of $1 + r + s$ *FO*-formulas.

We write $I = \langle \phi_0, \phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_s \rangle$

If $\mathcal{A} \in \mathrm{STRUCT}(\sigma)$ then

universe of $I(\mathcal{A})$: consists of all the k-tuples of $|\mathcal{A}|$ that satisfy ϕ_0

relations of $I(\mathcal{A})$: contain elements of $|I(\mathcal{A})|$ that satisfy $\phi_1, \ldots, \phi_r$

constants of $I(\mathcal{A})$: elements of $|I(\mathcal{A})|$ that satisfy $\psi_1, \ldots, \psi_s$

Example: A Unary Query (1/2)

Input Vocabulary: $\sigma = \langle F^1, P^2, S^2 \rangle$

For example $\mathcal{B} \in \mathrm{STRUCT}(\sigma)$, where $\mathcal{B} = \langle U, F, P, S \rangle$

$$U = \{\text{Abraham, Isaac, Rebekah, Sarah}, \ldots\}$$
$$F = \{\text{Rebekah, Sarah}, \ldots\}$$
$$P = \{(\text{Abraham, Isaac}), (\text{Sarah, Isaac}), \ldots\}$$
$$S = \{(\text{Abraham, Sarah}), (\text{Isaac, Rebekah}), \ldots\}$$

Example: A Unary Query (2/2)

Input vocabulary: $\sigma = \langle F^1, P^2, S^2 \rangle$

Output vocabulary: $\tau = \langle SI^2, AU^2 \rangle$

$$I = \langle true, \phi_{SI}(x,y), \phi_{AU}(x,y) \rangle$$

$$\phi_{SI}(x,y) = x \not\approx y \wedge (\exists f, m)\,[f \neq m \wedge P(f,x) \wedge P(m,x) \wedge P(f,y) \wedge P(m,y)]$$

$$\phi_{AU}(x,y) = F(x) \wedge (\exists p, s)\,[P(p,y) \wedge \phi_{SI}(p,s) \wedge (s = x \vee S(x,s))]$$

Example: The Addition Query

Let $\sigma = \langle A^1, B^1 \rangle$ and $\tau = S^1$.

Remember the addition formula ϕ_{add} that we defined earlier.

We have the unary query $I = \langle true, \phi_{add} \rangle$ that maps elements of STRUCT(σ) to elements of STRUCT(τ).

Example: A Binary Query From Graphs to Graphs

Input Vocabulary: $\sigma = \langle E^2 \rangle$.

Output Vocabulary: $\tau = \langle R^2 \rangle$.

$I = \langle \mathit{true},\ \phi((x,y),(x',y')) \rangle$ where

$\phi((x,y),(x',y')) = (x = x' \land E(y,y')) \lor (\mathrm{SUC}(x,y) \land x' = y = y')$

Let G be undirected. We can show that:

$$G \text{ is connected iff } (\max, \max) \text{ is reachable from } (0,0) \text{ in } I(G)$$

Boolean Queries

Complexity classes are usually defined for decision problems.

The descriptive analogue of a decision problem is the boolean query, i.e a mapping from STRUCT(τ) to $\{0, 1\}$. (Observe that the boolean query does not map structures to structures).

Any *FO*-sentence ϕ defines a boolean query as follows:

$$I_\phi(\mathcal{A}) = 1 \iff \mathcal{A} \models \phi.$$

E.g. if $\phi = \exists x, yE(x, y)$ then I_ϕ is the edge-existence query.

Higher Order Boolean Queries

$$\text{Let } \phi = (\exists A, B, C)(\forall x, y)\Big[E(x, y) \rightarrow \neg\Big((A(x) \wedge A(y)) \vee (B(x) \wedge B(y)) \vee (C(x) \wedge C(y))\Big)\Big]$$

Then $I_\phi(G) = 1$ iff G is 3-colorable.

Observe that ϕ is $\exists SO$ (Remember Fagin's Theorem).