Descriptive Complexity

Fagin’s Theorem

Ρωμανός Ασλάνης-Πέτρου

«ΑΛΓΟΡΙΘΜΟΙ, ΛΟΓΙΚΗ ΚΑΙ ΔΙΑΚΡΙΤΑ ΜΑΘΗΜΑΤΙΚΑ»

15 Δεκεμβρίου 2022

Introduction

Computational Complexity

The ”hardness” of a problem is related to the *computational* resources a *mechanical* procedure requires to solve the problem.

Descriptive Complexity

The ”hardness” of a problem is related to the *logical* resources that are required, inorder to formally express the problem.

Introduction

- An algorithm can be seen as a precise description of a mapping from inputs to outputs
- The most usual way to describe such a mapping, is via Turing machines.
- However we may choose to describe such a mapping with another, precise way.
- For example, using formal logic.

Preliminaries

Vocabulary

A vocabulary $\tau = \langle R_1^{a_1}, \ldots, R_k^{a_k}, c_1, \ldots, c_s, f_1^{r_1}, \ldots, f_t^{r_t} \rangle$ is a tuple of relation, constant and function symbols. R_i is a relation of arity a_i and f_j is a function of arity r_j.

Preliminaries

Vocabulary

A vocabulary $\tau = \langle R_1^{a_1}, \ldots, R_k^{a_k}, c_1, \ldots, c_s, f_1^{r_1}, \ldots, f_t^{r_t} \rangle$ is a tuple of relation, constant and function symbols. R_i is a relation of arity a_i and f_j is a function of arity r_j.

- We do not care about function symbols.
- (Our vocabularies will therefore be called relational.)

Preliminaries

Let $\tau = \langle R_1^{a_1}, \ldots, R_k^{a_k}, c_1, \ldots, c_s \rangle$ be a vocabulary.

τ-structure

A structure of vocabulary τ is a tuple $\mathcal{A} = \langle |A|, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_s^{\mathcal{A}} \rangle$, whose universe is the nonempty set $|A|$, for each relation R_i of arity a_i in τ, $\mathcal{A}$ has a relation $R_i^{\mathcal{A}}$ of arity a_i defined on $|A|$ and for each constant $c_j \in \tau$, $\mathcal{A}$ has a specified element of its universe.

Preliminaries

Let $\tau = \langle R_1^{a_1}, \dots, R_k^{a_k}, c_1, \dots, c_s \rangle$ be a vocabulary.

τ-structure

A structure of vocabulary τ is a tuple $\mathcal{A} = \langle |A|, R_1^{\mathcal{A}}, \dots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \dots, c_s^{\mathcal{A}} \rangle$, whose universe is the nonempty set $|A|$, for each relation R_i of arity a_i in τ, $\mathcal{A}$ has a relation $R_i^{\mathcal{A}}$ of arity a_i defined on $|A|$ and for each constant $c_j \in \tau$, $\mathcal{A}$ has a specified element of its universe.

- In descriptive complexity we study finite structures.
- To be more accurate, descriptive complexity is the connection of finite model theory and complexity.

Preliminaries

e.x: $\tau_s = \langle \leq^2, S^1 \rangle$ is the vocabulary of binary strings, where S is a unary relation that tells us which bit of the binary string is equal to 1.

Let $\mathcal{U}$ be a τ_s-structure such that:

$$\mathcal{U} = \langle |\mathcal{U}| = \{1, 2, 3, 4\}, \leq = \{\text{"}the\ usual\ order\text{"}\}, S = \{2, 3\} \rangle$$

Then $\mathcal{U}$ "represents" the binary string 0110.
($i \in S$ iff the ith bit of the string is equal to 1)

Second-order logic

Second-order logic = first-order logic + quantification $(\exists, \forall)$ over relations on the universe.

[1] R is a relation (or predicate) variable, while x is an individual variable (the usual variables we have in first-order logic $\ldots$)

Second-order logic

Second-order logic = first-order logic + quantification $(\exists, \forall)$ over relations on the universe.

e.x: Let $\mathcal{U}$ be a structure with universe $|\mathcal{U}| = \{0, 1, 2, 3\}$ and ϕ the second-order sentence $\exists R \forall x R(x, x)$.[1]

Then $\mathcal{U} \models \exists R \forall x R(x, x)$ iff there exist a relation $r \subseteq |\mathcal{U}| \times |\mathcal{U}|$ such that for all $x \in |\mathcal{U}|$, $(x, x) \in r$.

$$r = \{(0,0), (1,1), (2,2), (3,3)\}$$

[1] R is a relation (or predicate) variable, while x is an individual variable (the usual variables we have in first-order logic ...)

($\exists$)Second-order logic

Existential second-order logic ($SO(\exists)$) is a fragment of second-order logic. In particular $\phi \in SO(\exists)$ iff ϕ is of the following form:

$$\phi \equiv \exists R_1, \ldots, \exists R_n \psi$$

Where $R_i, i \in [n]$ relation symbols and ψ is a first-order formula.

$(\exists)$Second-order logic

Existential second-order logic ($SO(\exists)$) is a fragment of second-order logic. In particular $\phi \in SO(\exists)$ iff ϕ is of the following form:

$$\phi \equiv \exists R_1, \ldots, \exists R_n \psi$$

Where $R_i, i \in [n]$ relation symbols and ψ is a first-order formula.

e.x:

$$\exists R \exists G \exists B \forall x \forall y \Big(\big(R(x) \oplus G(x) \oplus B(x)\big) \wedge \big(E(x, y) \rightarrow \big(\neg(R(x) \wedge R(y)) \wedge \neg(G(x) \wedge G(y)) \wedge \neg(B(x) \wedge B(y))\big)\big)\Big)$$

Queries

Queries - boolean queries

Let C_{τ_1}, C_{τ_2} be classes of *finite* structures of vocabulary τ_1, τ_2 accordingly, that are closed under isomorphisms.

- A query is a mapping $Q : C_{\tau_1} \to C_{\tau_2}$.
- A boolean query is a mapping $Q_b : C_{\tau_1} \to \{0, 1\}$.

Where both Q and Q_b preserve isomorphisms.

Queries

Queries - boolean queries

Let C_{τ_1}, C_{τ_2} be classes of *finite* structures of vocabulary τ_1, τ_2 accordingly, that are closed under isomorphisms.

- A query is a mapping $Q : C_{\tau_1} \rightarrow C_{\tau_2}$.
- A boolean query is a mapping $Q_b : C_{\tau_1} \rightarrow \{0, 1\}$.

Where both Q and Q_b preserve isomorphisms.

e.x: Let $\sigma = \langle E \rangle$ be the vocabulary of graphs and C_σ the class of all structures of vocabulary σ ($\mathcal{A} \in C_\sigma \Rightarrow \mathcal{A} = \langle |\mathcal{A}|, E^{\mathcal{A}} \rangle$).

Then for all $G \in C_\sigma$ the disconnectivity query is:

$$DC(G) = \begin{cases} 1 & \textit{if } G \textit{ is a disconnected graph} \\ 0 & \textit{otherwise} \end{cases}$$

Queries - Definability

Let $\mathcal{L}$ be a logic and C a class of τ-structures.

$\mathcal{L}$-definability

A boolean query Q_b on C is $\mathcal{L}$-definable if there is a $\mathcal{L}$-sentence ϕ such that for all $\mathcal{U} \in C$:

$$Q_b(\mathcal{U}) = 1 \Leftrightarrow \mathcal{U} \models \phi$$

Queries - Definability

Let $\mathcal{L}$ be a logic and C a class of τ-structures.

$\mathcal{L}$-definability

A boolean query Q_b on C is $\mathcal{L}$-definable if there is a $\mathcal{L}$-sentence ϕ such that for all $\mathcal{U} \in C$:

$$Q_b(\mathcal{U}) = 1 \Leftrightarrow \mathcal{U} \models \phi$$

e.x: The boolean query *disconnectivity* is definable by the second-order sentence:

$$\exists S \Big(\exists x S(x) \land \exists y \neg S(y) \land \forall z \forall w \big(S(z) \land \neg S(w) \rightarrow \neg E(z,w) \big) \Big)$$

Which intuitively says "there are two disjoint, nonempty, sets of nodes with no edge between them".

Capturing complexity classes

$\mathcal{L}$ captures $\mathcal{C}$

We will say that a logic $\mathcal{L}$ captures a complexity class $\mathcal{C}$ on a domain of structures $\mathcal{D}$ if:

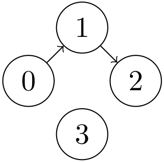
- For every fixed sentence $\phi \in \mathcal{L}$, the complexity of evaluating ϕ ($\mathcal{U} \models \phi$) on structures of $\mathcal{D}$ is a problem in class $\mathcal{C}$.
- Every boolean query from structures of $\mathcal{D}$ that can be decided in $\mathcal{C}$ is $\mathcal{L}$-definable

Binary strings

- Everything a TM does may be thought of as a query from binary strings to binary strings.
- In descriptive complexity we saw the vocabulary of binary strings.
- However we have a lot more structures that use other vocabularies.
- To relate them with Turing machines, we have to somehow encode them into binary strings.

Encoding structures

Define an encoding query $bin_\tau : C_\tau \to C_{\tau_s}$, where C_τ is the class of all (*ordered!*) structures of vocabulary τ and C_{τ_s} the class of all structures of the vocabulary of binary strings $\tau_s = \langle \leq^2, S^1 \rangle$.
Let $G = (V, E, s, t)$ where $V = \{0, 1, 2, 3\}$, $E^G = \{(0, 1), (1, 2)\}$ and $s^G = 0, t^G = 3$.



$E^G \subseteq V \times V =$
$\{(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), \ldots\}$
E is a 2-ary relation, V has 4 elements $\to$ we need 4^2 bits.

Binary encoding:

$$bin(G) = \overbrace{0100001000000000}^{E}\overbrace{00}^{s}\overbrace{10}^{t}, \text{ with } |bin(G)| = O(n^2).$$

Encoding structures

The particular choice of an encoding is not important. However there are some conditions that must be satisfied by the encoding.

An encoding query must be:

- Order-independent (identifies isomorphic structures).
- Polynomially bounded. (space efficient)
- First-order definable. (easy to encode and (…) to decode)

Encoding structures

The particular choice of an encoding is not important. However there are some conditions that must be satisfied by the encoding.

An encoding query must be:

- Order-independent (identifies isomorphic structures).
- Polynomially bounded. (space efficient)
- First-order definable. (easy to encode and (…) to decode)

Note that inorder to encode a structure into a binary string, an ordering must exist!

Fagin's Theorem

Fagin's Theorem

Existential second-order logic $SO(\exists)$ captures NP, on the domain of all(!) finite structures.

1. For a fixed $\phi \in SO(\exists)$, for all structures $\mathcal{U}$, checking if $\mathcal{U} \models \phi$ is in NP.
2. Every boolean query that can be decided in NP can be defined in $SO(\exists)$.

Fagin's Theorem

In other words …

1. For every $\phi \in SO(\exists)$ of some vocabulary τ and forall $\mathcal{A} \in C_\tau$ there exists a polynomial-time NTM M such that:

$$\mathcal{A} \models \phi \Leftrightarrow M(bin(\mathcal{A}) \downarrow_{yes})$$

Fagin's Theorem

In other words …

1. For every $\phi \in SO(\exists)$ of some vocabulary τ and forall $\mathcal{A} \in C_\tau$ there exists a polynomial-time NTM M such that:

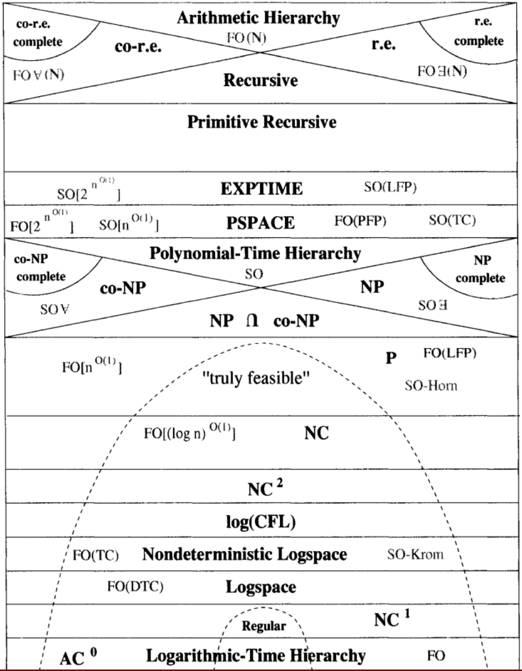
$$\mathcal{A} \models \phi \Leftrightarrow M(bin(\mathcal{A}) \downarrow_{yes})$$

2. For every polynomial-time NTM M that decides a language $L \in NP$, there exists a $SO(\exists)$ sentence ϕ such that:

$$w \in L \text{ iff } w = bin(\mathcal{A}) \text{ where } \mathcal{A} \text{ is a model of } \phi$$

The End

- By Fagin's theorem $SO(\exists) = NP$
- Corollary: $SO(\forall) = coNP$
- We know that $NP \neq coNP \to P \neq NP$.
- Corollary: $SO(\exists) \neq SO(\forall) \to P \neq NP$.


Arithmetic Hierarchy
FO(N)
co-r.e. complete
co-r.e.
r.e.
r.e. complete
FO∀(N)
FO∃(N)
Recursive
Primitive Recursive
SO[$2^{n^{O(1)}}$]
EXPTIME
SO(LFP)
FO[$2^{n^{O(1)}}$]
SO[$n^{O(1)}$]
PSPACE
FO(PFP)
SO(TC)
Polynomial-Time Hierarchy
SO
co-NP complete
co-NP
NP
NP complete
SO∀
SO∃
NP ∩ co-NP
FO[$n^{O(1)}$]
"truly feasible"
P
FO(LFP)
SO-Horn
FO[$(\log n)^{O(1)}$]
NC
NC^2
log(CFL)
FO(TC)
Nondeterministic Logspace
SO-Krom
FO(DTC)
Logspace
Regular
NC^1
AC^0
Logarithmic-Time Hierarchy
FO

Bibliography

[1] Ronald Fagin.
Generalized first-order spectra, and polynomial. time recognizable sets.
1974.

[2] Erich Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, Moshe Y. Vardi, Y. Venema, and Scott Weinstein.
Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series).
Springer-Verlag, Berlin, Heidelberg, 2005.

[3] Neil Immerman.
Descriptive Complexity.
Springer Verlag, 1998.