Descriptive Complexity: Preliminaries from Complexity

Stathis Zachos, Petros Potikas, Ioannis Kokkinis and Aggeliki Chalki

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

Overview

Overview

- The language accepted by a Turing machine M is denoted by $L(M) = \{w \in \{0,1\}^* | M(w) \downarrow\}$.

- Everything that a Turing machine does can be considered as a query from binary strings to binary strings.
- For each vocabulary τ, we define an encoding query,

$$bin_\tau : STRUC[\tau] \to STRUC[\tau_s]$$

where $\tau_s = \langle S^1 \rangle$ is the vocabulary of boolean strings.

The binary encoding of structures

- Let $\tau = \langle R_1^{a_1}, ..., R_r^{a_r}, c_1, ..., c_s \rangle$ be a vocabulary and $\mathcal{A} = \langle \{0, ..., n-1\}, R_1^{\mathcal{A}}, ..., R_r^{\mathcal{A}}, c_1^{\mathcal{A}}, ..., c_s^{\mathcal{A}} \rangle$ be an ordered τ-structure.
- Each $R_i^{\mathcal{A}}$ can be encoded as a binary string of length a_i:
  - We order the elements of $|\mathcal{A}|^{a_i}$ lexicographically.
  - The j-th bit of the binary string that encodes $R_i^{\mathcal{A}}$ is 1 iff $R(x_1^j, ..., x_{a_i}^j)$, where $x_1^j, ..., x_{a_i}^j$ is the j-th element in the defined ordering.
- Each constant $c_i^{\mathcal{A}}$ can be encoded as a binary string of length $\lceil logn \rceil$.

The binary encoding of structures

- The binary encoding of the structure $\mathcal{A}$ is the concatenation

$$bin_\tau(\mathcal{A}) = bin^{\mathcal{A}}(R_1)...bin^{\mathcal{A}}(R_r)bin^{\mathcal{A}}(c_1)...bin^{\mathcal{A}}(c_s)$$

  which has length $\hat{n}_\tau = n^{a_1} + ... + n^{a_r} + s\lceil logn \rceil$.

The coding $bin_\tau(\mathcal{A})$ presupposes an ordering on the universe!

Definition

Let $I : STRUC[\sigma] \to STRUC[\tau]$ be a query and T be a Turing machine.

We say that **T computes the query I** iff for any $\mathcal{A} \in STRUC[\sigma]$,

$$T(bin(\mathcal{A})) = bin(I(\mathcal{A})).$$

Complexity Classes

- We use the notation

$$DTIME[t(n)], NTIME[t(n)], DSPACE[s(n)], NSPACE[s(n)]$$

to denote the set of boolean queries that are computable by a TM in

deterministic time $\mathcal{O}(t(n))$, nondeterministic time $\mathcal{O}(t(n))$, deterministic space $\mathcal{O}(s(n))$, nondeterministic space $\mathcal{O}(s(n))$ respectively.

Complexity Classes

- L $= DSPACE[logn]$
- NL $= NSPACE[logn]$
- P $= \bigcup_{k=1}^{\infty} DTIME[n^k]$
- NP $= \bigcup_{k=1}^{\infty} NTIME[n^k]$
- PSPACE $= \bigcup_{k=1}^{\infty} DSPACE[n^k]$
- EXPTIME $= \bigcup_{k=1}^{\infty} DTIME[2^{n^k}]$

The Queries computable in C

Definition

Let $I : STRUC[\sigma] \rightarrow STRUC[\tau]$ be a query.
We say that **I is computable in C** iff the boolean query I_b is in C, where

$$I_b = \{(\mathcal{A}, i, a) | \text{ the } i^{th} \text{ bit of } bin(I(\mathcal{A})) \text{ is "}a\text{"}\}.$$

We define $Q(\mathrm{C}) = \mathrm{C} \cup \{I | I_b \in \mathrm{C}\}$ to be the set of all queries

computable in C.

Useful Inclusions and Relationships

Theorem (The Time Hierarchy Theorem)

If $f(n) \geq n$ is a proper complexity function, then the class $TIME[f(n)]$ is strictly contained within $TIME[f(n)log^2 f(n)]$.

Theorem (The Space Hierarchy Theorem)

If $f(n) \geq n$ is a proper complexity function, then $SPACE[f(n)]$ is strictly contained within $SPACE[f(n)log\, f(n)]$.

By Space and Time Hierarchy Theorems:

1. L $\subsetneq$ PSPACE and NL $\subsetneq$ NPSPACE
2. P $\subsetneq$ EXPTIME and NP $\subsetneq$ NEXPTIME

Useful Inclusions and Relationships

Theorem

Suppose that $f(n)$ is a proper complexity function. Then:

1. - $DSPACE[f(n)] \subseteq NSPACE[f(n)]$,
   - $DTIME[f(n)] \subseteq NTIME[f(n)]$.
2. $NTIME[f(n)] \subseteq DSPACE[f(n)]$.
3. $NSPACE[f(n)] \subseteq DTIME[2^{\mathcal{O}(f(n))}]$.

- L $\subseteq$ NL and P $\subseteq$ NP
- NP $\subseteq$ PSPACE
- NL $\subseteq$ P and NPSPACE $\subseteq$ EXPTIME

Useful Inclusions and Relationships

Theorem (Savitch's Theorem)

For any proper complexity function $f(n) \geq logn$, $NSPACE[f(n)] \subseteq DSPACE[f^2(n)]$.

In particular,

1. $\mathrm{L} \subseteq \mathrm{NL} \subseteq DSPACE[log^2 n]$
2. $\mathrm{PSPACE} = \mathrm{NPSPACE}$

Useful Inclusions and Relationships

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXPTIME}$$

Overview

Turing reductions

Definition

Given two boolean queries A, B and a complexity class C.
We say that **A is C-Turing reducible to B**, symb. $A \leq_{\mathsf{C}}^{T} B$, iff there exists an oracle Turing machine M such that M^B:

1. runs in complexity class C and
2. $L(M^B) = A$

An example is the polynomial-time Turing reduction, $\leq_{\mathsf{P}}^{T}$.

An example of a Turing reduction

Let $\tau_{gk} = \langle E^2, k \rangle$.

1. The boolean query $\textsc{Clique}$, decides if a given τ_{gk}-structure G has a clique of size k.
2. The query $\textsc{Max-Clique}$ computes the size of the largest clique in a given τ_{gk}-structure G.
3. $\textsc{Max-Clique}_b =$ $\{(G, i, a) |$ bit i of $bin(\textsc{Max-Clique}(G))$ is "a"$\}$.

$\textsc{Max-Clique}_b \leq_P^T \textsc{Clique}$: Given (G, i, a) perform binary search using an oracle for $\textsc{Clique}$ to determine the size s of the maximum clique for G. After *logn* queries to the oracle, s has been computed. Accept iff bit i of s is "a".

Many-One reductions

Recall that a boolean query $I : STRUC[\sigma] \to \{0,1\}$ can be considered as a subset of $STRUC[\sigma]$.

Definition

Let C be a complexity class, and $A \subseteq STRUC[\sigma]$, $B \subseteq STRUC[\tau]$ be boolean queries.
We say that **the query $\mathbf{I : STRUC[\sigma] \to STRUC[\tau]}$ is a C-many-one reduction from A to B** iff I is in $Q(\mathrm{C})$ with the property that for any $\mathcal{A} \in STRUC[\sigma]$,

$$\mathcal{A} \in A \Leftrightarrow I(\mathcal{A}) \in B$$

We say that **A is C-many-one reducible to B**, symb. $A \leq_{\mathrm{C}} B$.

Many-One reductions

Examples:

1. If I is a first order query, then it is a first-order reduction, symb. $\leq_{fo}$.
2. If $I \in Q(\mathrm{L})$, then it is a logspace reduction, symb. $\leq_{log}$.
3. If $I \in Q(\mathrm{P})$, then it is a polynomial-time reduction, symb. $\leq_p$.

A many-one reduction is a special case of a Turing reduction: To decide whether $\mathcal{A} \in A$, compute $I(\mathcal{A})$ and ask the oracle whether $I(\mathcal{A})$ is in B.

An example of a first-order reduction

1. PARITY is the boolean query on binary strings that is true iff a given string has an odd number of ones.
2. MULT is the multiplicative query that maps a pair of n-length binary strings to their $2n$-length product.
3. MULT_b is a boolean query on structures of $\tau_{abcd} = \langle A^1, B^1, c, d \rangle$ that is true iff bit c of the product $A \cdot B$ is "d".

An example of a first-order reduction

$$\textsc{Parity} \leq_{fo} \textsc{Mult}_b$$

The first-order reduction $I_{PM} : STRUC[\tau_s] \to STRUC[\tau_{abcd}]$ is given by the following formulas:

- $\phi_A(x, y) \equiv y = max \wedge S(x)$
- $\phi_B(x, y) \equiv y = max$
- $I_{PM} \equiv \lambda_{xy}\langle \mathbf{true}, \phi_A, \phi_B, \langle 0, max\rangle, \langle 0, 1\rangle\rangle$

$$\mathcal{A} \in \textsc{Parity} \Leftrightarrow I_{PM}(\mathcal{A}) \in \textsc{Mult}_b$$

Completeness for C

Definition

Let A be a boolean query, C be a complexity class and $\leq$ be a reducibility relation.
We say that **A is complete for C with respect to** $\leq$ iff

1. $A \in \mathrm{C}$ and
2. for all $B \in \mathrm{C}$, $B \leq A$.

We prove later that if a problem is complete with respect to first-order reductions, then it is complete with respect to logspace and polynomial-time reductions.

Complete problems

Complete for L:

- CYCLE : Given an undirected graph, does it contain a cycle?
- REACH_d: Given a directed graph, is there a deterministic path from vertex s to vertex t?

Complete for NL:

- REACH: Given a directed graph, is there a path from vertex s to vertex t?
- 2-SAT: Given a boolean formula in conjunctive normal form with only two literals per clause, is it satisfiable?

Complete problems

Complete for P:

- CIRCUITVALUEPROBLEM: Given an acyclic boolean circuit, with inputs specified, does its output gate have value one?
- NETWORKFLOW: Given a directed graph, with capacities on its edges, and a value V, is it possible to achieve a steady-state flow of value V through the graph?

Complete problems

Complete for NP:

- SAT: Given a boolean formula, is it satisfiable?
- 3-SAT: Given a boolean formula in conjunctive normal form with only three literals per clause, is it satisfiable?
- CLIQUE: Given an undirected graph and a value k, does the graph have a complete subgraph with k vertices?

Complete for PSPACE:

- QSAT: Given a quantified boolean formula, is it satisfiable?
- GO: Given a position in the game go, is there a forced win for the player whose move it is?

- The reductions $\leq_{fo}$, $\leq_{log}$, $\leq_{p}$ are transitive.
- Let $\leq_r$ be a transitive many-one reducibility relation and A be complete for C with respect to $\leq_r$.
  Let T be any boolean query.
  We can show that T is complete for C with respect to $\leq_r$ by showing that

$$T \in \mathrm{C} \text{ and } A \leq_r T.$$

Overview

Alternating Turing Machines

A **configuration** c of a Turing machine consists of the machine's state, work-tape contents and head positions.

Definition

An **alternating Turing machine** (ATM) is a Turing machine the states of which are divided into two groups: the existential states and the universal states.

The ATM in a given configuration c accepts iff

1. c is in a final accepting state, or
2. c is in an existential state and there exists a successor configuration c' that accepts, or
3. c is in a universal state, and all the successor configurations accept.

Definition

We define $ATIME[t(n)]$ ($ASPACE[s(n)]$) to be the set of boolean queries accepted by alternating Turing machines using $\mathcal{O}(t(n))$ time ($\mathcal{O}(s(n))$ space respectively).

Definition

We define $\text{AP} = \bigcup_{k=1}^{\infty} ATIME[n^k]$ and $\text{AL} = ASPACE[logn]$.

Definition

A **boolean circuit** is a directed acyclic graph (DAG)

$$C = (V, E, G_\wedge, G_\vee, G_\neg, I, r) \in STRUC[\tau_c]$$

where $\tau_c = \langle E^2, G^1_\wedge, G^1_\vee, G^1_\neg, I^1, r \rangle$.
Internal node w is:

- an and-gate iff $G_\wedge(w)$ holds
- an or-gate iff $G_\vee(w)$ holds
- a not-gate iff $G_\neg(w)$ holds
- called a leaf iff it has no incoming edges and leaf w is one iff $I(w)$ holds

Define **Circuit Value Problem** (CVP) to consist of those circuits the root gate of which evaluate to one.

- The Monotone Circuit Value Problem (MCVP) is the subset of CVP in which no negation gates occur.

MCVP is in AL

Proof. For a node a of a monotone circuit define $\mathrm{EVAL}(a)$ as follows:

- if $I(a)$ then accept
- if not $I(a)$ and a has no outgoing edges then reject
- if $G_\wedge(a)$ then in a universal state choose a child b of a and call $\mathrm{EVAL}(b)$
- if $G_\vee(a)$ then in an existential state choose a child b of a and call $\mathrm{EVAL}(b)$

The ATM calls $\mathrm{EVAL}(r)$ where r is the root gate.
The machine requires logspace.

Definition

The **quantified satisfiability problem** (QSAT) is the set of true formulas of the following form:

$$\psi = (Q_1 x_1)(Q_2 x_2)...(Q_r x_r)\phi$$

where ϕ is a boolean formula and each Q_i is either $\forall$ or $\exists$ and $x_1, ...x_r$ are the boolean variables occuring in ϕ.

QSAT is in $ATIME[n]$

Proof. For the formula

$$\Phi = (\exists x_1)(\forall x_2)...(Q_r x_r)\phi(\overrightarrow{x})$$

the alternating machine

- writes down a boolean value for x_1 in an existential state
- then it writes down a boolean value for x_2 in a universal state and so on

Eventually the machine evaluates the quantifier-free boolean formula ϕ given the nondeterministic choices for $x_1, x_2, ...x_r$.

- AP = PSPACE
- AL = P

These are special cases of the following theorem:

Theorem

For $s(n) \geq logn$, and for $t(n) \geq n$,

1. $$\bigcup_{k=1}^{\infty} ATIME[(t(n))^k] = \bigcup_{k=1}^{\infty} DSPACE[(t(n))^k]$$
2. $$ASPACE[s(n)] = \bigcup_{k=1}^{\infty} DTIME[k^{s(n)}]$$