Topics in Logic and Complexity
Handout 3

Anuj Dawar

MPhil Advanced Computer Science, Lent 2010

P-complete Problems

Game

Input: A directed graph $G = (V, E)$ with a partition $V = V_1 \cup V_2$ of the vertices and two distinguished vertices $s, t \in V$.

Decide: whether Player 1 can force a token from s to t in the game where when the token is on $v \in V_1$, Player 1 moves it along an edge leaving v and when it is on $v \in V_2$, Player 2 moves it along an edge leaving v.

Circuit Value Problem

A *Circuit* is a *directed acyclic graph* $G = (V, E)$ where each node has *in-degree* $0, 1$ or 2 and there is exactly one vertex t with no outgoing edges, along with a labelling which assigns:

- to each node of indegree 0 a value of 0 or 1
- to each node of indegree 1 a label $\neg$
- to each node of indegree 2 a label $\wedge$ or $\vee$

The problem CVP is, given a circuit, decide if the target node t evaluates to 1.

NP-complete Problems

SAT

Input: A Boolean formula ϕ

Decide: if there is an assignment of truth values to the variables of ϕ that makes ϕ true.

Hamiltonicity

Input: A graph $G = (V, E)$

Decide: if there is a cycle in G that visits every vertex exactly once.

co-NP-complete Problems

VAL

Input: A Boolean formula ϕ

Decide: if every assignment of truth values to the variables of ϕ makes ϕ true.

Non-3-colourability

Input: A graph $G = (V, E)$

Decide: if there is no function $\chi : V \to \{1, 2, 3\}$ such that the two endpoints of every edge are differently coloured.

PSPACE-complete Problems

Geography is very much like *Game* but now players are not allowed to visit a vertex that has been previously visitied.

HEX is a game played by two players on a graph $G = (V, E)$ with a source and target $s, t \in V$.

The two players take turns selecting vertices from V—neither player can choose a vertex that has been previously selected. Player 1 wins if, at any point, the vertices she has selected include a path from s to t. Player 2 wins if all vertices have been selected and no such path is formed.

The problem is to decide which player has a winning strategy.

Descriptive Complexity

Descriptive Complexity provides an alternative perspective on Computational Complexity.

Computational Complexity

- Measure use of resources (space, time, *etc.*) on a machine model of computation;
- Complexity of a language—i.e. a set of strings.

Descriptive Complexity

- Complexity of a class of structures—e.g. a collection of graphs.
- Measure the complexity of describing the collection in a formal logic, using resources such as variables, quantifiers, higher-order operators, *etc.*

There is a fascinating interplay between the views.

Signature and Structure

In general a *signature* (or *vocabulary*) σ is a finite sequence of *relation*, *function* and *constant* symbols:

$$\sigma = (R_1, \ldots, R_m, f_1, \ldots, f_n, c_1, \ldots, c_p)$$

where, associated with each relation and function symbol is an arity.

Structure

A structure $\mathbb{A}$ over the signature σ is a tuple:

$$\mathbb{A} = (A, R_1^{\mathbb{A}}, \dots, R_m^{\mathbb{A}}, f_1^{\mathbb{A}}, \dots, f_n^{\mathbb{A}}, c_1^{\mathbb{A}}, \dots, c_n^{\mathbb{A}}),$$

where,

- A is a non-empty set, the *universe* of the strucure $\mathbb{A}$,
- each $R_i^{\mathbb{A}}$ is a relation over A of the appropriate arity.
- each $f_i^{\mathbb{A}}$ is a function over A of the appropriate arity.
- each $c_i^{\mathbb{A}}$ is an element of A.

First-order Logic

Formulas of *first-order logic* are formed from the signature σ and an infinite collection X of variables as follows.

terms – c, x, $f(t_1, \dots, t_a)$

Formulas are defined by induction:

- *atomic formulas* – $R(t_1, \dots, t_a)$, $t_1 = t_2$
- *Boolean operations* – $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$
- *first-order quantifiers* – $\exists x \phi$, $\forall x \phi$

Queries

A formula ϕ with free variables among $x_1, \dots, x_n$ defines a map Q from structures to relations:

$$Q(\mathbb{A}) = \{\mathbf{a} \mid \mathbb{A} \models \phi[\mathbf{a}]\}.$$

Any such map Q which associates to every structure $\mathbb{A}$ a (n-ary) relation on A, and is isomorphism invariant, is called a *(n-ary) query*.

Q is *isomorphism invariant* if, whenever $f : A \to B$ is an isomorphism between $\mathbb{A}$ and $\mathbb{B}$, it is also an isomorphism between $(A, Q(\mathbb{A}))$ and $(B, Q(\mathbb{B}))$.

If $n = 0$, we can regard the query as a map from structures to $\{0, 1\}$—a *Boolean query*.

Graphs

For example, take the signature (E), where E is a binary relation symbol.

Finite structures (V, E) of this signature are directed graphs.

Moreover, the class of such finite structures satisfying the sentence

$$\forall x \neg Exx \wedge \forall x \forall y (Exy \to Eyx)$$

can be identified with the class of (*loop-free, undirected*) graphs.

Complexity

For a first-order sentence ϕ, we ask what is the *computational complexity* of the problem:

Input: a structure $\mathbb{A}$
Decide: if $\mathbb{A} \models \phi$

In other words, how complex can the collection of finite models of ϕ be?

In order to talk of the complexity of a class of finite structures, we need to fix some way of representing finite structures as strings.

Representing Structures as Strings

We use an alphabet $\Sigma = \{0, 1, \#, -\}$.

For a structure $\mathbb{A} = (A, R_1, \ldots, R_m, f_1, \ldots, f_l)$, fix a linear order $<$ on $A = \{a_1, \ldots, a_n\}$.

R_i (of arity k) is encoded by a string $[R_i]_<$ of 0s and 1s of length n^k.

f_i is encoded by a string $[f_i]_<$ of 0s, 1s and $-$s of length $n^k \log n$.

$$[\mathbb{A}]_< = \underbrace{1 \cdots 1}_{n} \# [R_1]_< \# \cdots \# [R_m]_< \# [f_1]_< \# \cdots \# [f_l]_<$$

The exact string obtained depends on the choice of order.

Naïve Algorithm

The straightforward algorithm proceeds recursively on the structure of ϕ:

- Atomic formulas by direct lookup.
- Boolean connectives are easy.
- If $\phi \equiv \exists x\, \psi$ then for each $a \in \mathbb{A}$ check whether

  $$(\mathbb{A}, c \mapsto a) \models \psi[c/x],$$

  where c is a new constant symbol.

This runs n time $O(ln^m)$ and $O(m \log n)$ space, where m is the nesting depth of quantifiers in ϕ.

$$\mathrm{Mod}(\phi) = \{\mathbb{A} \mid \mathbb{A} \models \phi\}$$

is in *logarithmic space* and *polynomial time*.

Reading List for this Handout

1. Papadimitriou. Chapters 8
2. Libkin Chapter 2.
3. Grädel et al. Sections 2.1–2.4 (Kolaitis).