Topics in Logic and Complexity

Handout 2

Anuj Dawar

http://www.cl.cam.ac.uk/teaching/2122/L15

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:

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

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, $f(t_1, \ldots, t_a)$

Formulas are defined by induction:

- *atomic formulas* – $R(t_1, \ldots, t_a)$, $t_1 = t_2$
- *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 $x_1, \ldots, x_n$ 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

$$\forall x \neg Exx \land \forall x \forall y (Exy \rightarrow Eyx)$$

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 in 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*.

Complexity of First-Order Logic

The following problem:

FO satisfaction

> *Input: a structure $\mathbb{A}$ and a first-order sentence ϕ*
>
> *Decide: if $\mathbb{A} \models \phi$*

is PSPACE-complete.

It follows from the $O(ln^m)$ and $O(m \log n)$ space algorithm that the problem is in PSPACE.

How do we prove completeness?

QBF

We define *quantified Boolean formulas* inductively as follows, from a set $\mathcal{X}$ of *propositional variables*.

- A propositional constant T or F is a formula
- A propositional variable $X \in \mathcal{X}$ is a formula
- If ϕ and ψ are formulas then so are: $\neg\phi$, $\phi \wedge \psi$ and $\phi \vee \psi$
- If ϕ is a formula and X is a variable then $\exists X\ \phi$ and $\forall X\ \phi$ are formulas.

Say that an occurrence of a variable X is *free* in a formula ϕ if it is not within the scope of a quantifier of the form $\exists X$ or $\forall X$.

QBF

Given a quantified Boolean formula ϕ and an assignment of *truth values* to its free variables, we can ask whether ϕ evaluates to *true* or *false*.
In particular, if ϕ has no free variables, then it is equivalent to either *true* or *false*.

QBF is the following decision problem:

> *Input:* *a quantified Boolean formula* ϕ *with no free variables.*
> *Decide:* *whether* ϕ *evaluates to* *true*.

Complexity of QBF

Note that a Boolean formula ϕ *without quantifiers* and with variables $X_1, \ldots, X_n$ is *satisfiable* if, and only if, the formula

$$\exists X_1 \cdots \exists X_n\, \phi \quad \text{is } \textit{true}.$$

Similarly, ϕ is *valid* if, and only if, the formula

$$\forall X_1 \cdots \forall X_n\, \phi \quad \text{is } \textit{true}.$$

Thus, SAT $\leq_L$ QBF and VAL $\leq_L$ QBF and so QBF is NP-hard and co-NP-hard.
In fact, QBF is PSPACE-complete.

QBF is in PSPACE

To see that QBF is in PSPACE, consider the algorithm that maintains a 1-bit register X for each Boolean variable appearing in the input formula ϕ and evaluates ϕ in the natural fashion.

The crucial cases are:

- If ϕ is $\exists X\ \psi$ then return T if *either* ($X \leftarrow$ T ; evaluate ψ) *or* ($X \leftarrow$ F ; evaluate ψ) returns T.
- If ϕ is $\forall X\ \psi$ then return T if *both* ($X \leftarrow$ T ; evaluate ψ) *and* ($X \leftarrow$ F ; evaluate ψ) return T.

PSPACE-completeness

To prove that QBF is PSPACE-complete, we want to show:

> *Given a machine M with a polynomial space bound and an input x, we can define a quantified Boolean formula ϕ_x^M which evaluates to true if, and only if, M accepts x.*
>
> *Moreover, ϕ_x^M can be computed from x in polynomial time (or even logarithmic space).*

The number of distinct configurations of M on input x is bounded by 2^{n^k} for some k ($n = |x|$).
Each configuration can be represented by n^k bits.

Constructing ϕ_x^M

We use tuples A, B of n^k Boolean variables each to encode *configurations* of M.

Inductively, we define a formula $\psi_i(A, B)$ which is *true* if the configuration coded by B is reachable from that coded by A in at most 2^i steps.

$$\begin{aligned}
\psi_0(A, B) &\equiv \text{“}A = B\text{''} \lor \text{“}A \to_M B\text{''} \\
\psi_{i+1}(A, B) &\equiv \exists Z \forall X \forall Y \ [(X = A \land Y = Z) \lor (X = Z \land Y = B) \\
&\qquad\qquad\qquad \Rightarrow \psi_i(X, Y)] \\
\phi &\equiv \psi_{n^k}(A, B) \land \text{“}A = \text{start''} \land \text{“}B = \text{accept''}
\end{aligned}$$

Reducing QBF to FO satisfaction

We have seen that *FO satisfaction* is in PSPACE.
To show that it is PSPACE-complete, it suffices to show that QBF $\leq_L$ FO sat.

The reduction maps a quantified Boolean formula ϕ to a pair $(\mathbb{A}, \phi^*)$ where $\mathbb{A}$ is a structure with two elements: 0 and 1 interpreting two constants f and t respectively.

ϕ^* is obtained from ϕ by a simple inductive definition.

Expressive Power of FO

For any *fixed* sentence ϕ of first-order logic, the class of structures $\mathrm{Mod}(\phi)$ is in L.

There are computationally easy properties that are not definable in first-order logic.

- There is no sentence ϕ of first-order logic such that $\mathbb{A} \models \phi$ if, and only if, $|A|$ is even.
- There is no formula $\phi(E, x, y)$ that defines the transitive closure of a binary relation E.

We will see proofs of these facts later on.

Second-Order Logic

We extend first-order logic by a set of *relational variables*.

For each $m \in \mathbb{N}$ there is an infinite collection of variables $\mathcal{V}^m = \{V_1^m, V_2^m, \ldots\}$ of *arity* m.

Second-order logic extends first-order logic by allowing *second-order quantifiers*

$$\exists X\, \phi \quad \text{for } X \in \mathcal{V}^m$$

A structure $\mathbb{A}$ satisfies $\exists X\, \phi$ if there is an m-ary relation R on the universe of $\mathbb{A}$ such that $(\mathbb{A}, X \to R)$ satisfies ϕ.

Existential Second-Order Logic

ESO—*existential second-order logic* consists of those formulas of second-order logic of the form:

$$\exists X_1 \cdots \exists X_k \, \phi$$

where ϕ is a first-order formula.

Examples

Evennness

This formula is true in a structure if, and only if, the size of the domain is even.

$$\begin{array}{ll} \exists B \exists S & \forall x \exists y B(x,y) \wedge \forall x \forall y \forall z B(x,y) \wedge B(x,z) \rightarrow y = z \\ & \forall x \forall y \forall z B(x,z) \wedge B(y,z) \rightarrow x = y \\ & \forall x \forall y S(x) \wedge B(x,y) \rightarrow \neg S(y) \\ & \forall x \forall y \neg S(x) \wedge B(x,y) \rightarrow S(y) \end{array}$$

Examples

Transitive Closure

This formula is true of a pair of elements a, b in a structure if, and only if, there is an E-path from a to b.

$$
\begin{array}{ll}
\exists P & \forall x \forall y\, P(x,y) \rightarrow E(x,y) \\
& \exists x P(a,x) \land \exists x P(x,b) \land \neg\exists x P(x,a) \land \neg\exists x P(b,x) \\
& \forall x \forall y (P(x,y) \rightarrow \forall z (P(x,z) \rightarrow y = z)) \\
& \forall x \forall y (P(x,y) \rightarrow \forall z (P(z,y) \rightarrow x = z)) \\
& \forall x ((x \neq a \land \exists y P(x,y)) \rightarrow \exists z P(z,x)) \\
& \forall x ((x \neq b \land \exists y P(y,x)) \rightarrow \exists z P(x,z))
\end{array}
$$

Examples

3-Colourability

The following formula is true in a graph (V, E) if, and only if, it is 3-colourable.

$$\begin{aligned}\exists R \exists B \exists G \quad & \forall x (Rx \lor Bx \lor Gx) \land \\ & \forall x (\quad \neg(Rx \land Bx) \land \neg(Bx \land Gx) \land \neg(Rx \land Gx)) \land \\ & \forall x \forall y (Exy \rightarrow (\quad \neg(Rx \land Ry) \land \\ & \qquad\qquad\qquad\quad \neg(Bx \land By) \land \\ & \qquad\qquad\qquad\quad \neg(Gx \land Gy)))\end{aligned}$$

Fagin's Theorem

Theorem (Fagin)
A class $\mathcal{C}$ of finite structures is definable by a sentence of *existential second-order logic* if, and only if, it is decidable by a *nondeterminisitic machine* running in polynomial time.

$$\mathrm{ESO} = \mathrm{NP}$$

One direction is easy: Given $\mathbb{A}$ and $\exists P_1 \ldots \exists P_m \phi$.

> *a nondeterministic machine can guess an interpretation for* $P_1, \ldots, P_m$ *and then verify* ϕ.

Fagin's Theorem

Given a machine M and an integer k, there is an ESO sentence ϕ such that $\mathbb{A} \models \phi$ if, and only if, M accepts $[\mathbb{A}]_<$, for some order $<$ in n^k steps.

We construct a *first-order* formula $\phi_{M,k}$ such that

$(\mathbb{A}, <, \mathrm{X}) \models \phi_{M,k} \quad \Leftrightarrow \quad$ X codes an accepting computation of M of length at most n^k on input $[\mathbb{A}]_<$

So, $\mathbb{A} \models \exists < \exists \mathrm{X}\, \phi_{M,k}$ if, and only if, there is some order $<$ on $\mathbb{A}$ so that M accepts $[\mathbb{A}]_<$ in time n^k.

Order

The formula $\phi_{M,k}$ is built up as the *conjunction* of a number of formulas. The first of these simply says that $<$ is a *linear order*

$$\begin{array}{l} \forall x(\neg x < x)\wedge \\ \forall x \forall y(x < y \rightarrow \neg y < x)\wedge \\ \forall x \forall y(x < y \vee y < x \vee x = y) \\ \forall x \forall y \forall z(x < y \wedge y < z \rightarrow x < z) \end{array}$$

We can use a linear order on the elements of $\mathbb{A}$ to define a lexicographic order on k-tuples.

Ordering Tuples

If $\mathsf{x} = x_1, \ldots, x_k$ and $\mathsf{y} = y_1, \ldots, y_k$ are k-tuples of variables, we use $\mathsf{x} = \mathsf{y}$ as shorthand for the formula $\bigwedge_{i<k} x_i = y_i$ and $\mathsf{x} < \mathsf{y}$ as shorthand for the formula

$$\bigvee_{i<k} \Big((\bigwedge_{j<i} x_j = y_j) \land x_i < y_i \Big)$$

We also write $\mathsf{y} = \mathsf{x} + 1$ for the following formula:

$$\mathsf{x} < \mathsf{y} \land \forall \mathsf{z} \big(\mathsf{x} < \mathsf{z} \to (\mathsf{y} = \mathsf{z} \lor \mathsf{y} < \mathsf{z}) \big)$$

Constructing the Formula

Let $M = (K, \Sigma, s, \delta)$.

The tuple $\mathbf{X}$ of second-order variables appearing in $\phi_{M,k}$ contains the following:

S_q	a k-ary relation symbol for each $q \in K$
T_σ	a $2k$-ary relation symbol for each $\sigma \in \Sigma$
H	a $2k$-ary relation symbol

Intuitively, these relations are intended to capture the following:

- $S_q(x)$ – the state of the machine at time x is q.
- $T_\sigma(x, y)$ – at time x, the symbol at position y of the tape is σ.
- $H(x, y)$ – at time x, the tape head is pointing at tape cell y.

We now have to see how to write the formula $\phi_{M,k}$, so that it enforces these meanings.

Initial state is s and the head is initially at the beginning of the tape.

$$\forall x \big((\forall y\, x \leq y) \rightarrow S_s(x) \wedge H(x,x) \big)$$

The head is never in two places at once

$$\forall x \forall y \big(H(x,y) \rightarrow (\forall z (y \neq z) \rightarrow (\neg H(x,z))) \big)$$

The machine is never in two states at once

$$\forall x \bigwedge_{q} (S_q(x) \rightarrow \bigwedge_{q' \neq q} (\neg S_{q'}(x)))$$

Each tape cell contains only one symbol

$$\forall x \forall y \bigwedge_{\sigma} (T_\sigma(x,y) \rightarrow \bigwedge_{\sigma' \neq \sigma} (\neg T_{\sigma'}(x,y)))$$

Initial Tape Contents

The initial contents of the tape are $[\mathbb{A}]_<$.

$$\begin{array}{ll} \forall x & x \leq n \rightarrow T_1(1,x) \wedge \\ & x \leq n^a \rightarrow (T_1(1, x+n+1) \leftrightarrow R_1(x|_a)) \\ \ldots & \end{array}$$

where,

$$x < n^a \quad : \quad \bigwedge_{i \leq (k-a)} x_i = 0$$

The tape does not change except under the head

$$\forall x \forall y \forall z (y \neq z \rightarrow (\bigwedge_{\sigma} (H(x,y) \wedge T_{\sigma}(x,z) \rightarrow T_{\sigma}(x+1,z)))$$

Each step is according to δ.

$$\forall x \forall y \bigwedge_{\sigma} \bigwedge_{q} (H(x,y) \wedge S_q(x) \wedge T_{\sigma}(x,y)) \rightarrow \bigvee_{\Delta} (H(x+1,y') \wedge S_{q'}(x+1) \wedge T_{\sigma'}(x+1,y))$$

where Δ is the set of all triples (q', σ', D) such that $((q, \sigma), (q', \sigma', D)) \in \delta$ and

$$y' = \begin{cases} y & \text{if } D = S \\ y - 1 & \text{if } D = L \\ y + 1 & \text{if } D = R \end{cases}$$

Finally, some accepting state is reached

$$\exists x\, S_{\text{acc}}(x)$$

NP

Recall that a languge L is in NP if, and only if,

$$L = \{x \mid \exists y R(x, y)\}$$

where R is *polynomial-time decidable* and *polynomially-balanced*.

Fagin's theorem tells us that polynomial-time decidability can, in some sense, be replaced by *first-order definability*.

co-NP

USO—*universal second-order logic* consists of those formulas of second-order logic of the form:

$$\forall X_1 \cdots \forall X_k \, \phi$$

where ϕ is a first-order formula.

A corollary of Fagin's theorem is that a class $\mathcal{C}$ of finite structures is definable by a sentence of *universal second-order logic* if, and only if, its complement is decidable by a *nondeterminisitic machine* running in polynomial time.

$$\text{USO} = \text{co-NP}$$

Second-Order Alternation Hierarchy

We can define further classes by allowing other second-order *quantifier prefixes*.

$\Sigma_1^1 = \text{ESO}$

$\Pi_1^1 = \text{USO}$

Σ_{n+1}^1 is the collection of properties definable by a sentence of the form: $\exists X_1 \cdots \exists X_k \, \phi$ where ϕ is a Π_n^1 formula.

Π_{n+1}^1 is the collection of properties definable by a sentence of the form: $\forall X_1 \cdots \forall X_k \, \phi$ where ϕ is a Σ_n^1 formula.

Note: every formula of second-order logic is Σ_n^1 and Π_n^1 for some n.

Polynomial Hierarchy

We have, for each n:

$$\Sigma_n^1 \cup \Pi_n^1 \quad \subseteq \quad \Sigma_{n+1}^1 \cap \Pi_{n+1}^1$$

The classes together form the *polynomial hierarchy* or PH.

$\mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE}$
$\mathrm{P} = \mathrm{NP}$ if, and only if, $\mathrm{P} = \mathrm{PH}$