Topics in Logic and Complexity
Handout 4

Anuj Dawar

MPhil Advanced Computer Science, Lent 2010

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* $(X \leftarrow \mathsf{T}\ \ ;\ \ \mathsf{evaluate}\ \psi)$ *or* $(X \leftarrow \mathsf{F}\ \ ;\ \ \mathsf{evaluate}\ \psi)$ returns T.
- If ϕ is $\forall X\ \psi$ then return T if *both* $(X \leftarrow \mathsf{T}\ \ ;\ \ \mathsf{evaluate}\ \psi)$ *and* $(X \leftarrow \mathsf{F}\ \ ;\ \ \mathsf{evaluate}\ \psi)$ 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 $\mathbf{A}, \mathbf{B}$ of n^k Boolean variables each to encode *configurations* of M.

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

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

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.

$$\begin{aligned}\exists B \exists S \quad & \forall x \exists y B(x,y) \wedge \forall x \forall y \forall z B(x,y) \wedge B(x,z) \to y = z \\ & \forall x \forall y \forall z B(x,z) \wedge B(y,z) \to x = y \\ & \forall x \forall y S(x) \wedge B(x,y) \to \neg S(y) \\ & \forall x \forall y \neg S(x) \wedge B(x,y) \to S(y)\end{aligned}$$

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.

$$\begin{aligned}\exists P \quad & \forall x \forall y\, P(x,y) \to E(x,y) \\ & \exists x P(a,x) \wedge \exists x P(x,b) \wedge \neg \exists x P(x,a) \wedge \neg \exists x P(b,x) \\ & \forall x \forall y (P(x,y) \to \forall z (P(x,z) \to y = z)) \\ & \forall x \forall y (P(x,y) \to \forall z (P(z,x) \to y = z)) \\ & \forall x ((x \neq a \wedge \exists y P(x,y)) \to \exists z P(z,x)) \\ & \forall x ((x \neq b \wedge \exists y P(y,x)) \to \exists z P(x,z))\end{aligned}$$

Examples

3-Colourability

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

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

Reading List for this Handout

1. Papadimitriou. Chapter 5. Section 19.1.
2. Grädel et al. Section 3.1