A Course in Logic and Complexity

Descriptive Complexity Theory

Tomáš Jirotka

March 27, 2010

Second-Order Logic

- Second-order allows quantification of relations.
- Formally, second-order formula can be built up from atomic formulae using the following rules:
  - If φ is a formula then so is $\neg\varphi$.
  - If φ and ψ are formulae then so is $\varphi \wedge \psi$.
  - If φ is a formula then so is $\exists x\varphi$ for any variable x.
  - If φ is a formula then so is $\exists R^n\varphi$ for any n and n-ary relation R.
- Example: Any first-order formula is a formula of second-order logic too.
- Example: $\exists P \forall x \forall y (P(x, y) \leftrightarrow P(y, x))$.
- We may naturally define $\forall R^n\varphi$ as $\neg\exists R^n\neg\varphi$.

Existential and Universal SO Sentences

- Let $\mathcal{V}$ be a finite vocabulary.
- We let $\exists$SO denote the set of second-order sentences of the form

$$\exists_1 R_1^{n_1} \exists_2 R_2^{n_2} \ldots \exists_k R_k^{n_k} \psi$$

where ψ is a first-order $\mathcal{V}$-sentence.

- Analogically, by $\forall$SO we mean the set of second-order sentences of the form

$$\forall_1 R_1^{n_1} \forall_2 R_2^{n_2} \ldots \forall_k R_k^{n_k} \psi$$

where ψ is a first-order $\mathcal{V}$-sentence.

Complexity Class **P**

- A *decision problem* is a question which has answer "yes" or "no". Usually we have a *language* $L \subseteq \{0,1\}^*$ and ask whether a word x falls into L.
- An algorithm is *polynomial-time* if there exists $k \geq 1$ such that, given any input of length n, the algorithm halts in fewer than n^k steps.
- A decision problem is polynomial-time if it can be decided by a polynomial-time algorithm. The set of all such problems is denoted **P**.
- Example: Given an integer n decide whether it is a prime. This problem is known to be in **P**.

Complexity Class **NP**

- An algorithm is called *nondeterministic* if it has more than one possible choice at each step.
- Class **NP** contains all nondeterministic polynomial-time decision problems.
- Alternative definition of **NP**: language L is in **NP** iff there exits a polynomial q, and a polynomial-time deterministic algorithm M, such that for all $x \in L$ there is a witness y of length at most $q(|x|)$ such that $M(x, y) = 1$, and for all $x \notin L$ the algorithm returns 0 for any witness y.
- Example: Given an integer n decide whether its factor lies between $\sqrt[4]{n}$ and $\sqrt{n}$. This problem is obviously is **NP**, but we do not know if it is in **P**.

Some More Classes

- In a similar way as we have defined **P** we define the class **EXPTIME**. It is a set of languages decidable in exponential time with respect to the length of input, i.e.

$$\mathbf{EXPTIME} = \bigcup_k \mathrm{Time}\left(2^{n^k}\right) .$$

- Example: Decide if a deterministic Turing machine halts in fewer than k steps.
- Analogically, **NEXPTIME** is a class of languages decidable in nondeterministic exponential time.
- Obviously, $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME}$.
- It is also known that $\mathbf{P} \subset \mathbf{EXPTIME}$ and $\mathbf{NP} \subset \mathbf{NEXPTIME}$.

Complements

- Consider a language $L \subseteq \{0,1\}^*$. The set $\{0,1\}^* \setminus L$ is called the *complement* of L.
- The set of complements to the languages which are in **NP** forms the class **coNP**.
- Example: Given a propositional formula φ decide whether it is unsatisfiable. The set of all unsatisfiable formulae lies in **coNP**.
- It holds: $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$.

NP-completeness

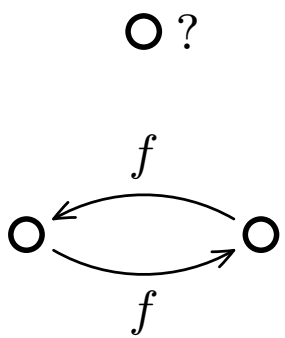
- For $A \subseteq \{0,1\}^m$ and $B \subseteq \{0,1\}^n$, we say that A is *polynomial-time reducible* to B, if there exists a polynomial-time algorithm that computes a function $f : A \to B$ such that $x \in A$ iff $f(x) \in B$.
- Let us have a language L. We say that L is **NP**-*complete* if (i) it is in **NP**, and (ii) any other language $\Lambda \in$ **NP** is polynomial-time reducible to L.
- Example: The language SAT containing all satisfiable first-order formulae is **NP**-complete.

Finite Spectrum

- A *finite structure* is a non-empty set, along with certain given functions and relations on that set.
- If σ is a sentence of first-order logic, then the *spectrum* of σ is the set of cardinalities of finite structures (i.e. subset of natural numbers) in which σ is true.
- Example: Let f be a unary function symbol and

$$\sigma \equiv \forall x\,(f(x) \neq x) \wedge \forall x \forall y\,(f(x) = y \leftrightarrow f(y) = x)\ .$$

Then the spectrum of σ is the set of all even numbers.



Spectrum Problem

- In 1952 Scholz posed the problem of characterising the class of spectra.
- Later, in 1956 Asser asked whether the complement of each spectrum is also a spectrum. This problem still remains open.

Theorem (Neil Jones, Alan Selman, 1974)

A set $S \subseteq \{0,1\}^*$ *is a spectrum iff* $S \in$ **NEXPTIME**.

- Hence, spectra are closed under complementation if and only if **NEXPTIME** = **coNEXPTIME**.

Generalised Spectrum

- Note that the spectrum of a first-order sentence ψ of relational vocabulary $\{R_1, \ldots, R_m\}$ can be viewed as the set of finite models of the $\exists$SO sentence $\exists R_1 \ldots \exists R_m \psi$.
- There is a one-to-one correspondence between the spectra of first-order sentences and the classes of finite models of $\exists$SO sentences over the empty vocabulary.
- A *generalised spectrum* is the class of finite models of a sentence in existential second-order logic.
- Example: The class of bipartite graphs is a generalised spectrum. It is defined by the sentence

$$\exists R \forall x \forall y (E(x, y) \rightarrow (R(x) \leftrightarrow \neg R(y))) .$$

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

The set of all properties expressible in existential second-order logic over some finite non-empty vocabulary equals precisely the complexity class **NP**.

Proof of $\exists SO \subseteq$ **NP**.

- Having the formula $\exists S_1 \ldots \exists S_t \phi$ we want to find an **NP**-machine M deciding whether this formula is satisfied.
- An accepting computation exists iff an interpretation of relations exists iff the sentence is satisfiable.
- M "guesses" the interpretations of relations S_i.
- Then, in polynomial-time, it decides if the formula holds.

Cook's Theorem

Corollary (Stephen Cook, 1971)

SAT is **NP***-complete.*

Proof.

- $SAT \in \mathbf{NP}$ is trivial.
- Consider $L \in \mathbf{NP}$. It is equivalent to some formula ϕ of the form $\exists R_1 \ldots \exists R_k \forall x_1 \ldots \forall x_n \psi(x_1, \ldots, x_n)$ where ψ is quantifier-free first-order formula in CNF.
- We are given a structure $\mathcal{M}$ and we ask if $\mathcal{M} \vDash \phi$.
- Rewrite ϕ to the form $\bigwedge_{\bar{a} \in M} \psi(a_1, \ldots, a_n)$.
- For each literal check whether it holds for $\bar{a}$ in $\mathcal{M}$. If so, delete each clause containing that literal. Otherwise, delete only that literal from the sentence $\psi(\bar{a})$.
- Introducing new variables for each quantified relation and its elements we obtain a formula ψ' and determine if it satisfiable.

Example

- Suppose we have a formula $\exists R \forall x \forall y (R(x, y) \vee P(x))$, and $\mathcal{M}$ is given by $M = \{a, b, c\}$, $P(a) = P(b) = 1$, $P(c) = 0$.
- The FO part we translate into formula $(R(a, a) \vee P(a)) \wedge (R(a, b) \vee P(a)) \wedge (R(a, c) \vee P(a)) \wedge (R(b, a) \vee P(b)) \wedge (R(b, b) \vee P(b)) \wedge (R(b, c) \vee P(b)) \wedge (R(c, a) \vee P(c)) \wedge (R(c, b) \vee P(c)) \wedge (R(c, c) \vee P(c))$.
- Since the relation R was quantified (and hence we do not know its thuth table) we introduce new variables r_{aa}, r_{ab}, r_{ac}, r_{ba}, r_{bb}, r_{bc}, r_{ca}, r_{cb}, r_{cc}, while we substitute the values of P.
- Now we are able to write a propositional formula $r_{ca} \wedge r_{cb} \wedge r_{cc}$ which is an input for SAT.