Chapter 7

Second-Order Logic and Fagin's Theorem

Second-order logic consists of first-order logic plus the power to quantify over relations on the universe. We prove Fagin's theorem which says that the queries computable in NP are exactly the second-order existential queries. A corollary due to Stockmeyer says that the second-order queries are exactly those computable in the polynomial-time hierarchy.

7.1 Second-Order Logic

Second-order logic consists of first-order logic plus new relation variables over which we may quantify. For example, the formula $(\forall A^r)\varphi$ means that for all choices of r-ary relation A, φ holds. Let SO be the set of second-order expressible boolean queries.

Any second-order formula may be transformed into an equivalent formula with all second-order quantifiers in front. If all these second-order quantifiers are existential, then we have a second-order existential formula. Let (SO$\exists$) be the set of second-order existential boolean queries. Consider the following example, in which R, Y, and B are unary relation variables. To indicate their arity, we place exponents on relation variables where they are quantified.

$$\Phi_{\text{3-color}} \equiv (\exists R^1)(\exists Y^1)(\exists B^1)(\forall x)\Big[\big(R(x) \vee Y(x) \vee B(x)\big) \wedge (\forall y)\Big(E(x,y) \rightarrow \neg\big(R(x) \wedge R(y)\big) \wedge \neg\big(Y(x) \wedge Y(y)\big) \wedge \neg\big(B(x) \wedge B(y)\big)\Big)\Big]$$

Observe that a graph G satisfies $\Phi_{\text{3-color}}$ iff G is 3-colorable. Three colorability of graphs is an NP complete problem (3-COLOR). In Section 7.2, we see that three colorability remains complete via first-order reductions. It will then follow that every query computable in NP is describable in SO$\exists$.

Second-order logic is extremely expressive. For this reason, it is very easy to write second-order specifications of queries. For the same reason, such specifications are not feasible to execute without further refinement (cf. Section 9.6). Recall that the first-order queries are those that can be computed on a CRAM in constant time, using polynomially many processors (Theorem 5.2). We will see that the second-order queries are those that can be computed in constant parallel time, but using exponentially many processors (Corollary 7.27).

Here are a few other examples of SO$\exists$ queries.

Example 7.1 SAT is the set of boolean formulas in conjunctive normal form (CNF) that admit a satisfying assignment (Example 2.18).

The boolean query SAT is expressible in SO$\exists$ as follows:

$$\Phi_{\text{SAT}} \equiv (\exists S)(\forall x)(\exists y)((P(x,y) \wedge S(y)) \vee (N(x,y) \wedge \neg S(y))) .$$

Φ_{SAT} asserts that there exists a set S of variables — the variables that should be assigned **true** — that is a satisfying assignment of the input formula. □

Example 7.2 Boolean query CLIQUE is the set of pairs $\langle G, k\rangle$ such that graph G has a complete subgraph of size k (Example 2.10). The vocabulary for CLIQUE is $\tau_{gk} = \langle E^2, k\rangle$.

The SO$\exists$ sentence Φ_{CLIQUE} says that there is a numbering of the vertices such that those vertices numbered less than k form a clique. In order to describe this numbering it is convenient to existentially quantify a function f. This can be replaced by a binary relation in the usual way (Exercise 7.3). Let Inj(f) mean that f is an injective function,

$$\begin{aligned} \text{Inj}(f) &\equiv (\forall xy)(f(x) = f(y) \rightarrow x = y) \\ \Phi_{\text{CLIQUE}} &\equiv (\exists f^1.\text{Inj}(f))(\forall xy)((x \neq y \wedge f(x) < k \wedge f(y) < k) \rightarrow E(x,y)) \end{aligned}$$

□

Exercise 7.3 Show how formula Φ_{CLIQUE} may be rewritten using an existentially quantified relation F of arity two, rather than function f. □

Exercise 7.4 Hamiltonian-Circuit (HC) is the boolean query that is true of an undirected graph iff it has a Hamiltonian circuit, i.e., a path that starts and ends at the same vertex and visits every other vertex exactly once. Write an SO$\exists$ sentence that expresses HC. [Hint: say that there exists a total ordering of the vertices that determines a Hamiltonian circuit.] □

Exercise 7.5 Write an SO$\exists$ sentence that expresses TSP — the traveling salesperson problem. Boolean query TSP has as input an undirected graph G with weights on its edges and an integer L. The TSP query is true iff G admits a Hamiltonian circuit whose total weight is at most L. In order to code TSP instances as logical structures, we must decide on an appropriate range for the integer weights. To be quite general, you should code these integers as binary strings. Let the vocabulary for TSP be $\tau_{\text{tsp}} = \langle W^3, L^1 \rangle$, consisting of a binary string $W(x, y, \cdot)$ for each potential edge (x, y) and a binary string $L(\cdot)$ representing limit L. For pairs (x, y) that are not edges, we can let the edge weight be the maximum value, i.e., the string of all 1's.

[Hint: you can assert the existence of the correct Hamiltonian-Circuit as in Exercise 7.4. To say that the total is at most L, you should assert the existence of a ternary relation R that maintains the running sum.] □

We finish this section by proving the easy direction of Fagin's Theorem.

Proposition 7.6 *The second-order existentially definable boolean queries are all computable in NP. In symbols,* SO$\exists$ $\subseteq$ NP.

Proof Given is a second-order existential sentence $\Phi \equiv (\exists R_1^{r_1}) \ldots (\exists R_k^{r_k})\psi$. Let τ be the vocabulary of Φ. Our task is to build an NP machine N such that for all $\mathcal{A} \in \text{STRUC}[\tau]$,

$$(\mathcal{A} \models \Phi) \quad \Leftrightarrow \quad (N(\text{bin}(\mathcal{A}))\downarrow) \tag{7.7}$$

Let $\mathcal{A}$ be an input structure to N and let $n = \|\mathcal{A}\|$. What N does is to nondeterministically write down a binary string of length n^{r_1} representing R_1, and similarly for R_2 through R_k. By nondeterministically write down a binary string,

we mean that at each step, N nondeterministically chooses whether to write a 0 or a 1. After this polynomial number of steps, we have an expanded structure $\mathcal{A}' = (\mathcal{A}, R_1, R_2, \ldots, R_k)$. N should accept iff $\mathcal{A}' \models \psi$. By Theorem 3.1, we can test if $\mathcal{A}' \models \psi$ in logspace, so certainly in NP. Notice that N accepts $\mathcal{A}$ iff there is some choice of relations R_1 through R_k such that $(\mathcal{A}, R_1, R_2, \ldots, R_k) \models \psi$. Thus, Equivalence 7.7 holds. □

7.2 Proof of Fagin's Theorem

The following theorem characterizes complexity class NP in an elegant and machine independent way. This was originally proved in Ron Fagin's 1973 doctoral thesis. It was the theorem that began the subject of descriptive complexity.

Theorem 7.8 (Fagin's Theorem) NP *is equal to the set of existential, second-order boolean queries,* NP = SO∃. *Furthermore, this equality remains true when the first-order part of the second-order formulas is restricted to be universal.*

Proof Let N be a nondeterministic Turing machine that uses time $n^k - 1$ for inputs bin($\mathcal{A}$) with $n = \|\mathcal{A}\|$. We write a second-order sentence,

$$\Phi \quad = \quad (\exists C_1^{2k} \ldots C_g^{2k} \Delta^k)\varphi$$

that says, "There exists an accepting computation $\overline{C}, \Delta$ of N." More precisely, first-order sentence φ will have the property that $(\mathcal{A}, \overline{C}, \Delta) \models \varphi$ iff $\overline{C}, \Delta$ is an accepting computation of N on input $\mathcal{A}$. That is, Equation 7.7 holds.

We describe how to code N's computation. $\overline{C}$ consists of a matrix $\overline{C}(\bar{s}, \bar{t})$ of n^{2k} tape cells with space $\bar{s}$ and time $\bar{t}$ varying between 0 and $n^k - 1$. We use k-tuples of variables $\bar{t} = t_1, \ldots, t_k$ and $\bar{s} = s_1, \ldots s_k$ each ranging over the universe of $\mathcal{A}$, i.e. from 0 to $n - 1$, to code these values. For each $\bar{s}, \bar{t}$ pair, $\overline{C}(\bar{s}, \bar{t})$ codes the tape symbol σ that appears in cell $\bar{s}$ at time $\bar{t}$ if N's head is not on this cell. If the head is present, then $\overline{C}(\bar{s}, \bar{t})$ codes the pair $\langle q, \sigma \rangle$ consisting of N's state q at time $\bar{t}$ and tape symbol σ. Let $\Gamma = \{\gamma_0, \ldots, \gamma_g\} = (Q \times \Sigma) \cup \Sigma$ be a listing of the possible contents of a computation cell. We will let C_i be a $2k$-ary relation variable for $0 \leq i \leq g$. The intuitive meaning of $C_i(\bar{s}, \bar{t})$ is that computation cell $\bar{s}$ at time $\bar{t}$ contains symbol γ_i.

At each step, the nondeterministic Turing machine will make one of at most two possible choices.[1] We encode these choices in k-ary relation Δ. Intuitively, $\Delta(\bar{t})$ is true, if step $\bar{t}+1$ of the computation makes choice "1"; otherwise it makes choice "0". Note that these choices can be determined from $\bar{C}$, but the formula is simplified when we explicitly quantify Δ. See Figure 7.9 for a view of N's computation.

It is now fairly straightforward to write the first-order sentence $\varphi(\overline{C}, \Delta)$ saying that $\overline{C}, \Delta$ codes a valid accepting computation of N. The sentence φ consists of four parts,

$$\varphi \quad \equiv \quad \alpha \wedge \beta \wedge \eta \wedge \zeta,$$

where α asserts that row 0 of the computation correctly codes input bin($\mathcal{A}$), β says that it is never the case that $C_i(\bar{s}, \bar{t})$ and $C_j(\bar{s}, \bar{t})$ both hold, for $i \neq j$, η says that for all $\bar{t}$, row $\bar{t}+1$ of $\overline{C}$ follows from row $\bar{t}$ via move $\Delta(\bar{t})$ of N, and ζ says that the last row of the computation includes the accept state. We can write sentence ζ explicitly. We may assume that when N accepts it clears its tape and moves all the way to the left and enters a unique accept state q_f. Let γ_{17} be the member of Γ corresponding to the pair $\langle q_f, 1\rangle$ of state q_f, looking at the symbol 1. Then ζ = $C_{17}(\overline{0}, \overline{max})$.

Sentence α must assert that the input is of length $I_\tau(n)$ for some n and that $\mathcal{A}$ has been correctly coded as bin($\mathcal{A}$) (cf. Exercise 2.3). For example, suppose that τ includes relation symbol R_1 of arity one. Assume that cell symbols γ_0, γ_1 are '0','1', respectively. Then α includes the following clauses, meaning that cell $0 \ldots 0 s_k$ contains 1 if $R_1(s_k)$ holds and 0 if it doesn't.

$$\begin{aligned} \cdots \quad & \wedge \Big(\bar{t} = 0 = s_1 = \ldots = s_{k-1} \wedge s_k \neq 0 \wedge R_1(s_k) \;\rightarrow\; C_1(\bar{s}, \bar{t})\Big) \\ \wedge \quad & \Big(\bar{t} = 0 = s_1 = \ldots = s_{k-1} \wedge s_k \neq 0 \wedge \neg R_1(s_k) \;\rightarrow\; C_0(\bar{s}, \bar{t})\Big) \;\wedge \cdots \end{aligned}$$

The following sentence η asserts that the contents of tape cell $(\bar{s}, \bar{t}+1)$ follows from the contents of cells $(\bar{s}-1, \bar{t})$, $(\bar{s}, \bar{t})$, and $(\bar{s}+1, \bar{t})$ via the move $\Delta(\bar{t})$ of N. Let $\langle a_{-1}, a_0, a_1, \delta\rangle \xrightarrow{N} b$ mean that the triple of cell contents a_{-1}, a_0, a_1 lead to cell b via move δ of N.

[1] A nondeterministic Turing machine can make one of at most a bounded number of choices at any step. By reducing this to a binary choice per step, we slow the machine down by a small constant factor and make the analysis simpler.

	Space 0	1	p	$n-1$	n		n^k-1	Δ
Time 0	$\langle q_0, w_0\rangle$	w_1	$\cdots$	w_{n-1}	$\sqcup$	$\cdots$	$\sqcup$	δ_0
1	w_0	$\langle q_1, w_1\rangle$	$\cdots$	w_{n-1}	$\sqcup$	$\cdots$	$\sqcup$	δ_1
	$\vdots$	$\vdots$	$\vdots$			$\vdots$		$\vdots$
t			a_{-1} a_0 a_1					δ_t
$t+1$			b					δ_{t+1}
	$\vdots$	$\vdots$	$\vdots$			$\vdots$		$\vdots$
n^k-1	$\langle q_f, 1\rangle$	$\cdots$	$\cdots$			$\cdots$		

Figure 7.9: An NP computation on input $w_0 w_1 \cdots w_{n-1}$; $\sqcup$ denotes blank

$$\eta_1 \equiv (\forall \bar{t}.\bar{t} \neq \overline{max})(\forall \bar{s}.\bar{0} < \bar{s} < \overline{max}) \bigwedge_{\langle a_{-1},a_0,a_1,\delta\rangle \xrightarrow{N} b} \Big(\neg^\delta \Delta(\bar{t}) \vee \neg C_{a_{-1}}(\bar{s}-1,\bar{t}) \vee \neg C_{a_0}(\bar{s},\bar{t}) \vee \neg C_{a_1}(\bar{s}+1,\bar{t}) \vee C_b(\bar{s},\bar{t}+1)\Big)$$

Here, $\neg^\delta$ is $\neg$ if $\delta = 1$ and is the empty symbol if $\delta = 0$.

Finally, let $\eta \equiv \eta_0 \wedge \eta_1 \wedge \eta_2$ where η_0 and η_2 encode the same information when $\bar{s} = \overline{0}$ and $\overline{max}$ respectively. □

Observe that the first-order part of formula Φ in the proof of Theorem 7.8 is universal and is in conjunctive normal form. Furthermore, if N is a deterministic polynomial-time machine, then we do not need choice relation Δ, so the first-order part of Φ is a Horn formula.[2] We obtain the following corollary, which is part of Grädel's Theorem (Theorem 9.32).

Corollary 7.10 *Every polynomial-time query is expressible as a second-order, existential Horn formula:* P $\subseteq$ SO$\exists$*-Horn.*

The proof of Theorem 7.8 shows that nondeterministic time n^k is contained in (SO$\exists$, arity $2k$). Lynch improved this to arity k. His proof uses the numeric predicate PLUS. Fagin's theorem holds even without numeric predicates, since we

[2] A Horn formula is a formula in conjunctive normal form with at most one positive literal per clause (Definition 9.26).

can existentially quantify binary relations and assert that they are $\leq$ and BIT. However, without the numeric predicates, we need an existential first-order quantifier to specify time $\bar{t}+1$, given time $\bar{t}$.

Theorem 7.11 (Lynch's Theorem)

$$\textit{For } k \geq 1, \quad \text{NTIME}[n^k] \quad \subseteq \quad \text{SO}\exists(\text{arity } k) \ .$$

Proof This is analogous to Lemma 5.31. We modify the proof of Fagin's theorem so that instead of guessing the entire tape at every step only a bounded number of bits per step is guessed. The following relations need to be guessed.

1. $Q_i(\bar{t})$ meaning that the state at move $\bar{t}$ is q_i,
2. $S_i(\bar{t})$ meaning that the symbol written at move $\bar{t}$ is σ_i,
3. $D(\bar{t})$ meaning that the head moves one space to the right after move $\bar{t}$; otherwise it moves one space to the left.

We must write a first-order formula asserting that $\overline{Q}, \overline{S}, D$ encode a correct accepting computation of N. The only difficulty in doing this is that for each move $\bar{t}$, we must ascertain the symbol $\rho_{\bar{t}}$ that is read by N. $\rho_{\bar{t}}$ is equal to σ_i where $S_i(\bar{t}')$ holds, and $\bar{t}'$ is the last time before $\bar{t}$ that the head was in its present location (or it is the corresponding input symbol if this is the first time the head is at this cell).

To express $\rho_{\bar{t}}$, we need to express the function $\bar{s} = p(\bar{t})$ meaning that at time $\bar{t}$, the head is at position $\bar{s}$. Since we are restricted to relations of arity k, we cannot guess the $k \log n$ bits per time needed to specify the function p. The solution to this problem is to do the next best thing and existentially quantify the current head position once every $\log n$ steps. We do this by quantifying k bits per step in relations $P_i(\bar{t})$, $i = 1, 2, \ldots, k$. When we string $\log n$ of these together, from time $r \log n$ through time $(r+1)\log n - 1$, we have a total of $k \log n$ bits which encode the head position at time $r \log n$.

The idea is similar to the proof of Bit Sum Lemma 1.18. Recall that numeric predicate BIT allows us to use each first-order variable to store $\log n$ bits. Furthermore, predicate BSUM(x, y), meaning that the number of one's in the binary expansion of x is y, is first-order (Lemma 1.18). This enables us to assert that relations $\overline{P}$ are consistent with the head movements given by D and thus correctly code the head position at $\log n$ step intervals. Finally, using BSUM again, we can ascertain the head position at any time $\bar{t}$. $\square$

The converse of Lynch's Theorem is an open question:

Open Problem 7.12 Prove or disprove: SO$\exists$(arity k) $=$ NTIME$[n^k]$

The subtlety in Open Problem 7.12 is that the first-order part of an SO$\exists$(arity k) formula may have more than k universal quantifiers. Thus, a first step in answering Open Problem 7.12 may be to answer:

Open Problem 7.13 Is there a fixed k such that FO $\subseteq$ DTIME$[n^k]$? Is there a fixed k such that FO $\subseteq$ NTIME$[n^k]$?

Grandjean gave a close relationship between nondeterministic time n^k and the class (SO$\exists$, fun, $k\forall$) of properties expressible by second-order existential sentences including function variables and containing only k universal first-order quantifiers.

Fact 7.14 *For $k \geq 2$,* NTIME$[n^k] \subseteq$ (SO$\exists$, *fun*, $k\forall$) $=$ (SO$\exists$, *fun*, $k\forall$, *arity* k) $\subseteq$ NTIME$[n^k(\log n)^2]$.

By considering the nondeterministic random access machine (NRAM) instead of the Turing machine, Grandjean later gave an exact bound,

Fact 7.15 *For $k \geq 1$,*

$$\textit{NRAM-TIME}[n^k] = (\text{SO}\exists, \textit{fun}, k\forall, \textit{arity } k) \ .$$

7.3 NP-Complete Problems

In 1971, Cook proved that SAT (Example 2.18) is NP-complete via polynomial-time Turing reductions [Coo71]. In fact, the problem is NP-complete via significantly weaker reductions:

Theorem 7.16 SAT *is complete for* NP *via first-order reductions.*

Proof This follows from Fagin's theorem. Given any boolean query $B \in$ NP, we know that $B =$ MOD$[\Phi]$ where $\Phi = (\exists S_1^{a_1} \cdots S_g^{a_g} \Delta^k)(\forall x_1 \cdots x_t)\psi(\bar{x})$, with ψ quantifier-free. We may assume that $\psi(\bar{x}) = \bigwedge_{j=1}^{r} C_j(\bar{x})$ is in conjunctive normal form.

For any input structure $\mathcal{A}$ with $n = \|\mathcal{A}\|$, define the boolean formula $\gamma(\mathcal{A})$ as follows: $\gamma(\mathcal{A})$ has boolean variables: $S_i(e_1, \ldots, e_{a_i})$ and $D(e_1, \ldots, e_k)$, $i =$

$1, \ldots, g$, $e_1, \ldots, e_{a_i} \in |\mathcal{A}|$. The clauses of $\gamma(\mathcal{A})$ are $C_j(\bar{e})$, $j = 1, \ldots, r$ as $\bar{e}$ ranges over all t-tuples from $|\mathcal{A}|$. In each $C_j(\bar{e})$, there may be some occurrences of numeric or input predicates: $P(\bar{e})$. These should be replaced by **true** or **false** according to whether they are true or false in $\mathcal{A}$.

It is clear from the construction that

$$\mathcal{A} \in B \qquad \Leftrightarrow \qquad \mathcal{A} \models \Phi \qquad \Leftrightarrow \qquad \gamma(\mathcal{A}) \in \text{SAT} \; .$$

Furthermore, the mapping from $\mathcal{A}$ to $\gamma(\mathcal{A})$ is a $t+1$-ary first-order query. □

Now that we know that SAT is NP-complete via first-order reductions, we can reduce SAT to other SO∃ boolean queries. This is possible iff these other problems are also NP-complete via first-order reductions (Exercise 2.15).

Proposition 7.17 *Let* 3-SAT *be the subset of* SAT *in which each clause has at most three literals. Then* 3-SAT *is* NP*-complete via first-order reductions.*

Proof We show that SAT $\leq_{\text{fo}}$ 3-SAT. Here is an example of the idea behind the reduction. Let $C = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_7)$ be a clause with more than three literals. Observe that $C \in$ SAT iff $C' \in$ 3-SAT, where C' is the following clause in which new variables $d_1, \ldots, d_4$ are introduced.

$$\begin{aligned} C' \;\equiv\; & (\ell_1 \vee \ell_2 \vee d_1) \wedge (\overline{d_1} \vee \ell_3 \vee d_2) \wedge (\overline{d_2} \vee \ell_4 \vee d_3) \wedge \\ & (\overline{d_3} \vee \ell_5 \vee d_4) \wedge (\overline{d_4} \vee \ell_6 \vee \ell_7) \end{aligned}$$

The first-order reduction from SAT to 3-SAT proceeds as follows. Let $\mathcal{A} \in$ STRUC$[\langle P^2, N^2 \rangle]$ be an instance of SAT with $n = \|\mathcal{A}\|$. Each clause c of $\mathcal{A}$ is replaced by $2n$ clauses as follows:

$$\begin{aligned} c' \;\equiv\; & ([x_1]^c \vee d_1) \wedge (\overline{d_1} \vee [x_2]^c \vee d_2) \wedge (\overline{d_2} \vee [x_3]^c \vee d_3) \wedge \cdots \wedge \\ & (\overline{d_n} \vee [\overline{x_1}]^c \vee d_{n+1})(\overline{d_{n+1}} \vee [\overline{x_2}]^c \vee d_{n+2}) \wedge \cdots \wedge (\overline{d_{2n-1}} \vee [\overline{x_n}]^c) \end{aligned}$$

Here $[\ell]^c$ means the literal ℓ if this occurs in c and **false** otherwise. It is not hard to see that c' is satisfiable iff c is satisfiable and that c' is definable in a first-order way from c. □

Proposition 7.18 3-COLOR *is* NP*-complete via first-order reductions.*

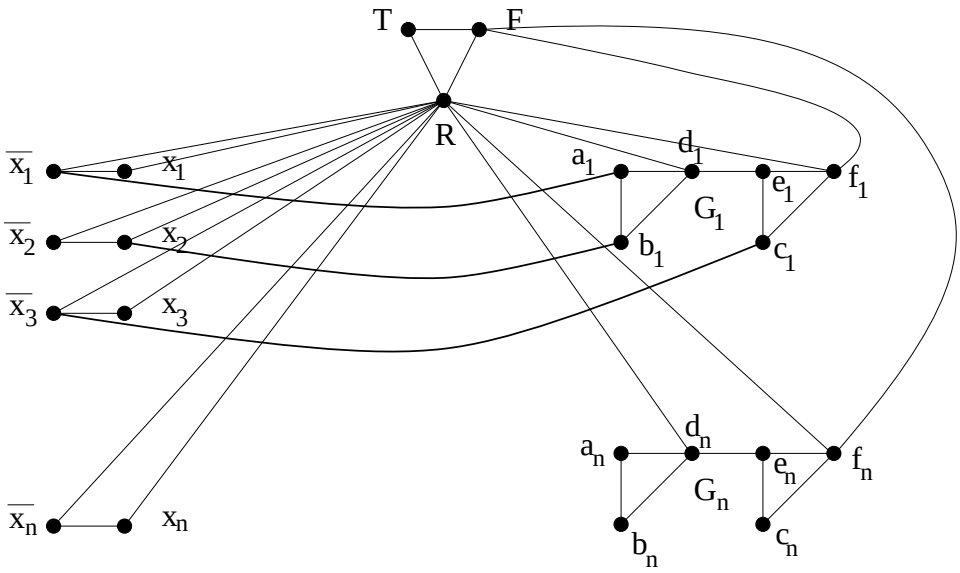

Figure 7.19: 3-SAT $\leq_{\text{fo}}$ 3-COLOR; G_1 encodes clause $C_1 = (\overline{x_1} \vee x_2 \vee \overline{x_3})$

Proof We will show that 3-SAT $\leq_{\text{fo}}$ 3-COLOR. We are given an instance $\mathcal{A}$ of 3-SAT and we must produce a graph $f(\mathcal{A})$ that is three colorable iff $\mathcal{A} \in$ 3-SAT. Let $n = \|\mathcal{A}\|$, so $\mathcal{A}$ is a boolean formula with at most n variables and n clauses.

The construction of $f(\mathcal{A})$ is shown in Figure 7.19. Notice the triangle, with vertices labeled T, F, R. Any three-coloring of the graph must color these three vertices distinct colors. We may assume without loss of generality that the colors used to color T, F, R are true, false, and red, respectively.

Graph $f(\mathcal{A})$ also contains a ladder each rung of which is a variable x_i and its negation $\overline{x_i}$. Each of these is connected to R, meaning that any valid three-coloring colors one of $x_i, \overline{x_i}$ true and the other false.

Finally, for each clause $C_i = \ell_1 \vee \ell_2 \vee \ell_3$, $f(\mathcal{A})$ contains the gadget G_i consisting of six vertices. G_i has three inputs a_i, b_i, c_i, connected to literals ℓ_1, ℓ_2, ℓ_3, respectively, and it has one output, f_i. See Figure 7.19 where gadget G_1 corresponds to clause $C_1 = \overline{x_1} \vee x_2 \vee \overline{x_3}$.

Observe that the triangle a_1, b_1, d_1 serves as an "or"-gate in that d_1 may be colored true iff at least one of its inputs $\overline{x_1}, x_2$ is colored true. Similarly, output f_1 may be colored true iff at least one of d_1 and the third input, $\overline{x_3}$ is colored true. Since f_i is connected to both F and R, f_i can only be colored true. It follows that a three coloring of the literals can be extended to color G_i iff the corresponding truth

assignment makes C_i true. Thus, $f(\mathcal{A}) \in$ 3-COLOR iff $\mathcal{A} \in$ 3-SAT.

The details of first-order reduction f are easy to fill in. $f(\mathcal{A})$ consists of one triangle, a ladder with n rungs, and n copies of the gadget. The only dependency on the input $\mathcal{A}$ — as opposed to its size — is that there is an edge from literal ℓ to input j of gadget G_i iff ℓ is the j^{th} literal occurring in C_i. □

7.4 The Polynomial-Time Hierarchy

We defined the polynomial-time hierarchy (PH) to be the set of boolean queries accepted in polynomial time by alternating Turing machines making a bounded number of alternations between existential and universal states (Equation (2.35)). The original definition of the polynomial-time hierarchy was via nondeterministic polynomial-time Turing reductions (Definition 2.9).

Definition 7.20 (Polynomial-Time Hierarchy via Oracles) Let $\Sigma_0^p =$ P be level 0 of the polynomial-time hierarchy. Inductively, let

$$\Sigma_{i+1}^p \quad = \quad \left\{ L(M^A) \;\middle|\; M \text{ is an NP oracle TM, } A \in \Sigma_i^p \right\}$$

Equivalently, Σ_{i+1}^p is the set of boolean queries that are nondeterministic polynomial-time Turing reducible to a set from Σ_i^p,

$$\Sigma_{i+1}^p \quad = \quad \left\{ B \;\middle|\; B \leq_{np}^t A, \text{ for some } A \in \Sigma_i^p \right\}$$

Define Π_i^p to be co-Σ_i^p, $\Pi_i^p = \left\{ \overline{A} \;\middle|\; A \in \Sigma_i^p \right\}$. Finally, PH $= \bigcup_{k=1}^{\infty} \Sigma_k^p$. □

The relationship between second-order boolean queries and the levels of the polynomial hierarchy are given by the following:

Theorem 7.21 *Let $S \subseteq$ STRUC$[\tau]$ be a boolean query, and let $k \geq 1$. The following are equivalent,*

1. *$S =$ MOD$[\Phi]$, for some $\Phi \in \Sigma_k^{\text{SO}}$. (Here Σ_k^{SO} is the set of all second-order sentences with second-order quantifier prefix $(\exists \overline{R_1})(\forall \overline{R_2}) \ldots (Q_k \overline{R_k})$.)*
2. *$S = \left\{ x \;\middle|\; (\exists y_1.|y_1| \leq |x|^c)(\forall y_2.|y_2| \leq |x|^c) \cdots (Q_k y_k.|y_k| \leq |x|^c) R(x, \overline{y}) \right\}$ where R is a deterministic polynomial-time predicate on $k+1$ tuples of binary strings and c is a constant.*

3. $S \in \text{ATIME-ALT}[n^{O[1]}, k]$.

4. $S \in \Sigma_k^p$.

Corollary 7.22 *A boolean query is in the polynomial-time hierarchy iff it is second-order expressible,*

$$\text{PH} \quad = \quad \text{SO} .$$

From Theorem 4.10 and Corollary 7.22, we obtain the following descriptive characterization of the P? = NP question: P is equal to NP iff every second-order query — over finite, ordered structures — is expressible as a first-order inductive definition.

Corollary 7.23 *The following conditions are equivalent:*

1. P = NP.

2. *Over finite, ordered structures,* FO(LFP) = SO.

Proof If FO(LFP) = SO, then P $\subseteq$ NP $\subseteq$ PH = P. Conversely, if P = NP, then PH = NP, so FO(LFP) = SO. □

Exercise 7.24 Prove Theorem 7.21. [Hint: By induction on k. The subtle part is relating Σ_k^p to the other conditions. For this, note that an NP machine with an oracle $A \in \Sigma_{k-1}^p$ can guess all the answers to its oracle queries. Then, at the end of its computation, it can check that these answers were all correct. This is a polynomial number of Σ_{k-1}^p and Π_{k-1}^p questions.] □

As seen in the following, the polynomial-time hierarchy is robust enough to finesse the difficulty that occurs in Open Problem 7.12,

Exercise 7.25 Prove that for any k,

$$\text{SO(arity } k) \;=\; \text{PH-TIME}[n^k] \;=\; \text{ATIME-ALT}[n^k, O(1)] \qquad \square$$

Exercise 7.26 Fagin's Theorem (Theorem 7.8) is a generalization of the Spectrum Theorem. Define the *spectrum* of a first-order sentence φ to be the set of cardinalities of the finite models of φ,

$$\text{spec}(\varphi) \;=\; \left\{ n \;\middle|\; n = |\mathcal{A}| \text{ for some } \mathcal{A} \in \text{MOD}[\varphi] \right\} .$$

As an example let φ_{field} be the conjunction of the field axioms, so spec(φ_{field}) is the set of prime powers. An interesting question is whether the set of spectra of first-order sentences is closed under complementation, i.e., if S is a spectrum then is $\overline{S} = \mathbf{Z}^+ - S$ one also? As we now see, this is equivalent to an important open question in complexity theory. The Spectrum Theorem says that a set $S \subseteq \mathbf{Z}^+$ is the spectrum of a first-order sentence iff $S \in$ NTIME[$2^{O[n]}$]. Fagin originally called the finite models of SO$\exists$ sentences "generalized spectra".

1. Write a first-order sentence whose spectrum is the set of even positive integers

2. Modify part 1 to get a first-order sentence whose spectrum is the set of odd positive integers.

3. Prove the Spectrum Theorem.

   [Hint: Show how it follows from Theorem 7.8. Note that a problem $S \subseteq \mathbf{Z}^+$ is assumed to be a set of binary strings coding natural numbers. Thus $S \in$ NTIME[$2^{O[n]}$] iff S coded in unary is in NP.]

4. Show using the Spectrum Theorem that $\overline{\text{spec}(\varphi_{\text{field}})}$ is a spectrum. □

As a corollary to the proof of Theorem 5.2, we obtain the following characterization of PH as a parallel complexity class. Up to this point, we had been assuming for notational simplicity that a CRAM has at most polynomially many processors. However, the class CRAM-PROC$[t(n), p(n)]$ still makes sense for numbers of processors $p(n)$ that are not polynomially bounded.

Corollary 7.27 PH *is equal to the set of boolean queries recognizable by a* CRAM *using exponentially many processors and constant time,*

$$\text{PH} \quad = \quad \bigcup_{k=1}^{\infty} \text{CRAM-PROC}[1, 2^{n^k}]$$

Proof The inclusion SO $\subseteq$ CRAM-PROC$[1, 2^{n^{O[1]}}]$ follows just as in the proof of Lemma 5.4. A processor number is now large enough to give values to all the relational variables as well as to all the first-order variables. Thus, as in Lemma 5.4, the CRAM can evaluate each first or second-order quantifier in three steps.

The inclusion CRAM-PROC$[1, 2^{n^{O[1]}}] \subseteq$ SO follows just as in the proof of Lemma 5.3. The only difference is that we use second-order variables to specify the processor number. □

In fact, Corollary 7.27 can be extended to,

Corollary 7.28 *For all constructible $t(n)$,*

$$\text{SO}[t(n)] \quad = \quad \text{CRAM-PROC}[t(n), 2^{n^{O[1]}}] \ .$$

Observe that Corollary 7.27 suggests that PH is a rather strange complexity class. No one would ever buy exponentially many processors and then use them only for constant time. See Corollary 10.30 for an interesting characterization of the much more robust complexity class PSPACE as exponentially many processors running in polynomial time.

Historical Notes and Suggestions for Further Reading

Theorem 7.8 (Fagin's theorem) was proved in Fagin's thesis, [Fag73, Fag74]. The idea of using choice relation Δ is due to Papadimitriou [Pap94]. The Spectrum Theorem discussed in Exercise 7.26 is due to Jones and Selman [JS74]. See [B82] for a history of the spectrum problem.

Theorem 7.16 was first proved by Lovász and Gács [LG77]. Dahlhaus proved that SAT is NP-complete via quantifier-free, first-order reductions [Da84].

The polynomial-hierarchy (PH) was defined by Stockmeyer [Sto77]. Corollary 7.22 appears there as well. Item 2 of Theorem 7.21 is due to Wrathall [Wra76].

Some of the simple ways to write NP-complete problems as SO$\exists$ formulas, like CLIQUE (Example 7.2) are due to Jose Antonio Medina [MI94].

Theorem 7.11 is due to Lynch, [Lyn82]. Facts 7.14 and 7.15 are due to Grandjean; see [Gra84, Gra85, Gra89] for their proofs. An interesting place to start investigating Open Problem 7.13 is to consider the deterministic time complexity of problem CLIQUE(k) — the set of graphs containing a k-clique — for a fixed k. The best known algorithm is due to Boppana and Halldórsson [BH92].

Exercise 7.25 is from [I83]. Corollary 7.27 is from [I89a].