1 NP Completeness

We will give several version of the first NP-completeness proof. First, we will briefly sketch the proof hat Circuit-SAT is NP-complete. Then we will do the original Cook's proof that SAT is NP-complete, and show how to modify it to obtain Fagin's theorem that existential second-order logic captures NP.

Circuit-SAT = $\{C \mid C$ is a satisfiable Boolean circuit$\}$

Theorem 1. *Circuit-SAT is* NP*-complete.*

Proof. We need to prove that

1. Circuit-SAT is in NP, and
2. Circuit-SAT is NP-hard (i.e., every language $L \in$ NP reduces to Circuit-SAT).

The fact that Circuit-SAT is in NP is easy: Given a circuit C on variables $x_1, \ldots, x_n$, nondeterministically guess an assignment to $x_1, \ldots, x_n$ and verify that this assignment is satisfying; this verification can be done in time polynomial in the size of the circuit. In other words,

$$Circuit - SAT = \{C \mid \exists x, |x| \leq |C|, R'(C, x)\}$$

where $R'(C, x)$ is True iff $C(x) = 1$ (i.e., C on input x evaluates to 1).

Now we prove NP-hardness. Take an arbitrary $L \in$ NP. Say

$$L = \{x \mid \exists y, |y| \leq |x|^c, R(x, y)\}$$

for some constant c and $R \in$ P. Let's suppose that $R(x, y)$ is computable in time $N = (|x| + |y|)^d$, for some constant d.

Consider N steps of computation of the Turing machine deciding R on input x, y. This computation can be pictured as a sequence of N configurations. A configuration at time t is a sequence of symbols $y_1 \ldots y_m$, where each y_j contains the following information: the contents of tape cell j at time t, whether or not tape cell is being scanned by the TM at time t, and if it is, then what is the state of a TM at time t.

The crucial observation is that the computation of a TM has the following "locality property": the value of symbol y_i at time $t + 1$ depends only on the values of symbols y_{i-1}, y_i, y_{i+1} at time t (as well as the transition function of the TM).

We can construct a constant-size circuit *Step* that computes the value of y_i at time $t+1$ from the values of y_{i-1}, y_i, y_{i+1} at time t. Now, we construct a big circuit $C(x, y)$ by replacing each symbol y_i in every configuration at time t by a copy of the circuit *Step* whose inputs are the outputs of the corresponding three copies of *Step* from the previous configuration. We also modify the circuit so that it outputs 1 on x, y iff the last configuration is accepting.

The size of the constructed circuit will be at most $N * N * |Step|$ (N configurations, at most N copies of *Step* in each), which is polynomial in $|x|$.

Our reduction from L to Circuit-SAT is the following: Given x, construct the circuit $C_x(y) = C(x, y)$ as explained above (with x hardwired into C). It is easy to verify that $x \in L$ iff there is y such that $C_x(y) = 1$. So this is a correct reduction. □

$$SAT = \{\phi \mid \phi \text{ is a satisfiable Boolean formula}\}$$

Theorem 2 (Cook-Levin). *SAT is* NP*-complete.*

Proof. One way to prove it would be show $Circuit - SAT \leq_p SAT$. But instead we will go through the proof as it was originally done, without any references to circuits.

Example 3. *Consider a very simple non-deterministic Turing machine with 3 states q_0, q_a, q_r and the following transition table:*

$$(q_0, 0) \to (q_0, \_, R) \quad (q_0, 1) \to (q_0, \_, R) \quad (q_0, 1) \to (q_a, \_, R) \quad (q_0, \_) \to (q_r, \_, R)$$

That is, TM accepts iff there is a symbol 1 in the input string. The non-determinism comes from the fact that any of the 1s would lead to an accept state in some computation.

To talk about Turing machine computations we will define a notion of *tableau*.

Definition 4. *A* tableau *for a Turing machine M on input w running in time n^k is a $n^k \times n^k$ table whose rows are configurations if a branch of the computation of M on input w.*

That is, any possible path in the computation tree of M on w can be encoded as a tableau. To simplify our formulae, we will assume that the first and the last column of the tableau contains a special symbol #.

The TM from example 3 has the following 3 tableau on string 0110:

T_1=

#	q_0	0	1	1	0	_	#
#	_	q_0	1	1	0	_	#
#	_	_	q_a	1	0	_	#

T_2=

#	q_0	0	1	1	0	_	#
#	_	q_0	1	1	0	_	#
#	_	_	q_0	1	0	_	#
#	_	_	_	q_a	0	_	#

T_3=

#	q_0	0	1	1	0	_	#
#	_	q_0	1	1	0	_	#
#	_	_	q_0	1	0	_	#
#	_	_	_	q_0	0	_	#
#	_	_	_	_	q_0	_	#
#	_	_	_	_	_	q_r	#

The first two correspond to the accepting branches, the last one to the rejecting. So a NTM accepts a string iff it has an accepting tableau on that string.

We will show how to encode a run of a NTM M on a string w by a propositional formula, in a sense that the formula will have a satisfying assignment iff there exists an accepting tableau of M on w. Our formula will have propositional variables $x_{i,j,s}$ where i is the name

of the row, j is the column and s is a symbol of $C = Q \cup \Gamma \cup \{\#\}$. The intention is that a variable $x_{i,j,s}$ is set to True iff symbol s is in the cell i, j.

The final formula will consist of a conjunction of 4 parts, call them ϕ_{cell}, ϕ_{start}, ϕ_{accept} and ϕ_{move}.

1. The first formula says that there is exactly one symbol in every cell:

$$\phi_{cell} = \bigwedge_{1 \le i,j < n^k} \left(\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right)$$

2. The second formula forces the values of starting configuration by taking a conjunction of the corresponding variables. In the example 3, on string 0110, the formula will look like this:

$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,0} \wedge x_{1,4,1} \wedge x_{1,5,1} \wedge x_{1,6,0} \wedge x_{1,7,\_} \wedge x_{1,8,\#}$$

3. The third formula says that the tableau corresponds to an accepting computation:

$$\phi_{accept} = \bigvee_{1 \le i,j < n^k} x_{i,j,q_a}$$

4. The last formula ϕ_{move} is the most complicated part. It has to encode the fact that every line in the tableau came from the previous line by a valid transition of M. This formula encodes, for every symbol $x_{i,j,s}$, all possible transitions that can lead to cell $[i, j]$ having a value s. The crucial property that allows us to encode that is that every symbol in the configuration can be determined by looking at the three symbols in the previous line: symbol above and symbols to the left and right of it. That is, to determine the value of $x_{i+1,j,s}$ it is sufficient to know, for all symbols, values of $x_{i,j-1,?}$, $x_{i,j,?}$ and $x_{i,j+1,?}$.Here, ? runs through all possible symbols. If there is no head position symbol among these three, then the value of $x_{i,j,s} = x_{i+1,j,s}$. In the example 3 there are two possibilities for the content of cell [3,4], either q_0 or q_a. This is encoded by the following formula

$$x_{2,3,q_0} \wedge x_{2,4,1} \wedge x_{2,5,1} \rightarrow (x_{3,4,q_0} \vee x_{3,4,q_a})$$

Notice here that the only time there is a $\vee$ on the right side of the implication is when there is a non-deterministic choice of a transition.

Now, suppose that the formula has a satisfying assignment. That assignment encodes precisely a tableau of an accepting computation of M. For the other direction, take an accepting computation of M and set exactly those $x_{i,j,s}$ corresponding to the symbols in the accepting tableau.

Finally, to show that our reduction is polynomial, we will show that the resulting formula is of polynomial size. The number of variables is $N = n^k \times n^k \times |Q \cup \Gamma \cup \{\#\}|$. In ϕ_{cell} there are $O(N)$ variables, same for ϕ_{accept}. Only n^k variables are in ϕ_{start}. Finally, ϕ_{move} has $O(N \times |\delta|)$ clauses. Therefore, the size of the formula is polynomial. □

A slight modification of this proof Fagin to adapt Cook's theorem to the finite model theory setting. In that setting, an input is a structure, and the question is whether a given formula (hardcoded) is true on this structure (this is called the *model checking* problem). A structure is encoded as a number which is the size of the structure, together with relational variables encoded as tables (k-dimensional, in general). An example of a structure is a graph: it is encoded by the number of vertices followed by the edge relation represented by its adjacency matrix.

Definition 5. *In finite model theory, a logic* $\mathcal{L}$ captures *a complexity class* C *on a class of structures* K *if 1) the model checking problem for these formulae on this class of structures can be done within the class, and 2) for a collection of structures from the class (closed under isomorphism and decidable in C) there is a formula in* $\mathcal{L}$ *true on all and only structures from that class.*

In simpler words, for each property (such as 3-colourability) of structures (such as graphs) decidable in C there is a formula from $\mathcal{L}$ true on all and only structures that have this property.

Definition 6. *Second-order existential logic (*$\exists SO$*) consists of formulae of the form* $\exists P_1 \dots P_k \phi(P_1, \dots, P_k)$*, where* ϕ *is a first-order formula that can have other free relational variables, and* P_i *are sets of* d_i*-tuples, where* d_i *it the arity of* P_i*, treated by* ϕ *as relational variables.*

Theorem 7 (Fagin). $\exists SO$ *logic captures* NP*.*

Proof. The bulk of Fagins proof was showing how to encode an order relation in $\exists SO$ logic. We will skip this here and only show how to modify Cook's theorem proof for the finite model theory setting, assume we have an order relation in the language. The order relation is necessary for saying things like "subsequent configuration", "next cell" and such.

Let n be the size of the structure; the length of the input and thus the running time of the machine M is polynomial in n. For each symbol $s \in Q \cup \Gamma \cup \{\#\}$ define a second-order variable C_s of arity 2k. Then $C_s(i, j)$ is equivalent to $x_{i,j,s}$. Now the new formulae ϕ_{cell}, ϕ_{start}, ϕ_{accept} and ϕ_{move} will be very similar to the old ones, just replacing $x_{i,j,s}$ with $C_s(i, j)$ and placing quantifiers instead of big $\vee$ and $\wedge$. For example, new $\phi_{accept} = \exists i, j \; C_{q_a}(i, j)$. We will leave it as an exercise to show that there exists an accepting tableau of M on w iff there exist values for C_s over a structure encoding the input string w as a unary relational variable W. □

Open Question: Is it possible to capture P without an order relation in the language?