Complexity Theory

Nondeterministic Polynomial Time

Daniel Borchmann, Markus Krötzsch

Computational Logic

2015-11-11

The Class NP

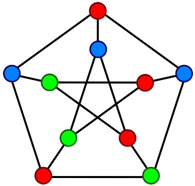
Beyond PTIME

- We have seen that the class PTIME provides a useful model of "tractable" problems
- This includes 2-SAT and 2-COLOURABILITY
- But what about 3-SAT and 3-COLOURABILITY?
- No polynomial time algorithms for these problems are known
- On the other hand . . .

Verifying Solutions

For many seemingly difficult problems, it is easy to verify the correctness of a "solution" if given.

p	q	r	$p \to q$
f	f	f	w
f	w	f	w
w	f	f	f
w	w	f	w
f	f	w	w
f	w	w	w
w	f	w	f
w	w	w	w



5		3				7		
			8					6
	7			6			4	
	4		1					
7		8		5		3		9
					9		6	
	5			1			7	
6					4			
		2				5		3

- SATISFIABILITY – a satisfying assignment
- k-COLOURABILITY – a k-colouring
- SUDOKU – a completed puzzle

Verifiers

Definition 7.1

- A Turing machine $\mathcal{M}$ which halts on all inputs is called a verifier for a language $\mathcal{L}$ if

$$\mathcal{L} = \{w \mid \mathcal{M} \text{ accepts } (w\#c) \text{ for some string } c\}$$

The string c is called a certificate (or witness) for w.

- $\mathcal{M}$ is a polynomial-time verifier for $\mathcal{L}$ if $\mathcal{M}$ is polynomially time bounded and

$$\mathcal{L} = \{w \mid \mathcal{M} \text{ accepts } (w\#c) \text{ for some string } c \text{ with } |c| \leq p(|w|)\}$$

for some fixed polynomial p.

Notation: $\#$ is a new separator symbol not used in words or certificates.

The Class NP

NP: "The class of dashed hopes and idle dreams."[1]

More formally:
the class of problems for which a possible solution can be verified in P

Definition 7.2

The class of languages that have polynomial-time verifiers is called NP.

In other words: NP is the class of all languages $\mathcal{L}$ such that:

- for every $w \in \mathcal{L}$, there is a certificate $c_w \in \Sigma^*$, where
- the length of c_w is polynomial in the length of w, and
- the language $\{(w\#c_w) \mid w \in \mathcal{L}\}$ is in P

[1] https://complexityzoo.uwaterloo.ca/Complexity_Zoo:N#np

More Examples of Problems in NP

HAMILTONIAN PATH

Input: An undirected graph G

Problem: Is there a path in G that contains each vertex exactly once?

k-CLIQUE

Input: An undirected graph G

Problem: Does G contain a fully connected graph (clique) with k vertices?

More Examples of Problems in NP

Subset Sum

Input: A collection of positive integers $S = \{a_1, \ldots, a_k\}$ and a target integer t.

Problem: Is there a subset $T \subseteq S$ such that $\sum_{a_i \in T} a_i = t$?

Travelling Salesperson

Input: A weighted graph G and a target number t.

Problem: Is there a simple path in G with weight $\leq t$?

Complements of NP are often not known to be in NP

NO HAMILTONIAN PATH

Input: An undirected graph G

Problem: Is there no path in G that contains each vertex exactly once?

Whereas it is easy to certify that a graph has a Hamiltonian path, there does not seem to be a polynomial certificate that it has not.

But we may just not be clever enough to find one.

More Examples

COMPOSITE (NON-PRIME) NUMBER

Input: A positive integer $n > 1$

Problem: Are there integers $u, v > 1$ such that $u \cdot v = n$?

PRIME NUMBER

Input: A positive integer $n > 1$

Problem: Is n a prime number?

Surprisingly: both are in NP (see Wikipedia "Primality certificate")

In fact: COMPOSITE NUMBER (and thus PRIME NUMBER) was shown to be in P

$\mathbb{N}$ is for Nondeterministic

Reprise: Nondeterministic Turing Machines

A nondeterministic Turing Machine (NTM) $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}})$ consists of

- a finite set Q of *states*,
- an *input alphabet* Σ not containing $\square$,
- a *tape alphabet* Γ such that $\Gamma \supseteq \Sigma \cup \{\square\}$.
- a *transition function* $\delta\colon Q \times \Gamma \to \mathfrak{P}(Q \times \Gamma \times \{L, R\})$
- an *initial state* $q_0 \in Q$,
- an *accepting state* $q_{\text{accept}} \in Q$.

Note

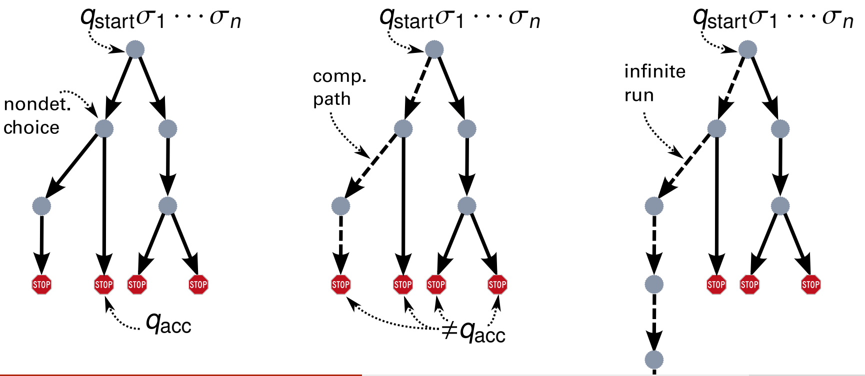
An NTM can halt in any state if there are no options to continue
$\leadsto$ no need for a special rejecting state

Reprise: Runs of NTMs

An (N)TM configuration can be written as a word uqv where $q \in Q$ is a state and $uv \in \Gamma^*$ is the current tape contents.

NTMs produce configuration trees that contain all possible runs:

accept: reject: reject (not halting):

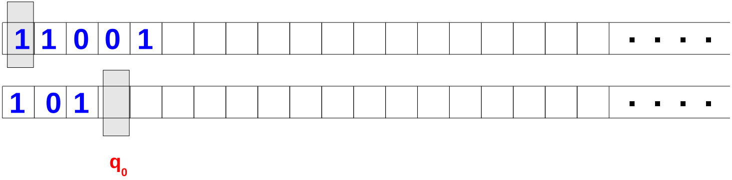


Example: Multi-Tape NTM

Consider the NTM $\mathcal{M} = \big(Q, \{0,1\}, \{0,1,\square\}, q_0, \Delta, q_{accept}\big)$ where

$$\Delta = \left\{ \begin{array}{l} \big(q_0,\ \binom{\_}{\_}, q_0, \binom{\_}{0}, \binom{N}{R}\big) \\ \big(q_0,\ \binom{\_}{\_}, q_0, \binom{\_}{1}, \binom{N}{R}\big) \\ \big(q_0,\ \binom{\_}{\_}, q_{check}, \binom{\_}{\_}, \binom{N}{N}\big) \\ \ldots \\ \text{transition rules for } \mathcal{M}_{check} \end{array} \right\}$$

and where $\mathcal{M}_{check}$ is a deterministic TM deciding whether number on second tape is > 1 and divides the number on the first.



Time and Space Bounded NTMs

Q: Which of the nondeterministic runs do time/space bounds apply to?
A: To all of them!

Definition 7.3

Let $\mathcal{M}$ be a nondeterministic Turing machine and let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

- $\mathcal{M}$ is f-time bounded if it halts on every input $w \in \Sigma^*$ and on every computation path after $\leq f(|w|)$ steps.
- $\mathcal{M}$ is f-space bounded if it halts on every input $w \in \Sigma^*$ and on every computation path using $\leq f(|w|)$ cells on its tapes.

  (Here we typically assume that Turing machines have a separate input tape that we do not count in measuring space complexity.)

Nondeterministic Complexity Classes

Definition 7.4

Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

- $\textsc{NTime}(f(n))$ is the class of all languages $\mathcal{L}$ for which there is an $O(f(n))$-time bounded nondeterministic Turing machine deciding $\mathcal{L}$, for some $k \geq 1$.
- $\textsc{NSpace}(f(n))$ is the class of all languages $\mathcal{L}$ for which there is an $O(f(n))$-space bounded nondeterministic Turing machine deciding $\mathcal{L}$.

All Complexity Classes Have a Nondeterministic Variant

$$\text{NPTime} = \bigcup_{d\geq 1} \text{NTime}(n^d)$$ nondet. polynomial time

$$\text{NExp} = \text{NExpTime} = \bigcup_{d\geq 1} \text{NTime}(2^{n^d})$$ nondet. exponential time

$$\text{N2Exp} = \text{N2ExpTime} = \bigcup_{d\geq 1} \text{NTime}(2^{2^{n^d}})$$ nond. double-exponential time

$$\text{NL} = \text{NLogSpace} = \text{NSpace}(\log n)$$ nondet. logarithmic space

$$\text{NPSpace} = \bigcup_{d\geq 1} \text{NSpace}(n^d)$$ nondet. polynomial space

$$\text{NExpSpace} = \bigcup_{d\geq 1} \text{NSpace}(2^{n^d})$$ nondet. exponential space

Equivalence of NP and NPTIME

Theorem 7.5

NP = NPTIME.

Proof.

- Suppose $\mathcal{L} \in \mathrm{NPTIME}$.
- Then there is an NTM $\mathcal{M}$ such that

$$w \in \mathcal{L} \iff \text{there is an accepting run of } \mathcal{M} \text{ of length } O(n^d)$$

  for some d.
- This path can be used as a certificate for w.
- A DTM can check in polynomial time that a candidate certificate is a valid accepting run.

Therefore NP $\supseteq$ NPTIME.

Equivalence of NP and $\mathrm{NPT{\small IME}}$

Proof of the converse direction:

- Assume $\mathcal{L}$ has a polynomial-time verifier $\mathcal{M}$ with certificates of length at most $p(n)$ for a polynomial p.
- Then we can construct an NTM $\mathcal{M}^*$ deciding $\mathcal{L}$ as follows:
  (1) $\mathcal{M}^*$ guesses a string of length $p(n)$
  (2) $\mathcal{M}^*$ checks in deterministic polynomial time if this is a certificate.

Therefore $\mathrm{NP} \subseteq \mathrm{NPT{\small IME}}$. □

NP and coNP

Note: Definition of NP is not symmetric

- there does not seem to be any polynomial certificate for Sudoku **un**solvability or propositional logic **un**satisfiability . . .
- converse of an NP problem is coNP
- similar for NExpTime and N2ExpTime

Other complexity classes are symmetric:

- Deterministic classes (coP = P etc.)
- Space classes mentioned above (esp. coNL = NL)

Deterministic vs. Nondeterminsitic Time

Theorem 7.6

$\mathrm{P} \subseteq \mathrm{NP}$, *and also* $\mathrm{P} \subseteq \mathrm{coNP}$.

(Clear since DTMs are a special case of NTMs)

It is not known to date if the converse is true or not.

- Put differently: “If it is easy to check a candidate solution to a problem, is it also easy to find one?”
- Unresolved since over 30 years of effort
- One of the major problems in computer science and math of our time
- 1,000,000 USD prize for resolving it (“Millenium Problem”); might not be much money at the time it is actually solved

Status of P vs. NP

- It is often said: "Most experts think $\mathrm{P} \neq \mathrm{NP}$"
  - Main argument: "If $\mathrm{NP} = \mathrm{P}$, someone ought to have found some polynomial algorithm by now."
  - "This is, in my opinion, a very weak argument. The space of algorithms is very large and we are only at the beginning of its exploration." (Moshe Vardi, 2002)
- Results of a poll among 100 experts [Gasarch 2002]:
  - $\mathrm{P} \neq \mathrm{NP}$: 61
  - $\mathrm{P} = \mathrm{NP}$: 9
  - No comment: 22
  - Other: independent (4), not independent (3), it depends (1)
- Over 100 "proofs" show $\mathrm{P} = \mathrm{NP}$ to be true/false/both/neither: `https://www.win.tue.nl/~gwoegi/P-versus-NP.htm`
- Many solutions conceivable, e.g., $\mathrm{P} = \mathrm{NP}$ could be shown with a non-constructive proof

A Simple Proof for $\mathrm{P} = \mathrm{NP}$

Clearly	$\mathcal{L} \in \mathrm{P}$	implies	$\mathcal{L} \in \mathrm{NP}$
therefore	$\mathcal{L} \notin \mathrm{NP}$	implies	$\mathcal{L} \notin \mathrm{P}$
hence	$\mathcal{L} \in \mathrm{coNP}$	implies	$\mathcal{L} \in \mathrm{coP}$
that is	$\mathrm{coNP} \subseteq \mathrm{coP}$		
using $\mathrm{coP} = \mathrm{P}$	$\mathrm{coNP} \subseteq \mathrm{P}$		
and hence	$\mathrm{NP} \subseteq \mathrm{P}$		
so by $\mathrm{P} \subseteq \mathrm{NP}$	$\mathrm{NP} = \mathrm{P}$		

q.e.d.?