TECHNISCHE
UNIVERSITÄT
DRESDEN

# COMPLEXITY THEORY

## Lecture 6: Nondeterministic Polynomial Time

**Markus Krötzsch**

**Knowledge-Based Systems**

TU Dresden, 7th Nov 2017

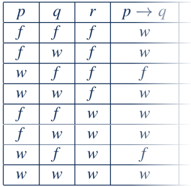
# The Class NP

# Beyond PTime

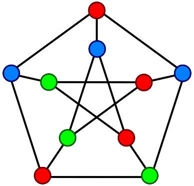
- We have seen that the class PTime provides a useful model of “tractable” problems
- This includes 2-Sat and 2-Colourability
- But what about 3-Sat and 3-Colourability?
- No polynomial time algorithms for these problems are known
- On the other hand . . .

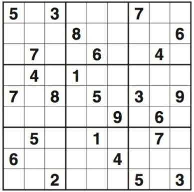
# Verifying Solutions

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

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

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

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

# Verifiers

**Definition 6.1:** A Turing machine $\mathcal{M}$ which halts on all inputs is called a verifier for a language **L** if

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

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

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

**Definition 6.2:** A Turing machine $\mathcal{M}$ is a polynomial-time verifier for **L** if $\mathcal{M}$ is polynomially time bounded and

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

for some fixed polynomial $p$.

# The Class NP

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

---

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

# 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 6.3:** The class of languages that have polynomial-time verifiers is called NP.

---

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

# 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 6.3:** The class of languages that have polynomial-time verifiers is called NP.

In other words: NP is the class of all languages **L** such that:

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

# 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”)

# 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

# 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** $\Sigma$ not containing $\square$,
- a **tape alphabet** $\Gamma$ such that $\Gamma \supseteq \Sigma \cup \{\square\}$.
- a **transition function** $\delta\colon Q \times \Gamma \to 2^{Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}}$
- an **initial state** $q_0 \in Q$,
- an **accepting state** $q_{\text{accept}} \in Q$.

## Note

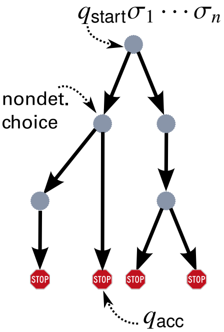
An NTM can halt in any state if there are no options to continue
$\rightsquigarrow$ no need for a special rejecting state

# Reprise: Runs of NTMs

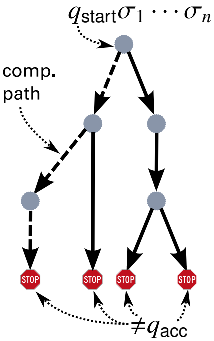
An (N)TM configuration can be written as a word $uqv$ where $q \in Q$ is a state and $uv \in \Gamma^*$ is the current tape contents.

NTMs produce configuration trees that contain all possible runs:

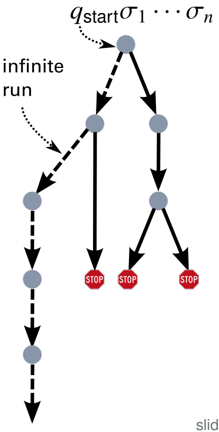
accept:



reject:



reject (not halting):

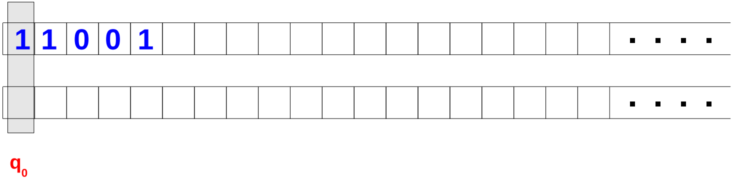


# Example: Multi-Tape NTM

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

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

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

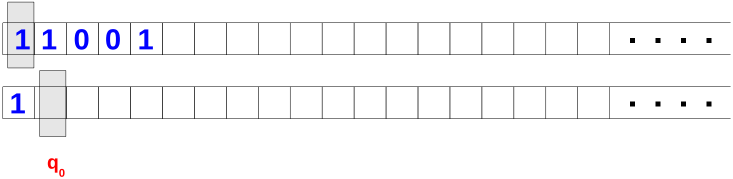


# Example: Multi-Tape NTM

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

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

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

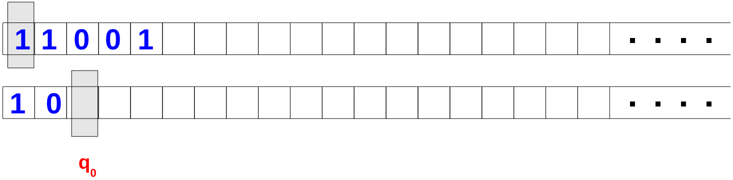


# Example: Multi-Tape NTM

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

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

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

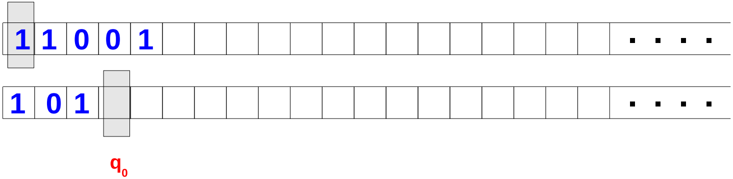


# Example: Multi-Tape NTM

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

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

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



# Example: Multi-Tape NTM

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

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

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

The machine $\mathcal{M}$ decides if the input is a composite number:

- guess a number on the second tape
- check if it divides the number on the first tape
- accept if a suitable number exists

# Time and Space Bounded NTMs

Q: Which of the nondeterministic runs do time/space bounds apply to?

# Time and Space Bounded NTMs

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

**Definition 6.4:** Let $\mathcal{M}$ be a nondeterministic Turing machine and let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

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

(2) $\mathcal{M}$ 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 6.5:** Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

(1) NTime($f(n)$) is the class of all languages **L** for which there is an $O(f(n))$-time bounded nondeterministic Turing machine deciding **L**.

(2) NSpace($f(n)$) is the class of all languages **L** for which there is an $O(f(n))$-space bounded nondeterministic Turing machine deciding **L**.

# All Complexity Classes Have a Nondeterministic Variant

| | |
|---|---|
| $\mathsf{NPTime} = \bigcup_{d\geq 1} \mathsf{NTime}(n^d)$ | nondet. polynomial time |
| $\mathsf{NExp} = \mathsf{NExpTime} = \bigcup_{d\geq 1} \mathsf{NTime}(2^{n^d})$ | nondet. exponential time |
| $\mathsf{N2Exp} = \mathsf{N2ExpTime} = \bigcup_{d\geq 1} \mathsf{NTime}(2^{2^{n^d}})$ | nond. double-exponential time |
| | |
| $\mathsf{NL} = \mathsf{NLogSpace} = \mathsf{NSpace}(\log n)$ | nondet. logarithmic space |
| $\mathsf{NPSpace} = \bigcup_{d\geq 1} \mathsf{NSpace}(n^d)$ | nondet. polynomial space |
| $\mathsf{NExpSpace} = \bigcup_{d\geq 1} \mathsf{NSpace}(2^{n^d})$ | nondet. exponential space |

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We first show NP $\supseteq$ NPTime:

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We first show NP $\supseteq$ NPTime:

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

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

for some $d$.

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We first show NP $\supseteq$ NPTime:

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

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

  for some $d$.
- This path can be used as a certificate for $w$.

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We first show NP $\supseteq$ NPTime:

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

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

  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 NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We now show NP $\subseteq$ NPTime:

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We now show NP $\subseteq$ NPTime:

- Assume **L** has a polynomial-time verifier $\mathcal{M}$ with certificates of length at most $p(n)$ for a polynomial $p$.

# Equivalence of NP and NPTime

**Theorem 6.6:** NP = NPTime.

**Proof:** We now show NP $\subseteq$ NPTime:

- Assume **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 **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 NP $\subseteq$ NPTime. ☐

# NP and coNP

Note: the 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 6.7:** P $\subseteq$ NP, and also P $\subseteq$ 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?”
- Exaggerated: “Can creativity be automated?” (Wigderson, 2006)
- Unresolved since over 35 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

Many people believe that P $\neq$ NP

- Main argument: "If NP = P, someone ought to have found some polynomial algorithm for an NP-complete problem 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)
- Another source of intuition: Humans find it hard to solve NP-problems, and hard to imagine how to make them simpler – possibly "human chauvinistic bravado" (Zeilenberger, 2006)
- There are better arguments, but none more than an intuition

# Status of P vs. NP

Many outcomes conceivable:

# Status of P vs. NP

Many outcomes conceivable:

- P = NP could be shown with a non-constructive proof

# Status of P vs. NP

Many outcomes conceivable:

- P = NP could be shown with a non-constructive proof
- The question might be independent of standard mathematics (ZFC)

# Status of P vs. NP

Many outcomes conceivable:

- $\mathsf{P} = \mathsf{NP}$ could be shown with a non-constructive proof
- The question might be independent of standard mathematics (ZFC)
- Even if $\mathsf{NP} \neq \mathsf{P}$, it is unclear if NP problems require exponential time in a strict sense – many super-polynomial functions exist . . .

# Status of P vs. NP

Many outcomes conceivable:

- $\mathsf{P} = \mathsf{NP}$ could be shown with a non-constructive proof
- The question might be independent of standard mathematics (ZFC)
- Even if $\mathsf{NP} \neq \mathsf{P}$, it is unclear if NP problems require exponential time in a strict sense – many super-polynomial functions exist . . .
- The problem might never be solved

# Status of P vs. NP

Current status in research:

- Results of a poll among 152 experts [Gasarch 2012]:
  - $P \neq NP$: 126 (83%)
  - $P = NP$: 12 (9%)
  - Don't know or don't care: 7 (4%)
  - Independent: 5 (3%)
  - And 1 person (0.6%) answered: "I don't **want** it to be equal."
- Experts have guessed wrongly in other major questions before
- Over 100 "proofs" show $P = NP$ to be true/false/both/neither: `https://www.win.tue.nl/~gwoegi/P-versus-NP.htm`

# A Simple Proof for P = NP

Clearly $\mathbf{L} \in \mathrm{P}$ implies $\mathbf{L} \in \mathrm{NP}$

therefore $\mathbf{L} \notin \mathrm{NP}$ implies $\mathbf{L} \notin \mathrm{P}$

hence $\mathbf{L} \in \mathrm{coNP}$ implies $\mathbf{L} \in \mathrm{coP}$

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.

# A Simple Proof for P = NP

Clearly $\mathbf{L} \in \mathrm{P}$ implies $\mathbf{L} \in \mathrm{NP}$
therefore $\mathbf{L} \notin \mathrm{NP}$ implies $\mathbf{L} \notin \mathrm{P}$
hence $\mathbf{L} \in \mathrm{coNP}$ implies $\mathbf{L} \in \mathrm{coP}$
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.?

# Summary and Outlook

NP can be defined using polynomial-time verifiers or polynomial-time nondeterministic Turing machines

Many problems are easily seen to be in NP

NTM acceptance is not symmetric: coNP as complement class, which is assumed to be unequal to NP

**What's next?**

- NP hardness and completeness
- More examples of problems
- Space complexities