### **Hilbert's Tenth Problem**

#### Hilbert, International Congress of Mathematicians, Paris, 1900

Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.

**Theorem** (Matiyasevich 1970) Hilbert's tenth problem is undecidable.

### Non-deterministic Turing machines

• Next move relation:

$$\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$$

- L(M) = set of words  $w \in \Sigma^*$  for which *there exists* a sequence of moves accepting w.
- **Proposition.** If *L* is accepted by a non-deterministic Turing machine *M*<sub>1</sub>, then *L* is accepted by some deterministic machine *M*<sub>2</sub>.

#### **Time complexity**

- *M* a (deterministic) Turing machine that halts on all inputs.
- Time complexity function  $T_M : \mathbb{N} \to \mathbb{N}$

 $T_M(n) = \max\{m \mid \exists w \in \Sigma^*, |w| = n \text{ such that the computation} \\ \text{of } M \text{ on } w \text{ takes } m \text{ moves} \}$ 

(assume numbers are coded in binary format)

- A Turing machine is *polynomial* if there exists a polynomial p(n) with  $T_M(n) \le p(n)$ , for all  $n \in \mathbb{N}$ .
- The *complexity class P* is the class of languages decided by a polynomial Turing machine.

#### Time complexity of non-deterministic Turing machines

- M non-deterministic Turing machine
- The running time of *M* on  $w \in \Sigma^*$  is
  - the length of a shortest sequence of moves accepting w if  $w \in L(M)$
  - 1, if  $w \notin L(M)$
- $T_M(n) = \max\{m \mid \exists w \in \Sigma^*, |w| = n \text{ such that the running time of } M \text{ on } w \text{ is } m\}$
- The *complexity class NP* is the class of languages accepted by a polynomial non-deterministic Turing machine.

# **Deciding languages in NP**

**Theorem.** If  $L \in NP$ , then there exists a deterministic Turing machine *M* and a polynomial p(n) such that

- M decides L and
- $T_M(n) \leq 2^{p(n)}$ , for all  $n \in \mathbb{N}$ .

*Proof:* Suppose *L* is accepted by a non-deterministic machine  $M_{nd}$  whose running time is bounded by the polynomial q(n).

To decide whether  $w \in L$ , the machine M will

- 1. determine the length *n* of *w* and compute q(n).
- 2. simulate all executions of  $M_{nd}$  of length at most q(n). If the maximum number of choices of  $M_{nd}$  in one step is r, there are at most  $r^{q(n)}$  such executions.
- 3. if one of the simulated executions accepts w, then M accepts w, otherwise M rejects w.

The overall complexity is bounded by  $r^{q(n)} \cdot q'(n) = O(2^{p(n)})$ , for some polynomial p(n).

### An alternative characterization of NP

• **Proposition.**  $L \in NP$  if and only if there exists  $L' \in P$  and a polynomial p(n) such that for all  $w \in \Sigma^*$ :

$$w \in L \iff \exists v \in (\Sigma')^* : |v| \le p(|w|) \text{ and } (w, v) \in L'$$

- Informally, a problem is in NP if it can be solved non-deterministically in the following way:
  - 1. guess a solution/certificate v of polynomial length,
  - 2. check in polynomial time whether v has the desired property.

## **Propositional satisfiability**

• Satisfiability problem SAT

Instance: A formula F in propositional logic with variables  $x_1, ..., x_n$ .

Question: Is *F* satisfiable, i.e., does there exist an assignment  $I : \{x_1, ..., x_n\} \rightarrow \{0, 1\}$  making the formula true ?

- Trying all possible assignments would require exponential time.
- Guessing an assignment *I* and checking whether it satisfies *F* can be done in (non-deterministic) polynomial time. Thus:
- Proposition. SAT is in NP.