A Crash Course on Complexity Theory

we are going to recall some fundamental notions from complexity theory that will be heavily used in the context of this course – details can be found in the standard textbooks

Deterministic Turing Machine (DTM)

$$M = (S, \Lambda, \Gamma, \delta, s_0, s_{accept}, s_{reject})$$

- S is the set of states
- Λ is the input alphabet, not containing the blank symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Lambda \subseteq \Gamma$
- $\delta : S \times \Gamma \rightarrow S \times \Gamma \times \{L,R\}$
- s_0 is the initial state
- s_{accept} is the accept state
- s_{reject} is the reject state, where $s_{accept} \neq s_{reject}$

Deterministic Turing Machine (DTM)

$$M = (S, \Lambda, \Gamma, \delta, s_0, s_{accept}, s_{reject})$$

$\delta(s_1, \alpha) = (s_2, \beta, R)$

IF at some time instant τ the machine is in sate s_1, the cursor points to cell κ, and this cell contains α

THEN at instant τ+1 the machine is in state s_2, cell κ contains β, and the cursor points to cell κ+1

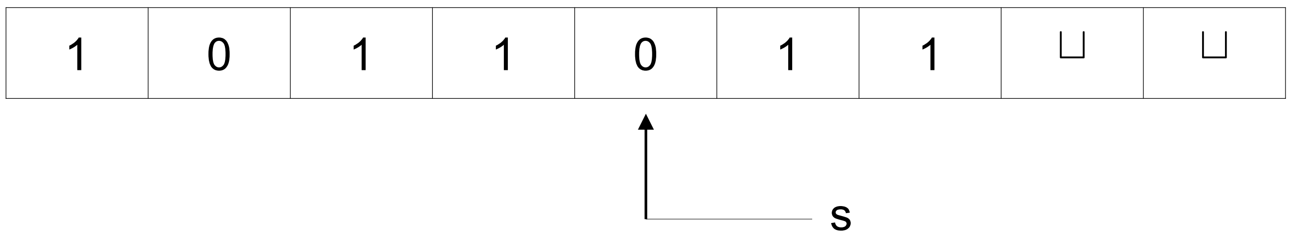
Nondeterministic Turing Machine (NTM)

$$M = (S, \Lambda, \Gamma, \delta, s_0, s_{accept}, s_{reject})$$

- S is the set of states
- Λ is the input alphabet, not containing the blank symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Lambda \subseteq \Gamma$
- **$\delta : S \times \Gamma \rightarrow 2^{S \times \Gamma \times \{L,R\}}$**
- s_0 is the initial state
- s_{accept} is the accept state
- s_{reject} is the reject state, where $s_{accept} \neq s_{reject}$

Turing Machine Configuration

A perfect description of the machine at a certain point in the computation

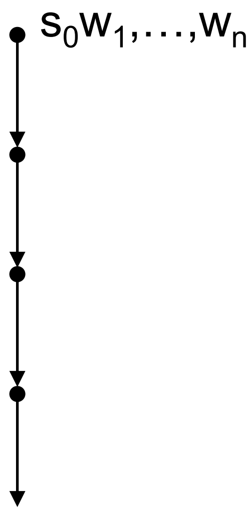


is represented as a string: **1011s011**

- Initial configuration on input $w_1,\ldots,w_n$ – $s_0 w_1,\ldots,w_n$
- Accepting configuration – $u_1,\ldots,u_k s_{accept} u_{k+1},\ldots,u_{k+m}$
- Rejecting configuration – $u_1,\ldots,u_k s_{reject} u_{k+1},\ldots,u_{k+m}$

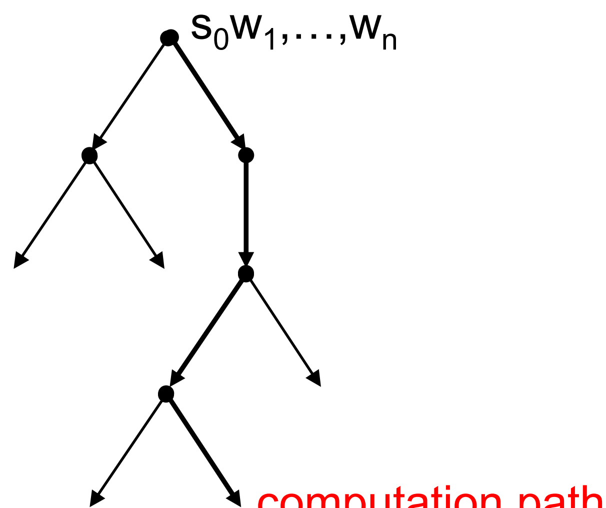
Turing Machine Computation

Deterministic



the next configuration is unique

Nondeterministic



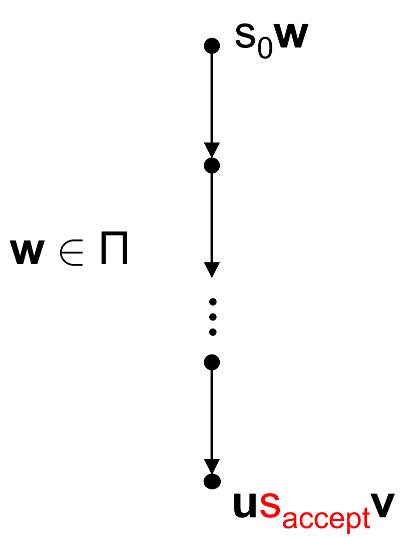
computation tree

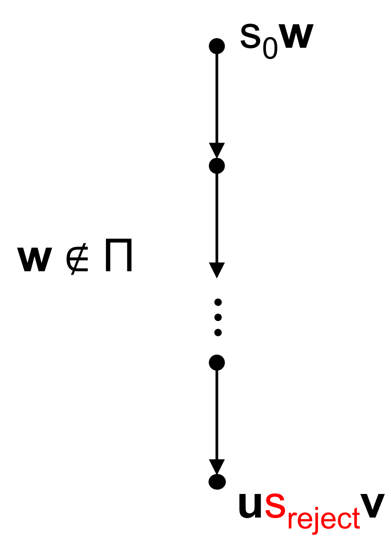
Deciding a Problem

(recall that an instance of a decision problem Π is encoded as a word over a certain alphabet Λ – thus, Π is a set of words over Λ, i.e., $\Pi \subseteq \Lambda^*$)

A DTM $M = (S, \Lambda, \Gamma, \delta, s_0, s_{accept}, s_{reject})$ decides a problem Π if, for every $\mathbf{w} \in \Lambda^*$:

- M on input **w** halts in s_{accept} if $\mathbf{w} \in \Pi$
- M on input **w** halts in s_{reject} if $\mathbf{w} \notin \Pi$

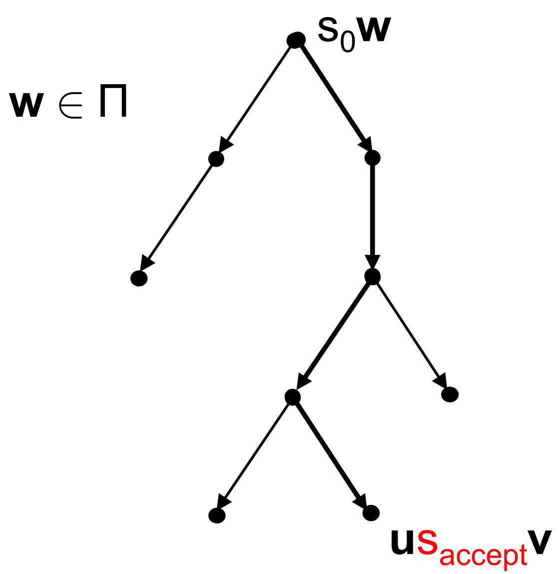


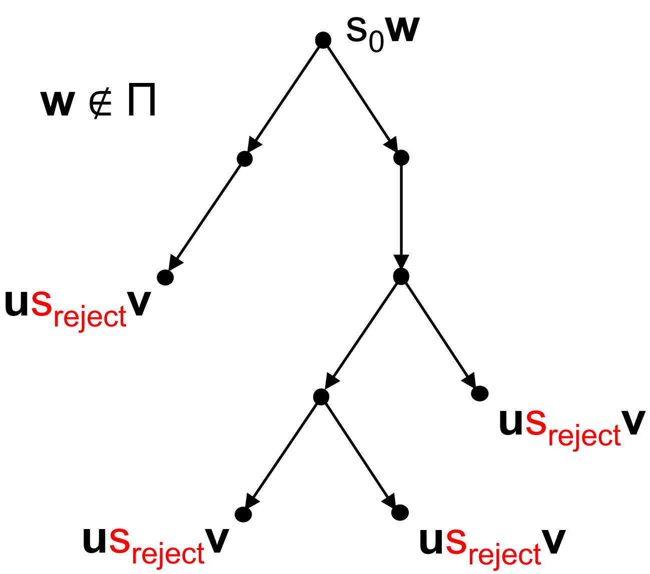


Deciding a Problem

A NTM $M = (S, \Lambda, \Gamma, \delta, s_0, s_{accept}, s_{reject})$ decides a problem Π if, for every $\mathbf{w} \in \Lambda^*$:

- The computation tree of *M* on input **w** is finite
- There exists at least one accepting computation path if $\mathbf{w} \in \Pi$
- There is no accepting computation path if $\mathbf{w} \notin \Pi$





Complexity Classes

Consider a function $f : N \rightarrow N$

$$\mathrm{TIME}(f(n)) = \{\Pi \mid \Pi \text{ is decided by some DTM in time } O(f(n))\}$$

$$\mathrm{NTIME}(f(n)) = \{\Pi \mid \Pi \text{ is decided by some NTM in time } O(f(n))\}$$

$$\mathrm{SPACE}(f(n)) = \{\Pi \mid \Pi \text{ is decided by some DTM using space } O(f(n))\}$$

$$\mathrm{NSPACE}(f(n)) = \{\Pi \mid \Pi \text{ is decided by some NTM using space } O(f(n))\}$$

Complexity Classes

- We can now recall the standard time and space complexity classes:

$$\text{PTIME} = \bigcup_{k>0} \text{TIME}(n^k)$$

$$\text{NP} = \bigcup_{k>0} \text{NTIME}(n^k)$$

$$\text{EXPTIME} = \bigcup_{k>0} \text{TIME}(2^{n^k})$$

$$\text{NEXPTIME} = \bigcup_{k>0} \text{NTIME}(2^{n^k})$$

$$\text{LOGSPACE} = \text{SPACE}(\log n)$$

$$\text{NLOGSPACE} = \text{NSPACE}(\log n)$$

these definitions are relying on two-tape Turing machines with a read-only and a read/write tape

$$\text{PSPACE} = \bigcup_{k>0} \text{SPACE}(n^k)$$

$$\text{EXPSPACE} = \bigcup_{k>0} \text{SPACE}(2^{n^k})$$

- For every complexity class C we can define its complementary class

$$\text{coC} = \{\Lambda^* \setminus \Pi \mid \Pi \in C\}$$

An Alternative Definition for NP

Theorem: Consider a problem $\Pi \subseteq \Lambda^*$. The following are equivalent:

- $\Pi \in$ NP
- There is a relation $R \subseteq \Lambda^* \times \Lambda^*$ that is polynomially decidable such that

  $\Pi = \{\mathbf{u} \mid$ there exists $\mathbf{w}$ such that $|\mathbf{w}| \leq |\mathbf{u}|^k$ and $(\mathbf{u},\mathbf{w}) \in R\}$

witness or certificate

$\{\mathbf{xy} \in \Lambda^* \mid (\mathbf{x},\mathbf{y}) \in R\} \in$ PTIME

Example:

3SAT = $\{\varphi \mid \varphi$ is a 3CNF formula that is satisfiable$\}$

= $\{\varphi \mid \varphi$ is a 3CNF for which $\exists$ assignment α such that $|\alpha| \leq |\varphi|$ and $(\varphi,\alpha) \in R\}$

where $R = \{(\varphi,\alpha) \mid \alpha$ is a satisfying assignment for $\varphi\} \in$ PTIME

Relationship Among Complexity Classes

LOGSPACE $\subseteq$ NLOGSPACE $\subseteq$ PTIME $\subseteq$ NP, coNP $\subseteq$

PSPACE $\subseteq$ EXPTIME $\subseteq$ NEXPTIME, coNEXPTIME $\subseteq$ …

Some useful notes:

- For a deterministic complexity class C, coC = C
- coNLOGSPACE = NLOGSPACE
- It is generally believed that PTIME ≠ NP, but we don't know
- PTIME $\subset$ EXPTIME $\Rightarrow$ at least one containment between them is strict
- PSPACE = NPSPACE, EXPSPACE = NEXPSPACE, etc.
- But, we don't know whether LOGSPACE = NLOGSPACE

Complete Problems

- These are the hardest problems in a complexity class
- A problem that is complete for a class C, it is unlikely to belong in a lower class

- A problem Π is complete for a complexity class C, or simply C-complete, if:
  1. $\Pi \in C$
  2. Π is C-hard, i.e., every problem $\Pi' \in C$ can be efficiently reduced to Π

there exists a polynomial time algorithm (resp., logspace algorithm) that computes a function f such that $\mathbf{w} \in \Pi' \Leftrightarrow f(\mathbf{w}) \in \Pi$ – in this case we write $\Pi' \leq_P \Pi$ (resp., $\Pi' \leq_L \Pi$)

- To show that Π is C-hard it suffices to reduce some C-hard problem Π' to it

Some Complete Problems

- NP-complete
  - SAT (satisfiability of propositional formulas)
  - Many graph-theoretic problems (e.g., 3-colorability)
  - Traveling salesman
  - etc.
- PSPACE-complete
  - Quantified SAT (or simply QSAT)
  - Equivalence of two regular expressions
  - Many games (e.g., Geography)
  - etc.