Non-deterministic TM (NTM)

extend $\delta: Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{\leftarrow, \rightarrow, \circ\}^k)$ ... choice of instruction from the set

- successor relation is not a function → multiple computations for a given output
- if $\delta(q,x) = \emptyset$, we assume rejection.
- input α accepted ≡ ∃ at least one accepting computation
- halting ≡ all computations halt
- time & space: maximum over all computations

↓ wlog $|\delta(q,x)| \leq 2$

encumeration works again (NM_α = NTM with code α), we have an universal NTM &c.

Exercise: k-tape → 2-tape NTM with only constant-factor slowdown

Df: Non-deterministic complexity classes: NTIME(f), NTIMEF(f) for functions
└ all accepting computations must agree on result, ∃ at least one accepting comp.

Theorem: $NP = NTIME(poly(n)) := \bigcup_{k \geq 0} NTIME(n^k)$

Proof: ⊆ the NTM guesses the certificate using non-determinism & then it runs the verifier
⊇ the certificate encodes non-deterministic choices

Example: The following problem is NP-complete:
$\{\langle \alpha, \beta, 1^t \rangle \mid NM_\alpha$ accepts input β within t steps$\}$ → ∈NP: simulate $NM_\alpha(\beta)$ using universal NTM with an "alarm clock" (reject after t steps)

↓ reduction: calculate $t = p(n)$, $\alpha :=$ code of NTM solving source problem, pass β

Space Complexity

We want to count only "work space" of the TM.

3 types of tapes:

- input tape: read-only, head doesn't leave more than 1 cell before/after input string
- k work tapes: read-write
- output tape: write-only, head cannot move left

} space used by computation := # visited cells on the work tapes (for NTM: max over computations)

This doesn't change time complexity classes: we can copy input → work → output with constant slowdown

We can encode information in position of head on input tape.
- this makes a difference if work space $\in o(\log n)$
- otherwise we can keep track of the head position in binary

Space classes: (defined using machines which always halt)

DSPACE(f), NSPACE(f) } decision problems
& DSPACEF(f), NSPACEF(f) } functions

$PSPACE = DSPACE(poly(n))$
$NPSPACE = NSPACE(poly(n))$

We want f to be:
1. non-decreasing
2. space-constructible
   ∈ $f(n)$ can be computed from 1^n, result in binary, in space $O(f(n))$
3. usually $f(n) \geq \log n$

} proper space-complexity function

Basic inclusions: $DTIME(f) \subseteq NTIME(f) \subseteq DSPACE(f) \subseteq NSPACE(f)$

↑ can try all certificates in space $O(f)$

So: $P \subseteq NP \subseteq PSPACE \subseteq NPSPACE$

Thm: $DSPACE(f) \subseteq DTIME(2^{O(f)})$ for every $f \geq \log n$.

Proof: First, let's bound # reachable configurations: $|Q| \cdot (n+2) \cdot (|\Gamma|+1)^{f(n)} \cdot f(n)^k$ (# tapes)

- $|Q|$: state
- $(n+2)$: pos. of head on input tape
- $(|\Gamma|+1)^{f(n)}$: contents of work tapes, extra character for "end of tape"
- $f(n)^k$: pos. of heads on work tapes

... this is $O(2^{O(f(n))})$

If a configuration repeats, the whole computation loops. (↑ this requires deterministic TM ⊛)

↑ so if it halts, it must do so within $2^{O(f(n))}$ steps

⇒ add a binary counter of $O(f(n))$ bits, use it as alarm clock. (increment in every step of the original TM). Alarm expires ⇒ reject.

In space-bounded computation with space $\geq \log n$, we can always make sure than the machine halts.

Corollary: $NPSPACE \subseteq EXPTIME := DTIME(2^{n^{O(1)}})$ — this is called EXPTIME or EXP

We want to prove the same for $NSPACE(f)$, but ⊛ makes it more complicated.

Reachability method

Df: Configuration graph of a given NTM on a given input is a directed graph with (within space $f(n)$):

- $V :=$ set of configurations (for input tape, consider only head position) ↑ limited by available space
- $E :=$ successor relation (depends on α)
- start $\in V$ ... initial config
- accept $\in V$ ... modify the TM to clean up before accepting ← clear working tapes, rewind input tape } unique accepting config

👁 $|V| \in O(2^{O(f)})$, $|E| \in O(|V|)$, graph can be generated in $O(poly(|V|))$ time & $O(f)$ space.

👁 Machine accepts ⇔ graph contains a (directed) path from start to accept.

Thm: $NSPACE(f) \subseteq DTIME(2^{O(f)})$ for every $f \geq \log n$. [therefore $NPSPACE \subseteq EXPTIME$]

Proof: Construct the reachability graph & run BFS on it.

↑ time $O(poly(|V|, |E|))$ ↑ also time $O(poly(|V|, |E|))$

↑ which is $O(2^{O(f)})$

Generally: Time-/space-efficient algorithms for REACH translate to inclusions of complexity classes. ↑ $\{\langle G, s, t\rangle \mid \exists$ path from s to t in $G\}$

Thm (Savitch's): $NSPACE(f) \subseteq DSPACE(f^2)$ for every $f \geq \log n$.

Corollary: $NPSPACE \subseteq PSPACE$, so $NPSPACE = PSPACE$.

↳ will be proven soon...

Lemma: REACH $\in \mathcal{O}(\log^2 n)$

Proof: Use "middle-first search".

Recursive function $D_k(x,y)$ computing "$\exists$ walk from x to y with at most 2^k edges".

We have: $D_0(x,y) = (x=y) \vee ((x,y) \in E)$

$D_k(x,y) = \exists z \forall (D_{k-1}(x,z) \wedge D_{k-1}(z,y))$ ... FOR loop & recursion

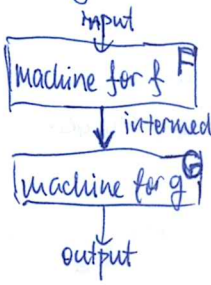
Every level of recursion requires $O(\log n)$ space for local variables, $\log n$ # levels suffice to find a path.

Now, we want to combine generator of config graph with this algorithm, but we don't have space to store the graph.

Lemma: If f can be computed in space $s(n)$ and g — " — $t(n)$, } $s,t \geq \log$ } also applies to composition of a language with a function (that is, a reduction)

Then $g(f(\ldots))$ can be computed in space $O(s(n) + t(2^{O(s(n))}))$

Proof:



Start G & keep track of position of head on G's input tape. $O(\log m)$ space

Whenever G moves its input head (& at the start of computation), re-run F to get the corresponding symbol of its output.

↳ modify F: reset work tapes on startup
reset input head — " —
keep track of output head position] $O(\log m)$ space
write to output tape: compare with G's input head pos = remember char in state
≠ discard written char (won't be read by F)

Total space:

$s(n)$ for F

$O(\log m)$ for head positions ... this is $O(t(m))$ $s(n)$

$t(m)$ for G

essentially, we combined a reachability alg. with an oracle for edges

Corollary: Savitch's thm.

↳ If $L \in NSPACE(f)$: graph generation requires $O(f)$ space, reachability needs $O(\log(2^{O(f)})^2) = O(f^2)$] combined by the lemma to $O(f^2)$

Corollary: DTIME$F(\log n)$ is closed under composition of functions.

Remark: REACH $\in O(\log n)$ would imply NSPACE(f) = DSPACE(f) ... but this is long open.

It's known that undirected UREACH $\in O(\log n)$ [Reingold 2004, non-trivial]

↳ this implies only SSPACE(f) = DSPACE(f)

↑ symmetric non-determinism (successor relation symmetric)

So we have: DTIME$(f) \subseteq$ NTIME$(f) \subseteq$ DSPACE$(f) \subseteq$ NSPACE$(f) \subseteq$ DTIME$(2^{O(f)})$

↳ $\subseteq$ DSPACE(f^2)

and: NSPACE$(\log n) \subseteq P \subseteq NP \subseteq$ PSPACE = NPSPACE $\subseteq$ EXPTIME

↑ this is also known as NL

↑ also = co-NPSPACE as PSPACE is closed under complements

More about PSPACE

Thm: QBF is PSPACE-complete. (with respect to $\leq_m^P$)

(QBF is sometimes called QSAT)

↑ the language of all true quantified Boolean formulas (all variables bound by quantifiers)

Proof: ① QBF ∈ PSPACE by the following recursive algorithm:

- $QBF(\forall x\, \varphi(x)) = QBF(\varphi(0)) \,\&\, QBF(\varphi(1))$
- $QBF(\exists x\, \varphi(x)) = QBF(\varphi(0)) \vee QBF(\varphi(1))$
- $QBF(\varphi \vee \psi) = QBF(\varphi) \vee QBF(\psi)$
- $QBF(\varphi \,\&\, \psi) = QBF(\varphi) \,\&\, QBF(\psi)$
- $QBF(\neg\varphi) = \neg QBF(\varphi)$

} $O(n)$ levels of recursion, $O(n)$ space per level. } $O(n^2)$ space

② QBF is PSPACE-hard: consider $L \in$ PSPACE, a TM M deciding L and its config. graph G.

- vectors of variables $\bar{x}$ encoding vertices ... $O(\text{poly}(n))$ bits [log of $|G|$]
- formula $\varphi(\bar{x}, \bar{y}) \equiv (\bar{x} = \bar{y}) \vee (x,y) \in E(G)$
  - can construct poly-sized circuit as in proof of Cook-Levin thm. & then reduce the circuit to an existentially-quantified formula as in Circuit-SAT $\leq_m^P$ SAT.
- mimic proof of Savitch's thm: $\varphi_k(\bar{x}, \bar{y}) \equiv$ "$\bar{y}$ is reachable from $\bar{x}$ within $\leq 2^k$ steps max."

  Failed attempt: $\varphi_k(\bar{x}, \bar{y}) \equiv \exists \bar{z}\, (\varphi_{k-1}(\bar{x}, \bar{z}) \,\&\, \varphi_{k-1}(\bar{z}, \bar{y}))$

  Double recursion ⇒ formula size grows exponentially !!

  Better: $\varphi_k(\bar{x}, \bar{y}) \equiv \exists \bar{z}\, \forall \bar{a}\, \forall \bar{b} \left(((\bar{a} = \bar{x} \,\&\, \bar{b} = \bar{z}) \vee (\bar{a} = \bar{z} \,\&\, \bar{b} = \bar{y})) \Rightarrow \varphi_{k-1}(\bar{a}, \bar{b}) \right)$
- G has size $O(2^{\text{poly}(n)}) \Rightarrow \log(\text{path len}) \in \text{poly}(n) \Rightarrow$ recursion has $\text{poly}(n)$ levels, formula size grows to $\text{poly}(n)$.

Intuition: PSPACE is the class of strategies for 2-player games with perfect information:

(∃ player 1's opening move) (∀ player 2's response) (∃ player 1's counter-response) (∀ ...) ...

Examples: these are known to be PSPACE-complete

- graph coloring game: undirected graph, finite set of k colors; each player colors an uncolored vertex; no edge must have both ends of the same color
- graph path game: building path edge by edge, ~~first player has his target~~, ~~2nd player~~ vertices must not repeat
- variants of Go, Checkers &c. (generalized to $N \times N$ boards)
- Sokoban (1-player, but enough internal state, which restricts future moves, but can be modified)

Alternating Turing Machine (ATM)

3 kinds of states:

- deterministic: config is accepting ⇔ next config is accepting
- existential: ~~accepts~~ config is accepting ⇔ ∃ non-det. choice ~~leading s.t. the rest~~ leading to accepting config.
- universal: is accepting ⇔ ∀ non-det. choice leads to accepting config.

We will require all computations to halt.

ATM → classes ATIME(f), ASPACE(f), AP ... [we won't define space-bounded classes as we require all computations to halt & alarm clocks don't help]

Theorem: AP = PSPACE.

Proof: $\supseteq$ is easy: QBF ∈ AP since we can execute quantifiers using corresponding state types. So if $L \leq_m^P$ QBF, we can first compute the reduction and then solve QBF.

$\subseteq$: simulate the ATM recursively as in QBF ∈ PSPACE.

- configurations take O(poly(n)) space – all computations are poly-time, so they are poly-space, too.
- recursion depth is bounded by time of the ATM.

Polynomial Hierarchy

Consider following restrictions of QBF:

- Σ_k-formulas (with no free variables): $(\exists x_1 \dots \exists x_{k_1})(\forall \dots)(\exists \dots) \dots \varphi(_)$ — k groups of quantifiers, starting with $\exists$; φ quantifier-free formula
- Π_k-formulas: similar, but starting with $(\forall \dots)$

} negation of a Σ_k-formula can be written as a Π_k-formula & vice versa

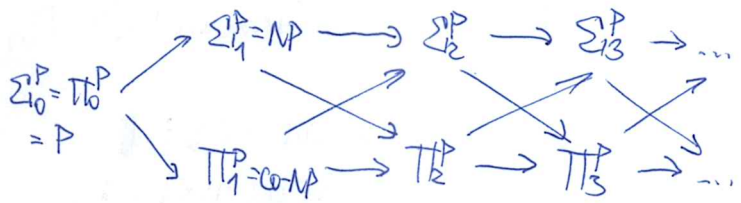
- Σ_k-SAT := $\{ \langle \varphi \rangle \mid \varphi$ is a true Σ_k-formula $\}$ (↑ encoding of) ... similarly Π_k-SAT.
- Σ_1-SAT is SAT (for general formulas, not only CNF), Π_1-SAT is TAUT.

We can use this to define new classes which generalize NP:

- $\Sigma_k^P := \{ L \mid L \leq_m^P \Sigma_k\text{-SAT} \}$ ... $\Sigma_1^P = NP$, $\Sigma_0^P = P$
- $\Pi_k^P := \{ L \mid L \leq_m^P \Pi_k\text{-SAT} \}$ ... $\Pi_1^P = co\text{-}NP$, $\Pi_0^P = P$, $\Pi_k^P = co\text{-}\Sigma_k^P$

} we defined classes using a problem complete for them

- generally, the following inclusions hold:



This is akin to the arithmetical hierarchy, but the inclusions are not known to be strict.

- $PH := \bigcup_k \Sigma_k^P = \bigcup_k \Pi_k^P$
- since every Σ_k/Π_k-SAT reduces trivially to QBF, we have PH ⊆ PSPACE (not known to be strict)

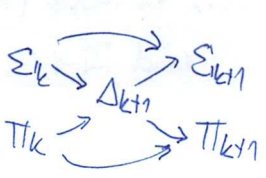
Example: "given Bool. formula φ, find a shorter ψ s.t. $\forall \bar{x}\ \varphi(\bar{x}) \Leftrightarrow \psi(\bar{x})$" $\in \Sigma_2^P$
(in fact, it's Σ_2^P-complete)

Remark: Definition using oracle machines also works:

$$\Sigma_{k+1}^P := NP[\Sigma_k^P] = NP[\Pi_k^P]$$

$$\Pi_{k+1}^P := co\text{-}NP[\Sigma_k^P] = co\text{-}NP[\Pi_k^P]$$

we can add $\Delta_{k+1}^P := P[\Sigma_k^P] = P[\Pi_k^P]$ ...



} this leads to the same hierarchy