# Complexity Theory
# WS 2009/10

Prof. Dr. Erich Grädel

Mathematische Grundlagen der Informatik
RWTH Aachen

# Contents

# 5 Alternating Complexity Classes

Alternating algorithms are a generalization of non-deterministic algorithms based on two-player games. Indeed, one can view non-deterministic algorithms as the restriction of alternating algorithms to solitaire (i.e., one-player) games. Since complexity classes are mostly defined in terms of Turing machines, we focus on the model of alternating Turing machines. But note that alternating algorithms can be defined in terms of other computational models, also.

**Definition 5.1.** An *alternating Turing machine* is a non-deterministic Turing machine whose state set $Q$ is divided into four classes $Q_\exists$, $Q_\forall$, $Q_{acc}$, and $Q_{rej}$. This means that there are existential, universal, accepting and rejecting states. States in $Q_{acc} \cup Q_{rej}$ are final states. A configuration of $M$ is called existential, universal, accepting, or rejecting according to its state.

The computation graph $G_{M,x}$ of an alternating Turing machine $M$ for an input $x$ is defined in the same way as for a non-deterministic Turing machine. Nodes are configurations (instantaneous descriptions) of $M$, there is a distinguished starting node $C_0(x)$ which is the input configuration of $M$ for input $x$, and there is an edge from configuration $C$ to configuration $C'$ if, and only if, $C'$ is a successor configuration of $C$. Recall that for *non-deterministic* Turing machines, the acceptance condition is given by the reachability problem: $M$ accepts $x$ if, and only if, in the graph $G_{M,x}$ some accepting configuration $C_a$ is reachable from $C_0(x)$. For *alternating* Turing machines, acceptance is defined by the GAME problem (see Sect. 3.3): the players here are called $\exists$ and $\forall$, where $\exists$ moves from existential configurations and $\forall$ from universal ones. Further, $\exists$ wins at accepting configurations and loses at rejecting ones. By definition, $M$ accepts $x$ if, and only if, Player $\exists$ has a winning strategy from $C_0(x)$ for the game on $G_{M,x}$.

When considering the computation tree $\mathfrak{T}_{M,x}$, which corresponds to the unraveling of the configuration graph from $C_0(x)$, we call a subtree $T_C$ accepting if $\exists$ has a winning strategy from $C$.

## 5.1 Complexity Classes

Time and space complexity are defined as for nondeterministic Turing machines. For a function $F : \mathbb{N} \to \mathbb{R}$, we say that an alternating Turing machine $M$ is $F$-time-bounded if for all inputs $x$, all computation paths from $C_0(x)$ terminate after at most $F(|x|)$ steps. Similarly, $M$ is $F$-space-bounded if no configuration of $M$ that is reachable from $C_0(x)$ uses more than $F(|x|)$ cells of work space. The complexity classes $\textsc{Atime}(F)$ and $\textsc{Aspace}(F)$ contain all problems that are decidable by, respectively, $F$-time bounded and $F$-space bounded alternating Turing machines.

The following classes are of particular interest:

- $\text{A}\textsc{Logspace} = \textsc{Aspace}(O(\log n))$,
- $\text{AP}\textsc{time} = \bigcup_{d\in\mathbb{N}} \textsc{Atime}(n^d)$,
- $\text{AP}\textsc{space} = \bigcup_{d\in\mathbb{N}} \textsc{Aspace}(n^d)$.

*Example* 5.2. $\textsc{qbf} \in \textsc{Atime}(O(n))$. We assume that, without loss of generality, negations appear only in front of variables. An alternating version of $\text{Eval}(\psi, \mathfrak{I})$ is the following:

**Algorithm 5.1.** Alternating $\text{Eval}(\psi, \mathfrak{I})$

**Input**: $(\psi, \mathfrak{I})$ where $\psi \in \textsc{qbf}$ und $\mathfrak{I}\colon \text{free}(\psi) \to \{0,1\}$
**if** $\psi = Y$ **then**
  **if** $\mathfrak{I}(Y) = 1$ **then** accept **else** reject
**endif**
**if** $\psi = \varphi_1 \vee \varphi_2$ **then** "$\exists$" guesses $i \in \{1,2\}$; **return** $\text{Eval}(\varphi_i, \mathfrak{I})$
**if** $\psi = \varphi_1 \wedge \varphi_2$ **then** "$\forall$" chooses $i \in \{1,2\}$; **return** $\text{Eval}(\varphi_i, \mathfrak{I})$
**if** $\psi = \exists X\varphi$ **then** "$\exists$" guesses $j \in \{0,1\}$; **return** $\text{Eval}(\varphi, \mathfrak{I}[X = j])$
**if** $\psi = \forall X\varphi$ **then** "$\forall$" chooses $j \in \{0,1\}$; **return** $\text{Eval}(\varphi, \mathfrak{I}[X = j])$





## 5.2 Alternating Versus Deterministic Complexity

There is a general slogan that parallel time complexity coincides with sequential space complexity.

**Theorem 5.3.** Let $S(n)$ be space-constructible with $S(n) \geq n$. Then,

$$\textsc{Nspace}(S) \subseteq \textsc{Atime}(S^2).$$

*Proof.* We use the same trick as in the proof of Savitch's Theorem: Let $L$ be decided by a nondeterministic Turing machine $M$ with space bounded by $S(n)$ and in time $2^{c\cdot S(n)}$. Let $\text{Conf}[S(n)]$ be the set of configurations of $M$ with space $\leq S(n)$. The alternating algorithm $\text{Reach}(C_1, C_2, t)$ (Algorithm 5.2) decides whether the configuration $C_2 \in \text{Conf}[S(n)]$ can be reached from configuration $C_1 \in \text{Conf}[S(n)]$ in at most $2^t$ steps. The algorithm is correct because $C_2$ is reachable from $C_1$ in at most $2^t$ steps if there is some $C$ such that $\text{Reach}(C_1, C, t-1)$ and $\text{Reach}(C, C_2, t-1)$ accept.

Let $f(t) = \max_{C_1,C_2\in\text{Conf}[S(n)]} \text{time}_{\text{Reach}}(C_1, C_2, t)$. Furthermore, $f(0) = O(S(n))$ and for all $t > 0$, $f(t) = O(S(n)) + f(t-1)$. Hence,

$$f(t) = (t+1)\cdot O(S(n)).$$

$L$ can then be decided as follows: At first, for an input $x$, the input configuration $C_0$ of $M$ on $x$ is constructed. Then, some accepting final configuration $C_a$ of $M$ is guessed. We will accept if $\text{Reach}(C_0, C_a, S(n))$

**Algorithm 5.2.** $\text{Reach}(C_1, C_2, t)$

**Input**: $C_1, C_2, t$
**if** $t = 0$ **then**
  **if** $C_1 = C_2$ *or* $C_2 \in \textit{Next}(C_1)$ **then** accept **else** reject
**else** /* $t > 0$ */
  existentially guess $C \in \text{Conf}[S(n)]$
  universally choose $(D_1, D_2) = (C_1, C)$ and $(D_1, D_2) = (C, C_2)$
  $\text{Reach}(D_1, D_2, t-1)$
**endif**

accepts. This algorithm needs

$$(c \cdot S(n) + 1)O(S(n)) = O(S^2(n))$$

steps. By the linear Speed-Up Theorem, which also applies to alternating Turing machines, $L \in \textsc{Atime}(S^2)$. Q.E.D.

**Theorem 5.4.** Let $T$ be space-constructible and $T(n) \geq n$. Then, $\textsc{Atime}(T) \subseteq \textsc{Dspace}(T^2)$.

*Proof.* Let $L \in \textsc{Atime}(T)$ and $M$ be some alternating Turing machine accepting $L$ in time bounded by $T(n)$. Then, there is some $r$ so that for all configurations $C$ of $M$: $|\mathrm{Next}(C)| \leq r$. Algorithm 5.3, $\mathcal{A}_T$, computes whether or not the subtree $T_C$ is accepting (output 1) or rejecting (output 0) for every configuration $C$ in $\mathfrak{T}_{M,x}$.

Obviously, this algorithm is working correctly. $\mathcal{A}_T(C_0(x))$ decides whether $M$ accepts $x$ and, hence, is a deterministic decision procedure for $L$.

---

**Algorithm 5.3.** $\mathcal{A}_T$, deterministic evaluation of $T_C$

---

**Input**: $C$
**if** $C$ *accepting* **then output** 1
**if** $C$ *rejecting* **then output** 0
**if** $C$ *existential* **then**
  **for** $i = 1, \ldots, r$ **do**
    compute $i$-th successor configuration $C_i$ of $C$
    **if** $F(C_i) = 1$ **then output** 1
  **endfor**
  **output** 0
**endif**
**if** $C$ *universal* **then**
  **for** $i = 1, \ldots, r$ **do**
    compute $i$-th successor configuration $C_i$ of $C$
    **if** $F(C_i) = 0$ **then output** 0
  **endfor**
  **output** 1
**endif**

---





How much space does this algorithm need? Let $C$ be some node of height $t$ in $\mathfrak{T}_{M,x}$, i.e., all computations of $M$ rooted at $C$ need at most $t$ steps. Then:

$$\mathrm{space}_{\mathcal{A}_T}(C) = \begin{cases} 0 & \text{if } t = 0 \\ \max_{C_i \in \mathrm{Next}(C)}(|C_i| + \mathrm{space}_{\mathcal{A}_T}(C_i)) & \text{if } t > 0\,. \end{cases}$$

Since $C_i \in \mathrm{Next}(C)$ is of height $t-1$, we obtain $\mathrm{space}_{\mathcal{A}_T}(C) \leq t \cdot T(n)$ and therefore $\mathrm{space}_{\mathcal{A}_T}(C_o) \leq T^2(n)$. Q.E.D.

In particular, we obtain

**Theorem 5.5** (Parallel time complexity = sequential space complexity)**.**

- $\textsc{APtime} = \textsc{Pspace}$.
- $\textsc{AExptime} = \textsc{Expspace}$.

*Proof.*

- $\textsc{Atime}(n^d) \subseteq \textsc{Dspace}(n^{2d}) \subseteq \textsc{Pspace}$,
  $\textsc{Dspace}(n^d) \subseteq \textsc{Nspace}(n^d) \subseteq \textsc{Atime}(n^{2d}) \subseteq \textsc{APtime}$.
- $\textsc{Atime}(2^{n^d}) \subseteq \textsc{Dspace}(2^{2n^d}) \subseteq \textsc{Expspace}$,
  $\textsc{Dspace}(2^{n^d}) \subseteq \textsc{Atime}(2^{2n^d}) \subseteq \textsc{AExptime}$. Q.E.D.

On the other hand, alternating space complexity corresponds to exponential deterministic time complexity.

**Theorem 5.6.** For any space-constructible function $S(n) \geq \log n$, we have that $\textsc{Aspace}(S) = \textsc{Dtime}(2^{O(S)})$.

*Proof.* The proof is closely associated with the GAME problem. For any $S$-space-bounded alternating Turing machine $M$, one can, given an input $x$, construct the computation graph $G_{M,x}$ in time $2^{O(S(|x|)}$ and then solve the GAME problem in order to decide the acceptance of $x$ by $M$.

For the converse, we shall show that for any $T(n) \geq n$ and any constant $c$, $\textsc{Dtime}(T) \subseteq \textsc{Aspace}(c \cdot \log T)$.

Let $L \in \textsc{Dtime}(T)$. Then there is a deterministic one-tape Turing machine $M$ that decides $L$ in time $T^2$. Let $\Gamma = \Sigma \cup (Q \times \Sigma) \cup \{*\}$ and

$t = G^2(n)$. Every configuration $C = (q, i, w)$ (in a computation on some input of length $n$) can be described by a word

$$\underline{c} = *w_0 \dots w_{i-1}(qw_i)w_{i+1} \dots w_t* \in \Gamma^{t+2}.$$

The $i$th symbol of the successor configuration depends only on the symbols at positions $i-1$, $i$, and $i+1$. Hence, there is a function $f_M : \Gamma^3 \to \Gamma$ such that, whenever symbols $a_{-1}$, $a_0$, and $a_1$ are at positions $i-1$, $i$ and $i+1$ of some configuration $\underline{c}$, the symbol $f_M(a_{-1}, a_0, a_1)$ will be at position $i$ of the successor configuration $\underline{c}'$.

The alternating algorithm $\mathcal{A}$ (Algorithm 5.4) decides $L$ using space $O(\log T(n))$. If $M$ accepts the input $x$, then Player $\exists$ has the following winning strategy for the game on $C_{\mathcal{A},x}$: the value chosen for $s$ is the time at which $M$ accepts $x$, and $(q^+a), i$ are chosen so that the configuration of $M$ at time $s$ is of the form $*w_0 \dots w_{i-1}(q^+a)w_{i+1} \dots w_t*$. At the $j$th iteration of the loop (that is, at configuration $s-j$), the symbols at positions $i-1, i, i+1$ of the configuration of $M$ at time $s-j$ are chosen for $a_{-1}, a_0, a_1$.

Conversely, if $M$ does not accept the input $x$, the $i$th symbol of the configuration at time $s$ is not $(q^+a)$. The following holds for all $j$: if, in the $j$th iteration of the loop, Player $\exists$ chooses $a_{-1}, a_0, a_1$, then

**Algorithm 5.4.** Alternating simulation of a determinisitc computation

existentially guess $s \le T^2(n) = t$
existentially guess $i \in \{0, \dots, s\}$
existentially guess $(q^+a) \in Q^+_{acc} \times \Sigma$
$b := (q^+a)$
**for** $j = 1, \dots, s$ **do**
    existentially guess $(a_{-1}, a_0, a_1) \in \Gamma^3$
    **if** $f_M(a_{-1}, a_0, a_1) \ne b$ **then** reject
    universally choose $k \in \{-1, 0, 1\}$
    $b := a_k$
    $i := i + k$
**endfor**
**if** *the i-th symbol of the input configuration of M on x equals b* **then** accept
**else** reject





either $f(a_{-1}, a_0, a_1) \ne b$, in which case Player $\exists$ loses immediately, or there is at least one $k \in \{-1, 0, 1\}$ such that the $(i+k)$th symbol of the configuration at time $s-j$ differs from $a_k$. Player $\forall$ then chooses exactly this $k$. At the end, $a_k$ will then be different from the $i$th symbol of the input configuration, so Player $\forall$ wins.

Hence $\mathcal{A}$ accepts $x$ if, and only if, $M$ does so. Q.E.D.

In particular, it follows that

- ALogspace = Ptime;
- APspace = Exptime.

The relationship between the major deterministic and alternating complexity classes is summarised in Fig. 5.1.

$$\begin{array}{ccccccccc}
\text{Logspace} & \subseteq & \text{Ptime} & \subseteq & \text{Pspace} & \subseteq & \text{Exptime} & \subseteq & \text{Expspace} \\
 & & \| & & \| & & \| & & \| \\
 & & \text{ALogspace} & \subseteq & \text{APtime} & \subseteq & \text{APspace} & \subseteq & \text{AExptime}
\end{array}$$

**Figure 5.1.** Relationship between deterministic and alternating complexity classes

## 5.3 Alternating Logarithmic Time

For time bounds $T(n) < n$, the standard model of alternating Turing machines needs to be modified a little by an indirect access mechanism. The machine writes down, in binary, an address $i$ on an separate index tape to access the $i$th symbol of the input. Using this model, it makes sense to define, for instance, the complexity class ALogtime = Atime($O(\log n)$).

Important examples of problems in ALogtime are

- the model-checking problem for propositional logic;
- the data complexity of first-order logic.

The results mentioned above relating alternating time and sequential space hold also for logarithmic time and space bounds. Note, however, that these do not imply that ALogtime = Logspace, owing to the quadratic overheads. It is known that ALogtime $\subseteq$ Logspace, but the converse inclusion is an open problem.

**Exercise 5.1.** Construct an ALogtime algorithm for the set of palindromes (i.e., words that are same when read from right to left and from left to right).