TECHNISCHE UNIVERSITÄT DRESDEN

COMPLEXITY THEORY

Lecture 16: Alternation

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 9th Dec 2019

Review

Baker, Gill, Solovay

Theorem 14.18 (Baker, Gill, Solovay, 1975): The answer to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ does not relativise: there are languages **A** and **B** such that $\mathsf{P}^{\mathbf{A}} = \mathsf{NP}^{\mathbf{A}}$ and $\mathsf{P}^{\mathbf{B}} \neq \mathsf{NP}^{\mathbf{B}}$.

In words: The P vs. NP problem does not relativise, and therefore cannot be solved by any techniques that do.

- Equality was shown using $\mathbf{A} =$ **True QBF**. It is so far not known that this oracle is not in P, so this might be the world we are living in.
- Inequality was shown using **B** that diagonalises against all polytime OTM to show that they cannot decide $\mathbf{L}_{\mathbf{B}}$.

Alternation

Alternating Computations

Non-deterministic TMs:

- Accept if there is an accepting run.
- Used to define classes like NP

Alternating Computations

Non-deterministic TMs:

- Accept if there is an accepting run.
- Used to define classes like NP

Complements of non-deterministic classes:

- Accept if all runs are accepting.
- Used to define classes like coNP

Alternating Computations

Non-deterministic TMs:

- Accept if there is an accepting run.
- Used to define classes like NP

Complements of non-deterministic classes:

- Accept if all runs are accepting.
- Used to define classes like coNP

We have seen that existential and universal modes can also alternate:

- Players take turns in games
- Quantifiers may alternate in QBF

Is there a suitable Turing Machine model to capture this?

Alternating Turing Machines

Definition 16.1: An alternating Turing machine (ATM) $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0)$ is a Turing machine with a non-deterministic transition function $\delta\colon Q \times \Gamma \to 2^{Q \times \Gamma \times \{\mathsf{L},\mathsf{R}\}}$ whose set of states is partitioned into existential and universal states:

$Q_\exists$: set of existential states $\qquad$ $Q_\forall$: set of universal states

- Configurations of ATMs are the same as for (N)TMs:
  tape(s) + state + head position
- A configuration can be universal or existential, depending on whether its state is universal or existential
- Possible transitions between configurations are defined as for NTMs

Alternating Turing Machines: Acceptance

Acceptance is defined inductively:

Definition 16.2: The set of accepting configurations of an ATM $\mathcal{M}$ is the least set of configurations C for which either of the following is true:

- C is existential and some successor configuration of C is accepting.
- C is universal and all successor configurations of C are accepting.

$\mathcal{M}$ accepts a word w if the start configuration on w is accepting.

Alternating Turing Machines: Acceptance

Acceptance is defined inductively:

Definition 16.2: The set of accepting configurations of an ATM $\mathcal{M}$ is the least set of configurations C for which either of the following is true:

- C is existential and some successor configuration of C is accepting.
- C is universal and all successor configurations of C are accepting.

$\mathcal{M}$ accepts a word w if the start configuration on w is accepting.

Note 1: configurations with no successor are a base case, since we have:

- An existential configuration without any successor configurations is rejecting.
- A universal configuration without any successor configurations is accepting.

Hence we don't need to specify accepting or rejecting states explicitly.

Note 2: defining this to be the least set implies that infinite runs are never enough to declare a configuration to be accepting.

Nondeterminism and Parallelism

ATMs can be seen as a generalisation of non-deterministic TMs:

> An NTM is an ATM where all states are existential (besides the single accepting state, which is always universal according to our definition).

Nondeterminism and Parallelism

ATMs can be seen as a generalisation of non-deterministic TMs:

An NTM is an ATM where all states are existential (besides the single accepting state, which is always universal according to our definition).

ATMs can be seen as a model of parallel computation:

In every step, fork the current process to create sub-processes that explore each possible transition in parallel

- for universal states, combine the results of sub-processes with AND
- for existential states, combine the results of sub-processes with OR

Alternative view: an ATM accepts if its computation tree, considered as an AND-OR tree, evaluates to true

Example: Alternating Algorithm for MinFormula

MinFormula

Input: A propositional formula φ.

Problem: Is φ the shortest formula that is satisfied by the same assignments as φ?

Example: Alternating Algorithm for MinFormula

MinFormula

Input: A propositional formula φ.

Problem: Is φ the shortest formula that is satisfied by the same assignments as φ?

MinFormula can be solved by an alternating algorithm:

```
MinFormula(formula φ) :
  universally choose ψ := formula shorter than φ
  existentially guess 𝓘 := assignment for variables in φ
  if φ^𝓘 = ψ^𝓘 :
    return false
  else :
    return true
```

Example: Alternating Algorithm for Geography

Recall the **Geography** game discussed in Lecture 10:

```
AltGeography(directed graph G, start node s) :
  Visited := {s} // visited nodes
  cur := s       // current node
  while true :
    // existential move:
    if all successors of cur are in Visited:
      return false
    existentially guess cur := unvisited successor of cur
    Visited := Visited ∪ {cur}
    // universal move:
    if all successors of cur are in Visited:
      return true
    universally choose cur := unvisited successor of cur
    Visited := Visited ∪ {cur}
```

Time and Space Bounded ATMs

As before, time and space bounds apply to any computation path in the computation tree.

Definition 16.3: Let $\mathcal{M}$ be an alternating 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.)

Defining Alternating Complexity Classes

Definition 16.4: Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

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

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

Common Alternating Complexity Classes

$$\text{AP} = \text{APTime} = \bigcup_{d\geq 1} \text{ATime}(n^d)$$ alternating polynomial time

$$\text{AExp} = \text{AExpTime} = \bigcup_{d\geq 1} \text{ATime}(2^{n^d})$$ alternating exponential time

$$\text{A2Exp} = \text{A2ExpTime} = \bigcup_{d\geq 1} \text{ATime}(2^{2^{n^d}})$$ alt. double-exponential time

$$\text{AL} = \text{ALogSpace} = \text{ASpace}(\log n)$$ alternating logarithmic space

$$\text{APSpace} = \bigcup_{d\geq 1} \text{ASpace}(n^d)$$ alternating polynomial space

$$\text{AExpSpace} = \bigcup_{d\geq 1} \text{ASpace}(2^{n^d})$$ alternating exponential space

Example 16.5: **Geography** $\in$ APTime.

Alternating Complexity Classes: Basic Properties

Nondeterminism:

ATMs can do everything that the corresponding NTMs can do, e.g., NP $\subseteq$ APTime

Alternating Complexity Classes: Basic Properties

Nondeterminism:
ATMs can do everything that the corresponding NTMs can do, e.g., NP $\subseteq$ APTime

Reductions: Polynomial many-one reductions can be used to show membership in many alternating complexity classes, e.g., if $\mathbf{L} \in$ APTime and $\mathbf{L}' \leq_p \mathbf{L}$ then $\mathbf{L}' \in$ APTime.

Alternating Complexity Classes: Basic Properties

Nondeterminism:
ATMs can do everything that the corresponding NTMs can do, e.g., NP $\subseteq$ APTime

Reductions: Polynomial many-one reductions can be used to show membership in many alternating complexity classes, e.g., if $\mathbf{L} \in$ APTime and $\mathbf{L}' \leq_p \mathbf{L}$ then $\mathbf{L}' \in$ APTime.

In particular: PSpace $\subseteq$ APTime (since **Geography** $\in$ APTime)

Alternating Complexity Classes: Basic Properties

Nondeterminism:
ATMs can do everything that the corresponding NTMs can do, e.g., NP $\subseteq$ APTime

Reductions: Polynomial many-one reductions can be used to show membership in many alternating complexity classes, e.g., if $\mathbf{L} \in$ APTime and $\mathbf{L}' \leq_p \mathbf{L}$ then $\mathbf{L}' \in$ APTime.

In particular: PSpace $\subseteq$ APTime (since **Geography** $\in$ APTime)

Complementation: ATMs are easily complemented:

- Let $\mathcal{M}$ be an ATM accepting language $\mathbf{L}(\mathcal{M})$
- Let $\mathcal{M}'$ be obtained from $\mathcal{M}$ by swapping existential and universal states
- Then $\mathbf{L}(\mathcal{M}') = \overline{\mathbf{L}(\mathcal{M})}$

For alternating algorithms this means: (1) negate all return values, (2) swap universal and existential branching points

Example: Complement of **MinFormula**

Original algorithm:

```
MinFormula(formula φ) :
  universally choose ψ := formula shorter than φ
  existentially guess 𝓘 := assignment for variables in φ
  if φ^𝓘 = ψ^𝓘 :
    return false
  else :
    return true
```

Complemented algorithm:

```
ComplMinFormula(formula φ) :
  existentially guess ψ := formula shorter than φ
  universally choose 𝓘 := assignment for variables in φ
  if φ^𝓘 = ψ^𝓘 :
    return true
  else :
    return false
```

Alternating Time vs. Deterministic Space

From Alternating Time to Deterministic Space

Theorem 16.6: For $f(n) \geq n$, we have $\text{ATime}(f) \subseteq \text{DSpace}(f^2)$.

From Alternating Time to Deterministic Space

Theorem 16.6: For $f(n) \geq n$, we have $\mathsf{ATime}(f) \subseteq \mathsf{DSpace}(f^2)$.

Proof: We simulate an ATM $\mathcal{M}$ using a TM $\mathcal{S}$:

- $\mathcal{S}$ performs a depth-first search of the configuration tree of $\mathcal{M}$
- The acceptance status of each node is computed recursively (similar to typical PSpace algorithms we have seen before)
- $\mathcal{M}$ accepts exactly if the root of the configuration tree is accepting

The maximum recursion depth is $f(n)$. The maximum size of a configuration is $O(f(n))$. Hence the claim follows. □

Note: The result can be strengthened to $\mathsf{ATime}(f) \subseteq \mathsf{DSpace}(f)$ by not storing the whole configuration. See [Sipser, Lemma 10.22].

From Nondeterministic Space to Alternating Time

Theorem 16.7: For $f(n) \geq n$, we have NSpace(f) $\subseteq$ ATime(f^2).

From Nondeterministic Space to Alternating Time

Theorem 16.7: For $f(n) \geq n$, we have NSpace(f) $\subseteq$ ATime(f^2).

Proof: We simulate an NTM $\mathcal{M}$ using an ATM $\mathcal{S}$.
Challenge: the computing paths of $\mathcal{M}$ might be up to $2^{df(n)}$ in length.

From Nondeterministic Space to Alternating Time

Theorem 16.7: For $f(n) \geq n$, we have $\mathsf{NSpace}(f) \subseteq \mathsf{ATime}(f^2)$.

Proof: We simulate an NTM $\mathcal{M}$ using an ATM $\mathcal{S}$.
Challenge: the computing paths of $\mathcal{M}$ might be up to $2^{df(n)}$ in length.
Solution: recursively solve Yieldability problems, as in Savitch's Theorem:

- We want to check if $\mathcal{M}$ can go from configuration C_1 to C_2 in at most k steps
- To do this, existentially guess an intermediate configuration C'.
- Universally check if $\mathcal{M}$ can go from C_1 to C' in $k/2$ steps, and from C' to C_2 in $k/2$ steps.

Storing one intermediate configuration C' takes space $O(f(n))$. Maximal recursion depth is $O(f(n))$. Hence the result follows. $\square$

Harvest: Alternating Time = Deterministic Space

For $f(n) \geq n$, we have shown

$$\text{ATime}(f) \subseteq \text{DSpace}(f^2) \quad \text{and} \quad \text{DSpace}(f) \subseteq \text{NSpace}(f) \subseteq \text{ATime}(f^2).$$

The quadratic increase is swallowed by (super)polynomial bounds:

Corollary 16.8 ("Alternating Time = Deterministic Space"): APTime = PSpace and AExpTime = ExpSpace.

Proof:

- $\text{ATime}(n^d) \subseteq \text{DSpace}(n^{2d}) \subseteq \text{PSpace}$
  $\text{DSpace}(n^d) \subseteq \text{NSpace}(n^d) \subseteq \text{ATime}(n^{2d}) \subseteq \text{APTime}$
- Second claim is left as an exercise □

One can also read this as "Parallel Time = Sequential Space."

Alternating Space vs. Deterministic Time

From Alternating Space to Deterministic Time

In this direction, the increase is exponential:

Theorem 16.9: For $f(n) \geq \log n$, we have $\mathsf{ASpace}(f) \subseteq \mathsf{DTime}(2^{O(f)})$.

From Alternating Space to Deterministic Time

In this direction, the increase is exponential:

Theorem 16.9: For $f(n) \geq \log n$, we have $\mathsf{ASpace}(f) \subseteq \mathsf{DTime}(2^{O(f)})$.

Proof: The proof is similar to the exponential deterministic simulation of space-bounded NTMs in Lecture 9 (Theorem 9.7):

- Construct configuration graph of ATM
- Iteratively compute acceptance status of each configuration
- Check if starting configuration is accepting

Each step can be done in exponential time (in particular, computing the acceptance condition in each step is no more difficult than for plain NTMs). □

From Deterministic Time To Alternating Space

The exponential blow-up can be reversed when going back to ATMs:

Theorem 16.10:
If $f(n) \geq \log n$ is space-constructible, then $\mathsf{DTime}(2^{O(f)}) \subseteq \mathsf{ASpace}(f)$.

From Deterministic Time To Alternating Space

The exponential blow-up can be reversed when going back to ATMs:

Theorem 16.10:
If $f(n) \geq \log n$ is space-constructible, then $\text{DTime}(2^{O(f)}) \subseteq \text{ASpace}(f)$.

Proof: We show: for any $g(n) \geq n$, we have $\text{DTime}(g) \subseteq \text{ASpace}(\log g)$.

We simulate a TM $\mathcal{M}$ using an ATM $\mathcal{S}$. This is not so easy

From Deterministic Time To Alternating Space

The exponential blow-up can be reversed when going back to ATMs:

Theorem 16.10:
If $f(n) \geq \log n$ is space-constructible, then $\text{DTime}(2^{O(f)}) \subseteq \text{ASpace}(f)$.

Proof: We show: for any $g(n) \geq n$, we have $\text{DTime}(g) \subseteq \text{ASpace}(\log g)$.

We simulate a TM $\mathcal{M}$ using an ATM $\mathcal{S}$. This is not so easy:

- A computation of $\mathcal{M}$ is exponentially longer than the space available to $\mathcal{S}$
  $\leadsto$ we solved this before with Yieldability

From Deterministic Time To Alternating Space

The exponential blow-up can be reversed when going back to ATMs:

Theorem 16.10:
If $f(n) \geq \log n$ is space-constructible, then $\text{DTime}(2^{O(f)}) \subseteq \text{ASpace}(f)$.

Proof: We show: for any $g(n) \geq n$, we have $\text{DTime}(g) \subseteq \text{ASpace}(\log g)$.

We simulate a TM $\mathcal{M}$ using an ATM $\mathcal{S}$. This is not so easy:

- A computation of $\mathcal{M}$ is exponentially longer than the space available to $\mathcal{S}$
  $\rightsquigarrow$ we solved this before with Yieldability
- A configuration of $\mathcal{M}$ is exponentially longer than the space available to $\mathcal{S}$
  $\rightsquigarrow$ this is more tricky . . .

There is a coarse proof sketch in [Sipser, Lemma 10.25]. We follow a more detailed proof from the lecture notes of Erich Grädel [Complexity Theory, WS 2009/10] (link).

From Deterministic Time To Alternating Space (2)

Notation: The proof is easier if we write a configuration $\sigma_1 \cdots \sigma_{i-1} q \sigma_i \sigma_{i+1} \cdots \sigma_m$ as a sequence

$$* \ \sigma_1 \ \cdots \ \sigma_{i-1} \ \langle q, \sigma_i \rangle \ \sigma_{i+1} \ \cdots \ \sigma_m \ *$$

of symbols from the set $\Omega = \{*\} \cup \Gamma \cup (Q \times \Gamma)$.

From Deterministic Time To Alternating Space (2)

Notation: The proof is easier if we write a configuration $\sigma_1 \cdots \sigma_{i-1} q \sigma_i \sigma_{i+1} \cdots \sigma_m$ as a sequence

$$* \; \sigma_1 \; \cdots \; \sigma_{i-1} \; \langle q, \sigma_i \rangle \; \sigma_{i+1} \; \cdots \; \sigma_m \; *$$

of symbols from the set $\Omega = \{*\} \cup \Gamma \cup (Q \times \Gamma)$.

Then the Ω-symbol (state and tape) at position i follows deterministically from the Ω-symbols at positions $i-1$, i, and $i+1$ in the previous step.
We write $\mathcal{M}(\omega_{i-1}, \omega_i, \omega_{i+1})$ for this symbol.

From Deterministic Time To Alternating Space (2)

Notation: The proof is easier if we write a configuration $\sigma_1 \cdots \sigma_{i-1} q \sigma_i \sigma_{i+1} \cdots \sigma_m$ as a sequence

$$* \; \sigma_1 \; \cdots \; \sigma_{i-1} \; \langle q, \sigma_i \rangle \; \sigma_{i+1} \; \cdots \; \sigma_m \; *$$

of symbols from the set $\Omega = \{*\} \cup \Gamma \cup (Q \times \Gamma)$.

Then the Ω-symbol (state and tape) at position i follows deterministically from the Ω-symbols at positions $i-1$, i, and $i+1$ in the previous step.
We write $\mathcal{M}(\omega_{i-1}, \omega_i, \omega_{i+1})$ for this symbol.

Proof idea:

- Only store a pointer to one cell in one configuration of $\mathcal{M}$
- Verify the contents of current cell i in step j by guessing the previous cell contents $\omega_{i-1}, \omega_i, \omega_{i+1}$ in step j.
- Check iteratively that the guessed symbols are correct

From Deterministic Time To Alternating Space (3)

Let $h : \mathbb{N} \to \mathbb{R}$ be a function in $O(g)$ that defines the exact time bound for $\mathcal{M}$ (no O-notation), and that can be computed in space $O(\log g)$.

```
AtmSimulateTm(TM M, input word w, time bound h) :
  existentially guess s ≤ h(|w|)       // halting step
  existentially guess i ∈ {0, ..., s}  // halting position
  existentially guess ω ∈ Q × Γ        // halting cell + state
  if M would not halt in ω :
     return false
  for j = s, ..., 1 do :
     existentially guess ⟨ω_{-1}, ω_0, ω_1⟩ ∈ Ω^3
     if M(ω_{-1}, ω_0, ω_{+1}) ≠ ω :
        return false
     universally choose ℓ ∈ {−1, 0, 1}
     ω := ω_ℓ
     i := i + ℓ
  // after tracing back s steps, check input configuration:
  return "input configuration of M on w has ω at position i"
```

Summary and Outlook

For $f(n) \geq \log n$, we have shown $\mathsf{ASpace}(f) = \mathsf{DTime}(2^{O(f)})$.

Corollary 16.11 ("Alternating Space = Exponential Deterministic Time"):
AL = P and APSpace = ExpTime.

Summary and Outlook

For $f(n) \geq \log n$, we have shown $\text{ASpace}(f) = \text{DTime}(2^{O(f)})$.

Corollary 16.11 ("Alternating Space = Exponential Deterministic Time"):
AL = P and APSpace = ExpTime.

We can sum up our findings as follows:

$$\begin{array}{ccccccccc} \text{L} & \subseteq & \text{PTime} & \subseteq & \text{PSpace} & \subseteq & \text{ExpTime} & \subseteq & \text{ExpSpace} \\ & & \parallel & & \parallel & & \parallel & & \parallel \\ & & \text{ALogSpace} & \subseteq & \text{APTime} & \subseteq & \text{APSpace} & \subseteq & \text{AExpTime} \end{array}$$

What's next?

- Alternation as a resource that can be bounded
- A hierarchy between NP and PSpace
- End-of-year consultation