

# COMPLEXITY THEORY

Lecture 16: Alternation

Markus Krötzsch Knowledge-Based Systems

TU Dresden, 10th Dec 2018

Baker, Gill, Solovay

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

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

- Equality was shown using **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  $L_B$ .



Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 2 of 25

## Alternation

Markus Krötzsch, 10th Dec 2018

## **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?

Markus Krötzsch, 10th Dec 2018

Complexity Theory

## 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.

slide 5 of 25

## 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  $\delta: Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$  whose set of states is partitioned into existential and universal states:

 $Q_{\exists}$ : set of existential states

 $Q_{\forall}$ : 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

Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 6 of 25

## 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

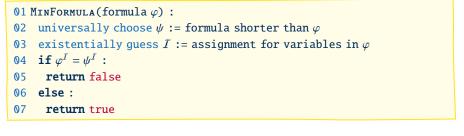
- for universal states, combine the results of sub-processes with AND
- for existential states, combine the results of sub-processes with OR

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 $\varphi$ .	
Problem:	Is $\varphi$ the shortest formula that is satisfied by the same assignments as $\varphi$ ?	

#### MINFORMULA can be solved by an alternating algorithm:



Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 9 of 25

## 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.)

Example: Alternating Algorithm for Geography

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

### **Q1** ALTGEOGRAPHY(directed graph G, start node s) :

- 02 Visited := {s} // visited nodes
- **03** cur := s // current node
- **04** while true :
- **05** // existential move:
- **06 if** all successors of cur are in Visited:
- **07** return false
- 08 existentially guess cur := unvisited successor of cur
- **09** Visited := Visited  $\cup$  {cur}
- 10 // universal move:
- 11 **if** all successors of cur are in Visited:
- 12 return true
- 13 universally choose cur := unvisited successor of cur
- 14 Visited := Visited  $\cup$  {cur}

Markus Krötzsch, 10th Dec 2018

Complexity Theory

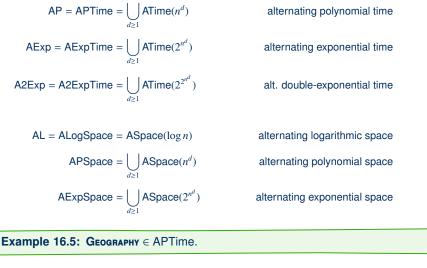
slide 10 of 25

## Defining Alternating Complexity Classes

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

- 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



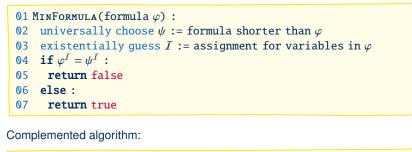
Markus Krötzsch, 10th Dec 2018

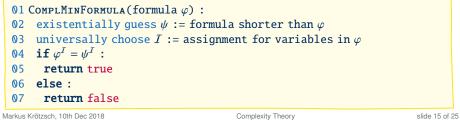
Complexity Theory

slide 13 of 25

## Example: Complement of MINFORMULA

### Original algorithm:





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  $L \in APT$  and  $L' \leq_p L$  then  $L' \in APT$  ime.

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

Complementation: ATMs are easily complemented:

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

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

Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 14 of 25

# Alternating Time vs. Deterministic Space

Markus Krötzsch, 10th Dec 2018

## From Alternating Time to Deterministic Space

**Theorem 16.6:** For  $f(n) \ge n$ , we have  $ATime(f) \subseteq DSpace(f^2)$ .

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

- 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)
- *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  $ATime(f) \subseteq DSpace(f)$  by not storing the whole configuration. See [Sipser, Lemma 10.22].

```
Markus Krötzsch, 10th Dec 2018
```

Complexity Theory

```
slide 17 of 25
```

# Harvest: Alternating Time = Deterministic Space

### For $f(n) \ge n$ , we have shown

 $ATime(f) \subseteq DSpace(f^2)$  and  $DSpace(f) \subseteq NSpace(f) \subseteq ATime(f^2)$ .

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

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

### Proof:

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

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

### From Nondeterministic Space to Alternating Time

**Theorem 16.7:** For  $f(n) \ge n$ , 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. Solution: recursively solve Yieldability problems, as in Savitch's Theorem:

- We want to check if 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.

```
Markus Krötzsch, 10th Dec 2018
```

Complexity Theory

slide 18 of 25

# Alternating Space vs. Deterministic Time

In this direction, the increase is exponential:

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

**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). 

Markus Krötzsch, 10th Dec 2018

Complexity Theory

```
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  $\Omega$ -symbol (state and tape) at position *i* follows deterministically from the  $\Omega$ -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 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

Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 21 of 25

### From Deterministic Time To Alternating Space

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

#### Theorem 16.10:

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

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

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  $\mathcal S$ → we solved this before with Yieldability
- A configuration of  $\mathcal{M}$  is exponentially longer than the space available to  $\mathcal{S}$  $\rightarrow$  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).

Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 22 of 25

# From Deterministic Time To Alternating Space (3)

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

	O1 ATMSIMULATETM(TM $M$ , input word $w$ , time bound $h$ ) :				
	02	existentially guess $s \le h( w )$ // halting step			
	03	existentially guess $i \in \{0, \ldots, s\}$ // halting position			
	04	existentially guess $\omega \in Q \times \Gamma$ // halting cell + state			
	05	if ${\mathcal M}$ would not halt in $\omega$ :			
	06	06 <b>return</b> false			
	07	<b>07</b> for $j = s,, 1$ do :			
	80	existentially guess $\langle \omega_{-1}, \omega_0, \omega_1  angle \in \Omega^3$			
	09	if $\mathcal{M}(\omega_{-1},\omega_0,\omega_{+1})\neq\omega$ :			
	10	<b>return</b> false			
	11	universally choose $\ell \in \{-1, 0, 1\}$			
	12	$\omega := \omega_\ell$			
	13	$i := i + \ell$			
	14	4 // after tracing back s steps, check input configuration:			
	15 <b>return</b> "input configuration of $\mathcal{M}$ on $w$ has $\omega$ at position $i$ "				
N	Markus Krötzsch, 10th Dec 2018 Complexity Theory		slide 24 of 2		

## Summary and Outlook

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

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

We can sum up our findings as follows:

### What's next?

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

Markus Krötzsch, 10th Dec 2018

Complexity Theory

slide 25 of 25