Complexity Theory

Alternation

Daniel Borchmann, Markus Krötzsch

Computational Logic

2016-01-05

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.										Result Variables.			
						$^1V_1$ 0 0 0 1 [1]	$^1V_2$ 0 0 0 2 [2]	$^1V_3$ 0 0 0 4 [n]	$^0V_4$ 0 0 0 0	$^0V_5$ 0 0 0 0	$^0V_6$ 0 0 0 0	$^0V_7$ 0 0 0 0	$^0V_8$ 0 0 0 0	$^0V_9$ 0 0 0 0	$^0V_{10}$ 0 0 0 0	$^0V_{11}$ 0 0 0 0	$^0V_{12}$ 0 0 0 0	$^0V_{13}$ 0 0 0 0	$^1V_{21}$ B_1 in a decimal fraction. [B_1]	$^1V_{22}$ B_3 in a decimal fraction. [B_3]	$^1V_{23}$ B_5 in a decimal fraction. [B_5]	$^0V_{24}$... 0 0 0 0 [B_7]
1	$\times$	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_3 = {}^1V_3 \right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	$-$	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{ {}^1V_4 = {}^2V_4 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	$+$	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{ {}^1V_5 = {}^2V_5 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	$\div$	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{ {}^2V_5 = {}^0V_5 \atop {}^2V_4 = {}^0V_4 \right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	$\div$	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{ {}^1V_{11} = {}^2V_{11} \atop {}^1V_2 = {}^1V_2 \right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	$-$	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{ {}^2V_{11} = {}^0V_{11} \atop {}^0V_{13} = {}^1V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	$-$	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{ {}^1V_3 = {}^1V_3 \atop {}^1V_1 = {}^1V_1 \right\}$	$= n - 1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	$+$	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^0V_7 = {}^1V_7 \right\}$	$= 2 + 0 = 2$	...	2	...	...	...	...	2										
9	$\div$	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{ {}^1V_6 = {}^1V_6 \atop {}^0V_{11} = {}^3V_{11} \right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	$\times$	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{ {}^1V_{21} = {}^1V_{21} \atop {}^3V_{11} = {}^3V_{11} \right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	....	B_1			
11	$+$	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{ {}^1V_{12} = {}^0V_{12} \atop {}^1V_{13} = {}^2V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{ -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2} \right\}$				
12	$-$	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{ {}^1V_{10} = {}^2V_{10} \atop {}^1V_1 = {}^1V_1 \right\}$	$= n - 2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	$-$	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{ {}^1V_6 = {}^2V_6 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n - 1$	1	...	...	...	...	$2n-1$											
14	$+$	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{ {}^1V_1 = {}^1V_1 \atop {}^1V_7 = {}^2V_7 \right\}$	$= 2 + 1 = 3$	1	...	...	...	...	...	3										
15	$\div$	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{ {}^2V_6 = {}^2V_6 \atop {}^2V_7 = {}^2V_7 \right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	$\times$	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{ {}^1V_8 = {}^0V_8 \atop {}^3V_{11} = {}^4V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	$-$	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{ {}^2V_6 = {}^3V_6 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n - 2$	1	...	...	...	...	$2n-2$											
18	$+$	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{ {}^2V_7 = {}^3V_7 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 3 + 1 = 4$	1	...	...	...	...	...	4										
19	$\div$	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{ {}^3V_6 = {}^3V_6 \atop {}^3V_7 = {}^3V_7 \right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{ \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = A_3 \right\}$						
20	$\times$	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{ {}^1V_9 = {}^0V_9 \atop {}^4V_{11} = {}^5V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	$\times$	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{ {}^1V_{22} = {}^1V_{22} \atop {}^0V_{12} = {}^2V_{12} \right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	....	.....	B_3		
22	$+$	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{ {}^2V_{12} = {}^0V_{12} \atop {}^2V_{13} = {}^3V_{13} \right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{ A_3 + B_1 A_1 + B_3 A_3 \right\}$				
23	$-$	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{ {}^2V_{10} = {}^3V_{10} \atop {}^1V_1 = {}^1V_1 \right\}$	$= n - 3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							
					Here follows a repetition of Operations thirteen to twenty-three.																	
24	$+$	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{ {}^4V_{13} = {}^0V_{13} \atop {}^0V_{24} = {}^1V_{24} \right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	....	....	....	.....	.....	.....	B_7
25	$+$	$^1V_1 + {}^1V_3$	$^1V_3$	$\left\{ {}^1V_1 = {}^1V_1 \atop {}^1V_3 = {}^1V_3 \right\}$ $\left\{ {}^5V_6 = {}^0V_6 \atop {}^5V_7 = {}^0V_7 \right\}$	$= n + 1 = 4 + 1 = 5$ .... by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0										

(early computation path written by Ada Lovelace)

Extended New Year's Review: Lectures 15–19

Alternation

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 14.1

An alternating Turing machine (ATM) $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0)$ is a Turing machine with a non-deterministic transition function $\delta\colon Q \times \Gamma \to \mathfrak{P}(Q \times \Gamma \times \{\, L, R \,\})$ 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

Alternating Turing Machines: Acceptance

Acceptance is defined recursively:

Definition 14.2

A configuration C of an ATM $\mathcal{M}$ is accepting if one 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 recursively:

Definition 14.2

A configuration C of an ATM $\mathcal{M}$ is accepting if one 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: configurations with no successor are the 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.

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
  exist. guess I := assignment for variables in φ
  if φ^I = ψ^I :
    return FALSE
  else :
    return TRUE
```

Example: Alternating Algorithm for Geography

```
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 14.3

Let $\mathcal{M}$ be an alternating Turing machine and let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

- $\mathcal{M}$ is f-time bounded if it halts on every input $w \in \Sigma^*$ and on every computation path after $\leq f(|w|)$ steps.
- $\mathcal{M}$ 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 14.4

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

- $\textsc{ATime}(f(n))$ is the class of all languages $\mathcal{L}$ for which there is an $O(f(n))$-time bounded alternating Turing machine deciding $\mathcal{L}$, for some $k \geq 1$.
- $\textsc{ASpace}(f(n))$ is the class of all languages $\mathcal{L}$ for which there is an $O(f(n))$-space bounded alternating Turing machine deciding $\mathcal{L}$.

Common Alternating Complexity Classes

$$\mathrm{AP} = \mathrm{APT{\scriptsize IME}} = \bigcup_{d \geq 1} \mathrm{AT{\scriptsize IME}}(n^d)$$ alternating polynomial time

$$\mathrm{AE{\scriptsize XP}} = \mathrm{AE{\scriptsize XP}T{\scriptsize IME}} = \bigcup_{d \geq 1} \mathrm{AT{\scriptsize IME}}(2^{n^d})$$ alternating exponential time

$$\mathrm{A2E{\scriptsize XP}} = \mathrm{A2E{\scriptsize XP}T{\scriptsize IME}} = \bigcup_{d \geq 1} \mathrm{AT{\scriptsize IME}}(2^{2^{n^d}})$$ alt. double-exponential time

$$\mathrm{AL} = \mathrm{AL{\scriptsize OG}S{\scriptsize PACE}} = \mathrm{AS{\scriptsize PACE}}(\log n)$$ alternating logarithmic space

$$\mathrm{APS{\scriptsize PACE}} = \bigcup_{d \geq 1} \mathrm{AS{\scriptsize PACE}}(n^d)$$ alternating polynomial space

$$\mathrm{AE{\scriptsize XP}S{\scriptsize PACE}} = \bigcup_{d \geq 1} \mathrm{AS{\scriptsize PACE}}(2^{n^d})$$ alternating exponential space

Example: GEOGRAPHY $\in \mathrm{APT{\scriptsize IME}}$

Alternating Complexity Classes: Basic Properties

Nondeterminism: ATMs can do everything that the corresponding NTMs can do, e.g., $\mathrm{NP} \subseteq \mathrm{AP}\textsc{Time}$

Alternating Complexity Classes: Basic Properties

Nondeterminism: ATMs can do everything that the corresponding NTMs can do, e.g., $\mathrm{NP} \subseteq \mathrm{APT{\small IME}}$

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

Alternating Complexity Classes: Basic Properties

Nondeterminism: ATMs can do everything that the corresponding NTMs can do, e.g., $\mathrm{NP} \subseteq \mathrm{APT\textsc{ime}}$

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

In particular: $\mathrm{PS\textsc{pace}} \subseteq \mathrm{APT\textsc{ime}}$ (since GEOGRAPHY $\in \mathrm{APT\textsc{ime}}$)

Alternating Complexity Classes: Basic Properties

Nondeterminism: ATMs can do everything that the corresponding NTMs can do, e.g., $\mathrm{NP} \subseteq \mathrm{APT{\small IME}}$

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

In particular: $\mathrm{PS{\small PACE}} \subseteq \mathrm{APT{\small IME}}$ (since GEOGRAPHY $\in \mathrm{APT{\small IME}}$)

Complementation: ATMs are easily complemented:

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

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
  exist. 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
  univ. choose 𝓘 := assignment for variables in φ
  if φ^𝓘 = ψ^𝓘 :
    return TRUE
  else :
    return FALSE
```