

Computational Complexity Theory - Course Notes

CCT Team

December 22, 2006

8 Lecture 7, 20/9-2006

8.1 Garden of Eden game

Definition 8.1 *The garden of eden game is a 2-player game played with a finite directed graph as a playing board. Three playing pieces are used: Adam, Eve and an apple. The two players are named snake and Burns. We will use the abbreviations A , E , app , S , B respectively. S controls the apple, B have limited control of Adam and Eve.*

- A and E are initially placed on the board and the apple is off. S makes the first move.
- When S is to move, app is off the board and he is free to insert it on any non-occupied vertex he chooses
- When B is to move, he swaps either A and E with app and removes app from the board.
- S wins if A sees E , meaning there is an edge from A to E .
- B wins if A never sees E

Proposition 8.1 (Who wins the game-theorem) *Assuming that both S and B plays optimally the following holds*

1. S wins the game from any position where there is a path from A to E . S wins in $\lceil \log_2 k \rceil$ rounds of moves, where k is the length of the path from A to E .
2. B wins the game from any position where there is no path from A to E .

If there is a path from A to E of length k ,

Proof.

1. Assume that there is a path v_1, \dots, v_k from A to E in a given board situation. S now places App at the vertex $v_{\lceil k/2 \rceil}$. B either moves A or E to $v_{\lceil k/2 \rceil}$ and removes the apple. No matter whether A or E is chosen there is still a path from A to E, and furthermore the distance is now half as big as before. S pulls the same trick once again inserting the apple at vertex $v_{\lceil (k/2)/2 \rceil}$ etc. From this it is clear that the snake can win in $\lceil \log_2 k \rceil$ rounds.

2. Assume that there is no path from A to E in a given board situation. The idea in the proof here is to look at one such possible situation:

$$\overset{A}{v_1} \leftarrow v_2 \rightarrow v_3 \rightarrow \overset{E}{v_4}$$

Snake chooses either v_2 or v_3 , let us say v_2 :

$$\overset{A}{v_1} \leftarrow \overset{app}{v_2} \rightarrow v_3 \rightarrow \overset{E}{v_4}$$

Now the trick is that B always replaces E with app, this means that A stays on his place. That gives

$$\overset{A}{v_1} \leftarrow \overset{E}{v_2} \rightarrow v_3 \rightarrow v_4$$

This can be repeated forever without S ever winning. If the game have gone on for more than $\lceil \log_2 n \rceil$ rounds, where n is the number of vertices's in the board graph it is clear that B is the winner.

□

Corollary 8.1 *We have just proved that $GAP \in ASPACE(\log n)$, and $GAP \in ATIME(\log n)$ where n denotes the size of the input graph.*

Proof. We can play the game sketched above using an alternating Turing machine. Given an instance $\langle G, u, v \rangle$ of GAP. We will use four working tapes. One for the position of Adam, one for Eve and one for the apple. The contents of each of these tapes are simply a vertex from G.

The snake and Burns does moves alternating. We will let the snake be an OR player and Burns be an AND player. First the snake non-deterministically choses a vertex for the apple. Then Burns non-deterministically swaps either the apple with Eve or apple with Adam. The game continues. Snake wins if Adam becomes adjacent to eve. Now the fourth tape is a tape which initially holds the value zero. For each round of this game a counter on this tape is incremented. If a any time the counter is greater than $\log_2(n)$ then the game is aborted and won by Burns (there

should have been a winner within this playing time).

Now the space used is $O(\log n)$ for the representation of the counter and the three others work tapes for Adam, Eve and the apple. The time used is at most $O(\log_2 n)$ as shown in the above theorem.

□

Theorem 8.1 *The monotone circuit value problem is P-complete.*

Theorem 8.2 $AL = P$

Proof.

The idea is to find a complete problem for P and show it is complete for AL. We pick the Circuit Value Problem (CVP).

CVP \in AL:

CVP is a member of AL since a given circuit can be checked using massive parallelism (see lecture notes for lecture 6). The input to the alternating machine is a variable free circuit represented as a list of edges and corresponding sorts (0, 1, and, or, and not) of nodes. Now we start from the output node and traverse the circuit backwards. If we encounter an AND (respectively OR) gate then we split the machine and leave an AND-gate (respectively OR). Sooner or later we eventually end up with nodes which is either 0 or 1. If a 0 is encountered we go into reject state. If 1 is encountered we go to accept state. The result of the computation of this alternating machine corresponds to the value of the circuit. If the alternating machine accepts the value of the circuit is 1 (respectively for reject and 0).

We only use logarithmic space since each machine uses one pointer which points to the gate we currently are at now. Since CVP is complete for P we get $AL \subseteq P$.

$\forall L \in AL : L \leq CVP$:

Given any language L in AL. Let L be decided by an alternating Turing machine M using only logarithmic space. For any input x we will describe a reduction mapping input strings to circuits $r(x)$ so that $r(x) \in CVP$ iff M accepts x . Now let the gates of the circuit be of the form C where C is the a configuration of M on x . There is an edge from C_1 to C_2 if and only if C_2 yields C_1 in one step. The sort of a gate C depends on the state of the configuration C . If C is in OR state the gate is an OR gate. If it is an AND gate is is an AND gate. If it is an accepting state the sort is 1. If it is a rejecting state the sort of the gate is 0. Since there are at most

polynomially many configurations we will create a circuit which is of polynomial size.

Corollary 8.2 *For any space constructible bound $s(n) \geq \log n$ it holds*

$$\text{ASPACE}(O(s(n))) = \text{DTIME}[2^{O(s(n))}].$$

Proof. By up-wards translation.

□

Theorem 8.3 $\text{AP} = \text{PSPACE}$.

Proof.

“ \subseteq “: Given a language in AP. There is an alternating polynomial time Turing Machine M which decides L . We can look at the running of M on any input using a two-player game perspective. If the first state is an AND state, then the AND player should have a winning strategy no matter how the rest of the game proceeds. If the first game is an OR state the OR player should be have a winning strategy. The winning strategy of an AND player involves the fact that no matter how the and player choses the next branch he will eventually win. The winning strategy of an OR player has the property that for a least one choice of a branch he will eventually win.

We will now use a deterministic Turing machine running in polynomial space to decide who wins the above game. We can mimic the computation of M since for each configuration of M we can use the transition function to discover the succeeding configurations. We will not expand the whole computation graph of G since this might be to large. Instead we will traverse the configuration graph in a depth first manner using a stack. We will propagate accepts and rejects according to whether the parent node is an AND or and OR gate.

“ \supseteq “: Given a language $L \in \text{PSPACE}$ and assume L is decided by the Turing machine M . Look at the configuration graph of the machine. We would like to determine whether for some x , machine M ends up in an accepting state. So

- We play the garden of Eden game on the configuration graph of the PSPACE-machine (we do not need the entire configuration graph, it is implicitly given).

- Since $\text{PSPACE} \subseteq \text{EXP}$ the configuration graph can at most be of exponential size say 2^{cn} for some constant c . By corollary 8.1 the time usage is then at most $O(\log 2^{cn})$ which is $O(n)$.

It therefore holds that M can be simulated in alternating polynomial time.

□

Corollary 8.3 *By up-wards translation it holds that if $t(n) \geq n$ is a time constructible bound then $\text{ATIME}[t^{O(1)}] = \text{DSPACE}[t^{O(1)}]$.*

8.2 Alternating Turing machines and time and space

Open question 8.1 *Are the following true or false $P \neq \text{NP}$, $\text{NL} \neq \text{NP}$, $L \neq \text{NP}$, $\neg(\text{NP} \subseteq \text{PSIZE})$, $\neg(\text{NP} \subseteq \text{SIZE}(O(n)))$, $\text{SAT} \notin \text{SIZE}(O(n))$.*

Theorem 8.4 (Fortnow '96) $\text{SAT} \notin \text{NL} \cap \text{SIZE}(n^{1+O(1)})$.

Proof. See lecture notes for lecture 8.

□