STAT 155 Notes
Professor: Adrian Gonzalez Casanova

Neo Lee

Spring 2024

CONTENTS

Chapter 1

Combinatorial games

1.1 Lecture 1 (skipped)

1.2 Lecture 2 (skipped)

1.3 Lecture 3

Definition 1.3.1: Combinatorial game

A two-player game of perfect information where the players alternate moves and the game ends in a finite number of moves. At each position $v \in V$, there is a set of moves E_v available to the player whose turn it is. The game ends when $E_v = \emptyset$.

Note:

We can think of a combinatorial game as a directed graph $G = (V, E)$ where V is the set of positions and E is the set of edges. The game starts at some vertex v_0 and ends when v_0 has no outgoing edges.

Definition 1.3.2: Impartial game

A game where the set of moves available to each player is the same at each position. The terminal reward to each player is the same. Players alternate moves.

Definition 1.3.3: Partisan game

The complement definition of an impartial game.

Example 1.3.1 (Impartial games)

1. Tic-tac-toe

2. Chomp

3. Subtraction games

4. Nim

> **Definition 1.3.4: Proressively bounded combinatorial game**
>
> A combinatorial game is progressively bounded if for every starting position v_0, there is a finite bound $B(v_0)$ on the number of moves in the game before the game ends.

> **Definition 1.3.5: Strategy**
>
> Let $V_{NT} \subseteq V$ is the set of non-terminal positions. A strategy for a player is a function that assigns a legal move to each non-terminal position $v \in V_{NT}$
>
> $$\sigma : V_{NT} \to \cup_{v \in V_{NT}} E_v.$$

> **Definition 1.3.6: Winning strategy**
>
> A winning strategy for a player from position $v \in V$ is a strategy that is guaranteed to result in a win for that player.

Theorem 1.3.1 Existence of a winning strategy

In a progressively bounded, impartial combinatorial game, $V = F \sqcup S$. That is, from any initial position, one of the players has a winning strategy.

> **Note:**
> Denote F, S as the wining positions for first and second player respectively.

Proof: We prove this by induction on the finite bound $B(v)$ on the number of moves in the game before the game ends. If $B(v) = 0$, then the game ends immediately and v is a winning position either for the first or second player. Hence, for $B(v) = 0$, the theorem holds.

Now suppose that the theorem holds for all $B(v) < k$ and consider a game with $B(v) = k$. Let $v \in V$ be the initial position. Without loss of generality, suppose it is the first player's turn. The player can move along some edges $e \in E_v$ to v'. By inductive hypothesis, $V = F \sqcup S$ for $B(v) = k - 1$. If $v' \in F$, then the first player has a winning strategy at v, else if $v' \in S$, then the second player has a winning strategy at v. Hence, for all v, either $v \in F$ or $v \in S$. This proves the theorem.   ☺

Theorem 1.3.2 Existence of winning strategy for first player in rectangular Chomp

In a non-terminal rectangular Chomp game with $m \times n$ for board, the first player always has a winning strategy.

Proof: First player eats the top right corner to move to v'. Then, by *Theorem 1.3.1*, the new position v' is either a winning position for the first player or the second player. Suppose $v' \in S$, and the next move by the second player moves to v''. Then, in fact, the first player can choose to not eat the right top corner in the first move, but instead directly move to v'', which would make v the winning position for the first player.

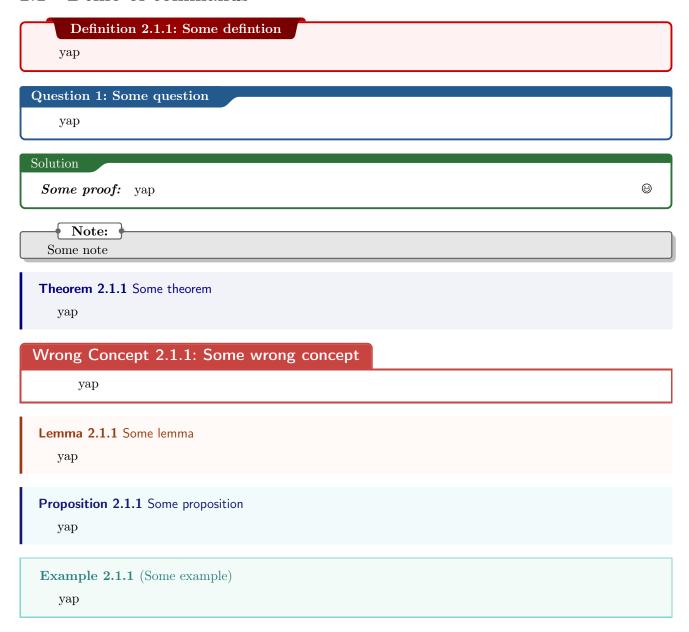> **Note:**
> - v' is the rectangular board with the top right corner eaten.
>
> - This is called a *strategy stealing*.
>
> ☺

Chapter 2

Starting a new chapter

2.1 Demo of commands

Definition 2.1.1: Some defintion

yap

Question 1: Some question

yap

Solution

Some proof: yap ☺

Note:

Some note

Theorem 2.1.1 Some theorem

yap

Wrong Concept 2.1.1: Some wrong concept

yap

Lemma 2.1.1 Some lemma

yap

Proposition 2.1.1 Some proposition

yap

Example 2.1.1 (Some example)

yap

Claim 2.1.1 Some claim

yap

Corollary 2.1.1 Some corollary

yap

Some unlabeled theorem

This is a new paragraph

Algorithm 1: Some algorithm

Input: input
Output: output

```
    /* This is a comment */
1   This is first line ;                                    // This is also a comment
2   if x > 5 then
3   |   do nothing
4   else if x < 5 then
5   |   do nothing
6   else
7   |   do nothing
8   end
9   while x == 5 do
10  |   still do nothing
11  end
12  foreach x = 1 : 5 do
13  |   do nothing
14  end
15  return return nothing
```