# 4701 pre-presentation

Team '); DROP TABLE Teams; --

# The Team

Alex Slover slover.alex@gmail.com
Andrew Francis agf33@cornell.edu
Haruki Yukawa hy97@cornell.edu

# Problem Statement and Motivation

- AI system for Modified Chinese Checkers
- Limited in terms of computation time and memory space
- Choose optimal move on each turn to win
  - conditional on the opponent's behavior

# I/O Specification

Inputs:

- Game state information from the environment
  - I/O protocol: http://cs4701.wikispaces.com/Modified+Chinese+Checkers
- Pre-calculated analytics (if any) that the agent will use

Outputs:

- The move for this turn
- Logs for analysis
- GUI for testing/analysis

# Background Reading

We've identified several texts that look helpful, and plan to look for more as we go


- Russell, Norvig (2009). Artificial Intelligence: A Modern Approach 2nd Ed.
- Mitchell (1997). Machine Learning
- Hastie, Tibshirani, Friedman (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction
- Bellman, R. E. (1965). On the application of dynamic programming to the determination of optimal play in chess and checkers. PNAS, 53, 244–246.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210–229.
- Samuel, A. L. (1967). Some studies in machine learning using the game of checkers II—Recent progress. IBM Journal of Research and Development, 11(6), 601–617.
- Schaeffer, J. (2008). One Jump Ahead: Computer Perfection at Checkers. Springer-Verlag. Schaeffer, J., Burch, N., Bjornsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers is solved. Science, 317, 1518–1522.

# General Approach

Where's the AI?

- Alpha Beta Search
- Learning from logs (potentially against itself) to improve heuristics
- Recognizing opponents and augmenting play-style
- Search Tree Pruning heuristics
- Special behaviors for beginning and ending games
- Different modes for different boards

# Systems Architecture and Work Plan

(a) Components
- Modular architecture for agent
- Analytic GUI

(b) Testing
- Unit tests! Unit tests!
- Automated QA harness

(c) Distribution of work
- Total code ownership
- Weekly "sprints"

# Data Sources

- Production (i.e. nightly runs against other agents) logfiles
- Logfiles generated by playing against itself in a dev environment

# Evaluation Plans

(a) Performance

Programmatical performance

- Plys searched per second?
- Memory consumption (the smaller the better)

Optimality of solution

- Results of games played against other agents, as well as previous iterations of itself

(b) Simple “toy” problem

- Move legality
- Search for the optimal solution for a miniature game with a sufficiently small state space (?)

(c) “Hard” test case

- Playing against the best agent (judged by proportion of wins) available

# Schedule

| Deliverable | Target Date |
| --- | --- |
| Define interfaces for core classes | 9/15/2011 |
| Prototype 2 - alpha-beta search bot with simple evaluation fxns, updated GUI | 10/13/2011 |
| Prototype 3 - offline ML models for eval fxn, pruning heuristics | 10/31/2011 |
| Prototype 4 - online ML models for opponent behavior, adjusting own behavior | 11/10/2011 |