PSPACE

Slides by Dan Fu

Outline

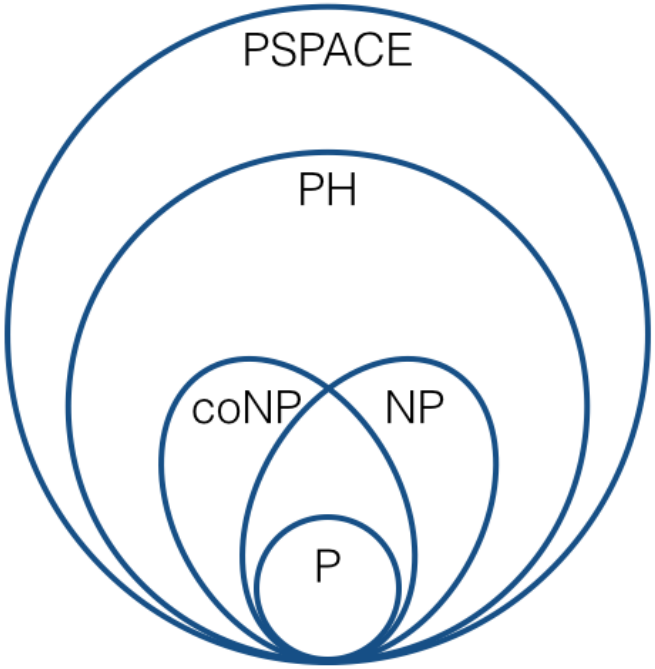
- PSPACE
- PSPACE-Completeness
- QBF and TQBF
- Chess AI

PSPACE

- Languages that can be solved with a Turing Machine that uses space (total # cells) that is polynomial to the size of the input (# cells in the input string)
- Time complexity: P and NP
- Space complexity: PSPACE and NPSPACE
- NPSPACE = PSPACE
  - Savitch's Theorem, a deterministic machine will only use at most the space of a nondeterministic machine, squared.
  - NSPACE(S (n)) ⊆ SPACE(S$(n)^2$)

PSPACE

- w.r.t the other classes we've seen, it should look like this



PSPACE-Complete

- A PSPACE-Complete problem is PSPACE and every other PSPACE problem can be transformed to it in polynomial time.
- Hardest problems of PSPACE

Poll: What makes PSPACE problems difficult?

A. SPACE includes many more problems than NP problems
B. SPACE can use as much time as it wants… SPACE can even use an exponential amount of time
C. The solution to a SPACE problem may not be efficiently verifiable

What makes PSPACE-Complete difficult?

A. PSPACE-Complete describes space bounded problems, rather than time bounded problems
B. Problems are not easy to solve (like NP-Complete), but **also difficult to verify** even if given a solution
C. The working memory for the solution could use only polynomial space, but it can be an exponentially long solution.

NP ⊆ PSPACE

- Write a series of for loops for searching the tree of possible NP execution paths.
- It may take exponential time to try all certificates to an NP-Complete problem, but they each only use polynomial space
- PSPACE-Complete seems (but has not been proved) to be harder than NP-Complete

Quantified Boolean Formula (QBF)

- SAT (archetypal NP-Complete problem)
  - $\exists x_1 \exists x_2 \exists x_3 \ldots : (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge \cdots$
  - Looks for the existence of a single solution

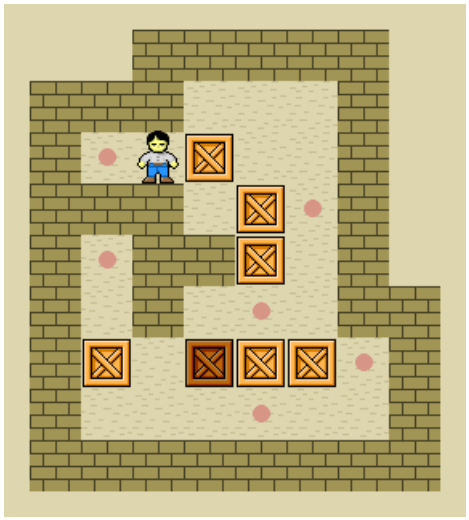
- QBF (archetypal PSPACE-Complete problem)
  - $\exists x_1 \forall x_2 \exists x_3 \ldots : (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge \cdots$
  - Can include ∀ quantifiers
  - Captures the idea of “games” between 2 players
  - x_{odd} for player 1, x_{even} for player 2
  - So does some move exist for player 1, such that for all moves by player 2, another move exists for player 1, etc…where the game will reach a determinable result?

Discussion

What are examples of games that match the QBF description?

PSPACE-Complete Games

Checkers



Sokoban

Chess

Images from Wikipedia

Sciencexpress Research Article

Checkers Is Solved

Jonathan Schaeffer,* Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, Steve Sutphen

Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada.

*To whom correspondence should be addressed. E-mail: jonathan@cs.ualberta.ca

The game of checkers has roughly 500 billion billion possible positions (5×10^{20}). The task of solving the game, determining the final result in a game with no mistakes made by either player, is daunting. Since 1989, almost continuously, dozens of computers have been working on solving checkers, applying state-of-the-art artificial intelligence techniques to the proving process. This paper announces that checkers is now solved: perfect play by both sides leads to a draw. This is the most challenging popular game to be solved to date, roughly one million times more complex than Connect Four. Artificial intelligence technology has been used to generate strong heuristic-based game-playing programs, such as DEEP BLUE for chess. Solving a game takes this to the next level, by replacing the heuristics with perfection.

Enumerate all end-game positions (≤10 pieces on the board) = 39 trillion positions, and figure out how to get there from 19 starting positions

Took 13 years to calculate, using 50-200 computers

Looked for convergences in play, simplifying equivalent board states

How many chess moves?

- Shannon number - 10^{120}
- Hardy - $10^{10^{50}}$

- https://www.youtube.com/watch?v=Km024eldY1A

- Bound the # of useful moves before a game ends (or else a game of chess could go on exponentially long)
- Artificial extensions to an $n \times n$ board
- Initially believed to be PSPACE, but actually harder than PSPACE (EXPTIME)
- But the actual game of chess is $O(1)$ using lookup tables, because there is a finite number of moves

Turochamp, 1948 – 1st chess computer program

- Developed by Alan Turing and David Champernowne in 1948
- A set of algorithms that could be implemented if a sufficiently powerful computer existed
- Could only look ahead 2 moves
- Scored actions such as check, threat to opponent's pieces or safety of own pieces, taking pieces according to their value, etc.
- Turing executed the program by hand and lost to computer scientist Alick Glennie
  - Each move took 30 minutes

Ferranti Mark 1 – early 1950s

- One of the first commercially available general-purpose digital computers
- Dr. Dietrich Prinz wrote the first running chess program
- Only worked for mate-in-two problems
- Took 15-20 minutes
- Mostly based on Turochamp

From University of Manchester CS

Minimax Algorithm, 1950s

Los Alamos National Lab, Computer History Museum

- John von Neumann, game theory
- Score each piece
  - Pawn = 1
  - Knight = 3
  - Bishop = 3.5
  - Rook = 5
  - Queen = 10
  - King = 1000
- Build a search tree
- Ran on the MANIAC I computer, 6x6 board, won 1 out of 3 games against a chess player with only 1 week of experience

Chess Opening Moves Library

- Mac Hac VI, 1967
  - First computer to beat a human in a tournament
  - First computer to get a chess rating
  - Used lookup tables for common openings, scoring them to turn 5
- Belle, 1979
  - Used a hash map to make things faster
  - Could generate up to 200,000 chess board states per second, searching 9 moves ahead
  - Won a lot of *computer* chess tournaments

Deep Blue, 1990s – 1st computer to beat a world chess champion !

- Feng-hsiung Hsu @ IBM
- Defeated Garry Kasparov (#1 chess player of the previous generation) in 1997
- Utilized parallel computing
- Still using opening libraries, deep search algorithms, heuristics, typically 6-20 moves ahead depending on situation
- Still a brute force algorithm, searching 200 million board states per second

AlphaZero, 2015 – used AI and self-training to construct it's strategy

- No databases
- Defeated Stockfish (155W-6L-839Draws)
  - Currently the best brute force chess program
- However, only trained for <24 hours of playing itself
- Searches only ~100,000 positions per second but is "smarter" at selecting promising moves