A Search-based Approach for Solving Sum Games

CGTC V, Lisbon 2025

Martin Müller, University of Alberta
Work with Taylor Folkersen, Henry Du, Zahra Bashir, Fatemeh Tavakoli, all University of Alberta

Overview

- Combinatorial games - mathematics vs computing science
- Goals of this research - Why another CGT solver?
- Work on single games
- MCGS - A **M**inimax-based **C**ombinatorial **G**ame **S**olver
  - Framework
  - Current state and future plans

Combinatorial Games Research

in Mathematics vs Computing Science

- In Mathematics:
  - Build theories of CGT, deeply understand games with specific structure
  - Prove theorems about whole (infinite) ***classes*** of games
- In Computing Science:
  - Solve ***specific*** game positions
  - Find efficient and ***general*** algorithms to solve games

Mathematics + Computing Science for CGT

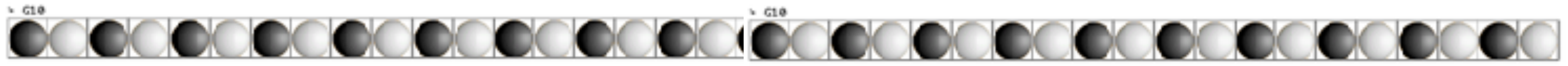
- **Computing Science helps** mathematicians:
  - Explore (large) numbers of games in a class
  - Help to find patterns, verify or refute conjectures
- **Math helps** computing scientists:
  - Improve efficiency of computation
  - Rules to simplify games
  - Basis to exploit sum structure, and partial order of games

Project Background and (Short) History

- 2017 and 2022: taught graduate courses - Algorithms in Combinatorial Game Theory
- Several course assignments and research projects: solve Clobber by search
- Strongest solver by Taylor Folkersen's team. Later in 2022: Taylor improves it further
- 2023/24: Henry Du creates two solvers for NoGo. The newer one is based on CGT
- Goal: build general search-based solver system
    - MCGS, Minimax-based Combinatorial Game Solver
- Fall 2024: started implementation.
- December 2024: hired Taylor Folkersen

Single Game Results - Linear Clobber

- Conjecture (Albert et al 2005, [1]): $(BW)^n$ is an N position for n != 3
  - Previously [1]: proven up to n=19



  - Now proven up to **n = 33** (!) (Folkersen et al 2022, Folkersen 2022)

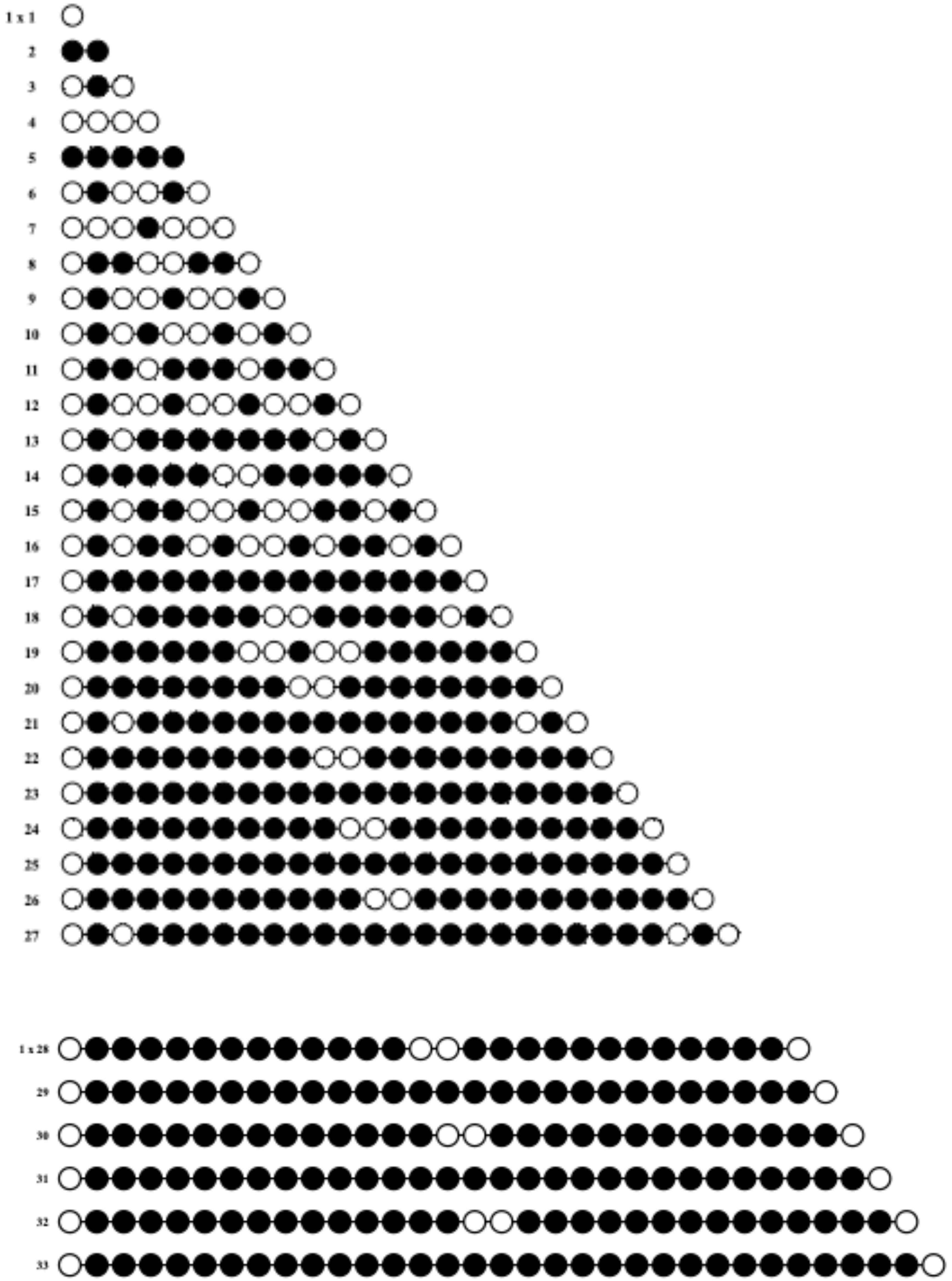
  - Efficient solver uses many CGT-inspired techniques

[1] M. Albert, J.P. Grossman, R. Nowakowski, and D. Wolfe. An introduction to Clobber. INTEGERS: The Electronic Journal of Combinatorial Number Theory, 5(2), 2005.

Note: at CGTC V, Hayward et all announced a general proof of the conjecture!

Single Game Results - Linear NoGo

- Solved all empty boards **up to 1x39**
  (Du et al 2023, Du et al 2024)
  - Only 1x1 and 1x4 are P positions
  - All others are N positions
- Solved all opening moves as win or loss (N or P) up to 1x33 - wins = white stones in figure, losses = black stones
- Up to 1x27 (Du et al 2023)
  used traditional (full board) minimax search
- Larger boards: CGT-based techniques (Du et al 2024)
  - Reduces the search by about two orders of magnitude for large boards

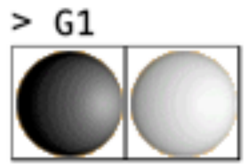
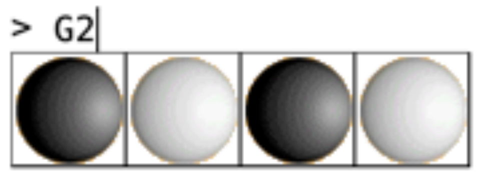
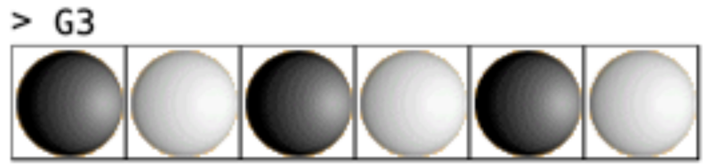
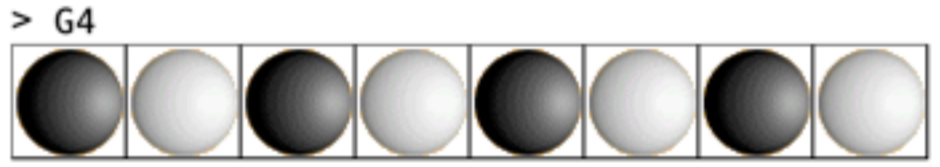
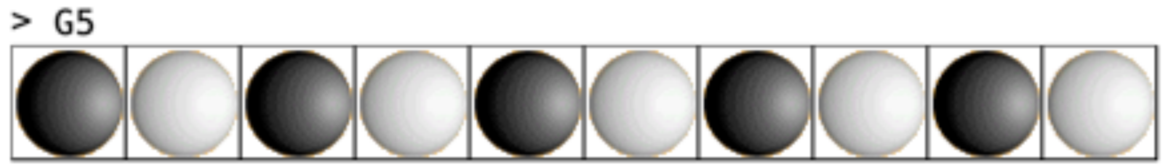
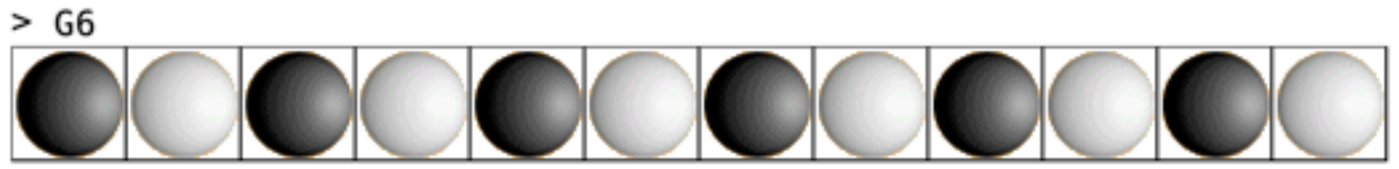


Goals of This Research

- **General** combinatorial game solver using the techniques learned from solving single games
- Answer the basic **question: Who wins game G**, if player p goes first?
- For short combinatorial games (finite, normal play)
- Efficient minimax search-based solver
    - Use **sum** game structure, **equality** and **partial ordering** of games for simplification
    - Work from first principles
    - **Avoid** computing canonical forms

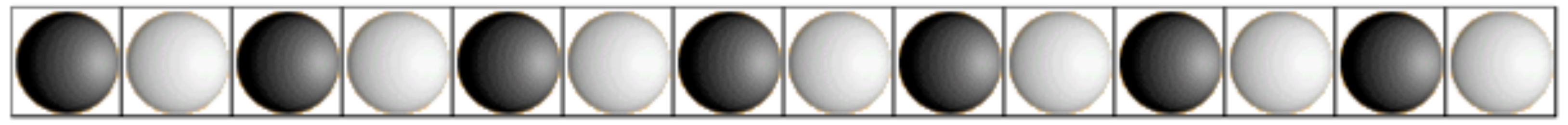
Why Avoid Canonical Forms? (1)

Canonical form of 1xn Clobber boards (computed with CGSuite)

<table>
<tr><td>
> G1</td><td>*</td></tr>
<tr><td>
> G2</td><td>+-{*,^}</td></tr>
<tr><td>
> G3</td><td>0</td></tr>
<tr><td>
> G4</td><td>+-{^[2],{^*,^^|0,^*,+-{0,^*}}}</td></tr>
<tr><td>
> G5</td><td>+-{^<2>*,^,{0|^,+-{*,^}},{^^*|v,+-{*,^}}}</td></tr>
<tr><td>
> G6</td><td>+-{0,^*}</td></tr>
</table>

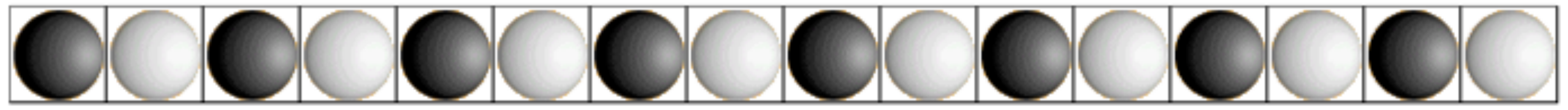
Why Avoid Canonical Forms? (2)

```
> G7
```



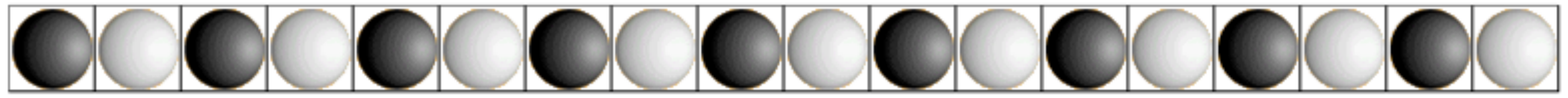
```
+-{{^[2]*,^^,{0|{0|^*,+-{0,^*}},{^*,^^|0,^*,+-{0,^*}}},{^3*|0,{^*,^^|0,^*,+-{0,^*}}}|0,{^^|0,^*},{0,{^*,^^|0,^*,+-{0,^*}}||v*,+-{0,^*}|
0},+-{^<2>,^*,{0|^*,+-{0,^*}},{^^|v*,+-{0,^*}}},{^^,{^*,^^|0,^*,+-{0,^*}}|0,v*}},{{^^*|^},{^3*|0,{^*,^^|0,^*,+-{0,^*}}}|0,*,{0,*,{^,^*|
0,v*}|vv,v*}},{0|{0|^*,+-{0,^*}},{^*,^^|0,^*,+-{0,^*}}||+-{^[2],{^*,^^|0,^*,+-{0,^*}}},{^,^^*|*,^,+-{*,^}||{0,v*,+-{0,^*}|vv,v*},+-
{0,^*}}},{0|{0|^*,+-{0,^*}},{^*,^^|0,^*,+-{0,^*}}||{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}},{0|^*,+-{0,^*}||0,{0,v*,+-{0,^*}|vv,v*}},{+-{0,^*},
{^*,^^|0,^*,+-{0,^*}}||*,v,+-{*,^}|vv*,v}},{{0|+-{0,^*}},{^^*,{^^|^*}|^,^*,{^,^[2]*|0,*,{0,^*|v,v*}},{^*,^^|0,*,{0,^*|v,v*}}}|+-{0,^*},
{*,^<2>,{^,^[2]*|0,*,{0,^*|v,v*}}|v,v*,{0,v*,{0,^*|v,v*}|vv,vv*}},+-{*,^},{+-{0,^*},{^*,^^|0,^*,+-{0,^*}}||*,v,+-{*,^}|vv*,v}}}
```

```
> G8
```



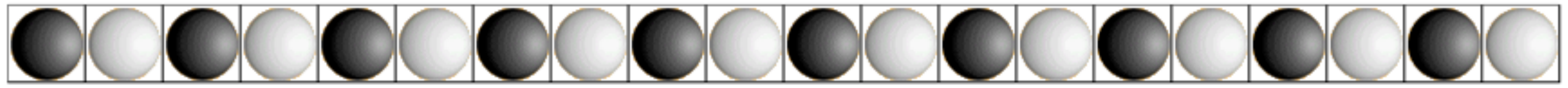
```
+-{{^,^^*|*,^,+-{*,^}},{^^*|+-{^<2>*,^,{0|^,+-{*,^}},{^^*|v,+-{*,^}}},{^[2]*,{0|^,+-
{*,^}},{^^,^^*|*,^,{^,^*|*,v}}|*,{*,v,+-{*,^}|vv*,v}}}}
```

```
> G9
```



```
+-{{{0|^,+-{*,^}},{0|{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}||^,{0|^,+-{*,^}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||*,{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}}}}|*,{{0|^,+-{*,^}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}},
{0|^*,+-{0,^*}||0,{0,v*,+-{0,^*}|vv,v*}},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}}|{{*,v,+-{*,^}|vv*,v},+-{*,^},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}||vv,v*,{0,v*,+-{0,^*}|vv,v*}|v3*,vv},{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}|||vv*,v,{*,v,+-{*,^}|vv*,v}|
v3,vv*},{0|^*,+-{0,^*}||0,{0,v*,+-{0,^*}|vv,v*}|||vv,v*,{0,v*,+-{0,^*}|vv,v*}|v3*,vv}},{0|^*,+-{0,^*}||0,v*}},{{{^^*|^},^3,{0|{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}},{^4*|^,{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}}|^,{^3|^,^^*},{^[2],^^*,{0|{0|
^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^3|*,{^,^^*|*,^,+-{*,^}}}|*,{^^*|*,^},{*,{^,^^*|*,^,+-{*,^}}||v,+-{*,^}|0},+-{^<2>*,^,{0|^,+-{*,^}},{^^*|v,+-{*,^}}},{^^*,{^,^^*|*,^,+-{*,^}}|*,v}},{^3,{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}|*,^},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}|||*,{^,^^*|*,^,
+-{*,^}}||v,+-{*,^}|0}},{{^3*|^^},{^4*|^,{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}}}|^,^*,{^,^*|*,v}},{0|{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}||{0|^,+-{*,^}},{0|^*,+-{0,^*}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}},
{0|^*,+-{0,^*}||0,{0,v*,+-{0,^*}|vv,v*}},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}},{{^,^^*|*,^,+-{*,^}},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}|+-{0,^*},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}}},{0|{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^^*,^3|^,^^*,
{^,^^*|*,^,+-{*,^}}}||{{^^|^*},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}|v<2>,+-{*,^},+-{^[2]*,{^,^^*|*,^,+-{*,^}}}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},+-{^[2]*,{^,^^*|*,^,+-{*,^}}}},{^^,^3*|^*,^^,{^*,^^|0,^*,+-{0,^*}}||{0,
{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},+-{*,^},{^,^^*|*,^,+-{*,^}}}},{{0||^,^^*|*,^,+-{*,^}},{^3*,{^3|^^*}|^^,^^*,{^^,{^^*|^}|^,^*,{^,^^*|0,*,{*,^|v,v*}},{^*,^[2]|0,*,{*,^|v,v*}}},{^^*,^3|^,^*,{^,^^*|0,*,{*,^|v,v*}}}}|{^,^^*|*,^,+-{*,^}},{^*,^[2]|0,*,{*,^|v,v*}},
{^*,^^|0,^*,+-{0,^*}},{{0|+-{*,^}},{^^,{^^*|^}|^,^*,{^,^^*|0,*,{*,^|v,v*}},{^*,^[2]|0,*,{*,^|v,v*}}}|{0,^<2>*,{^*,^[2]|0,*,{*,^|v,v*}}|v,v*,{*,v,{*,^|v,v*}|vv,vv*}},+-{0,^*},+-{*,^},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}},{{^,^^*|*,^,+-{*,^}},{^^*,^3|^,^^*,
{^,^^*|*,^,+-{*,^}}}|+-{0,^*},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}}}|+-{^<2>*,^,{0|^,+-{*,^}},{^^*|v,+-{*,^}}},{^,{^^*,^3|^,^*,{^,^^*|0,*,{*,^|v,v*}}}|v[2]*,{^,+-{*,^}|vv*}},{{^,^^*|*,^,+-{*,^}},{^*,^^|0,^*,+-{0,^*}},{^^,^^*,{^3,^3*|^,^^*,{^^,^^*|*,^,
{^,^*|*,v}}}|^*,{^^*|*,^},{^,^*,{^^,^^*|*,^,{^,^*|*,v}}|0,v<2>*,{0,*,{^,^*|*,v}|v*,v[2]}}},{^^,^3*|^*,^^,{^*,^^|0,^*,+-{0,^*}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},+-{*,^},{^,^^*|*,^,+-{*,^}}}|{0,*,{^,^*|*,v}|v*,v[2]},{+-{0,^*},{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,
+-{*,^}|vv*,v},+-{*,^}},+-{*,^},{^,^*,{^^,^^*|*,^,{^,^*|*,v}}|0,v<2>*,{0,*,{^,^*|*,v}|v*,v[2]}}|{v,v*,{0,*,{^,^*|*,v}|vv*,v},{0,*,{^,^*|*,v}|v*,v[2]}|vv,{v|vv*}},{+-{*,^}|0}},{^,^^*|*,^,+-{*,^}||0,*}},+-{{^[2],^^*,{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^3|*,{^,^^*|*,^,+-{*,^}}}|*,
{^^*|*,^},{*,{^,^^*|*,^,+-{*,^}}||v,+-{*,^}|0},+-{^<2>*,^,{0|^,+-{*,^}},{^^*|v,+-{*,^}}},{^^*,{^,^^*|*,^,+-{*,^}}|*,v}},{{^^|^*},{^3|*,{^,^^*|*,^,+-{*,^}}}|0,*,{0,*,{^,^*|*,v}|vv*,v}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},+-
{^[2]*,{^,^^*|*,^,+-{*,^}}}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}},{0|^*,+-{0,^*}||0,{0,v*,+-{0,^*}|vv,v*}},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}},{{0|+-{*,^}},{^^,{^^*|^}|^,^*,{^,^^*|0,*,{*,^|v,v*}},{^*,^[2]|0,*,{*,^|v,v*}}}|
{0,^<2>*,{^*,^[2]|0,*,{*,^|v,v*}}|v,v*,{*,v,{*,^|v,v*}|vv,vv*}},+-{0,^*},+-{*,^},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}}},{{{^^|^*},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}|v<2>,+-{*,^},+-{^[2]*,{^,^^*|*,^,+-{*,^}}}},{{^,^^*|*,^,+-{*,^}},{^^*,^3|^,^^*,{^,^^*|*,^,+-
{*,^}}}|+-{0,^*},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}}|{v,+-{*,^}|0},{+-{^[2]*,{^,^^*|*,^,+-{*,^}}},{+-{*,^},{^,^^*|*,^,+-{*,^}}|0,{0,v*,+-{0,^*}|vv,v*}}||{*,v,+-{*,^}|vv*,v},{v,+-{*,^}|0}|0}},{{^^,^3*|^*,^^,{^*,^^|0,^*,+-{0,^*}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,
+-{*,^}|vv*,v},+-{*,^}},+-{*,^},{^,^^*|*,^,+-{*,^}}}},{^^,^3*|^*,^^,{^*,^^|0,^*,+-{0,^*}}|||0,{^*,^^|0,^*,+-{0,^*}}||v*,+-{0,^*}|0},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}|||*,{^,^^*|*,^,+-{*,^}}||v,+-{*,^}|0}|{v,+-{*,^}|0},{{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},{0,
{^*,^^|0,^*,+-{0,^*}}||v*,+-{0,^*}|0},{*,{^,^^*|*,^,+-{*,^}}||v,+-{*,^}|0}||{*,v,+-{*,^}|vv*,v},{v,+-{*,^}|0}|0}}},{0|{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}||{{^^|^*},{^^*,^3|^,^^*,{^,^^*|*,^,+-{*,^}}}|v<2>,+-{*,^},+-{^[2]*,{^,^^*|*,^,+-
{*,^}}}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},+-{^[2]*,{^,^^*|*,^,+-{*,^}}}},{^^,^3*|^*,^^,{^*,^^|0,^*,+-{0,^*}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|vv*,v},+-{*,^}},+-{*,^},{^,^^*|*,^,+-{*,^}}}|||+-{0,^*},{{0|+-{0,^*}},
{{^^|^*},{^3,^3*|^,^^*,{^^,^^*|*,^,{^,^*|*,v}}}|+-{*,^},{^[2]*,{0|^,+-{*,^}},{^^,^^*|*,^,{^,^*|*,v}}|*,{*,v,+-{*,^}|vv*,v}}},{0|{0|^,+-{*,^}},{^,^^*|*,^,+-{*,^}}||*,{0|^,+-{*,^}||*,{*,v,+-{*,^}|vv*,v}}},{^^,^3*|^*,^^,{^*,^^|0,^*,+-{0,^*}}||{0,{^*,^^|0,^*,+-{0,^*}}|{*,v,+-{*,^}|
vv*,v},+-{*,^}},+-{*,^},{^,^^*|*,^,+-{*,^}}}|v,+-{*,^},{^<2>,{^,^*|*,v},{^[2]*,{0|^,+-{*,^}},{^^,^^*|*,^,{^,^*|*,v}}|*,{*,v,+-{*,^}|vv*,v}}||vv*,v,{*,v,+-{*,^}|vv*,v}|v3,vv*}}}}
```

```
> G10
```



[illegible]

Why Avoid Canonical Forms? (3)

- We just want to solve for win/loss
- Computing canonical forms of all intermediate positions incurs a huge overhead
- Example: 1 ×16 empty NoGo board
    - Result in CGSuite: massive canonical form, 1201194 stops, takes several minutes
    - 1x17 in CGSuite: Java heapspace overflow
    - Solver from (Du et al 2024), from scratch without using precomputed database:
        - Solves 1x16 in less than 30 milliseconds, scales much further

Why use Minimax Search ?
Relation to Outcome Classes in CGT

- Two searches, one with each player (Left, Right) going first
- Solve yes/no question: Can the first player win?
- The total 2x2=4 results determine the outcome class
  - Black can always win, no matter who goes first: $G > 0$, in class L
  - White can always win: $G < 0$, in class R
  - First player wins, $G \ngtrless 0$, in class N
  - Second player wins: $G = 0$, in class P

The Many Uses of Search in Solving Games

- Not just to directly solve games
- Many other uses make overall solution more efficient
- Prove if games are equal: $G = H$ iff $G - H = 0$, is second—player win
- Prove that two subgames are inverses and cancel: $G + H = 0$
- Dominated options: to show $G \geq H$,
  solve $G - H \geq 0$ by a **single** search: Right going first loses
- Replace subgame in sum by simplest equal game
- **Main Challenge**: **when** to try such searches for simplification?

MCGS - A Minimax-based Combinatorial Game Solver

- **Version 1 released!**
- Game-independent, for all short games
- Basic framework, not many optimisations so far
- Sample games implemented: linear Clobber, linear NoGo, Nim, Elephants and Rhinos
- Abstract games: integer, dyadic rational, nimber, up-star, switch
- Documentation and extensive test framework, simple file format
- Version 2 will add support for building and using databases of small games

Approach and Techniques Used..

..(or planned)..

- Abstract **game** class, specific games extend game, e.g. implement rules, decomposition
- Stack-based **sum game** data structure
- Efficient **play, undo move** on both subgame and sum level
- Supports game-specific rules to **split** game into more subgames
- (future) Simplify sumgame
- (future) Remove zeroes, and pairs of game+inverse
- (future) Replace game by simplest equal game
- (future) Static evaluation for early win/loss detection, use bounds on game values

Example - How to Add your Own Game to MCGS

```
class elephants : public strip
   elephants(game_as_string);
   void play(move m, bw to_play);
   void undo_move();
   split_result split_implementation();
   move_generator* create_move_generator(bw to_play);
   void print(ostream);
   game* inverse();
```

Future Plans

- Version 2: add database, optimisations
- Future Versions and extensions
    - Many more optimisations
    - Support thermographs
    - Add specialised algorithms for impartial games
    - Implement more games including Clobber, NoGo on a grid

Thank you!