# Solving Two-player Games with QBF Solvers in General Game Playing

**Yifan He**
Abdallah Saffidine
Michael Thielscher

UNSW Sydney, Australia

# General Game Playing

## Game Playing AI

- Deep Blue (1996), AlphaGo (2016)
- AlphaZero (2018)

# General Game Playing

## Game Playing AI

- Deep Blue (1996), AlphaGo (2016)
- AlphaZero (2018)
- Cannot play **all** perfect information games

# General Game Playing

## Game Playing AI

- Deep Blue (1996), AlphaGo (2016)
- AlphaZero (2018)
- Cannot play **all** perfect information games

## General Game Playing Challenge

- Game Description Language GDL (2006)
  - Logic language similar to Prolog
  - $next(cell(X,Y,P)) \;:-\; true(cell(X,Y,blank)),\; does(P, mark(X,Y)).$
- Some successful players
  - FluxPlayer (2006), CadiaPlayer (2007), GAZ (2020)

# General Game Playing

## Game Playing AI

- Deep Blue (1996), AlphaGo (2016)
- AlphaZero (2018)
- Cannot play **all** perfect information games

## General Game Playing Challenge

- Game Description Language GDL (2006)
  - Logic language similar to Prolog
  - $next(cell(X,Y,P))\ :-\ true(cell(X,Y,blank)),\ does(P,mark(X,Y)).$
- Some successful players
  - FluxPlayer (2006), CadiaPlayer (2007), GAZ (2020)
- Play well, not solve

# Game Solving (with logic) in GGP

**Solving 1-player game with ASP (Thielscher, 2009)**

- Can the player win the game within $T_{max}$ steps.

# Game Solving (with logic) in GGP

## Solving 1-player game with ASP (Thielscher, 2009)

- Can the player win the game within $T_{max}$ steps.
- Convert GDL $G$ to Time-extended ASP $Ext(G)$

$$next(cell(X,Y,P)) \;:-\; true(cell(X,Y,blank)),\; does(P,mark(X,Y)).$$

$$true(cell(X,Y,P),T+1) :- true(cell(X,Y,blank),T), time(T),$$
$$does(P,mark(X,Y),T).$$

# Game Solving (with logic) in GGP

## Solving 1-player game with ASP (Thielscher, 2009)

- Can the player win the game within $T_{max}$ steps.
- Convert GDL $G$ to Time-extended ASP $Ext(G)$

$$next(cell(X,Y,P)) \;:-\; true(cell(X,Y,blank)),\; does(P,mark(X,Y)).$$

$$true(cell(X,Y,P),T+1) :- true(cell(X,Y,blank),T), time(T),$$
$$does(P,mark(X,Y),T).$$

- Additional ASP clauses $P$
  - 1 legal move per step before termination
  - The player must reach terminal within $T_{max}$ steps
  - The player must achieve its goal when termination

# Game Solving (with logic) in GGP

## Solving 1-player game with ASP (Thielscher, 2009)

- Can the player win the game within $T_{max}$ steps.
- Convert GDL $G$ to Time-extended ASP $Ext(G)$

$$next(cell(X,Y,P)) \ :- \ true(cell(X,Y,blank)),\ does(P,mark(X,Y)).$$

$$true(cell(X,Y,P),T+1) :- true(cell(X,Y,blank),T), time(T),$$
$$does(P,mark(X,Y),T).$$

- Additional ASP clauses $P$
  - 1 legal move per step before termination
  - The player must reach terminal within $T_{max}$ steps
  - The player must achieve its goal when termination
- Use ASP planner Clingo to solve $Ext(G) \cup P$
- ASP approach is comparable to forward search

# Solving Games with QBF Solvers

## Two-player Zero-sum Turn-taking games

- Chess, Go, Connect-4, Generalized Tic-Tac-Toe, Breakthrough, Dots and Boxes...

# Solving Games with QBF Solvers

## Two-player Zero-sum Turn-taking games

- Chess, Go, Connect-4, Generalized Tic-Tac-Toe, Breakthrough, Dots and Boxes...
- Encode to Quantified Boolean Formula
  - Example: $\exists a.\forall b.\exists c.\ (a \vee \neg b) \wedge (c \vee \neg a)$

# Solving Games with QBF Solvers

## Two-player Zero-sum Turn-taking games

- Chess, Go, Connect-4, Generalized Tic-Tac-Toe, Breakthrough, Dots and Boxes...
- Encode to Quantified Boolean Formula
  - Example: $\exists a.\forall b.\exists c.\ (a \vee \neg b) \wedge (c \vee \neg a)$
  - Connect-4 (Gent, 2003)
  - Generalized Tic-Tac-Toe (Diptarama et al., 2016)
  - Positional board games (Saffidine et al., 2020)
  - Positional + some non-positional board games in BDDL (Shaik et al., 2023)
- QBF method outperforms Proof Number Search in Generalized Tic-Tac-Toe

# Our Work

## Motivation

- Solving 1-p games with ASP works well in GGP, and solving 2-p games with QBF is promising
- Solving 2-p games with QBF in GGP is natural

## Overall approach

- Encode GDL to QBF such that the QBF is true iff player 1 can force a win within $T_{max}$ steps
- GDL $\xrightarrow{\textit{Directly}}$ QBF ✗
  - GDL stable model vs. QBF classical model
- **GDL** $\Longrightarrow$ **QASP** $\xrightarrow{\textit{Converter}}$ QBF $\xrightarrow{\textit{QBF Solver}}$ W/L
  - GDL stable model, QASP stable model
  - QASP to QBF (Fandinno et al., 2021)

# QASP Review

$P$ is a logic program with ground atoms **A**.

$$\psi = Q_1\ X_1.Q_2\ X_2.\dots Q_n\ X_n.\ \ P \qquad Q_i \in \{\exists, \forall\}$$

- Example: $\forall x.\ \{\{x\}.\quad :-a.\quad a:-x.\}.$
- Satisfiable if and only if both programs have a stable model.
  - $\{\{x\}.\quad :-a.\quad a:-x.\quad :-not\ x.\}.$ ✗
  - $\{\{x\}.\quad :-a.\quad a:-x.\quad :-\ x.\}.$ Stable model: $\{\}$

# QASP Review

$P$ is a logic program with ground atoms **A**.

$$\psi = Q_1\ X_1.Q_2\ X_2.\dots Q_n\ X_n.\ \ P \qquad Q_i \in \{\exists, \forall\}$$

- Example: $\exists x.\ \{\{x\}.\quad :-a.\quad a:-x.\}.$
- Satisfiable iff either of the program have a stable model.
  - $\{\{x\}.\quad :-a.\quad a:-x.\quad :-not\ x.\}.$ ✗
  - $\{\{x\}.\quad :-a.\quad a:-x.\quad :-\ x.\}.$ Stable model: $\{\}$

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn
- Main theoretical result:
  - **Q** $Ext(G) \cup P$ is satisfiable iff player 1 can win within $T_{max}$ steps

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn
- (2) and (3) looks similar?
  - They are handled quite differently

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
     - If player 1 makes illegal moves or makes 0 or 2+ moves per step, $Ext(G) \cup P$ is falsified immediately
  3. Before the game terminates player 2 must make a legal move per turn

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
     - If player 1 makes illegal moves or makes 0 or 2+ moves per step, $Ext(G) \cup P$ is falsified immediately
  3. Before the game terminates player 2 must make a legal move per turn
     - If player 2 makes illegal moves or makes 0 or 2+ moves per step, $Ext(G) \cup P$ is falsified immediately ✗

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
     - If player 1 makes illegal moves or makes 0 or 2+ moves per step, $Ext(G) \cup P$ is falsified immediately
  3. Before the game terminates player 2 must make a legal move per turn
     - If player 2 makes illegal moves or makes 0 or 2+ moves per step, $Ext(G) \cup P$ is falsified immediately ✗
     - Player 2 cannot force $Ext(G) \cup P$ to be false by:
     - making illegal moves,
     - or making 0 or 2+ moves

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints:
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints:
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn

  - One possible way: create cheating variables: QBF encoding of Connect-4 (Gent, 2003)
  - Intuition: If player 2 is not making exactly 1 legal move, player 2 cheats
  - Player 1 wins if and only if it wins or player 2 cheats
  - Not very efficient

# GDL to QASP

- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints:
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn

# GDL to QASP

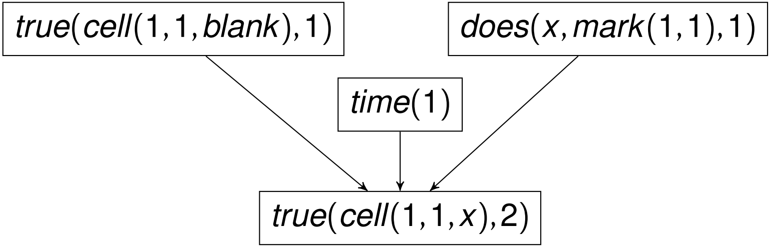
- Convert the game $G$ to $Ext(G)$
- Use an ASP $P$ to model the Constraints:
  1. The game must terminate within $T_{max}$ steps, and when the game terminates, player 1 wins
  2. Before the game terminates player 1 must make a legal move per turn
  3. Before the game terminates player 2 must make a legal move per turn
  - Logarithmic encoding in positional games (Saffidine et al., 2020)
  - Example: use 3 bits to represent 8 possible actions
    - $\forall b_0 b_1 b_2.$
    - $b_0 = \bot; b_1 = \bot; b_2 = \bot \rightarrow player\_2\_action(1)$
    - $b_0 = \top; b_1 = \bot; b_2 = \bot \rightarrow player\_2\_action(2)$
    - $\dots$
    - $b_0 = \bot; b_1 = \top; b_2 = \top \rightarrow player\_2\_action(7)$
    - $b_0 = \top; b_1 = \top; b_2 = \top \rightarrow player\_2\_action(8)$

# GDL to QASP (cont.)

- Final step: add quantifiers to $Ext(G) \cup P$

# GDL to QASP (cont.)

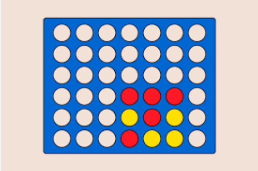
- Final step: add quantifiers to $Ext(G) \cup P$
- Quantify each atom as early as possible, based on atom dependency of $Ext(G) \cup P$
  - Example $true(cell(1,1,x),2) :- true(cell(1,1,blank),1), time(1), does(x, mark(1,1),1).$

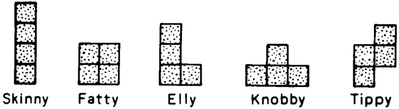


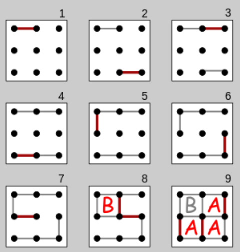

  - `true(cell(1,1,x),2)` should be quantified no earlier than `time(1)`, `does(x,mark(1,1),1)`, and `true(cell(1,1,blank),1)`.

# Experiments

- Connect-4, Breakthrough, Generalized Tic-Tac-Toe, Dots and Boxes



# Experiments

- Connect-4, Breakthrough, Generalized Tic-Tac-Toe, Dots and Boxes

# Experiments

- Connect-4, Breakthrough, Generalized Tic-Tac-Toe, Dots and Boxes

1. Convert GDL games at a certain depth to QBF
   - QBF solver DepQBF and Caqe + bloqqer preprocessor
2. Minimax + Transposition table solver in C++

# Experiments

- Connect-4, Breakthrough, Generalized Tic-Tac-Toe, Dots and Boxes

1. Convert GDL games at a certain depth to QBF
   - QBF solver DepQBF and Caqe + bloqqer preprocessor
2. Minimax + Transposition table solver in C++

- Iterative increase depth $T_{max}$, we record
  - Solving time (time limit 1000s)
  - $T_{max}$: depth of the game
    - maximum depth at least one method can solve
    - Red: the first player winnable within $T_{max}$ steps
    - Blue: the first player cannot win at depth $T_{max}$
  - $\mu_G$ length of the longest playing sequence that the first player wins

# Experiments Results

| Game | Config | $\mu_G$ | $T_{max}$ | DepQBF | Caqe | Minx |
|---|---|---|---|---|---|---|
| Connect-4 | 4×4 | 15 | 15 | 1.48 | **1.21** | 1.42 |
| | 5×5 | 25 | 21 | 372.85 | **137.77** | 517.50 |
| | 6×6 | 35 | 19 | * | **597.56** | * |
| GTTT-1-1 | elly | 15 | 7 | 6.91 | **4.38** | 9.75 |
| | fat. | 15 | 15 | **204.11** | 411.91 | 307.38 |
| | knob. | 15 | 15 | **379.34** | 705.57 | * |
| | skin. | 15 | 15 | 394.47 | * | **206.59** |
| | tip. | 15 | 9 | 16.99 | **8.42** | 30.94 |
| GTTT-2-2 | fat. | 14 | 14 | **171.36** | 313.55 | * |
| | skin. | 14 | 14 | **390.11** | 548.99 | 662.32 |
| Breakthrough | 2×5 | 21 | 21 | 6.66 | 5.95 | **0.36** |
| | 2×6 | 29 | 15 | 12.49 | 11.78 | **2.86** |
| | 3×4 | 19 | 19 | 9.98 | 9.50 | **1.09** |
| | 3×5 | 31 | 19 | * | 847.31 | **92.41** |
| | 4×4 | 25 | 25 | 159.73 | **69.63** | 106.20 |
| D&B | 2×2 | 12 | 12 | 6.70 | 6.46 | **0.63** |
| | 2×3 | 17 | 17 | * | 605.09 | **15.06** |

- Both Caqe and DepQBF can solve most instances to a reasonable depth
- QBF is comparable with Minimax search

# Summary and Future Work

## Contribution

- Convert from 2-player games in GDL to QBF
- Comprable with forward search in some games
  - Inline with 1-player games while generalizing it to 2-player zero-sum games
- Strong winnability of multi-player games

## Future Work

- Embed the translation into a GGP player
- Obtain a smaller encoding
  - Our encoding size proportional to $O(A \cdot T_{max})$
  - Lifted-encoding technique used in BDDL to QBF $O(log(Board\_Size) \cdot T_{max})$ (Shaik et al., 2023)