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

Yifan He[1]
Abdallah Saffidine [1]
Michael Thielscher [1]

[1] 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)
- Can an AI play **all** perfect information games?

General Game Playing

Game Playing AI

- Deep Blue (1996), AlphaGo (2016)
- AlphaZero (2018)
- Can an AI play **all** perfect information games?

General Game Playing Challenge

- Rules described in Game Description Language (GDL)
  - $\texttt{next}(\textit{cell}(X, Y, P))\ \texttt{:-}\ \texttt{true}(\textit{cell}(X, Y, \textit{blank})),\ \texttt{does}(P, \textit{mark}(X, Y)).$
- FluxPlayer, CadiaPlayer, Ary, GGPZero...

General Game Playing

Game Playing AI

- Deep Blue (1996), AlphaGo (2016)
- AlphaZero (2018)
- Can an AI play **all** perfect information games?

General Game Playing Challenge

- Rules described in Game Description Language (GDL)
  - $\texttt{next}(\textit{cell}(X, Y, P)) \texttt{ :- } \texttt{true}(\textit{cell}(X, Y, \textit{blank})), \texttt{ does}(P, \textit{mark}(X, Y)).$
- FluxPlayer, CadiaPlayer, Ary, GGPZero...
- Play well not play perfectly

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 $Ext(G)$

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

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 $Ext(G)$

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

- An additional ASP program P
  - 1 legal move per step before termination
    - $1\ \{does(r,M,T)\ :\ move(M)\}\ 1\ :-\ not\ terminated(T).$
    - $:-not\ legal(r,M,T),\ does(r,M,T).$
  - 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 $Ext(G)$

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

- An additional ASP program P
  - 1 legal move per step before termination
    - $1\ \{does(r,M,T)\ :\ move(M)\}\ 1\ :-\ not\ terminated(T).$
    - $:-not\ legal(r,M,T),\ does(r,M,T).$
  - 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 > forward search in some games

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
  - 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
- Solving 2-p games with QBF is promising

Overall approach

- Can x win a 2-player game within T_{max} steps no matter what o does
- GDL $\xrightarrow{Directly}$ QBF ✗
  - GDL stable model VS QBF classical model
- **GDL** $\Longrightarrow$ **QASP** $\xrightarrow{Converter}$ QBF $\xrightarrow{QBF\ \ Solver}$ 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**.

$$Q_1\ X_1\ \ldots\ Q_n\ X_n\ \ P$$

$Q_i \in \{\exists, \forall\}$. $\mathit{fix}(X, Y)$, where $Y \subseteq X \subseteq \mathbf{A}$, as the logic program $\{:\!-\ \mathit{not}\ x.\mid x \in Y\} \cup \{:\!-\ x.\mid x \in X \setminus Y\}$.

1. P is satisfiable iff it has a stable model.
2. If the QASP has form $\exists\ X\ \mathbf{Q}\ P$ (resp. $\forall\ X\ \mathbf{Q}\ P$), the program is satisfiable iff **there exists** (resp. **for all**) $Y \subseteq X$ such that the program $\mathbf{Q}\ (P \cup \mathit{fix}(X, Y))$ is satisfiable.

GDL to QASP

- Convert the game G to $Ext(G)$
- Use an ASP P to model the Constraints:
  1. The game must terminates within T_{max} steps
  2. When the game terminates, player 1 wins
  3. Before the game terminates player 1 must make a legal move per turn
  4. 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 terminates within T_{max} steps
  2. When the game terminates, player 1 wins
  3. Before the game terminates player 1 must make a legal move per turn
  4. Before the game terminates player 2 must make a legal move per turn
     - logarithmic encoding in positional games
     - Actions of the player 2: $does(o, mark(1..3, 1..3), T)$.
     - Introduce: $moveL(o, 0..3, T)$

$$\{moveL(o, 0..3, T)\} :- time(T).$$

$$does(o, mark(1,1), T) :- moveL(o, 0, T),\ not\ moveL(o, 1, T),$$
$$not\ moveL(o, 2, T), not\ moveL(o, 3, T), legal(o, mark(1,1), T).$$

GDL to QASP

- Convert the game G to $Ext(G)$
- Use an ASP P to model the Constraints:
  1. The game must terminates within T_{max} steps
  2. When the game terminates, player 1 wins
  3. Before the game terminates player 1 must make a legal move per turn
  4. Before the game terminates player 2 must make a legal move per turn
- All the stable models of $Ext(G) \cup P$ corresponds to a playing sequence of G
  - Terminates within T_{max} steps
  - The first player wins

GDL to QASP

- All the stable models of $Ext(G) \cup P$ corresponds to a playing sequence of G
  - Terminates within T_{max} steps
  - The first player wins

GDL to QASP

- All the stable models of $Ext(G) \cup P$ corresponds to a playing sequence of G
  - Terminates within T_{max} steps
  - The first player wins
- Add quantifiers to $Ext(G) \cup P$
  - A quantifier prefix **Q**, **Q** $(Ext(G) \cup P)$ is satisfiable iff the game G is a win for player 1 within T_{max} steps.

GDL to QASP

- All the stable models of $Ext(G) \cup P$ corresponds to a playing sequence of G
  - Terminates within T_{max} steps
  - The first player wins
- Add quantifiers to $Ext(G) \cup P$
  - A quantifier prefix **Q**, **Q** $(Ext(G) \cup P)$ is satisfiable iff the game G is a win for player 1 within T_{max} steps.

$$\exists does(x, M_1, 1), does(x, M_2, 1), ..., does(x, M_n, 1)$$

$$\forall moveL(o, 1, 1), moveL(1, 2, 1), ..., moveL(o, K, 1)$$

$$\ldots$$

$$\exists does(x, M_1, T_{max}), does(x, M_2, T_{max}), ..., does(x, M_n, T_{max})$$

$$\forall moveL(o, 1, T_{max}), moveL(1, 2, T_{max}), ..., moveL(o, K, T_{max})$$

GDL to QASP

- All the stable models of $Ext(G) \cup P$ corresponds to a playing sequence of G
  - Terminates within T_{max} steps
  - The first player wins
- Add quantifiers to $Ext(G) \cup P$
  - A quantifier prefix **Q**, **Q** $(Ext(G) \cup P)$ is satisfiable iff the game G is a win for player 1 within T_{max} steps.

    $\exists does(x, M_1, 1), does(x, M_2, 1), ..., does(x, M_n, 1)$

    $\forall moveL(o, 1, 1), moveL(1, 2, 1), ..., moveL(o, K, 1)$

    ...

    $\exists does(x, M_1, T_{max}), does(x, M_2, T_{max}), ..., does(x, M_n, T_{max})$

    $\forall moveL(o, 1, T_{max}), moveL(1, 2, T_{max}), ..., moveL(o, K, T_{max})$
  - Quantify remaining atoms (e.g., $legal(x, M, T)$) as early as possible, based on atom dependency
    - Example: $a :- b,\ c.$
    - a is quantified no earlier than b and c.

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++
- We record
  - μ_G length of the longest playing sequence that the first player wins
  - T_{max} the depth of the game
    - Red: the first player winnable within T_{max} steps
    - Blue: the first player cannot win at any depth
    - Orange: the first player cannot win within $T_{max} < \mu_G$ depth
  - Solving time of DepQBF, Caqe, and Minimax
    - Time limit 1000s/game

Experiments

Game	Config	μ_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
- Outperforms 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

- Obtain a smaller encoding
  - Lifted encoding technique (Shaik et al., 2023)
- Embed the translation into a GGP player