HyperPlay: A Solution to General Game Playing with Imperfect Information

M. Schofield, T. Cerexhe and M. Thielscher

Karpathiotaki Maria
M1279

karpathiotaki.maria@gmail.com
Intelligent agents that can automatically learn how to skillfully play a wide variety of games, given only the descriptions of the game rules.

Agents have to learn diverse game-playing strategies without any game-specific knowledge being provided by their developers.

Relevant game-specific knowledge required for expert-level play, must be effectively discovered during play!!
Players don’t know exactly the actions chosen by other players.

They know who the other players are, what their possible strategies/actions are, and the preferences/payoffs of these other players.

Hence, information about the other players is imperfect.

E.g.: Card games, like bridge and poker.
A first-order logic based language (variant of Datalog) for defining discrete games with **complete information**!

The expressiveness of GDL allows a large range of deterministic, perfect information, simultaneous-move games to be described, with any number of adversary or cooperating players.

- **Turn-based** games are modeled by having the players who do not have a turn return a special no operation move.

```
role(?r)       ?r is a player
init(?f)       ?f holds in the initial position
true(?f)       ?f holds in the current position
legal(?r, ?m)  ?r can do ?m in the current position
does(?r, ?m)   player ?r does move ?m
next(?f)       ?f holds in the next position
terminal       the current position is terminal
goal(?r, ?v)    ?r gets payoff ?v
```
GDL-II: GDL with incomplete/imperfect information

- GDL has recently been extended.
- Two new keywords to describe arbitrary (finite) games with randomized moves and imperfect information:
  - `sees`
  - `random`

<table>
<thead>
<tr>
<th>role(?r)</th>
<th>?r is a player</th>
</tr>
</thead>
<tbody>
<tr>
<td>init(?f)</td>
<td>?f holds in the initial position</td>
</tr>
<tr>
<td>true(?f)</td>
<td>?f holds in the current position</td>
</tr>
<tr>
<td>legal(?r, ?m)</td>
<td>?r can do ?m in the current position</td>
</tr>
<tr>
<td>does(?r, ?m)</td>
<td>player ?r does move ?m</td>
</tr>
<tr>
<td>next(?f)</td>
<td>?f holds in the next position</td>
</tr>
<tr>
<td>terminal</td>
<td>the current position is terminal</td>
</tr>
<tr>
<td>goal(?r, ?v)</td>
<td>?r gets payoff ?v</td>
</tr>
<tr>
<td>sees(?r, ?p)</td>
<td>?r perceives ?p in the next position</td>
</tr>
<tr>
<td>random</td>
<td>the random player (aka. Nature)</td>
</tr>
</tbody>
</table>
General approach that can be used by any general game player.

The intuition is to translate imperfect-information games into a format suitable for simpler, perfect-information players.
We maintain a bag $H$ of HyperGames (random samples or “guesses” of the current true game state).

In each round $n$, a perfect-information player can select a next move $a_n$ suitable for each of these isolated models.

Our move selection is then submitted to the game controller and a new set of percepts in for this round is received.

Each model $M$ in our bag of samples $H$ is then propagated forward to reflect the deeper game tree.

If we select a path that the last percepts reveal to be impossible, then we reach a state where no consistent joint move can be found, so:

- we backtrack by adding the guilty move vector to a set of bad moves for that state,
- and we call forward on this earlier game node, effectively undoing the move and attempting to push forward again.

This process repeats until a consistent model is found for the current round.
The HyperPlay algorithm is **agnostic** of the move selection process.

Move selection should be based on:

- the *expected* value of a move in a HyperGame
- the *propability* that the HyperGame is the true game
Several games were selected:
  • Monty Hall,
  • Krieg-TicTacToe,
  • Blind BreakThrough

The HyperPlayer opposed a Cheat, a HyperPlayer with access to the true game, and fully resourced so that it made the best move choices within the limitations of the move selection process.
  • The method for calculating the Cheat’s resources was to play one Cheat against another Cheat with different resources.
Results

- The results showed a successful implementation of the HyperPlay technique for playing imperfect information games.

- The collection of models:
  - can be very accurate,
  - is a credible substitute for perfect information about the true game,
  - can be competitive even against a Cheat.
References


Thank you!
HyperPlay: The Algorithm

```
procedure main()
begin
  \mathcal{H} := \{\langle 0 \rangle, \ldots, \langle 0 \rangle\}
  n := 1
repeat
  a_n := select_move(\mathcal{H})
  I_n := submit_move(a_n)
  for all M \in \mathcal{H} do
    forward(M, n + 1)
  n := n + 1
until end_of_game
end

procedure forward(M = \langle B_1, \vec{m}_1, \ldots, B_{k-1}, \vec{m}_{k-1}, B_k \rangle, n)
begin
  if k < n then
    if choose \vec{m} \in \mathcal{L}(M) \setminus B_k
      with \vec{m}|_{I_r} = a_k \& \& I(\vec{m}, M) = I_k
        then
          M := M \oplus \vec{m}
          forward(M, n)
        else
          backtrack(M, n)
  else
    backtrack(M, n)
end

procedure backtrack(\langle B_1, \vec{m}_1, \ldots, B_{k-1}, \vec{m}_{k-1}, B_k \rangle, n)
begin
  B_{k-1} := B_{k-1} \cup \{\vec{m}_{k-1}\}
  forward(\langle B_1, \vec{m}_1, \ldots, B_{k-1} \rangle, n)
end

HyperPlay [Schofield, 2012].
\[ P(HG_i | \text{Percepts}) = \frac{P(\text{Percepts} | HG_i) \cdot P(HG_i)}{P(\text{Percepts})} \]

\[ P(HG_i | \text{Percepts}) \sim P(HG_i) \]
Move Selection

- \( \text{ChoiceFactor}_i = \prod_j \text{Choices}_{i,j} \)

- \( P(HG_i) = \frac{1}{\text{ChoiceFactor}_i} \cdot \frac{1}{\sum_n 1/ \text{ChoiceFactor}_n} \)

- \( E(\text{Move}_j) = \sum \text{E}(\text{Move}_{i,j}) \cdot P(HG_i) \)