KR-Techniques for General Game Playing

Michael Thielscher
Roadmap

1. General Game Playing – a Grand AI Challenge

2. KR-Aspects
   - Formalizing game rules:
     Compact representations of state machines
   - **Challenge I:**
     Mapping game descriptions to efficient representations
   - Extracting useful knowledge from game descriptions
   - **Challenge II:**
     Proving properties of games

3. Further Aspects: Search + Learning
The Turk (18th Century)
Alan Turing & Claude Shannon (~1950)
Deep-Blue Beats World Champion (1997)
In the early days, game playing machines were considered a key to Artificial Intelligence (AI).

But chess computers are highly specialized systems. Deep-Blue's intelligence was limited. It couldn't even play a decent game of Tic-Tac-Toe or Rock-Paper-Scissors.

A General Game Player is a system that
• understands formal descriptions of arbitrary strategy games
• learns to play these games well without human intervention
Rather than being concerned with a specialized solution to a narrow problem, General Game Playing encompasses a variety of AI areas.
## General Game Playing and AI

<table>
<thead>
<tr>
<th>Agents</th>
<th>Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competitive environments</td>
<td>Deterministic, complete information</td>
</tr>
<tr>
<td>Uncertain environments</td>
<td>Nondeterministic, partially observable</td>
</tr>
<tr>
<td>Unknown environment model</td>
<td>Rules partially unknown</td>
</tr>
<tr>
<td>Real-world environments</td>
<td>Robotic player</td>
</tr>
</tbody>
</table>

- **Agents**
  - Competitive environments
  - Uncertain environments
  - Unknown environment model
  - Real-world environments

- **Games**
  - Deterministic, complete information
  - Nondeterministic, partially observable
  - Rules partially unknown
  - Robotic player
Knowledge Representation for Games
-
The Game Description Language
Games as State Machines
Initial Position and End of Game
Simultaneous Moves

Diagram of a network where nodes are labeled with letters (a, b, c, d, e, f, g, h, i, j, k) and the edges are labeled with a/b or b/a, indicating the possible moves between nodes.
Every finite game can be modeled as a state transition system

But direct encoding impossible in practice

19,683 states

\(~ 10^{43}\) legal positions
Modular State Representation: Fluents

\[ \text{cell}(X, Y, M) \]

\[ X, Y \in \{1, 2, 3\} \]
\[ M \in \{x, o, b\} \]

\[ \text{control}(P) \]
\[ P \in \{xplayer, oplayer\} \]
Actions

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
</tbody>
</table>
```

- `mark(X, Y)`
- `x, y ∈ {1, 2, 3}`
- `noop`
### Tic-Tac-Toe Game Model

<table>
<thead>
<tr>
<th>Symbolic expressions:</th>
<th><code>{xplayer, oplayer, cell(1, 1, b),noop,...}</code></th>
</tr>
</thead>
</table>

- **roles** `{xplayer, oplayer}`
- **initial** $s_1 = \{\text{cell}(1, 1, b), \ldots, \text{cell}(3, 3, b), \text{control}(oplayer)\}$
- **legal actions** \{\{(xplayer, mark(1, 1), s_1), \ldots, (oplayer, noop, s_1), \ldots\}\}
- **update** $\langle\langle xplayer \mapsto \text{mark}(1, 1), oplayer \mapsto \text{noop}\rangle, s_1\rangle$
  $\mapsto \langle\{\text{cell}(1, 1, x), \ldots, (\text{cell}(3, 3, b), \text{control}(oplayer)\rangle, \ldots\}$
- **terminals** $t_1 = \{\text{cell}(1, 1, x), \text{cell}(1, 2, x), \text{cell}(1, 3, x), \ldots\}, \ldots$}
- **goal** \{(xplayer, $t_1$, 100), (oplayer, $t_1$, 0), \ldots\}
Symbolic Game Model

Let \( \Sigma \) be a countable set of ground expressions.

A game is a structure \((R, l, u, s_1, t, g)\)

- \( R \in 2^\Sigma \) roles
- \( l \subseteq R \times \Sigma \times 2^\Sigma \) legal actions
- \( u: (R \mapsto \Sigma) \times 2^\Sigma \mapsto 2^\Sigma \) update
- \( s_1 \in 2^\Sigma \) initial position
- \( t \subseteq 2^\Sigma \) terminal positions
- \( g \subseteq R \times 2^\Sigma \times \mathbb{N} \) goal relation

where \( 2^\Sigma := \) finite subsets of \( \Sigma \)
A game description is a **stratified, allowed logic program** whose signature includes the following game-independent vocabulary:

- `role(player)`
- `init(fluent)`
- `true(fluent)`
- `does(player, move)`
- `next(fluent)`
- `legal(player, move)`
- `goal(player, value)`
- `terminal`
Describing a Game: Roles

A GDL description $P$ encodes the roles $R = \{\sigma \in \Sigma : P \models \text{role}(\sigma)\}$

$\text{role}(\text{xplayer}) \leq$

$\text{role}(\text{oplayer}) \leq$
Describing a Game: Initial Position

A GDL description $P$ encodes $s_1 = \{\sigma \in \Sigma : P \vdash \text{init}(\sigma)\}$

\[
\begin{align*}
\text{init}(\text{cell}(1,1,b)) & \leq \\
\text{init}(\text{cell}(1,2,b)) & \leq \\
\text{init}(\text{cell}(1,3,b)) & \leq \\
\text{init}(\text{cell}(2,1,b)) & \leq \\
\text{init}(\text{cell}(2,2,b)) & \leq \\
\text{init}(\text{cell}(2,3,b)) & \leq \\
\text{init}(\text{cell}(3,1,b)) & \leq \\
\text{init}(\text{cell}(3,2,b)) & \leq \\
\text{init}(\text{cell}(3,3,b)) & \leq \\
\text{init}(\text{control}(xplayer)) & \leq
\end{align*}
\]
Preconditions

For $S \subseteq \Sigma$ let $S^{\text{true}} := \{\text{true}(\sigma) : \sigma \in S\}$
then $P$ encodes $l = \{(r, \sigma, S) : P \cup S^{\text{true}} \models \text{legal}(r, \sigma)\}$

$\text{legal}(P, \text{mark}(X,Y)) \iff \text{true}(\text{cell}(X,Y,b)) \land \text{true}(\text{control}(P))$

$\text{legal}(\text{xplayer}, \text{noop}) \iff \text{true}(\text{cell}(X,Y,b)) \land \text{true}(\text{control}(\text{oplayer}))$

$\text{legal}(\text{oplayer}, \text{noop}) \iff \text{true}(\text{cell}(X,Y,b)) \land \text{true}(\text{control}(\text{xplayer}))$
Update

For $A : R \leftrightarrow \Sigma$ let $A^{\text{does}} := \{\text{does}(r, A(r)) : r \in R\}$
then $P$ encodes $u(A, S) = \{\sigma : P \cup A^{\text{does}} \cup S^{\text{true}} \models \text{next}(\sigma)\}$

\[
\text{next} (\text{cell}(M,N,x)) \leq \text{does} (\text{xplayer}, \text{mark}(M,N)) \\
\text{next} (\text{cell}(M,N,o)) \leq \text{does} (\text{oplayer}, \text{mark}(M,N)) \\
\text{next} (\text{cell}(M,N,W)) \leq \text{true} (\text{cell}(M,N,W)) \land \neg W=b \\
\text{next} (\text{cell}(M,N,b)) \leq \text{true} (\text{cell}(M,N,b)) \land \text{does} (P, \text{mark}(J,K)) \land (\neg M=J \lor \neg N=K) \\
\text{next} (\text{control} (\text{xplayer})) \leq \text{true} (\text{control} (\text{oplayer})) \\
\text{next} (\text{control} (\text{oplayer})) \leq \text{true} (\text{control} (\text{xplayer}))
\]
Termination

\[ P \text{ encodes } t = \{ S \subseteq \Sigma : P \cup S_{true} \models \text{terminal} \} \]

\[
\begin{align*}
\text{terminal} & \leq \text{line}(x) \lor \text{line}(o) \\
\text{terminal} & \leq \neg \text{open} \\
\text{line}(W) & \leq \text{row}(M,W) \\
\text{line}(W) & \leq \text{column}(N,W) \\
\text{line}(W) & \leq \text{diagonal}(W) \\
\text{open} & \leq \text{true} (\text{cell}(M,N,b))
\end{align*}
\]
Auxiliary Clauses

\[
\begin{align*}
\text{row}(M, W) & \leq \text{true}(\text{cell}(M, 1, W)) \land \\
& \quad \text{true}(\text{cell}(M, 2, W)) \land \\
& \quad \text{true}(\text{cell}(M, 3, W)) \\
\text{column}(N, W) & \leq \text{true}(\text{cell}(1, N, W)) \land \\
& \quad \text{true}(\text{cell}(2, N, W)) \land \\
& \quad \text{true}(\text{cell}(3, N, W)) \\
\text{diagonal}(W) & \leq \text{true}(\text{cell}(1, 1, W)) \land \\
& \quad \text{true}(\text{cell}(2, 2, W)) \land \\
& \quad \text{true}(\text{cell}(3, 3, W)) \\
\text{diagonal}(W) & \leq \text{true}(\text{cell}(1, 3, W)) \land \\
& \quad \text{true}(\text{cell}(2, 2, W)) \land \\
& \quad \text{true}(\text{cell}(3, 1, W))
\end{align*}
\]
Goals

\[ P \text{ encodes } g = \{(r, S, n) : P \cup S_{\text{true}} \models \text{goal}(r, n)\} \]

\begin{align*}
\text{goal}(\text{xplayer}, 100) &\leq \text{line}(x) \\
\text{goal}(\text{xplayer}, 50) &\leq \neg \text{line}(x) \wedge \neg \text{line}(o) \wedge \neg \text{open} \\
\text{goal}(\text{xplayer}, 0) &\leq \text{line}(o) \\
\text{goal}(\text{oplayer}, 100) &\leq \text{line}(o) \\
\text{goal}(\text{oplayer}, 50) &\leq \neg \text{line}(x) \wedge \neg \text{line}(o) \wedge \neg \text{open} \\
\text{goal}(\text{oplayer}, 0) &\leq \text{line}(x)
\end{align*}
Reasoning

Game descriptions are a good example of knowledge representation with formal logic.

**Automated reasoning** about actions necessary to
- determine legal moves
- update positions
- recognize end of game
Challenge I: Efficient Descriptions
GDL and the Frame Problem

$\text{next}(\text{cell}(M,N,x)) \leq \text{does}(\text{xplayer}, \text{mark}(M,N))$

$\text{next}(\text{cell}(M,N,o)) \leq \text{does}(\text{oplayer}, \text{mark}(M,N))$

$\text{next}(\text{cell}(M,N,W)) \leq \text{true}(\text{cell}(M,N,W)) \land \neg W=b$

$\text{next}(\text{cell}(M,N,b)) \leq \text{true}(\text{cell}(M,N,b)) \land$

$\text{does}(P, \text{mark}(J,K)) \land (\neg M=J \lor \neg N=K)$

$\text{next}(\text{control}(\text{xplayer})) \leq \text{true}(\text{control}(\text{oplayer}))$

$\text{next}(\text{control}(\text{oplayer})) \leq \text{true}(\text{control}(\text{xplayer}))$
GDL and the Frame Problem

Effect Axioms
\[
\text{next}(\text{cell}(M,N,x)) \leq \text{does}(\text{xplayer}, \text{mark}(M,N))
\]
\[
\text{next}(\text{cell}(M,N,o)) \leq \text{does}(\text{oplayer}, \text{mark}(M,N))
\]

Frame Axioms
\[
\text{next}(\text{cell}(M,N,W)) \leq \text{true}(\text{cell}(M,N,W)) \land \neg W=b
\]
\[
\text{next}(\text{cell}(M,N,b)) \leq \text{true}(\text{cell}(M,N,b)) \land
\quad \text{does}(P, \text{mark}(J,K)) \land (\neg M=J \lor \neg N=K)
\]

Action-Independent Effects
\[
\text{next}(\text{control(xplayer)}) \leq \text{true}(\text{control(oplayer)})
\]
\[
\text{next}(\text{control(oplayer)}) \leq \text{true}(\text{control(xplayer)})
\]
A More Efficient Encoding (PDDL)

(:action noop
  :effect (and (when (control xplayer) (control oplayer))
               (when (control oplayer) (control xplayer)))
)

(:action mark
  :parameters (?p ?m ?n)
  :effect (and (not cell(?m ?n b))
               (when (= ?p xplayer) (cell(?m ?n x)))
               (when (= ?p oplayer) (cell (?m ?n o)))
               (when (control xplayer) (control oplayer))
               (when (control oplayer) (control xplayer)))
)
How to Get There?

Using **Situation Calculus**, the completion of the GDL clauses entails

\[
\text{cell}(M,N,W,\text{do}(\text{mark}(\text{xplayer},J,K),S)) \leftrightarrow
\begin{align*}
W &= x \land M = J \land N = K \\
\lor \quad \text{cell}(M,N,W,S) \land \neg W = b \\
\lor \quad \text{cell}(M,N,W,S) \land W = b \land (\neg M = J \lor \neg N = K)
\end{align*}
\]

This is equivalent to the (instantiated) **Successor State Axiom**

\[
\text{cell}(M,N,W,\text{do}(\text{mark}(\text{xplayer},J,K),S)) \leftrightarrow
\begin{align*}
W &= x \land M = J \land N = K \\
\lor \\
\text{cell}(M,N,W,S) \land \neg (M = J \land N = K \land W = b)
\end{align*}
\]
A More Difficult Example

\begin{align*}
\text{succ}(0, 1) & \leq \\
\text{succ}(1, 2) & \leq \\
\text{succ}(2, 3) & \leq \\
\text{init}(\text{step}(0)) & \leq \\
\text{next}(\text{step}(N)) & \leq \text{true}(\text{step}(M)) \land \text{succ}(M, N)
\end{align*}

The equivalence

\begin{align*}
\text{step}(N, \text{do}(P, A, S)) & \iff \text{step}(M, S) \land \text{succ}(M, N)
\end{align*}

does not entail the positive and negative(!) effects

\begin{align*}
(\text{when } (\text{and } (\text{step } ?m) (\text{succ } ?m ?n)) (\text{step } ?n)) \\
(\text{when } (\text{step } ?n) (\text{not } (\text{step } ?n)))
\end{align*}
Challenge I

Translate GDL effect clauses into an efficient action representation!

- Which formalism?
  Successor state axioms, state update axioms (Fluent Calculus), PDDL, causal laws, ...
- May require to prove state constraints
- Concurrency (for $n$-player games w/ $n \geq 2$)
Challenge II: Proving State Constraints
The Value of Knowledge

Not only are state constraints helpful for better encodings, structural knowledge of a game is crucial for good play.

Examples

- A game is turn-based.
- Each board cell \((X, Y)\) has a unique contents \(M\).
- Markers \(\times\) and \(\circ\) in Tic-Tac-Toe are permanent.
- A game is weakly (strongly) winnable.

Game properties like these can be formalized using \textit{ATL}; see [W. v. d. Hoek, J. Ruan, M. Wooldridge; 2008]
Claim

Fluent \texttt{control} has a unique argument in every reachable position.

\begin{verbatim}
P: init(control(xplayer)) <=
next(control(xplayer)) <= true(control(oplayer))
next(control(oplayer)) <= true(control(xplayer))
\end{verbatim}

The claim holds if

- uniqueness holds initially, and
- uniqueness holds \texttt{next}, provided it is \texttt{true} (and every player makes a legal move).
Answer Set Programming

We can use ASP to prove both an induction base and step.

\[
P \cup h_0 \leq 1\{\text{init(control}(X)) : \text{controldomain1}(X)\}1 \\
\leq h_0
\]

admits no answer set;

same for

\[
P \cup 1\{\text{true(control}(X)) : \text{controldomain1}(X)\}1 \leq \\
h \leq 1\{\text{next(control}(X)) : \text{controldomain1}(X)\}1 \\
\leq h
\]
Another Example

Claim
Every board cell has a unique contents.

Let $P$ be the GDL clauses for Tic-Tac-Toe.

$P \cup h_0(X,Y) \leq 1\{\text{init(control}(X,Y,Z)):\neg \text{celldomain3}(Z)\}1$

$h_0 \leq \neg h_0(X,Y)$

$\leq \neg h_0$

admits no answer set.
For the induction step, uniqueness of control must be known!

\[
P \cup \begin{align*}
1\{\text{true}(\text{control}(X)) &: \text{controldomain}1(X)\} & \leq & 1\{\text{does}(R,A) &: \text{doesdomain}2(A)\} \\
 & \leq & \text{does}(R,A) \land \neg\text{legal}(R,A) \\
1\{\text{true}(\text{cell}(X,Y,Z)) &: \text{celldomain}3(Z)\} & \leq & h(X,Y) \\
& \leq & 1\{\text{next}(\text{cell}(X,Y,Z)) &: \text{celldomain}3(Z)\} \\
& \leq & \neg h(X,Y) \\
& \leq & \neg h
\end{align*}
\]

admits no answer set.
Challenge II

Induction proofs using ASP work fine for reasonably small games.

For complex games, the grounded program becomes too large.

Find a more abstract proof method for GGP!
Planning and Search
Game Tree Search (General Concept)
A General Architecture

Game Description → Reasoner → Compiled Theory

- Reasoner:
  - Move List
  - State Update
  - Termination & Goal

Search
Learning
Towards Good Play

Besides efficient inference and search algorithms, the ability to automatically generate a good evaluation function distinguishes good from bad General Game Playing programs.

Existing approaches:
- Mobility and Novelty Heuristics
- Structure Detection
- Fuzzy Goal Evaluation
- Monte-Carlo Tree Search
Mobility

- More moves means better state

- Advantage:
  In many games, being cornered or forced into making a move is quite bad
  - In Chess, having fewer moves means having fewer pieces, pieces of lower value, or less control of the board
  - In Chess, when you are in check, you can do relatively few things compared to not being in check
  - In Othello, having few moves means you have little control of the board

- Disadvantage: Mobility is bad for some games
Worldcup 2006: Cluneplayer vs. Fluxplayer

Playclock:

Roles:

- Red
  - CLUNEPLAYER

- Black
  - FLUXPLAYER

Last Moves (step 2):

- Red
  - noop

- Black
  - move(bp,c,c6,d,c5)
Designing Evaluation Functions

- Typically designed by programmers/humans
- A great deal of thought and empirical testing goes into choosing one or more good functions
- E.g.
  - piece count, piece values in chess
  - holding corners in Othello
- But this requires knowledge of the game's structure, semantics, play order, etc.
Fuzzy Goal Evaluation: Example

Value of intermediate state = Degree to which it satisfies the goal

\[
goal(x_{\text{player}}, 100) \leq \text{line}(x)
\]
\[
\text{line}(P) \leq \text{row}(P)
\]
\[
\lor \text{col}(P)
\]
\[
\lor \text{diag}(P)
\]
Full Goal Specification

\[
\text{goal(xplayer,100) } \leq \text{ line(x)}
\]

\[
\text{line(P) } \leq \text{ row(P) } \lor \text{ col(P) } \lor \text{ diag(P)}
\]

\[
\text{row(P) } \leq \text{ true(cell(1,Y,P)) } \land \text{ true(cell(2,Y,P)) } \land \\
\text{ true(cell(3,Y,P))}
\]

\[
\text{col(P) } \leq \text{ true(cell(X,1,P)) } \land \text{ true(cell(X,2,P)) } \land \\
\text{ true(cell(X,3,P))}
\]

\[
\text{diag(P) } \leq \text{ true(cell(1,1,P)) } \land \text{ true(cell(2,2,P)) } \land \\
\text{ true(cell(3,3,P))}
\]

\[
\text{diag(P) } \leq \text{ true(cell(3,1,P)) } \land \text{ true(cell(2,2,P)) } \land \\
\text{ true(cell(1,3,P))}
\]
After Unfolding

\[
goal(x,100) \leq true(cell(1,Y,x)) \land true(cell(2,Y,x)) \land true(cell(3,Y,x)) \\
\lor true(cell(X,1,x)) \land true(cell(X,2,x)) \land true(cell(X,3,x)) \\
\lor true(cell(1,1,x)) \land true(cell(2,2,x)) \land true(cell(3,3,x)) \\
\lor true(cell(3,1,x)) \land true(cell(2,2,x)) \land true(cell(1,3,x))
\]

3 literals are true after \texttt{does}(x,\texttt{mark}(1,1))
2 literals are true after \texttt{does}(x,\texttt{mark}(1,2))
4 literals are true after \texttt{does}(x,\texttt{mark}(2,2))
Our t-norms: Instances of the Yager family (with parameter $q$)

$$T(a,b) = 1 - S(1-a,1-b)$$
$$S(a,b) = (a^q + b^q)^{1/q}$$

Evaluation function for formulas

$$
eval(f \land g) = T'(\eval(f),\eval(g))$$
$$
eval(f \lor g) = S'(\eval(f),\eval(g))$$
$$
eval(\neg f) = 1 - \eval(f)$$
Advanced Fuzzy Goal Evaluation: Example

$\text{init}(\text{cell(green, j, 13)}) \land \ldots$

$\text{goal(green, 100)} \leq \text{true(cell(green, e, 5))} \land \ldots$

Truth degree of goal literal = $(\text{Distance to current value})^{-1}$
Identifying Metrics

- **Order relations**  Binary, antisymmetric, functional, injective

  
  \[
  \text{succ}(1,2). \quad \text{succ}(2,3). \quad \text{succ}(3,4).
  \]
  
  \[
  \text{file}(a,b). \quad \text{file}(b,c). \quad \text{file}(c,d).
  \]

- Order relations define a **metric** on **functional** features

  \[
  \Delta (\text{cell}(\text{green}, j, 13), \text{cell}(\text{green}, e, 5)) = 13
  \]
Degree to which $f(x,a)$ is true given that $f(x,b)$:

$$(1-p) - (1-p) * \Delta(b,a) / |\text{dom}(f(x))|$$

With $p=0.9$, eval($\text{cell(green, e, 5)}$) is

0.082 if $\text{true(cell(green, f, 10))}$

0.085 if $\text{true(cell(green, j, 5))}$
A General Architecture

Game Description

Compiled Theory

Reasoner

Move List

State Update

Termination & Goal

Search

Evaluation Function
Fuzzy goal evaluation works particularly well for games with

- independent sub-goals
  - 15-Puzzle
- converge to the goal
  - Chinese Checkers
- quantitative goal
  - Othello
- partial goals
  - Peg Jumping, Chinese Checkers with >2 players
The GGP Challenge

Much like RoboCup, General Game Playing
- combines a variety of AI areas
- fosters developmental research
- has great public appeal
- has the potential to significantly advance AI

In contrast to RoboCup, GGP has the advantage to
- focus on the high-level knowledge aspect of intelligence
- poses a number of interesting challenges for KRR
- make a great hands-on course for AI+KR students
A Vision for GGP

Uncertainty
- Nondeterministic games with incomplete information

Natural Language Understanding
- Rules of a game given in natural language

Computer Vision
- Vision system sees board, pieces, cards, rule book, ...

Robotics
- Robot playing the actual, physical game
Resources

- Stanford GGP initiative games.stanford.edu
  - GDL specification
  - Basic player

- GGP in Germany general-game-playing.de
  - Game master

- Palamedes palamedes-ide.sourceforge.net
  - GGP/GDL development tool
Recommended Papers

- J. Clune
  Heuristic evaluation functions for general game playing, AAAI 2007
- H. Finnsson, Y. Björnsson
  Simulation-based approach to general game playing, AAAI 2008
- M. Genesereth, N. Love, B. Pell
  General game playing, AI magazine 26(2), 2006
- W. v. d. Hoek, J. Ruan, M. Wooldridge
  Verification of games in the game description language, 2008 (submitted)
- S. Schiffel, M. Thielscher
  Fluxplayer: a successful general game player, AAAI 2007
- S. Schiffel, M. Thielscher
  Specifying multiagent environments in the Game Description Language, 2008 (submitted)