Computer Game Playing
General Game Playing

General Game Players are systems
- able to accept a formal description of arbitrary games
- able to use such descriptions to play the games effectively

Cognitive Information Processing Technologies
- knowledge representation
- reasoning
- learning
- rational behavior

Unlike specialized game players (e.g. Deep Blue), they do not use algorithms designed in advance for specific games.
Tic-Tac-Toe

X

O

X
Chess
Kriegspiel
Bughouse Chess
Towers of Hanoi
Maze World
Monopoly
International Activities

Website – http://games.stanford.edu
- Games
- Game Manager
- Reference Players

Annual Competition
- Pittsburgh 2005
- Boston 2006
Finite Synchronous Games

Finite environment
- Environment with finitely many states
- One initial state and one or more terminal states

Finite Players
- Fixed finite number of players
- Each with finitely many “actions”
- Each with one or more goal states

Synchronous Update
- All players move on all steps (some no-ops)
- Environment changes only in response to moves
Games as State Machines
Initial State and Terminal States
Goal States
Simultaneous Actions
Correct Game Descriptions

In a correct game description
- Each player has a legal move in every non-terminal state
- Every terminal state is a goal state: a goal value is specified for each player
Encoding Alternatives

**State Machines.** Astronomically large state spaces, e.g. \( \sim 5000 \) states in Tic-Tac-Toe, \( \sim 10^{30} \) states in Chess.

**Lists and Tables.** Still the same size. Just switching to database states does not decrease the size of direct representation.

**Programs.** One possibility is to write a program to generate legal moves and successor states and to evaluate goals and termination. However, which language? Java, Lisp, C? What if a player wants to reason about the structure of a game in general? This is difficult if the game is encoded in procedural form.

**Logic.** There are existing interpreters / compilers. Logic is easier to use for analysis than procedural encodings.
Formal Game Descriptions

Whatever form is used, the description must give all information necessary to determine legality of moves, state transition, termination, and goals.

Nothing is assumed except for logic.

- No arithmetics
- No physics
- No common sense

(To emphasize this, game descriptions can be written in terms of nonsense symbols.)
Logical Reasoning

Minimal Capabilities
- Computing legality of moves
- Computing consequences of actions
- Knowing when a goal is achieved
- Knowing when a game is over

Other Potential Uses
- Simplification
- Optimization
- Game Theory
Simple Games

Simple Games
- Single Player
- Moderate state space

Examples
- Towers of Hanoi
- Blocks World
- Buttons and Light

Approach
- Planning
Indeterminacy

[Diagram of a complex graph with nodes labeled a, b, c, d, e, f, g, h, i, j, k, and edges labeled with symbols such as a/a, a/b, b/a, b/b.]

General Game Playing

Introduction

21
Dealing with Indeterminacy

Coercive Sequential Plans

Conditional Plans

Interleaved Planning and Execution

~> Trial-and-error

(reversibility desirable)
Dealing with Size

Partial Search
- Minimax
- Evaluation function for non-terminal states

Reformulation
~> Solve abstract problem
~> Refine abstract solution to complete solution
Learning

Performance improvement during a match
- Recognizing futile actions/strategies
- Modeling other players

Performance improvement over multiple matches
~> Must recognize that it is the same game

Performance improvement over games
- Lemmas
- Concepts (e.g. “pinned pieces”)
Infrastructure

Parameterized collection of Games
- Complete information
- One player or \( n \) players
- Competition and cooperation

Services
- Sample players (including human interface)
- Game definition support
- Game Manager

Records
- Players
- Performance data
Game Manager

- Graphics for Spectators
- Game Manager
- Temporary State Data
- Player
- Game Descriptions
- Match Records
- TCP/IP
Game Playing Protocol

Start
- Manager sends Start message to players
  \texttt{Start(match,role,description,startclock,playclock)}

Play
- Manager sends Play messages to players
  \texttt{Play(match,actions)}
- Receives plays in response

Stop
- Manager sends Stop message to players
  \texttt{Stop(match,actions)}
Misplay

Player errs
- Illegal move
- No move before clock runs out

~> Manager selects a random legal move
Variety of Games

Number of players
- Single player, e.g. Towers of Hanoi, 15-Puzzle, Maze World
- 2-player competitive games, e.g. Tic-Tac-Toe, Chess
- $n$-player games with allies and enemies, e.g. Monopoly

Information
- Incomplete, e.g. Maze World, Monopoly
- Complete, e.g. Chess

Scale
- Exhaustively searchable, e.g. Tic-Tac-Toe
- Not exhaustively searchable, e.g. Chess
Game Theory

Like General Game Theory, traditional Game Theory is concerned with games in general.

In traditional Game Theory, the game tree is known to the game programmer in advance.

In General Game Theory, the game “tree” is communicated at runtime. Hence there is an emphasis on finite description and use of this description.
Automatic Programming

The goal of automatic programming is the automatic generation of programs that achieve formally stated specifications. Ditto for GGP.
Critique

The main downside of general game playing is the possible criticism of game playing as being a frivolous pursuit.

Sample Application:
  Enterprise Management Behavior:
  Laws, Business Rules, Contracts
  Planning under Behavioral Constraints
  Monitoring and Conformance Testing
  Business Process Mediation
  Automated Legislation
Critique

Human Intelligence is arguably the product of eons of evolution. We are, to some extent at least, “wired” to function well in this world. General Game Players have nothing at their disposal but mathematics.

A key indicator of intelligence is the ability to function in radically new environments.
Critique

*Real* intelligence requires the ability to *figure out* new environments.

Well, no question that that ability is essential. However, real intelligence also requires the ability to *use* theories once they are formed. This is the domain of *specification-based systems / declarative systems* and so forth; and interest in this problem dates to the beginning of the field.
The main advantage we expect the advice taker to have is that its behavior will be improvable merely by making statements to it, telling it about its [...] environment and what is wanted from it. To make these statements will require little, if any, knowledge of the program or the previous knowledge of the advice taker.
The potential use of computers by people to accomplish tasks can be “one-dimensionalized” into a spectrum representing the nature of the instruction that must be given the computer to do its job. Call it the what-to-how spectrum. At one extreme of the spectrum, the user supplies his intelligence to instruct the machine with precision exactly how to do his job step-by-step. [...] At the other end of the spectrum is the user with his real problem. [...] He aspires to communicate what he wants done [...] without having to lay out in detail all necessary subgoals for adequate performance.
Robert Heinlein

A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects.
## Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday, October 12th, 2006</td>
<td>Introduction</td>
</tr>
<tr>
<td>19th,</td>
<td>Game Description Language</td>
</tr>
<tr>
<td>26th,</td>
<td>Automated Reasoning</td>
</tr>
<tr>
<td>November 2nd,</td>
<td>State-Space Search and Planning</td>
</tr>
<tr>
<td>9th,</td>
<td>Incomplete Information</td>
</tr>
<tr>
<td>16th,</td>
<td>Game Theory</td>
</tr>
<tr>
<td>Beginning of summer term 2007</td>
<td>Final Contest</td>
</tr>
</tbody>
</table>