UNSW

THE UNIVERSITY OF NEW SOUTH WALES

General Game Playing in AI Research and Education

Michael Thielscher

GGP in AI Research & Education

- What is General Game Playing (GGP)?
- The Many Facets of GGP Research
- The Potential for GGP in AI Teaching

General Game Playing

General Game Players are systems that

- understand new game descriptions
- learn to play these games effectively

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

General Game Playing

General Game Players are systems that

- understand new game descriptions
- learn to play these games effectively

Translation:

They don't know the rules until the game starts.

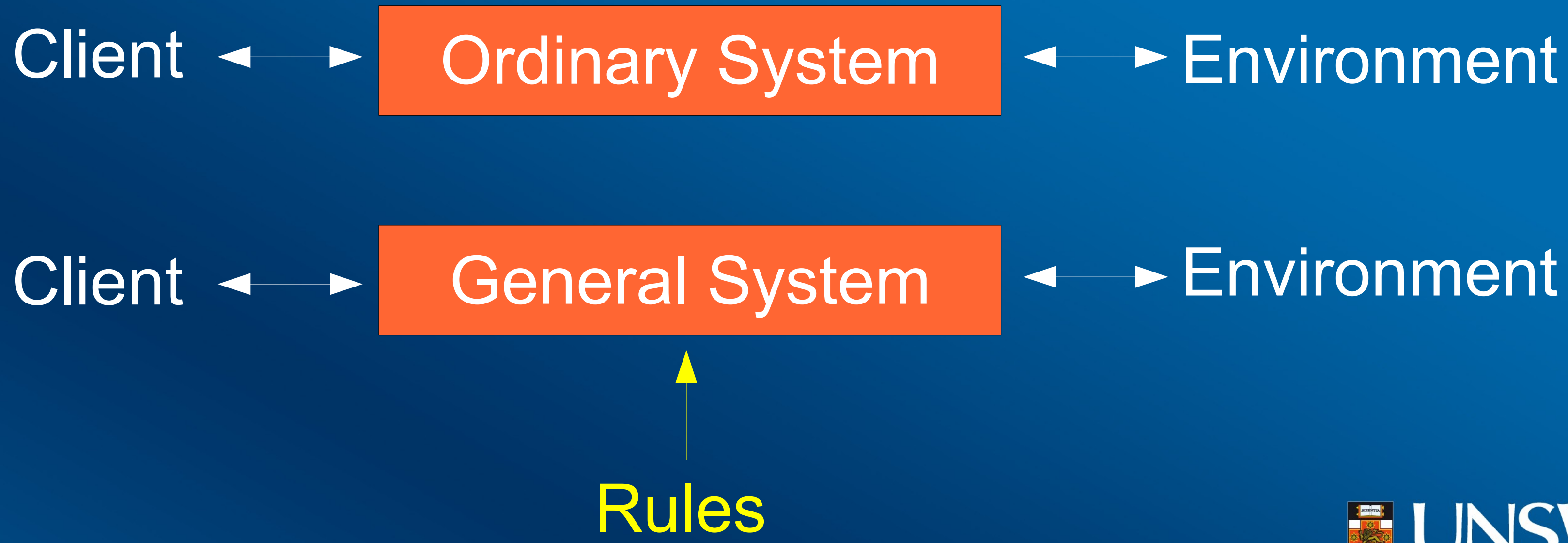
Unlike specialised programs (Deep Blue) they can't use game-specific algorithms.

Why GGP?

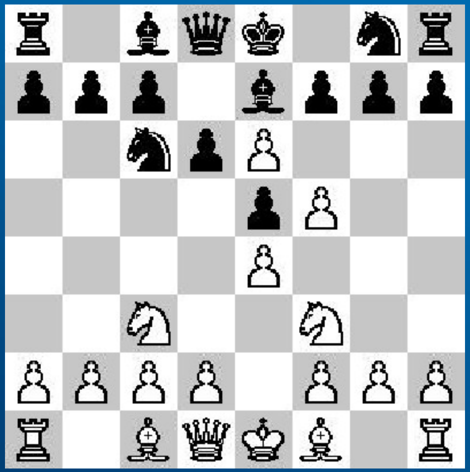
- Many biological, economic, political, social processes can be formalised as games.

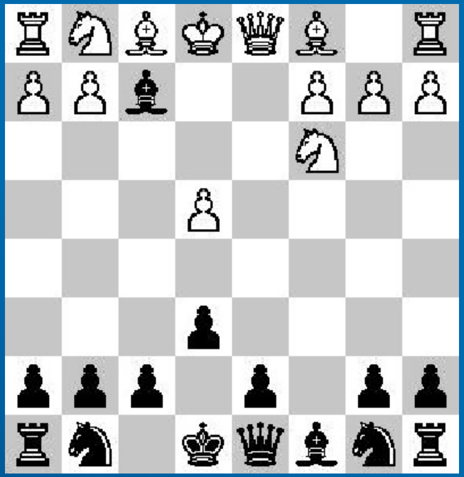
Why GGP?

- Many biological, economic, political, social processes can be formalised as games.
- GGP ⊆ General Problem Solving

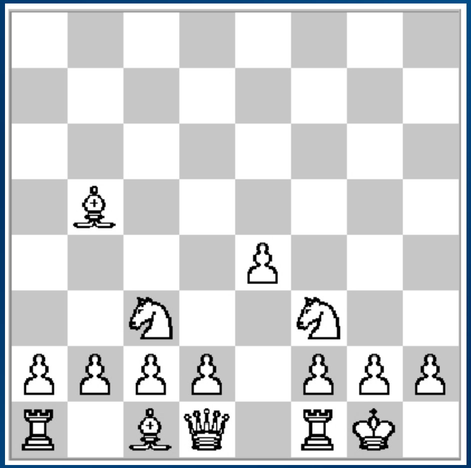


Application: General Chess Player





Bughouse Chess



Kriegspiel

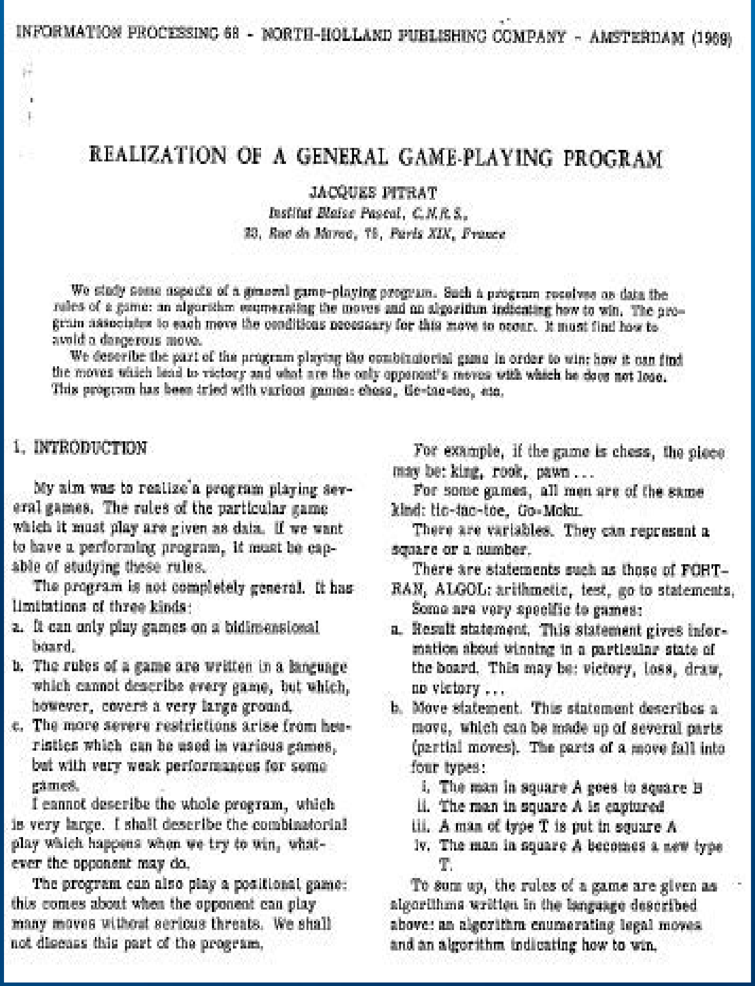
Application:
Universal AI Game Engine

Entertainment

Security Games

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

A Brief History of GGP



INFORMATION PROCESSING 68 - NORTH-HOLLAND PUBLISHING COMPANY - AMSTERDAM (1969)

REALIZATION OF A GENERAL GAME-PLAYING PROGRAM

JACQUES PITRAT
Institut Blaise Pascal, C.N.R.S.,
20, Rue du Maroc, 75, Paris XIX, France

We study some aspects of a general game-playing program. Such a program receives as data the rules of a game: an algorithm enumerating the moves and an algorithm indicating how to win. The program associates to each move the conditions necessary for this move to occur. It must find how to avoid a dangerous move.

We describe the part of the program playing the combinatorial game in order to win: how it can find the moves which lead to victory and what are the only opponent's moves with which he does not lose. This program has been tried with various games: chess, tic-tac-toe, etc.

1. INTRODUCTION

My aim was to realize a program playing several games. The rules of the particular game which it must play are given as data. If we want to have a performing program, it must be capable of studying these rules.

The program is not completely general. It has limitations of three kinds:

a. It can only play games on a bidimensional board.
b. The rules of a game are written in a language which cannot describe every game, but which, however, covers a very large ground.
c. The more severe restrictions arise from heuristics which can be used in various games, but with very weak performances for some games.

I cannot describe the whole program, which is very large. I shall describe the combinatorial play which happens when we try to win, whatever the opponent may do.

The program can also play a positional game: this comes about when the opponent can play many moves without serious threats. We shall not discuss this part of the program.

For example, if the game is chess, the piece may be: king, rook, pawn . . .

For some games, all men are of the same kind: tic-tac-toe, Go-Moku.

There are variables. They can represent a square or a number.

There are statements such as those of FORTRAN, ALGOL: arithmetic, test, go to statements.

Some are very specific to games:

a. Result statement. This statement gives information about winning in a particular state of the board. This may be: victory, loss, draw, no victory . . .
b. Move statement. This statement describes a move, which can be made up of several parts (partial moves). The parts of a move fall into four types:
   i. The man in square A goes to square B
   ii. The man in square A is captured
   iii. A man of type T is put in square A
   iv. The man in square A becomes a new type T.

To sum up, the rules of a game are given as algorithms written in the language described above: an algorithm enumerating legal moves and an algorithm indicating how to win.

J. Pitrat (1968): Realization of a General Game-Playing Program

"AI systems are dumb because AI researchers are too clever."

AAAI GGP Competition

- 2005 Cluneplayer (USA)
- 2006 Fluxplayer (Germany)
- 2007/08 Cadiaplayer (Iceland)
- 2009/10 Ary (France)
- 2011 TurboTurtle (USA)

AAAI GGP Competition

- 2005 Cluneplayer (USA)
- 2006 Fluxplayer (Germany)
- 2007/08 Cadiaplayer (Iceland)
- 2009/10 Ary (France)
- 2011 TurboTurtle (USA)

1st German Open, Berlin 2011

- Ary (France) Classic Track
- Fluxii (Iceland) Imperfect-Information Track

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

GGP Since 2005

- $>$100 Publications
- GGP Workshops at IJCAI since 2009
- GGP Special Issue of "KI-Journal" Feb 2011
- Google hits for "General Game Playing":

120,000

GGP Around the World

- Stanford
  Michael Genesereth
- U of Alberta
  Jonathan Schaeffer, N. Sturtevant
- Reykjavík U
  Yngvi Björnsson, H. Finnsson, S. Schiffel
- U of Bremen
  Stefan Edelkamp, P. Kissmann
- U of Potsdam
  Torsten Schaub, etal.

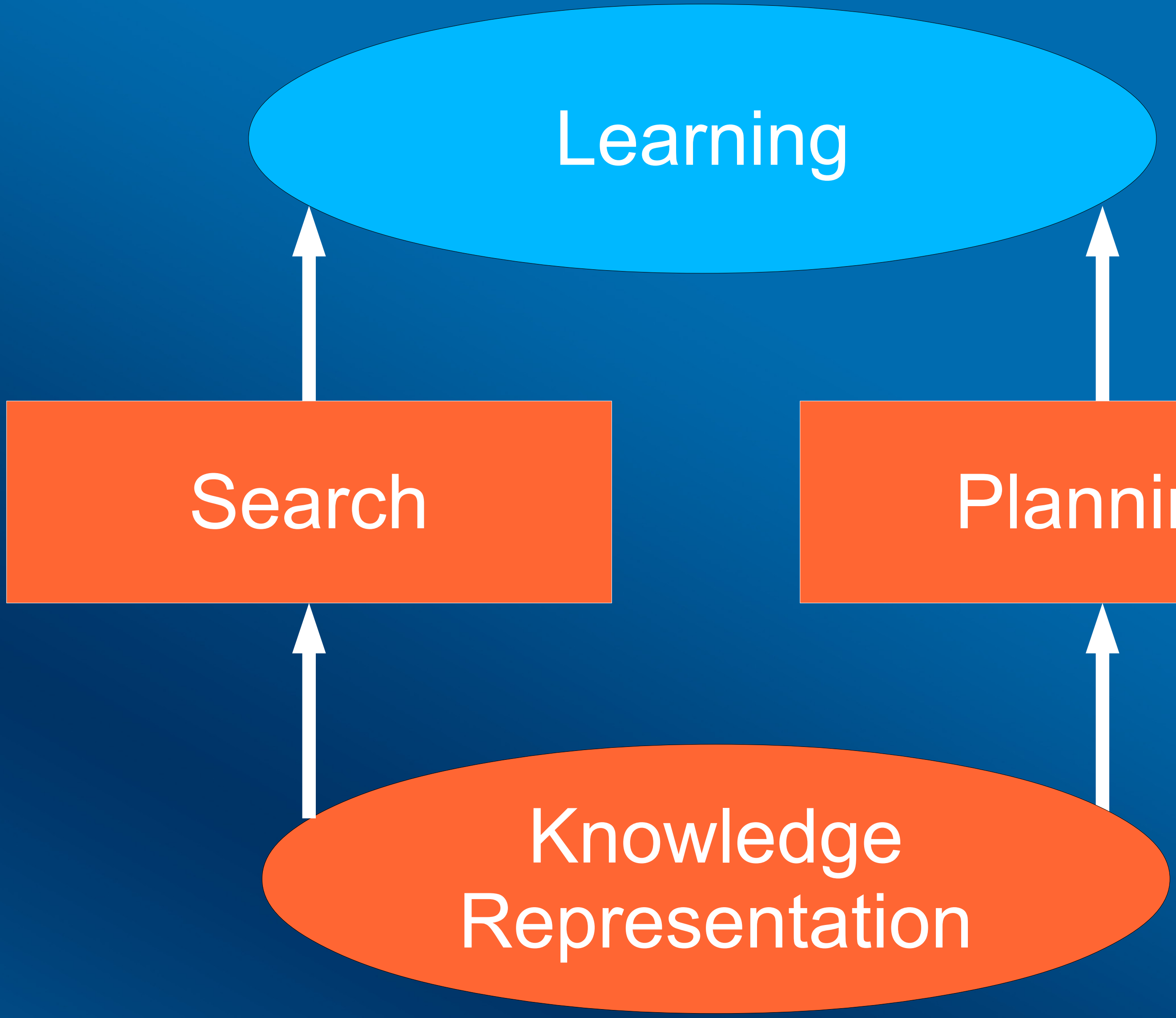
UNSW
THE UNIVERSITY OF NEW SOUTH WALES

Part II

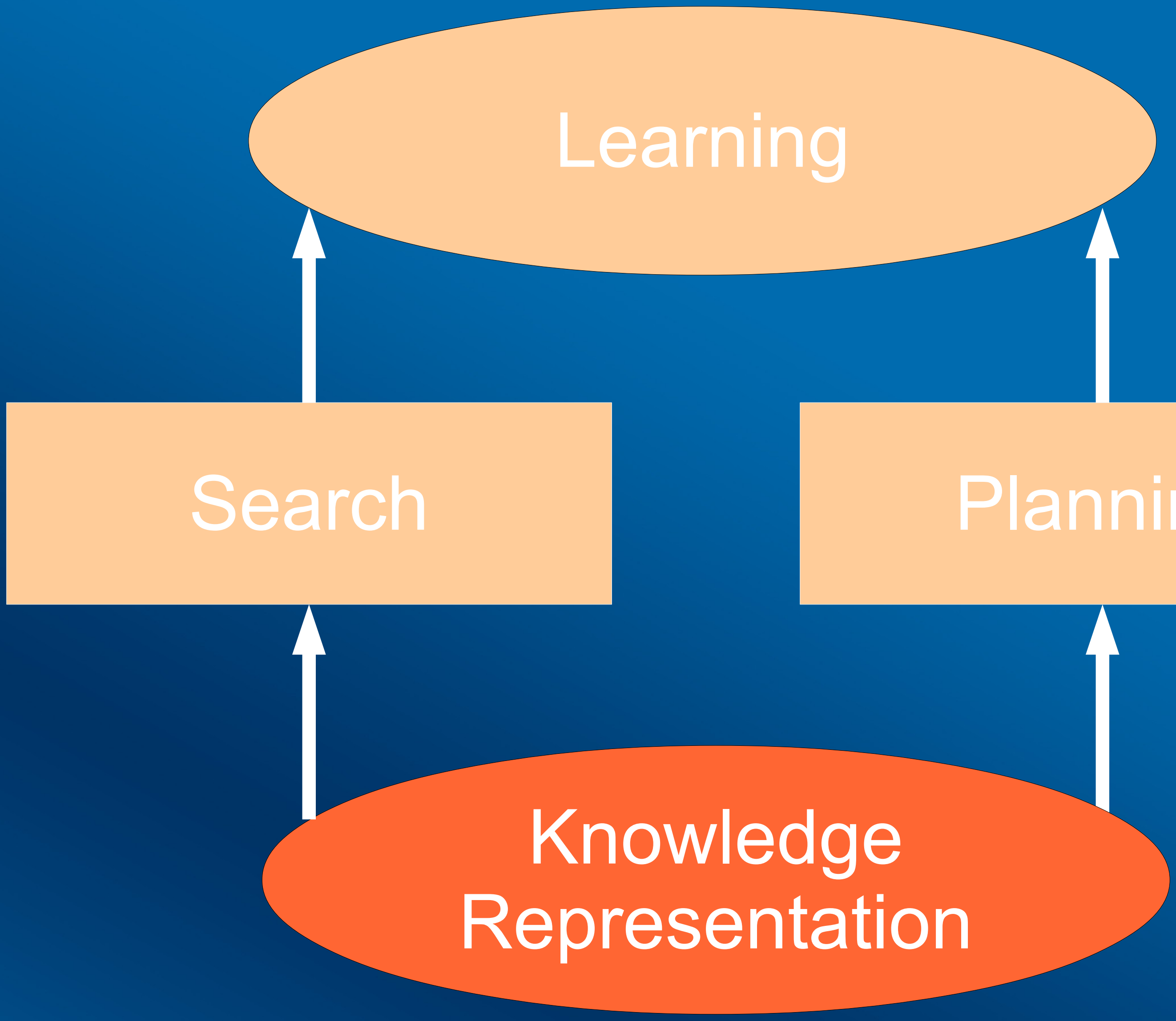
—

The Many Facets of General Game Playing Research

GGP Research Landscape



GGP Research Landscape



KR: Foundations

Game Description Language GDL

```
init(cell(a,1,wr))   …   init(cell(h,8,br))
```

Initial position

KR: Foundations

Game Description Language GDL

```
init(cell(a,1,wr))  …  init(cell(h,8,br))

legal(white,castle(?dir)) <= true(cell(e,1,wk))∧ …
next(cell(g,1,wk)) <= does(white,castle(kside))∧ …
```

Moves

UNSW THE UNIVERSITY OF NEW SOUTH WALES

KR: Foundations

Game Description Language GDL

```
init(cell(a,1,wr))  …  init(cell(h,8,br))

legal(white,castle(?dir)) <= true(cell(e,1,wk)) ∧ …
next(cell(g,1,wk)) <= does(white,castle(kside)) ∧ …

goal(white,100) <= checkmate ∧ true(control(black))
```

Objective

KR: Results

- Automatic translations (→ efficient reasoning)

  Automata, BDDs, C++, OCAML

KR: Results

- Automatic translations (→ efficient reasoning)

  Automata, BDDs, C++, OCAML

- Automated theorem proving

```
true(φ).

false :- next(φ).
```

ASP

KR: Results

- Automatic translations (→ efficient reasoning)

  Automata, BDDs, C++, OCAML

- Automated theorem proving

```
true(φ).
false :- next(φ).
```

ASP

- Describing imperfect-information games

```
sees(?p,your_card(?c))
    <= does(random,deal(?p,?c))
```

KR: Open Questions

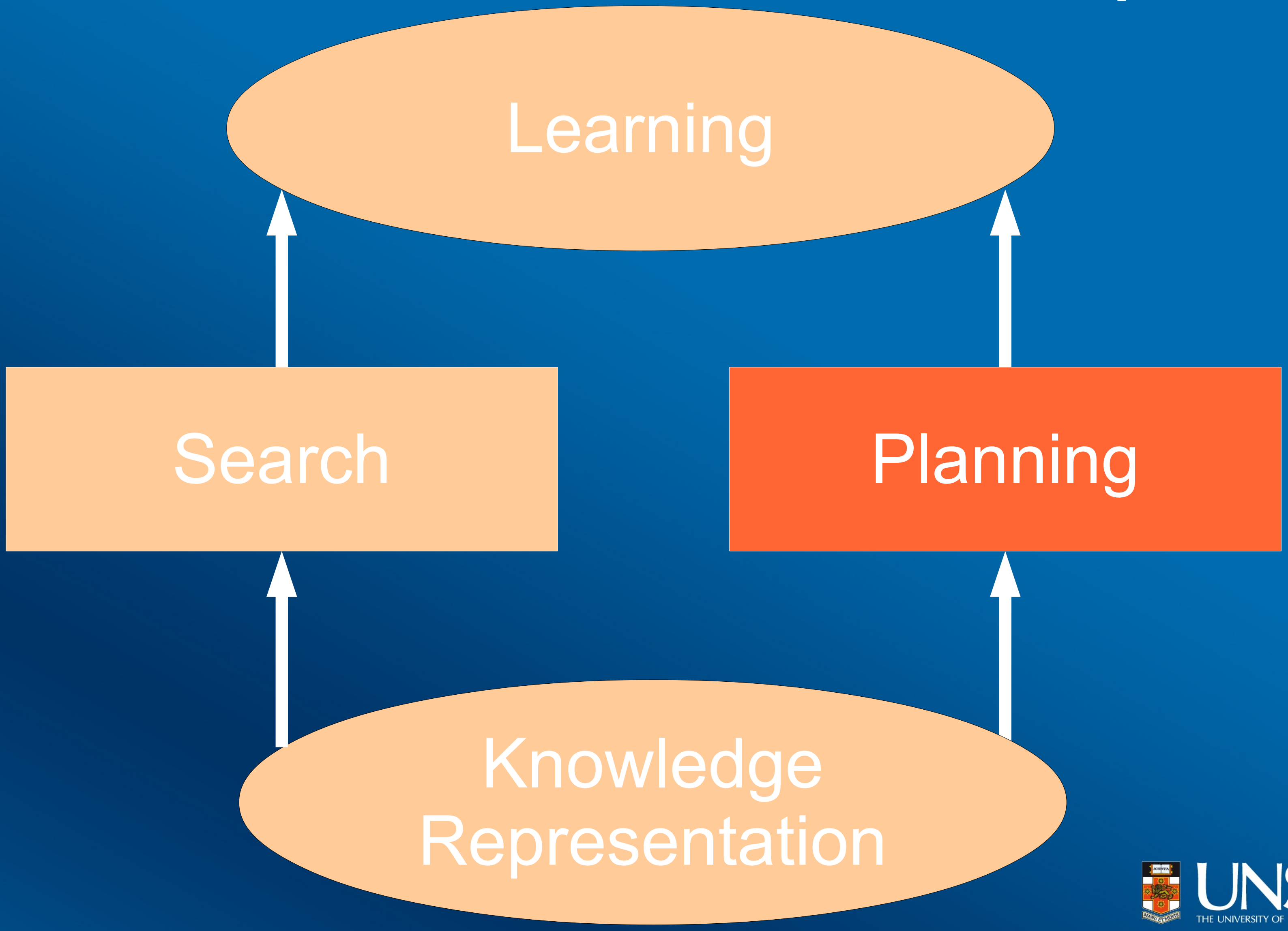
- Efficient non-propositional encodings
- Efficient first-order theorem proving
- Efficient reasoning with imperfect information

KR: Open Questions

- Efficient non-propositional encodings
- Efficient first-order theorem proving
- Efficient reasoning with imperfect information
- Other general problem description languages
- Belief Revision for GGP
- Spatial Reasoning for GGP

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

GGP Research Landscape



Planning: Results

- Partial translation GDL → PDDL

  Problem: determine negative effects

- CSP / ASP for single-player games

- Techniques known from Planning adapted to GGP: decomposition & symmetry detection

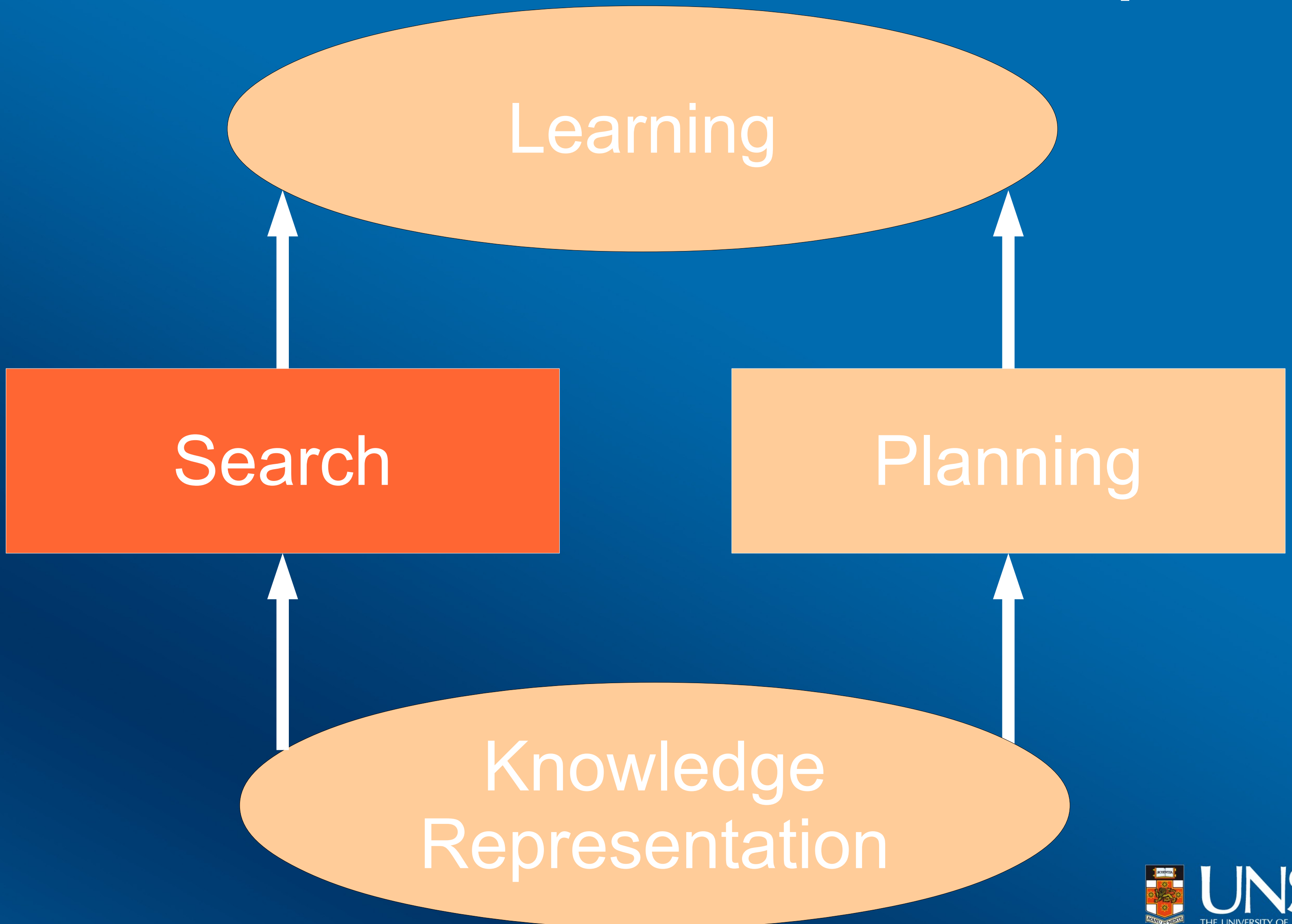
Planning: Open Questions

- Full translation GDL → Planning Language
- Planning heuristics (eg. relaxation) for GGP
- Other techniques (eg. hierarchical planning) for GGP

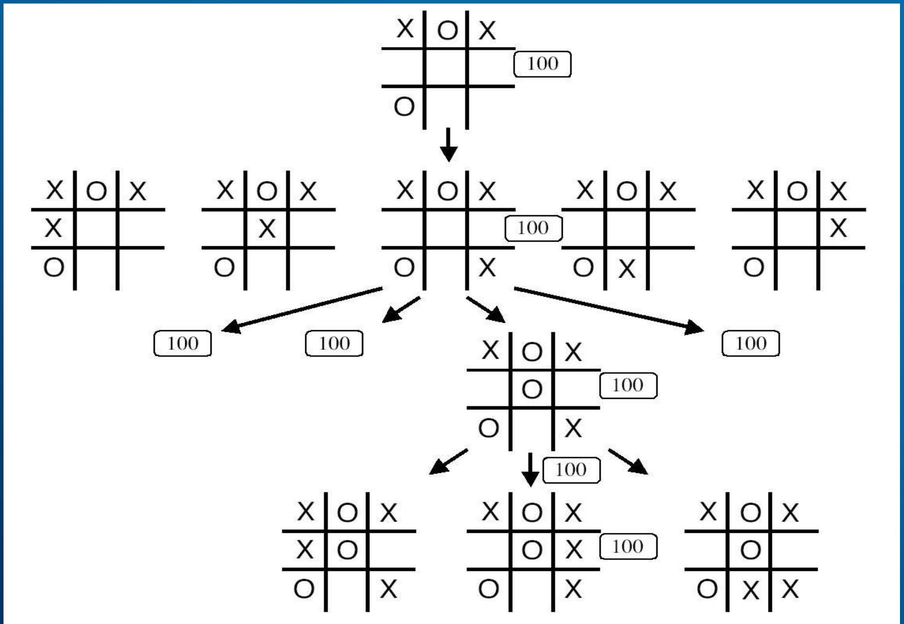
Planning: Open Questions

- Full translation GDL → Planning Language
- Planning heuristics (eg. relaxation) for GGP
- Other techniques (eg. hierarchical planning) for GGP
- Conversely: GGP techniques (eg. opponent modelling) for adversarial planning

GGP Research Landscape

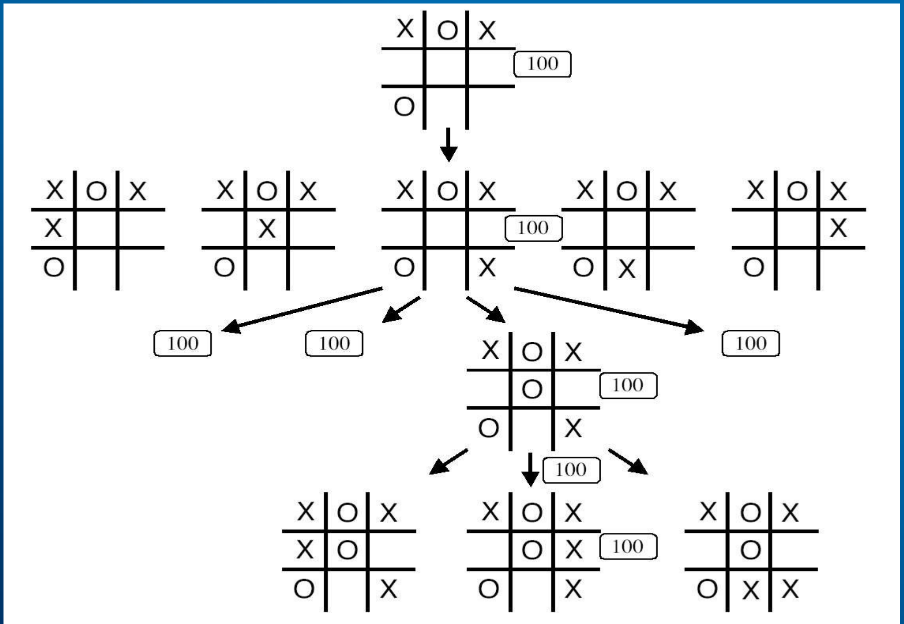


Search(1): Adversarial Search



- Minimax with α/β
- Transposition tables

Search(1): Adversarial Search



- Minimax with α/β
- Transposition tables

Automatically generated evaluation functions

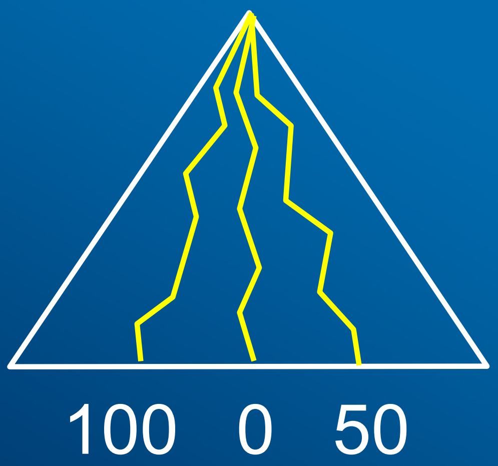
General heuristics mobility, novelty

Fuzzy logic truth-degree of goal formula

Learning piece count, piece values

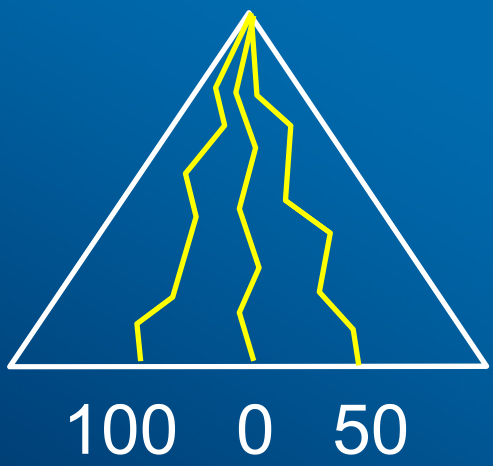
UNSW THE UNIVERSITY OF NEW SOUTH WALES

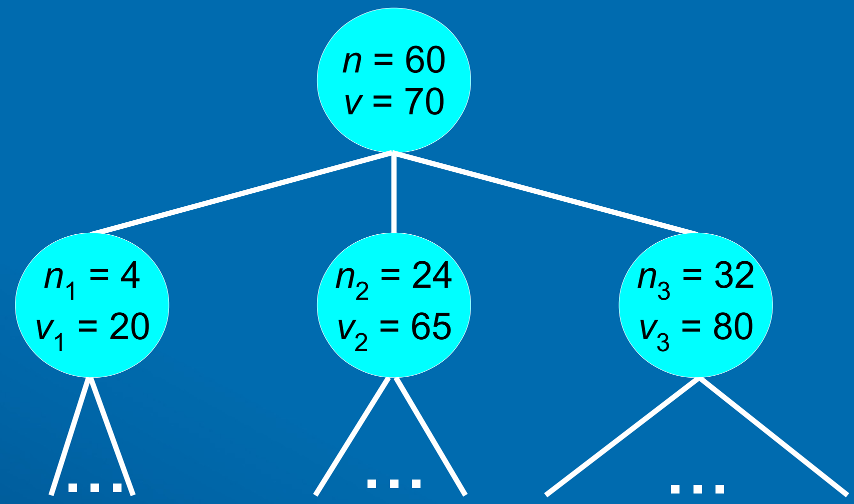
Search(2): Monte Carlo Search



- Random sampling

Search(2): Monte Carlo Search

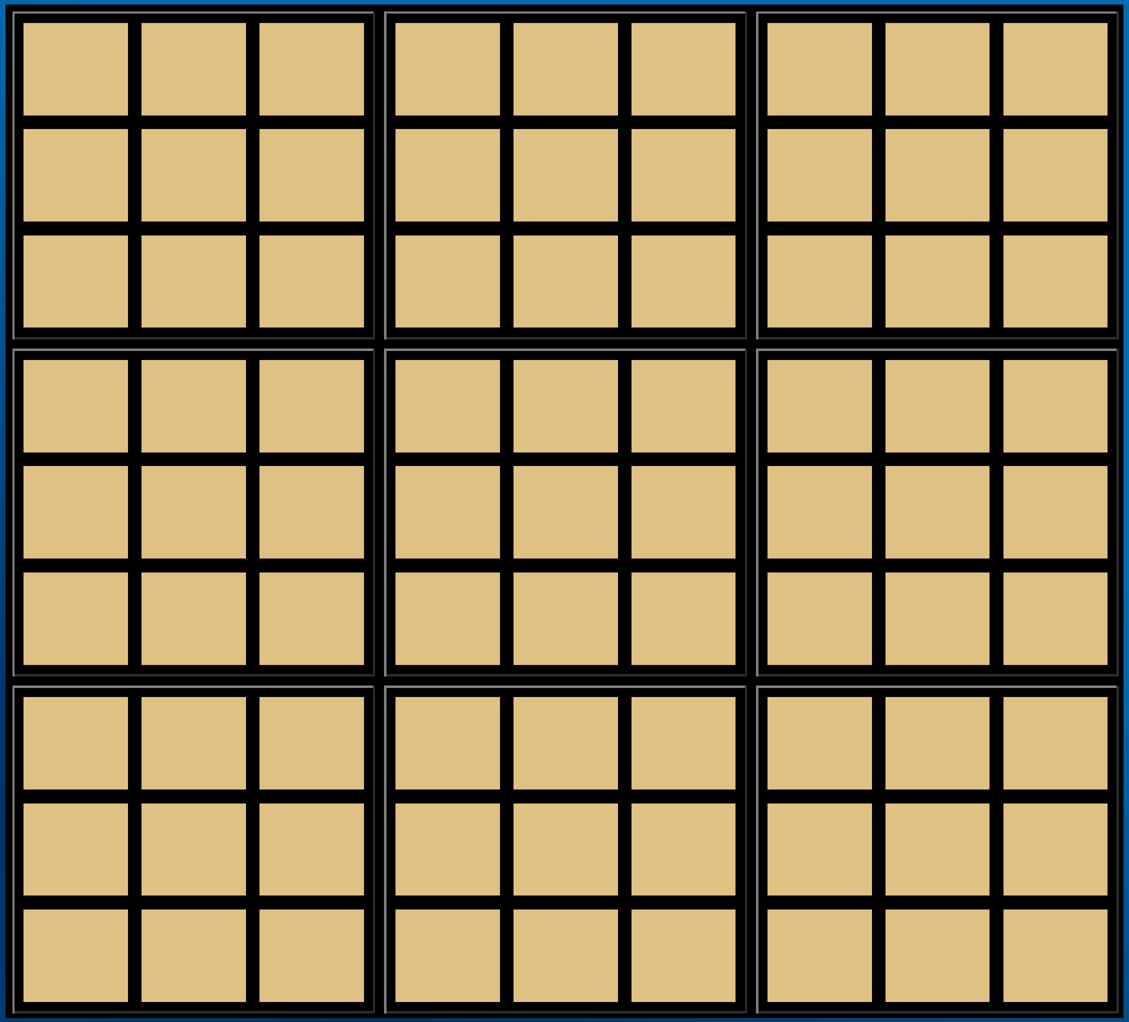




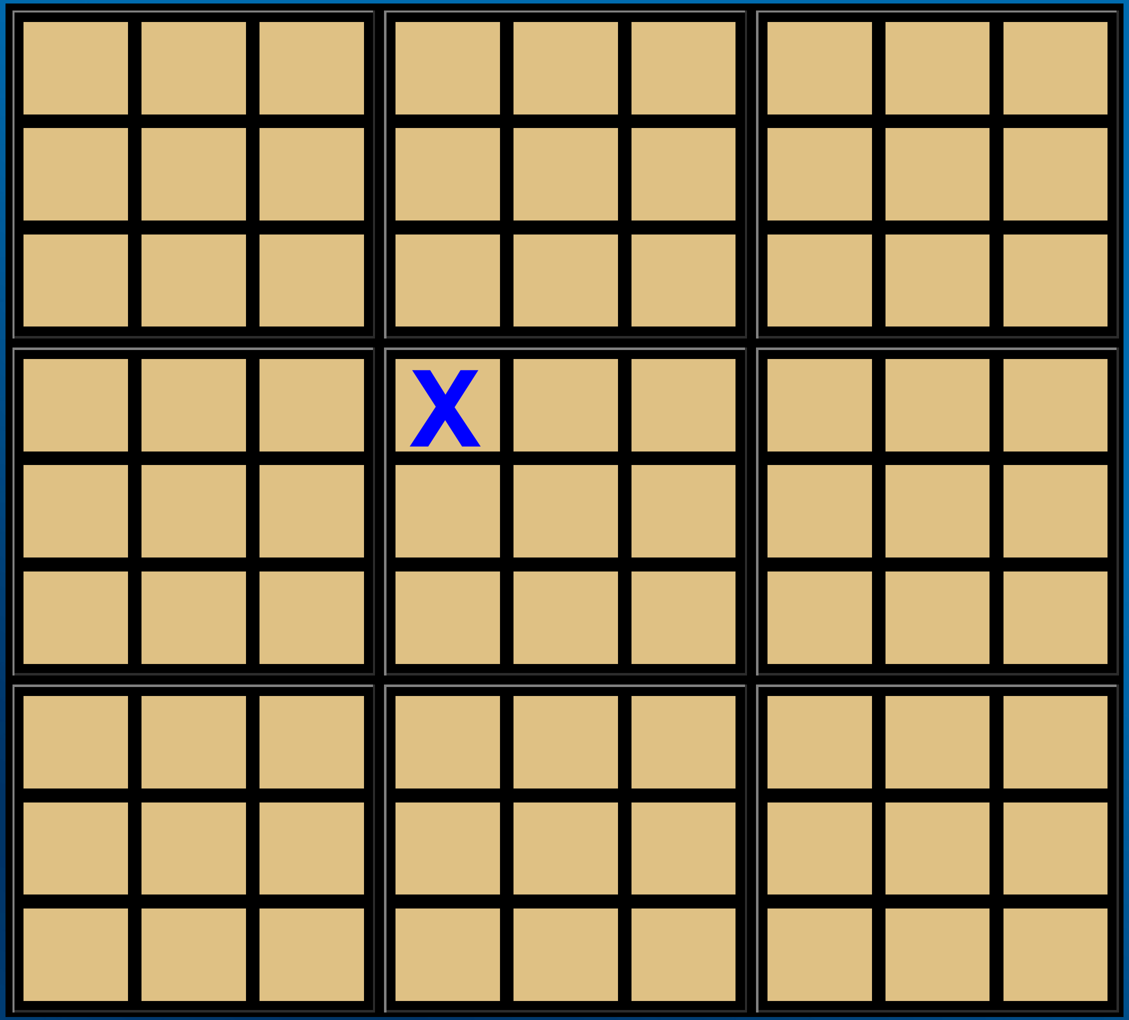
- Random sampling
- UCT (exploration vs. exploitation)

Game-specific knowledge to bias move selection

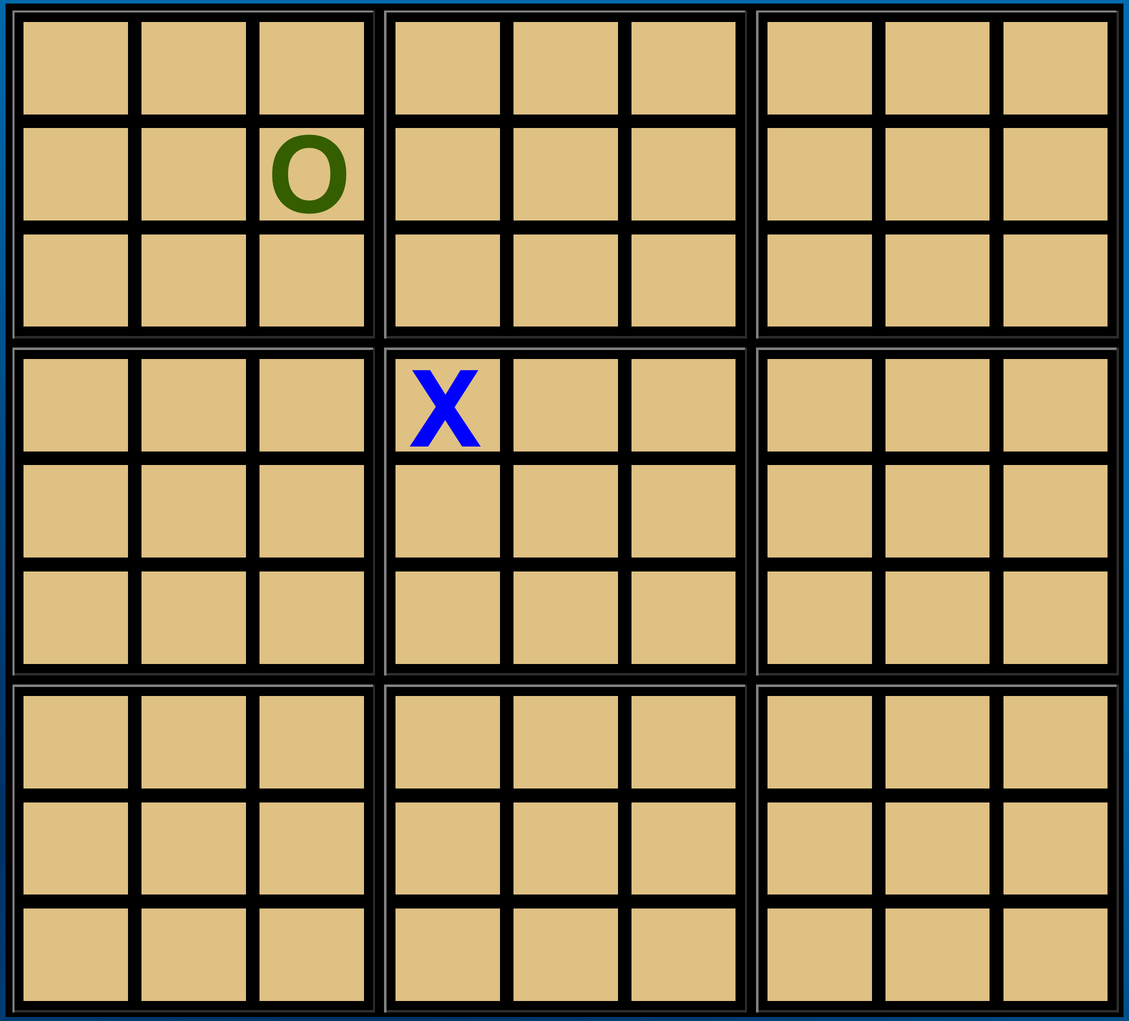
Example



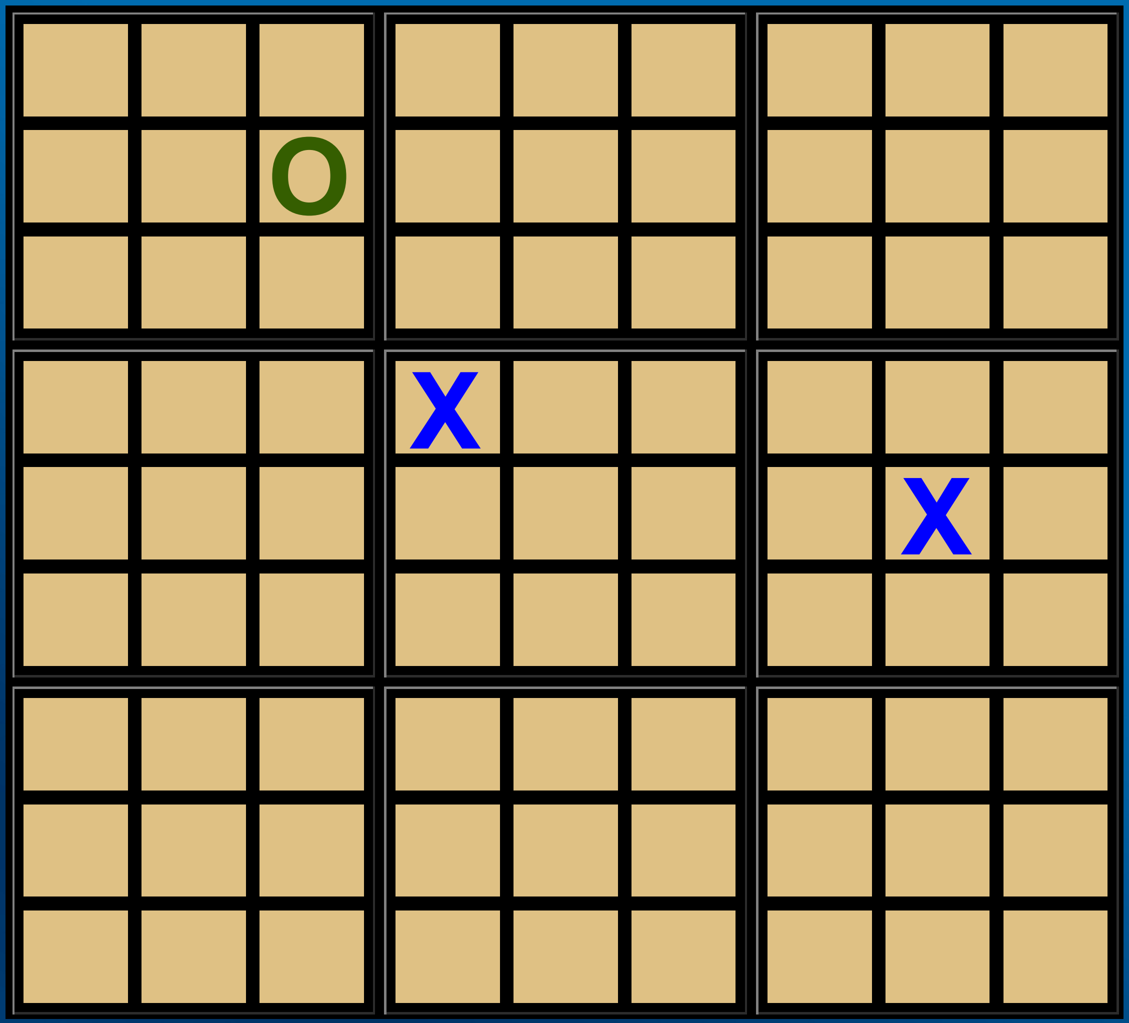
Example



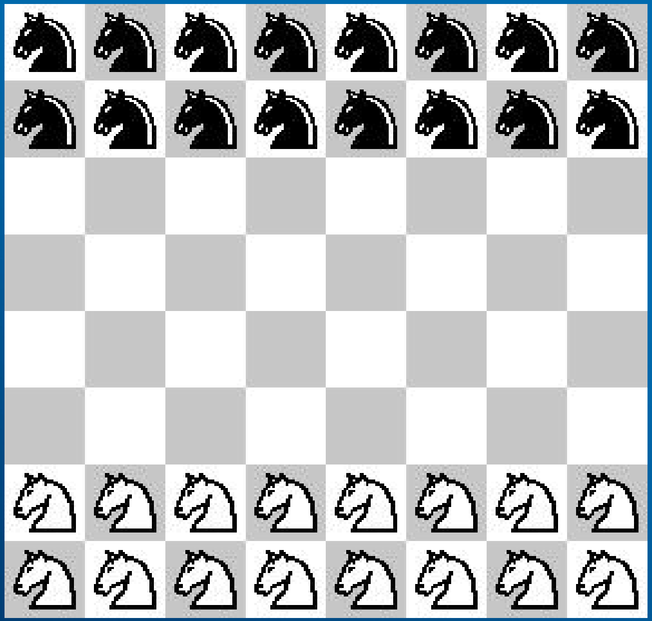
Example



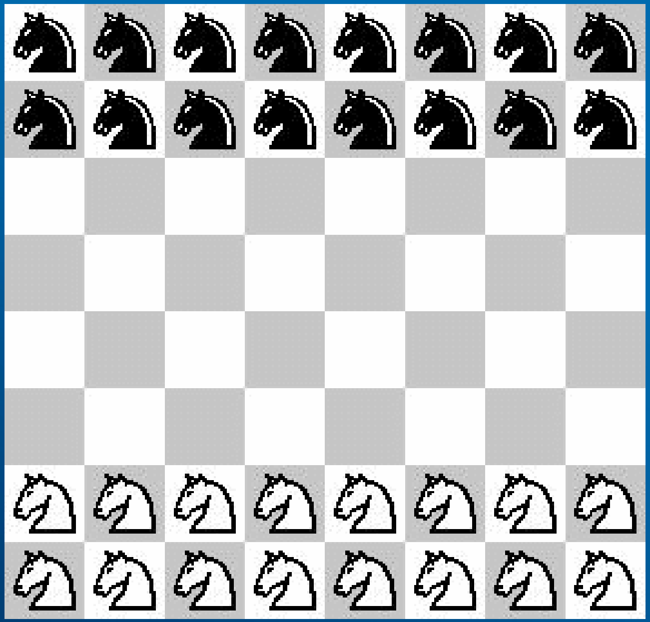
Example



UCT vs Minimax (Example)



UCT vs Minimax (Example)



Search: Open Questions

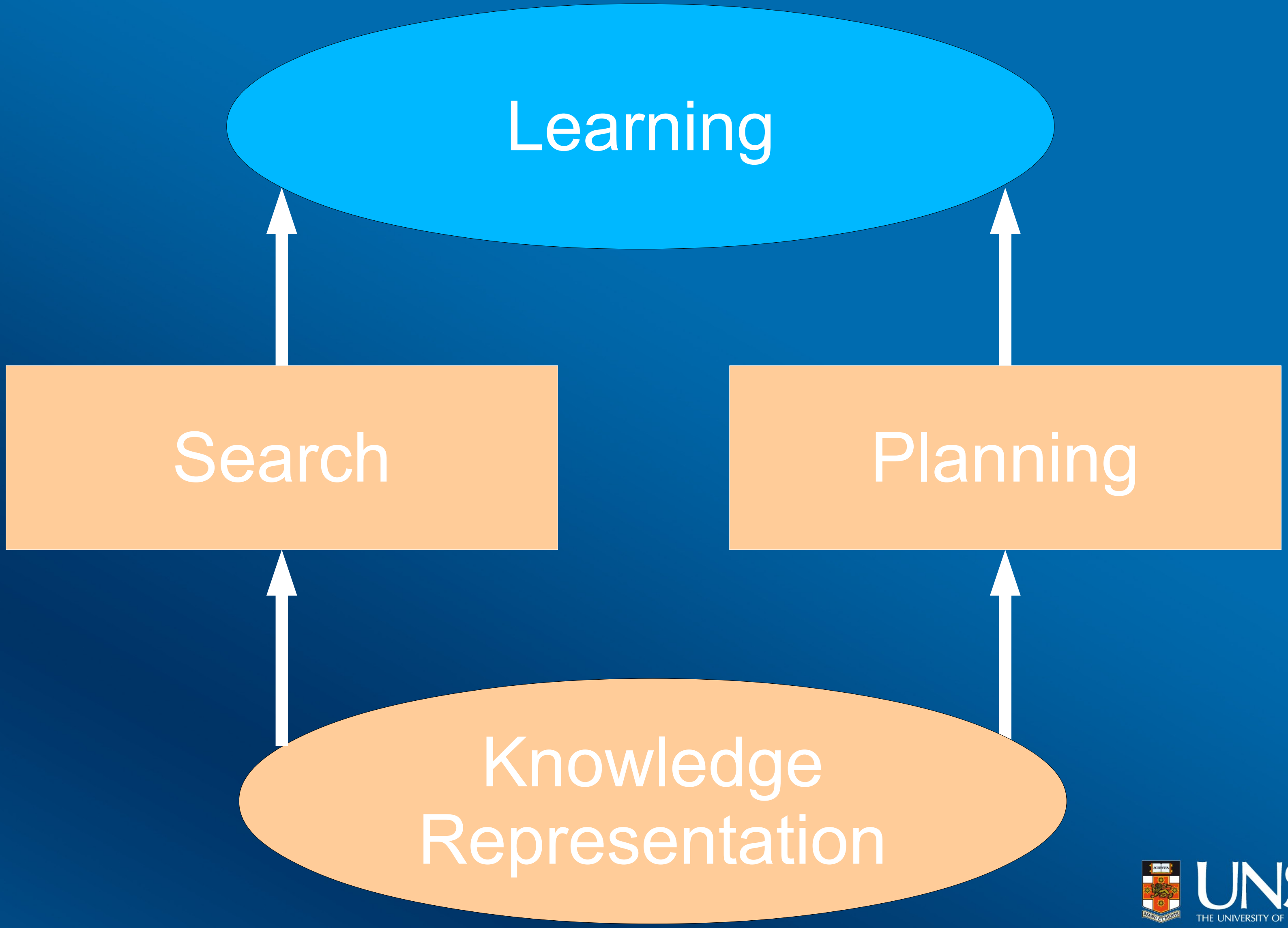
- Learning knowledge to guide UCT search
- Decide search technique at runtime
- Search for imperfect-information games

Search: Open Questions

- Learning knowledge to guide UCT search
- Decide search technique at runtime
- Search for imperfect-information games
- Automatic generation of search heuristics
- Informed search / local search for GGP

UNSW THE UNIVERSITY OF NEW SOUTH WALES

GGP Research Landscape



Learning: Results / Open Questions

- Transfer learning (between similar games)
- Neural nets to improve evaluation functions
- TD-learning to construct relevant features
- Statistical learning to improve biasing in UCT

Learning: Results / Open Questions

- Transfer learning (between similar games)
- Neural nets to improve evaluation functions
- TD-learning to construct relevant features
- Statistical learning to improve biasing in UCT
- More learning requires more data & time

General General Game Playing

NLP

- Systems understand game rules in English

General General Game Playing

NLP

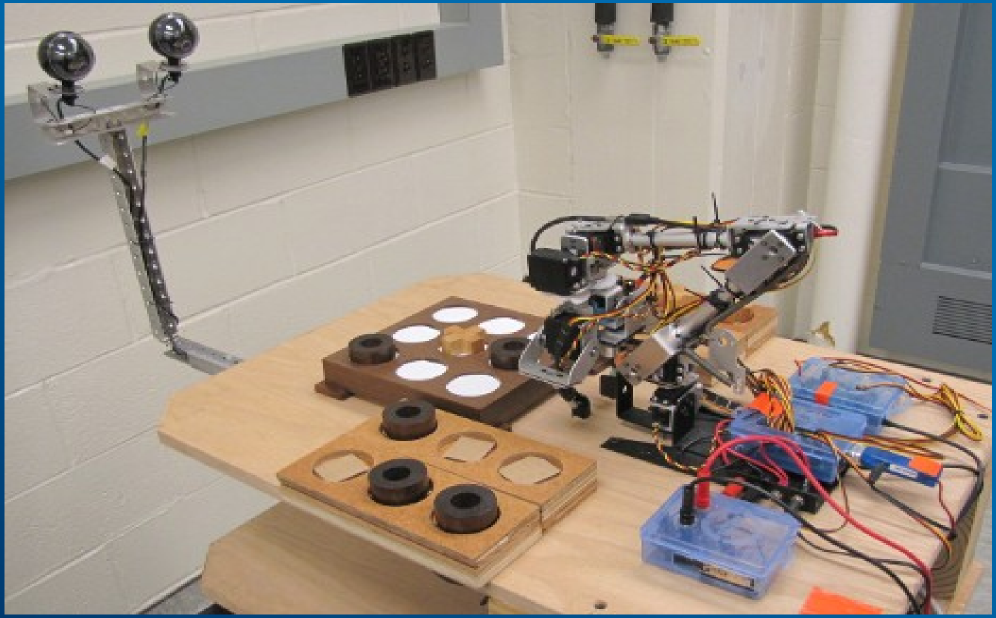
- Systems understand game rules in English

Computer vision

- Boards, pieces etc

Robotics

- GGPlaying robot



(Purdue University 2010)

Part III

—

The Potential for GGP in AI Teaching

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

Options

Several ways to add GGP to AI curricula

1. Single lecture (2hrs) as part of AI-related course

   Introduction to AI, Intelligent Systems, KR, ...

2. Graduate course (full-fledged, hands-on)

   Student teams build their own player

3. Integral part of Introduction to AI

UNSW THE UNIVERSITY OF NEW SOUTH WALES

Available Material

www.general-game-playing.de

- Slides (pdf, odp) for
  - Single lectures
  - Multiple lectures
  - Full-fledged courses
- Sample tutorials / assignments

UNSW THE UNIVERSITY OF NEW SOUTH WALES

Available Material

`www.general-game-playing.de`

- Slides (pdf, odp) for
  - Single lectures
  - Multiple lectures
  - Full-fledged courses
- Sample tutorials / assignments
- Software
  - Game Controller, Game Checker, Player
- Links to GGP courses around the world

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

GGP and AI

Search

↓

Logic / Logic Programming

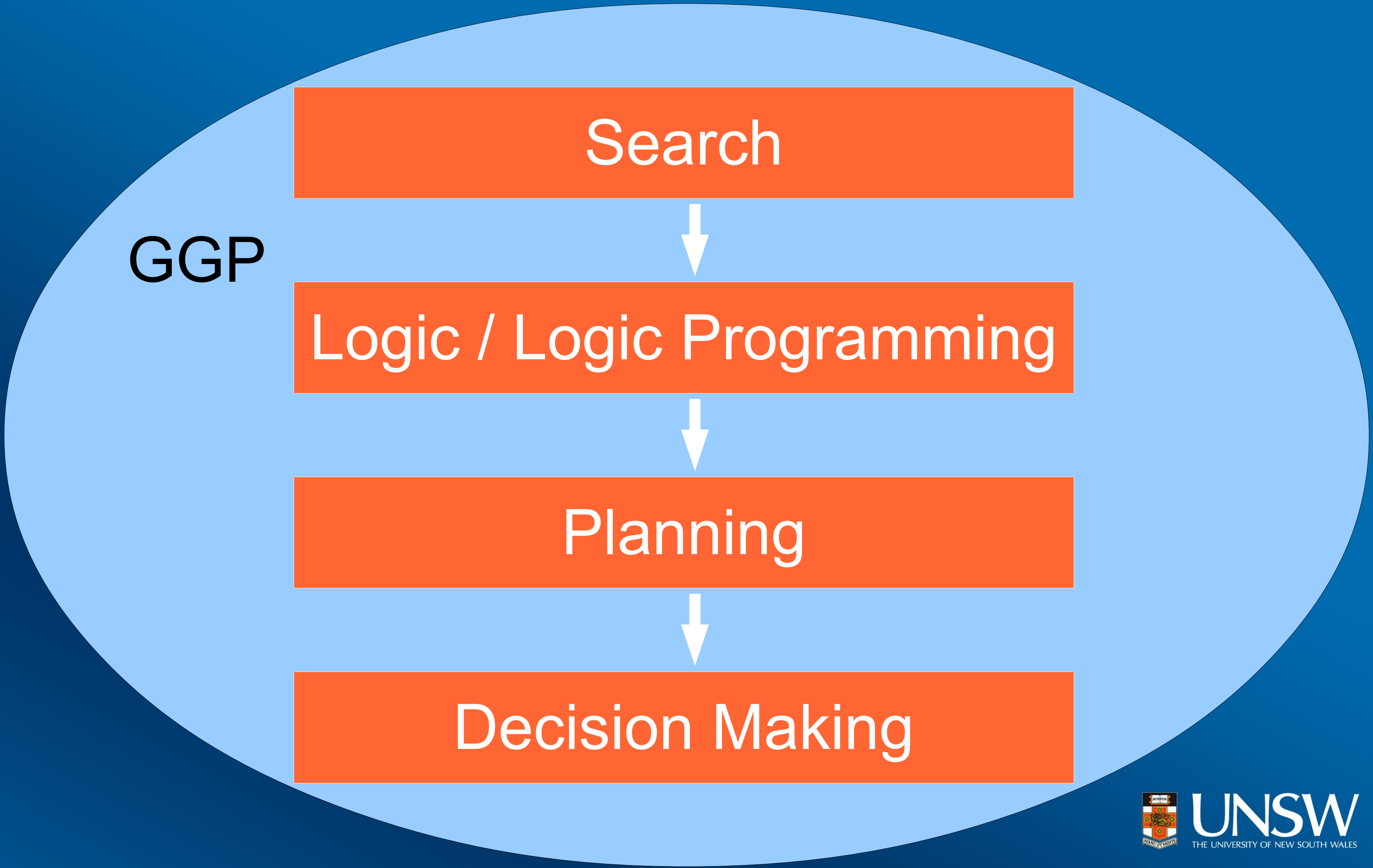
↓

Planning

↓

Decision Making

GGP and AI


GGP
Search
Logic / Logic Programming
Planning
Decision Making

Search

Task: Solve search problems

1. Search trees
2. Blind search (BFS vs. DFS, IDS)
3. Adversarial search (Minimax)
4. Informed search (assume given a heuristics)

Logic / Logic Programming

Task: Axiomatise games

1. Propositional / first-order logic

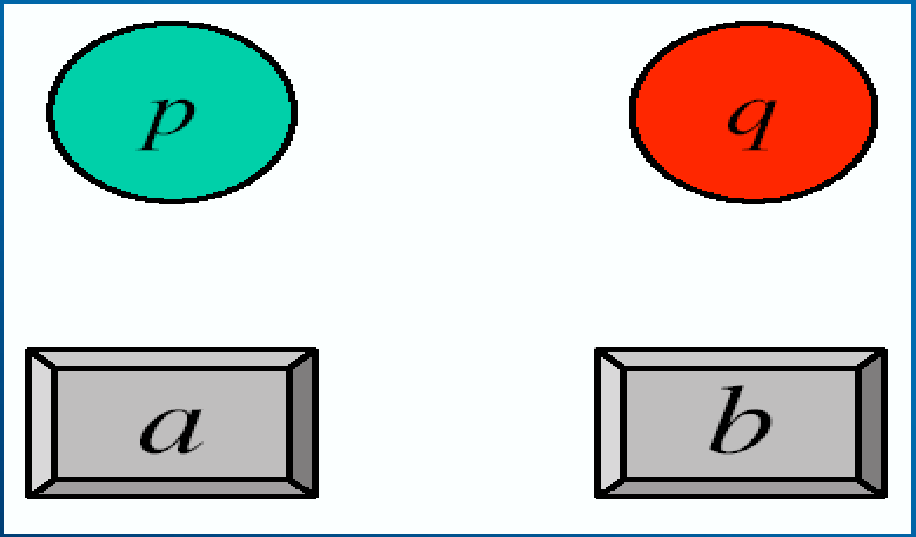
2. Logic programs

Task: Understand rules (ie. draw inferences)

1. Unification, Resolution, NAF

2. Optional: CSP / ASP (1-player games)

Example: Propositional Logic



Pressing button *a* toggles *p*.

Pressing button *b* interchanges *p* and *q*.

Initially: *p* and *q* are off.

Goal: *p* and *q* are on.

Planning

Task: Play games

1. Classic planning
2. Continuous planning
3. Conditional planning
4. Multi-agent and adversarial planning

Decision Making

Task: Play better

1. Utility functions
2. MDPs, POMDPs
3. Game theory

UNSW THE UNIVERSITY OF NEW SOUTH WALES

The End

Conclusion

GGP Research

- poses interesting research challenges for different AI sub-disciplines
- requires integration of AI methods & systems
- is an example of General Problem Solving

Conclusion

GGP Research

- poses interesting research challenges for different AI sub-disciplines
- requires integration of AI methods & systems
- is an example of General Problem Solving

GGP Teaching

- covers many aspects of AI
- has motivating competitive aspect
- GGP is "cool"