CS 440
Introduction to Artificial Intelligence

Lecture 9:

CPSs and Intro to Logic

13 February 2030

Entropy

- S is a set of examples
- $p_{\oplus}$ is the proportion of examples in class $\oplus$
- $p_{\ominus} = 1 - p_{\oplus}$ is the proportion of examples in class $\ominus$

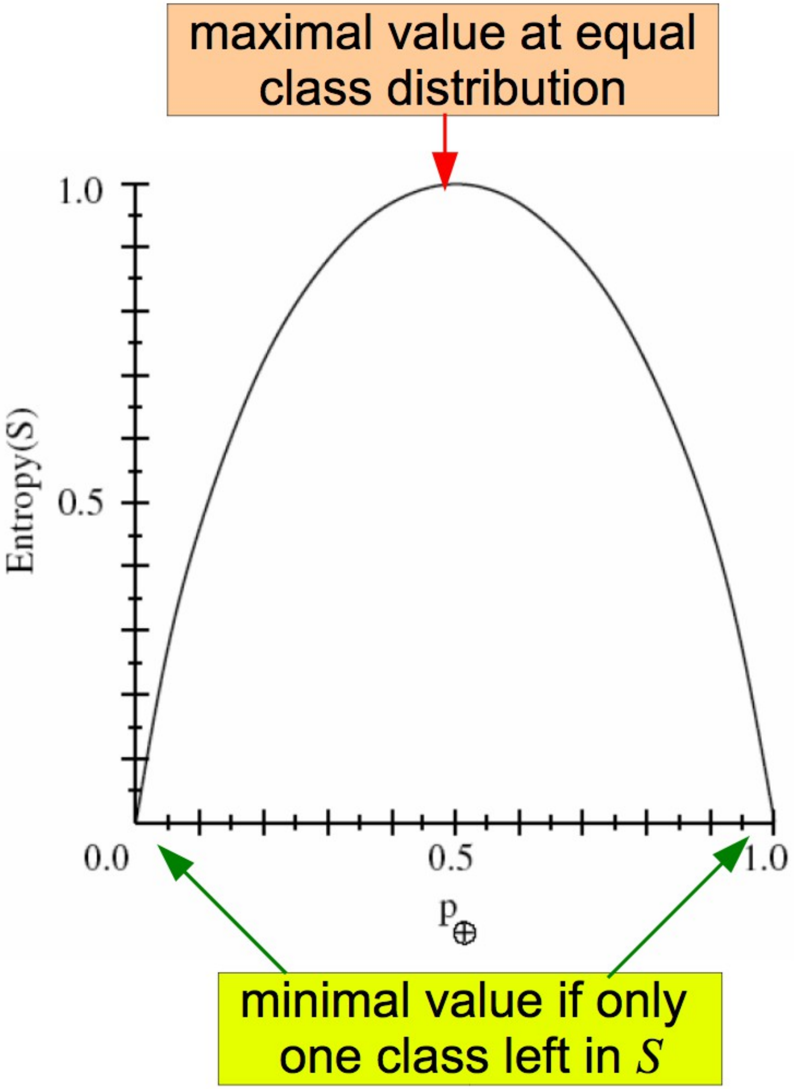
Entropy:

$$E(S) = -p_{\oplus} \cdot \log_2 p_{\oplus} - p_{\ominus} \cdot \log_2 p_{\ominus}$$

- Interpretation:
  - amount of unorderedness in the class distribution of S

For a complex (non-binary) set

$$E(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$



Split on Information Gain

- When an attribute A splits the set S into subsets S_i
  - we compute the average entropy
  - and compare the sum to the entropy of the original set S

Information Gain for Attribute A

$$Gain(S, A) = E(S) - I(S, A) = E(S) - \sum_i \frac{|S_i|}{|S|} \cdot E(S_i)$$

- The attribute that maximizes the difference is selected
  - i.e., the attribute that reduces the unorderedness most!

- **Note**:
  - maximizing information gain is equivalent to minimizing average entropy, because $E(S)$ is constant for all attributes A

Building Decision Trees

- Greedy approach
  - Recursively select trait that maximizes information gain.

Build_Decision_Tree(S, Attributes)
  - If S is monotone (all same type) or Attributes = {}
    - return
  - //Find best attribute
  - IG_{best}=-1
  - for all attributes a
    - *S'* = set of subsets from dividing S by a
      - if IG(*S'*) > IG_{best}
        - IG_{best} = IG(*S*)
        - a_{best} = a
  - *S'* = set of subsets from dividing S by a
  - For all S' in
    - Build_Decision_Tree(S', Attributes \ a_{best})

Discussion

- Will this algorithm find the optimal decision tree?
    - What would we define as an optimal decision tree?
- How could this method be modified to accommodate sets with missing traits?
- How could this method be modified to accommodate non-discrete data
  - Example:
    - Wait times = 8 min, 9 min, 11 min, 19 min
    - Exact prices

Constraint- Satisfaction Problems

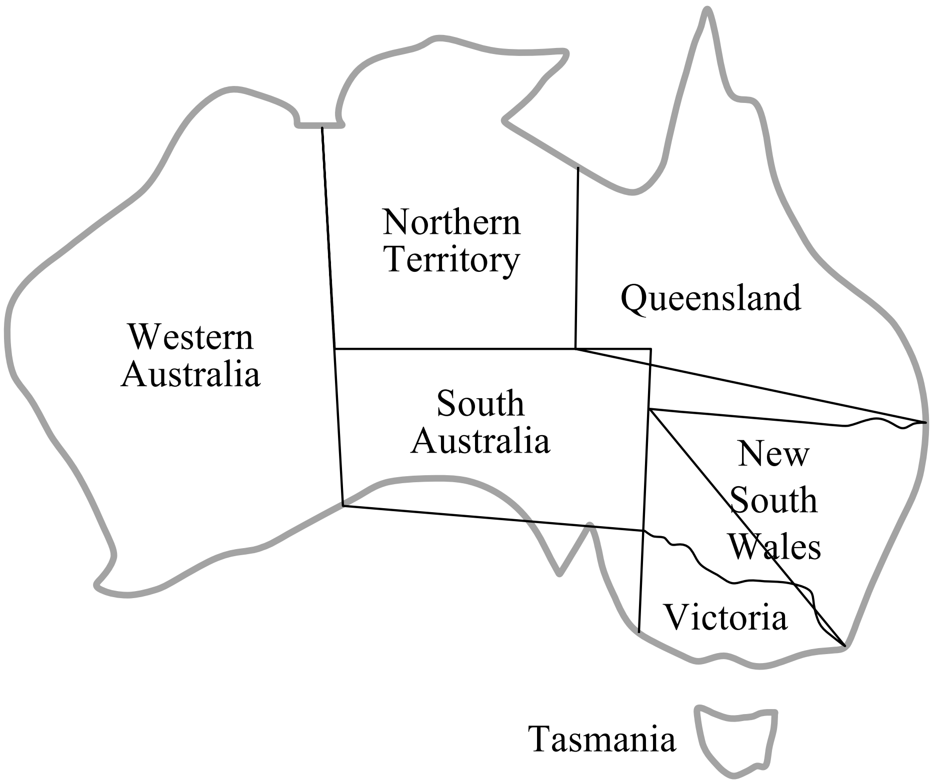
- Discrete and Finite Domains
  - E.g., map-Coloring, 8-queens puzzle
- Boolean CSPs
  - Satisfiability problems (prototypical NP-Complete problem)
- Discrete and Infinite Domains
  - Scheduling over the set of integers (e.g., all the days after today)
- Continuous Domains
  - Scheduling over continuous time
  - Linear Programming problems
    - Constraints are linear inequalities over the variables

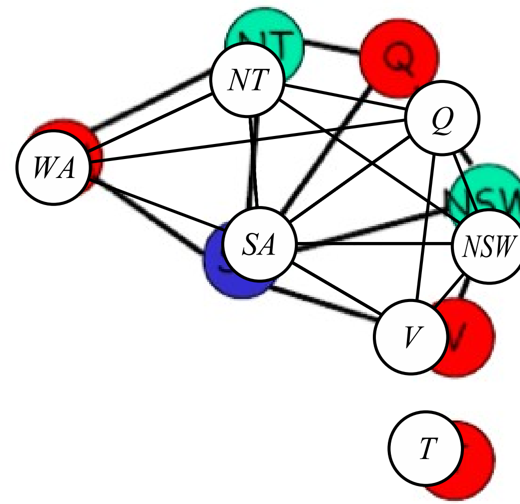
Additional examples:

- crossword puzzles, cryptography problems, Sudoku
- and many classical NP-Complete problems:
  - clique problems, vertex-cover, traveling salesman, subset-sum, hamiltonian-cycle

Example

Color the map of Australia so that no two neighboring regions have the same color





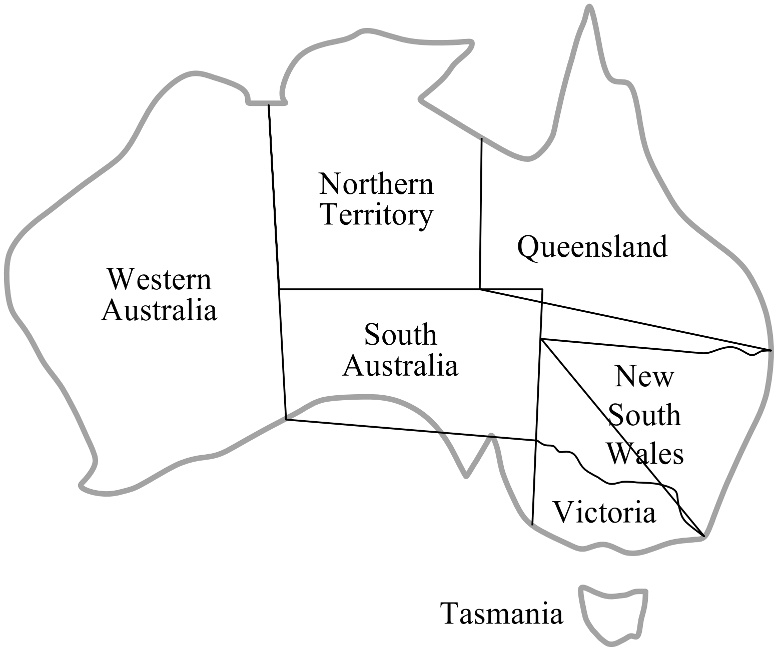
<u>Variables:</u> { WA, NT, Q, NSW, V, SA, T}

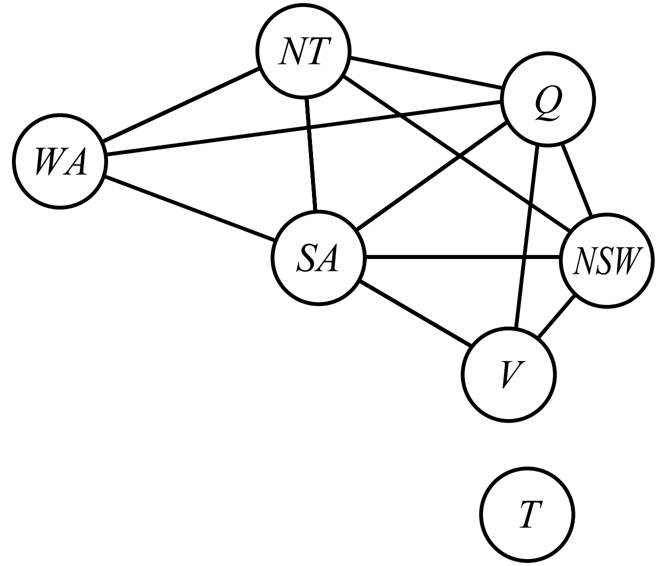
<u>Domain for each variable:</u> {red, green, blue}

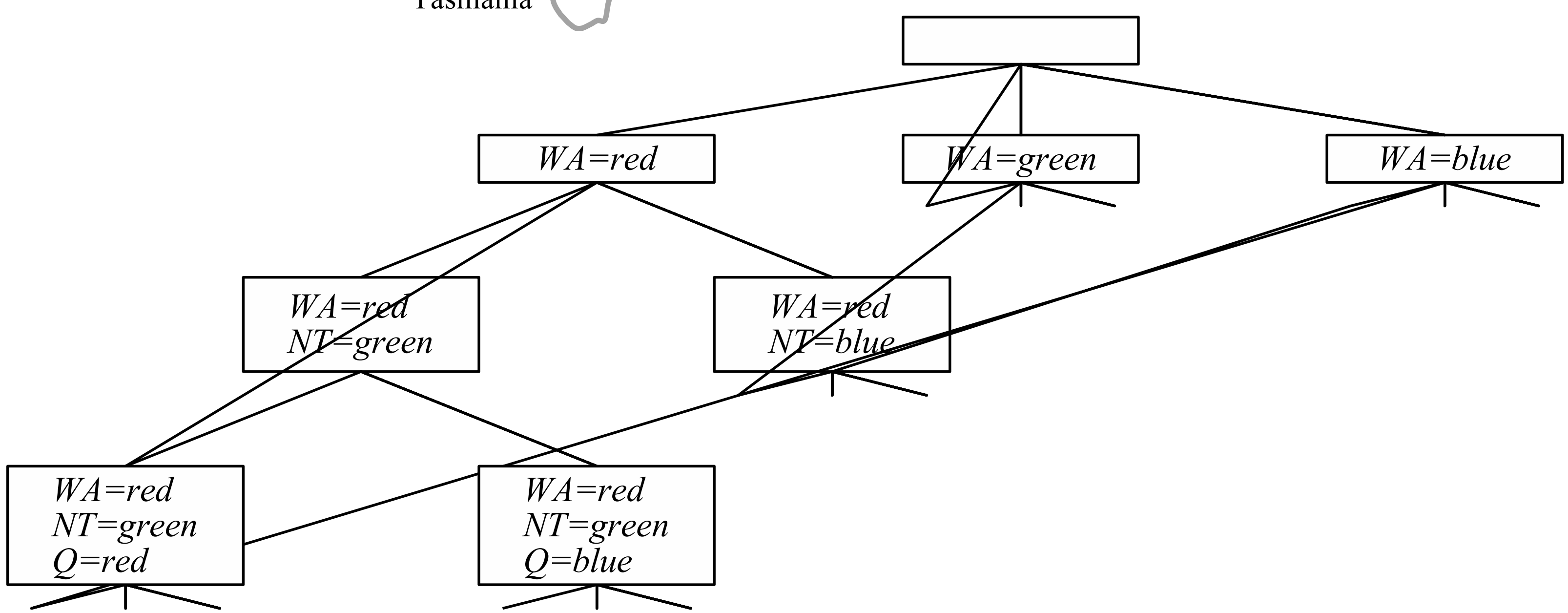
<u>Constraints:</u>

WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q,
SA ≠ Q, SA ≠ NSW, SA ≠ V,
Q ≠ NSW, NSW ≠ V

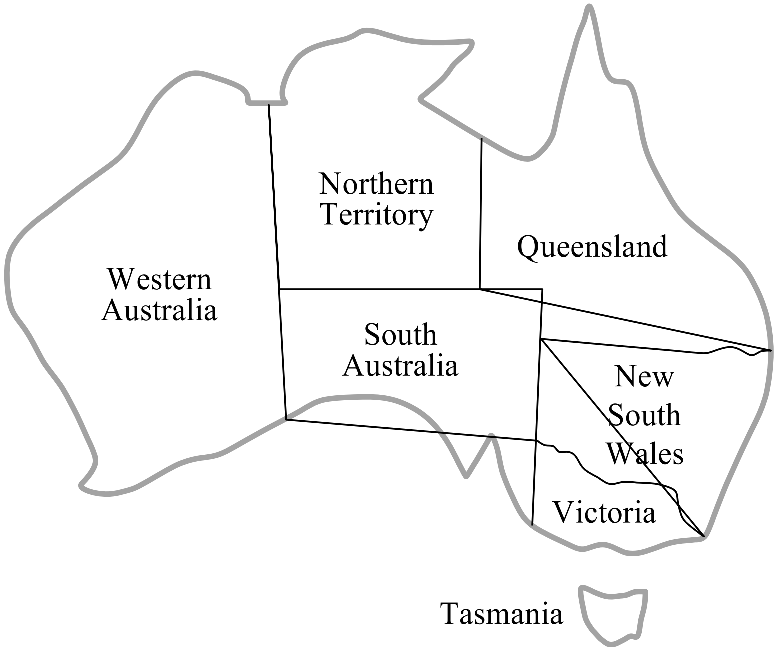
Backtracking Search

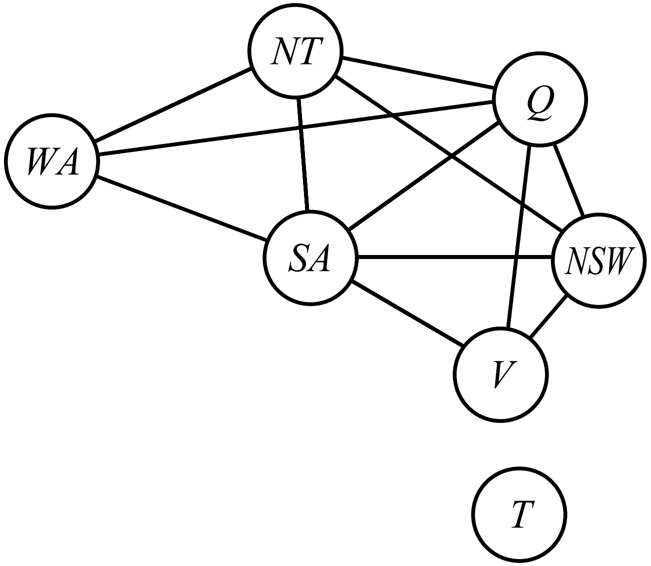






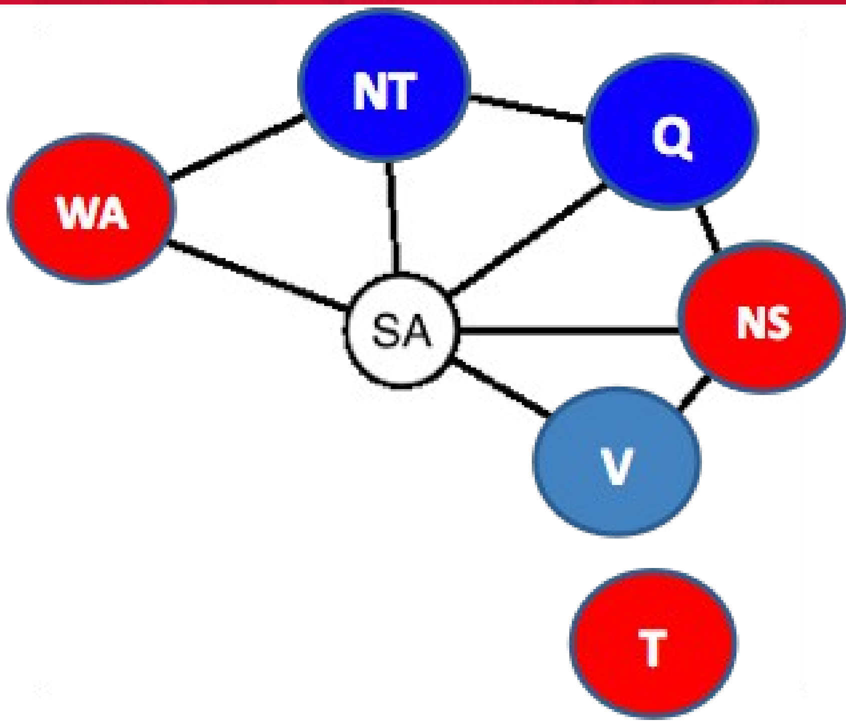
Backtracking: Forward Checking





	WA	NT	Q	NSW	V	SA	T
Initially	RGB	RGB	RGB	RGB	RGB	RGB	RGB
After WA=R	R	G B	RGB	RGB	RGB	G B	RGB
After Q = G	R	B	G	R B	RGB	B	RGB
After V=B	R	B	G	R	B		RGB

Backtracking: Intelligent Backjumping



- Order: Q, NS, V, T, SA
- Failure when trying to assign SA
- SA's conflict set {Q, NS, V}
- *Backjump* to the latest node in the conflict set: V
- Skip Tasmania

Assume WA=red and NSW =red, then assign T, NT, Q, SA

SA will cause a conflict, whatever we do...

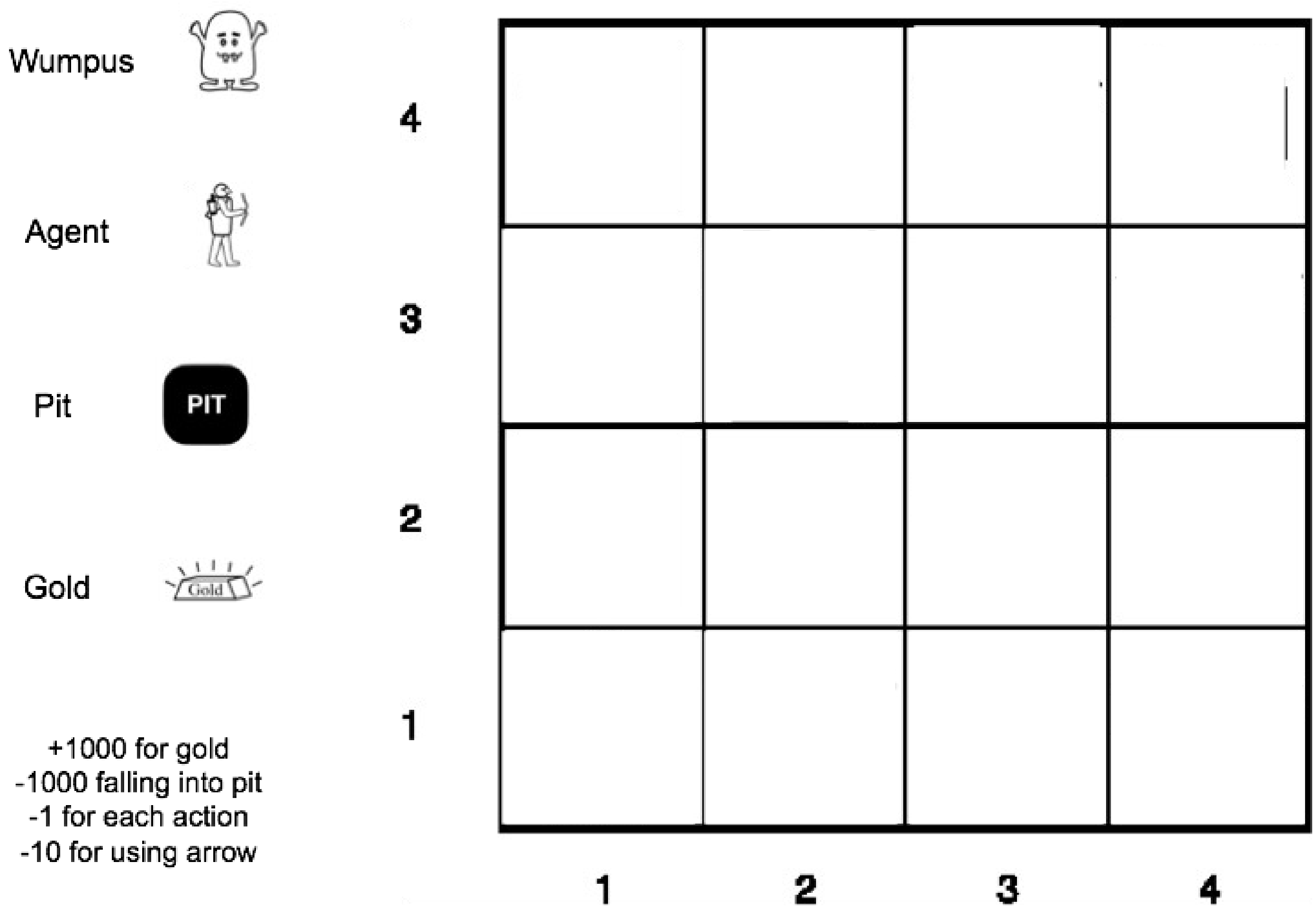
- Where should the algorithm backjump?

CSP Examples: Robotics

- **Robot task planning**
  - **Conditions needed to accomplish task**
  - **Example - door must be open to enter room**

RUTGERS

Binary CSP example: Wumpus World

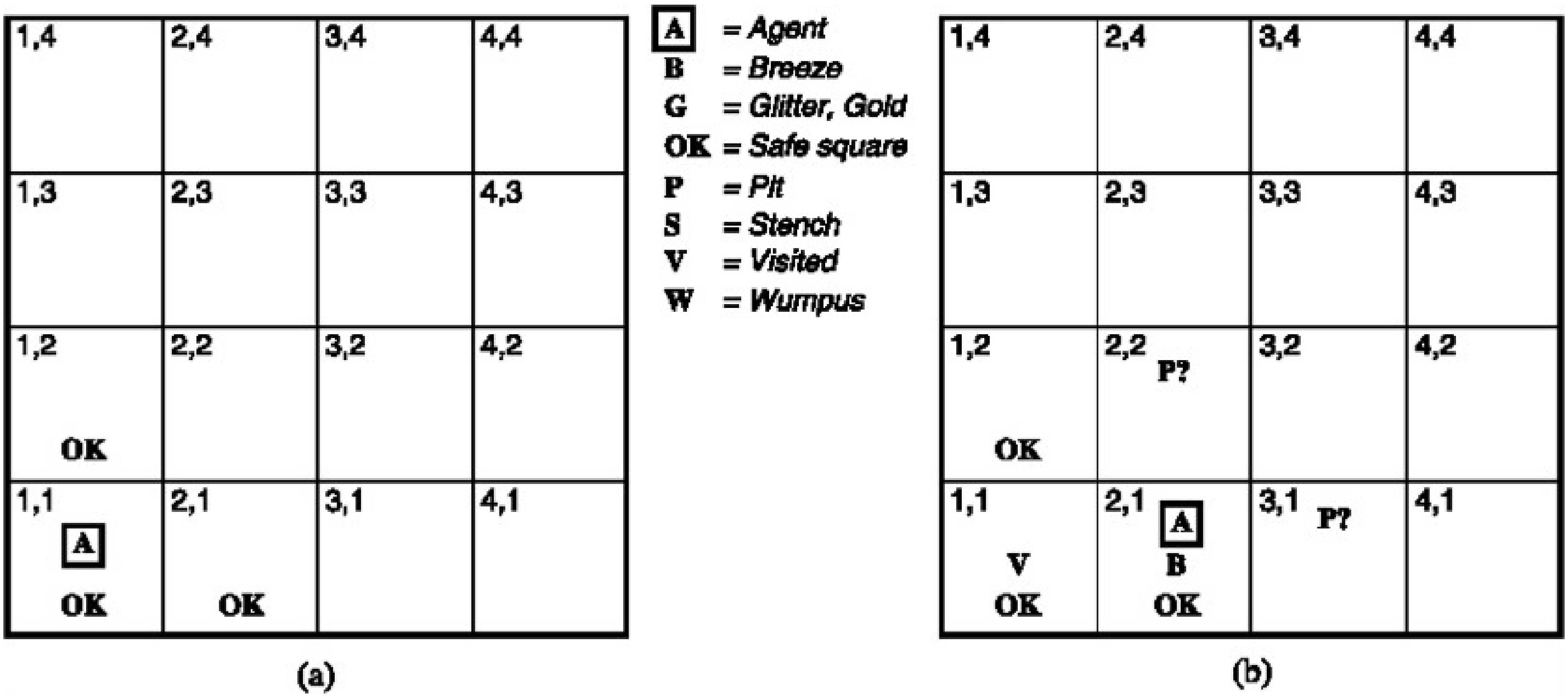


RUTGERS

Truth Tables

[redacted]	[redacted]	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
[redacted]	[redacted]	TRUE	FALSE	FALSE	TRUE	TRUE
[redacted]	[redacted]	TRUE	FALSE	TRUE	TRUE	FALSE
[redacted]	[redacted]	FALSE	FALSE	TRUE	FALSE	FALSE
[redacted]	[redacted]	FALSE	TRUE	TRUE	TRUE	TRUE

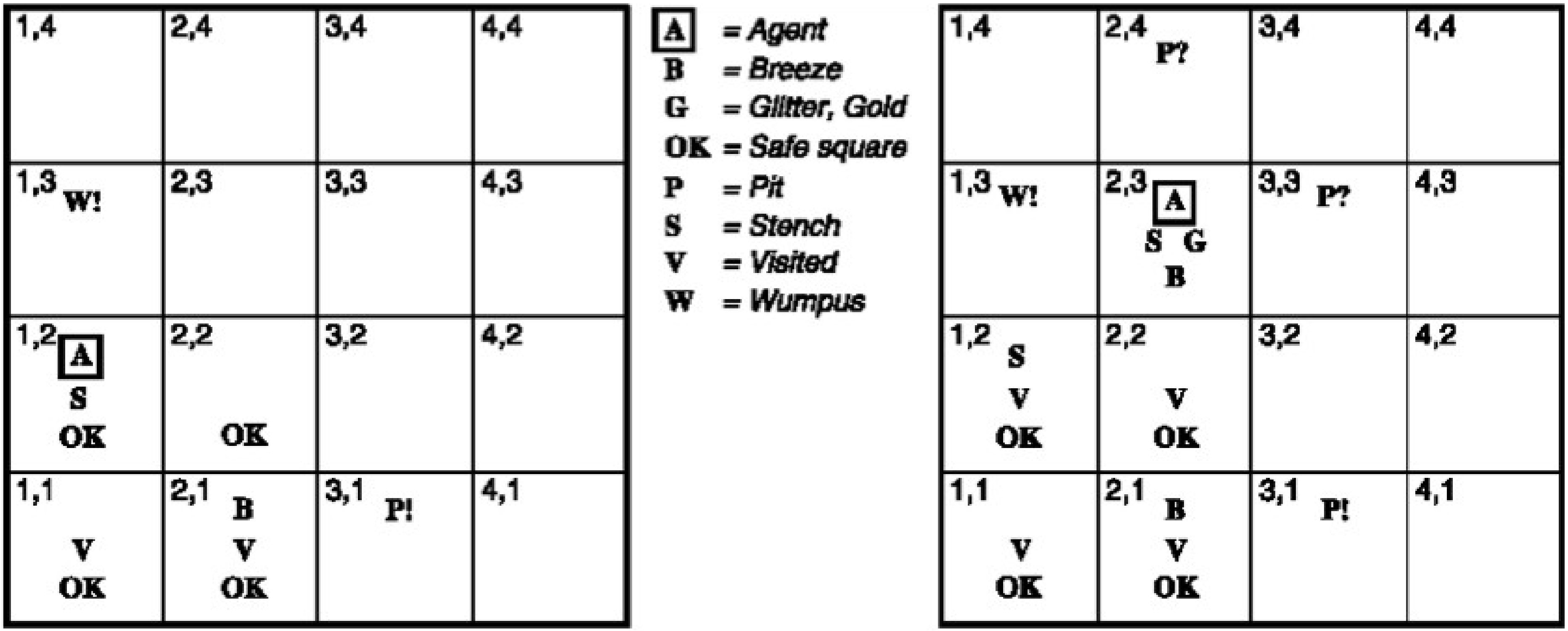
Wumpus World



Truth Table for Wumpus World

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Wumpus World



(a)

(b)

Wumpus World

+1000 for gold
-1000 falling into pit
-1 for each action
-10 for using arrow