CS 440
Introduction to Artificial Intelligence

Lecture 10:

Backtracking and Logic

17 February 2020

Constraint- Satisfaction Problems

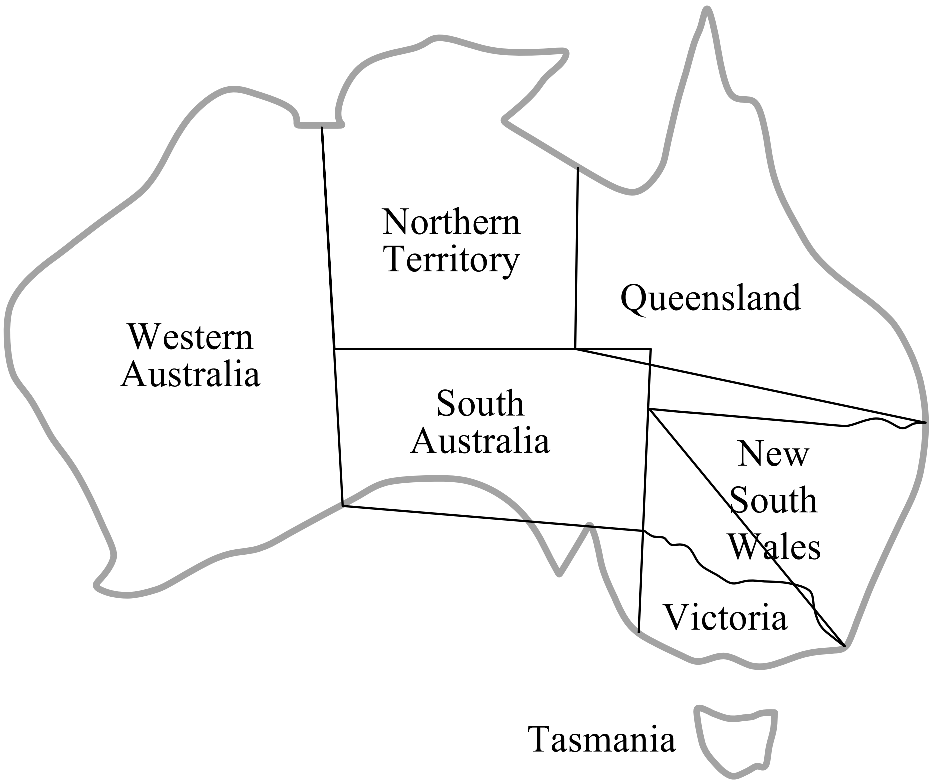
- Discrete and Finite Domains
  - E.g., map-Coloring, 8-queens puzzle
- Boolean CSPs
  - Satisfiability problems (prototypical NP-Complete problem)
- Discrete and Infinite Domains
  - Scheduling over the set of integers (e.g., all the days after today)
- Continuous Domains
  - Scheduling over continuous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

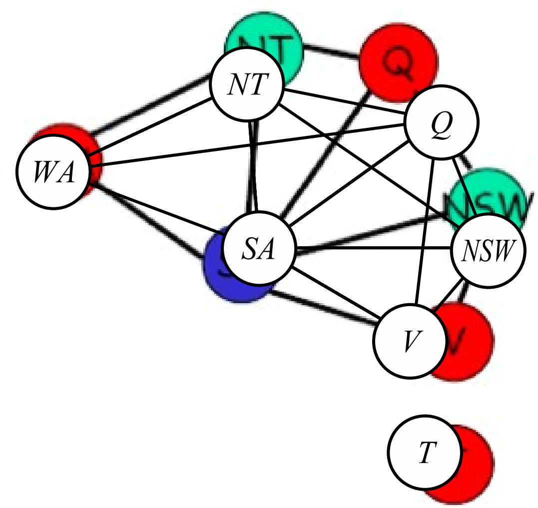
Additional examples:

- 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





Variables: { WA, NT, Q, NSW, V, SA, T}

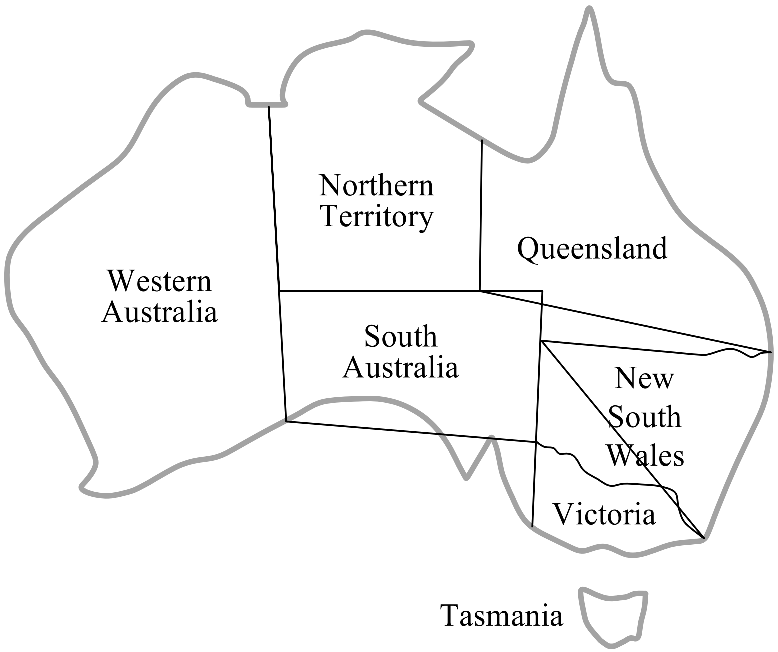
Domain for each variable: {red, green, blue}

Constraints:

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

Backtracking: Intelligent Backjumping

- Iteratively set each variable to valid value
  - Valid given constraints and values of other variables
- Set variables that have only one valid value
  - All variables set $\Rightarrow$ solution found
- If any variable has no valid values
  - Current setting cannot produce solution
  - Backtrack
    - Roll back variable settings



Backtracking Algorithm

Backtracking(X, C, S)

Input: A set of variables **X**, a set of constraints **C** and a partial setting **S**

While ∃ **x** ∈ **X** that has only one valid value, v

Set **x** to **v** in **S**

If all variables have been set

Return **S**

If ∃ **x** ∈ **X** with no valid values

return **FAIL**

Let x = an unset value in **X**

For all valid settings of **x** as **v**

S' = **S**

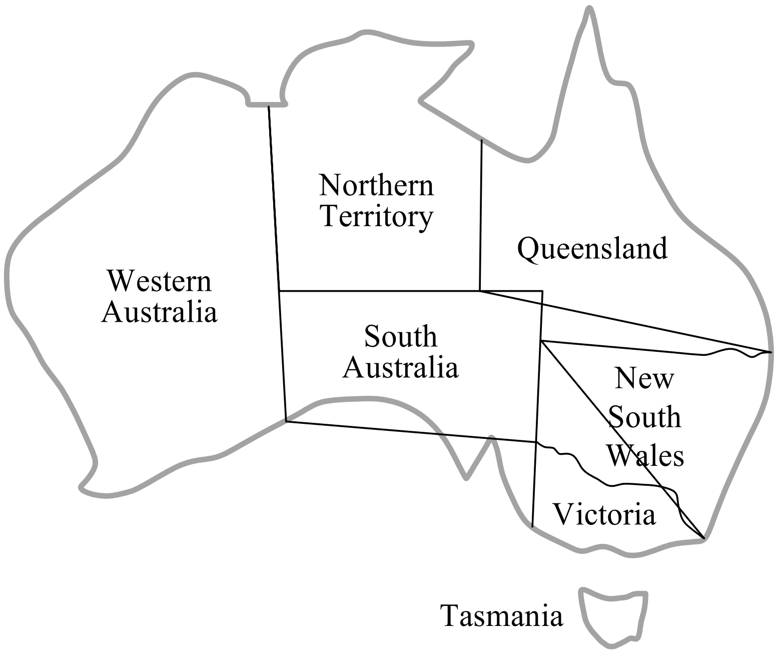
Set x to v in S'

result = Backtracking(**X, C, S'**)

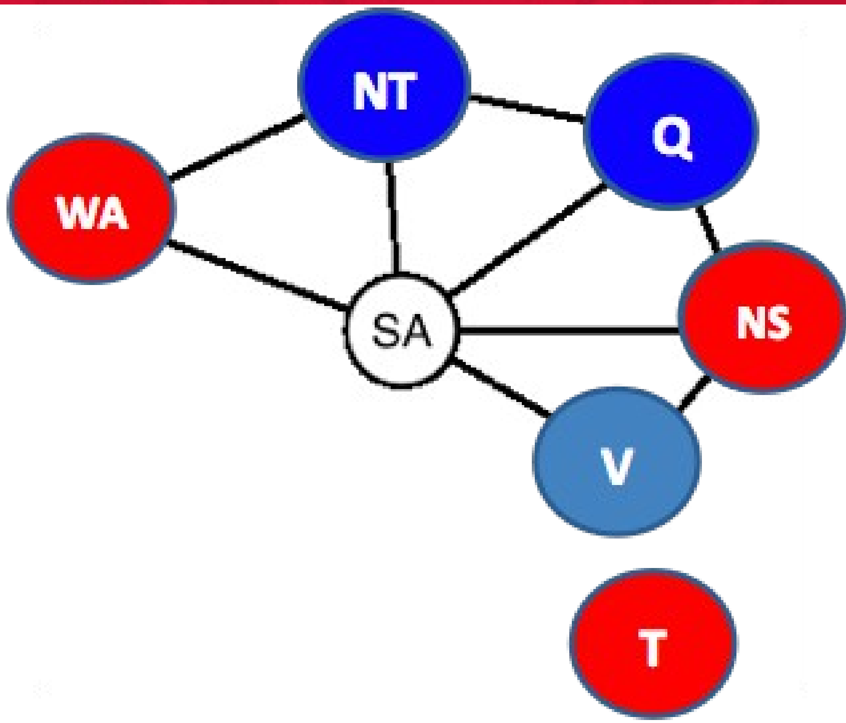
if(result ≠ **FAIL**)

Return **result**

Return **FAIL**



Backtracking:Example



- 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?

Backtracking

Will find a solution if one exists

Will return FAIL if one does not exist

Complexity: $O(k^n)$

Constraint satisfaction problem

Variables can be set to **true** or **false**

Denote variables as $\mathbf{X} = \{x_1, x_2, \ldots\}$

Constraints introduced by set of logical statements

Examples:

- $f(\mathbf{X}) = \neg\, x_1$
- $g(\mathbf{X}) = x_1 \vee x_2$
- $h(\mathbf{X}) = f(\mathbf{X}) \wedge g(\mathbf{X})$
  - $h(\mathbf{X}) = \neg x_1 \wedge (x_1 \vee x_2)$
- $j(X) = x_1 \rightarrow x_2$
- $k(X) = l(X) \leftrightarrow m(X)$

Hint: Think about what sets of variable settings satisfy a statement

DNF/CNF

Disjunctive Normal Form (DNF) i

OR of ANDs

Example: $(x_1 \wedge \neg x_2) \vee (x_3 \wedge \neg x_2 \wedge x_4 \wedge x_1) \vee (x_5 \wedge \neg x_6)$

Conjunctive Normal Form (CNF)

ANDs of ORs

Example: $(x_1 \vee x_2) \wedge (x_3 \vee x_2 \vee \neg x_4 \vee x_1) \wedge (\neg x_5 \vee x_6)$

Any logical statement can be reduced to either form

Allow backtracking algorithm to be applied to problem

- Exponential complexity

Axioms

- Statements that are always true
- True for all possible settings of variables

$x_1 \vee \neg x_1$

Commutativity:

- $f(X) \vee g(X) \Leftrightarrow g(X) \vee f(X)$
- $f(X) \wedge g(X) \Leftrightarrow g(X) \wedge g(X)$

Transitivity:

- $(f(X) \rightarrow g(X) \wedge g(X) \rightarrow h(X)) \rightarrow (f(X) \rightarrow h(X))$

Distributive:

- $f(X) \vee (g(X) \wedge h(X)) \Leftrightarrow (f(X) \vee g(X)) \wedge (f(X) \vee h(X))$
- $f(X) \wedge (g(X) \vee h(X)) \Leftrightarrow (f(X) \wedge g(X)) \vee (f(X) \wedge h(X))$

How would you prove something is an axiom?

How would you prove something isn't an axiom?

Axioms

How would you prove something is an axiom?

- Reduction
  - Reduce to a statement that is trivially true
- Contradiction
  - Assume axiom if false and show contradiction
    - Variable that can neither be true of false

How would you prove something isn't an axiom?

- Setting of variables for which statement is false

How would an AI agent prove something is an axiom?

Entailment

f(X) entails g(X)

- $f(X) \models g(X)$
- For all variable settings where f(X) is true, g(X) is also true
  - $g(X) \vee \neg f(X)$
- f(X) is sufficient to show g(X)
- f(X) implies g(X)

Examples:

- $\neg x_1 \wedge (x_1 \vee x_2) \models x_2$

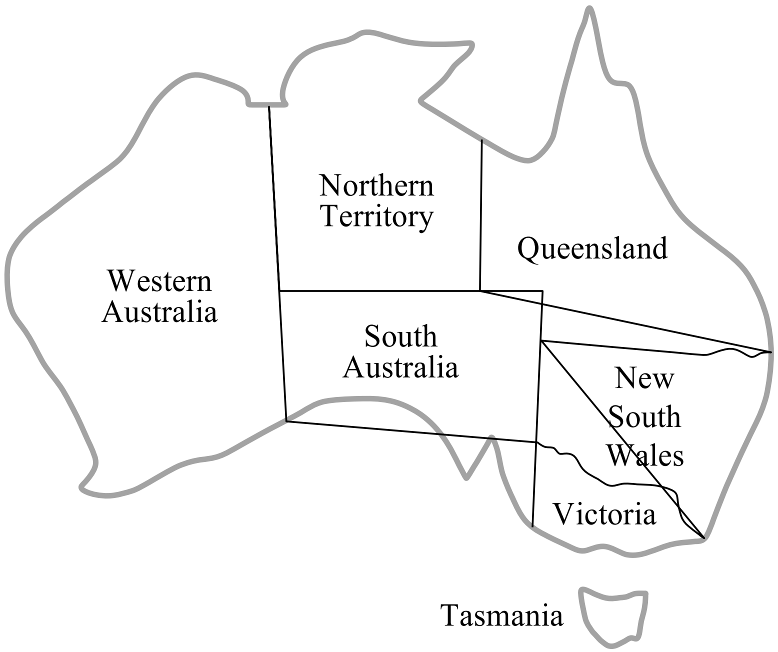
Entailment vs implies

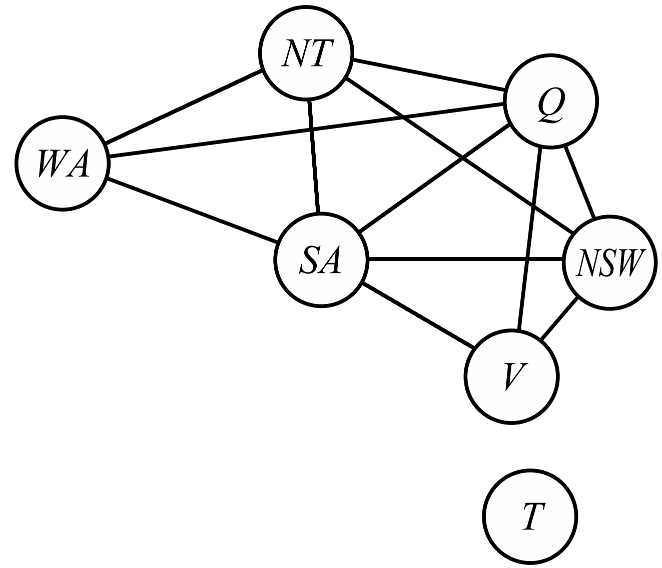
- $\models$ **vs** $\rightarrow$

Modus Ponens

- **If x implies y and x is true then you can entail that y must also be true**
  - $x \rightarrow y,\ y \models y$
- **Example**
  - **If you don't study then you will fail the exam**
  - **You did not study**
  - **Therefore, you will fail the exam**

Backtracking: Forward Checking





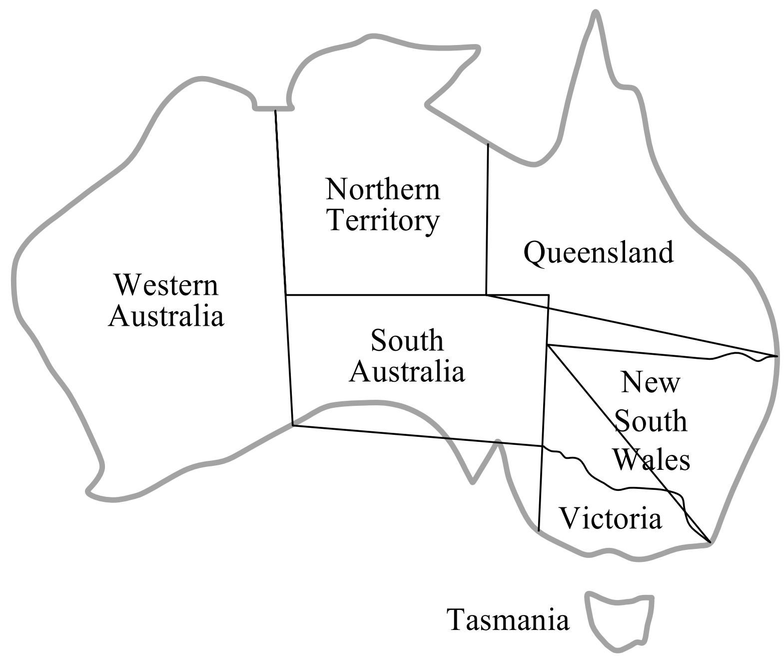
Initially	RGB	RGB	RGB	RGB	RGB	RGB	RGB
After WA=R	R	G B	RGB	RGB	RGB	G B	RGB
After Q = G	R	B	G	R B	RGB	B	RGB
After V=B	R	B	G	R	B		RGB

Backtracking Algorithm

Backtracking(X, C, S)

Input: A set of variables **X**, a set of constraints **C** and a partial setting **S**

```
While ∃ x ∈ X that has only one valid value, v
    Set x to v in S
If all variables have been set
    Return S
If ∃ x ∈ X with no valid values
    return FAIL
Let x = an unset value in X
For all valid settings of x as v
    S' = S
    Set x to v in S'
    result = Backtracking(X, C, S')
    if(result ≠ FAIL)
        Return result
Return FAIL
```



Binary CSP example: Wumpus World

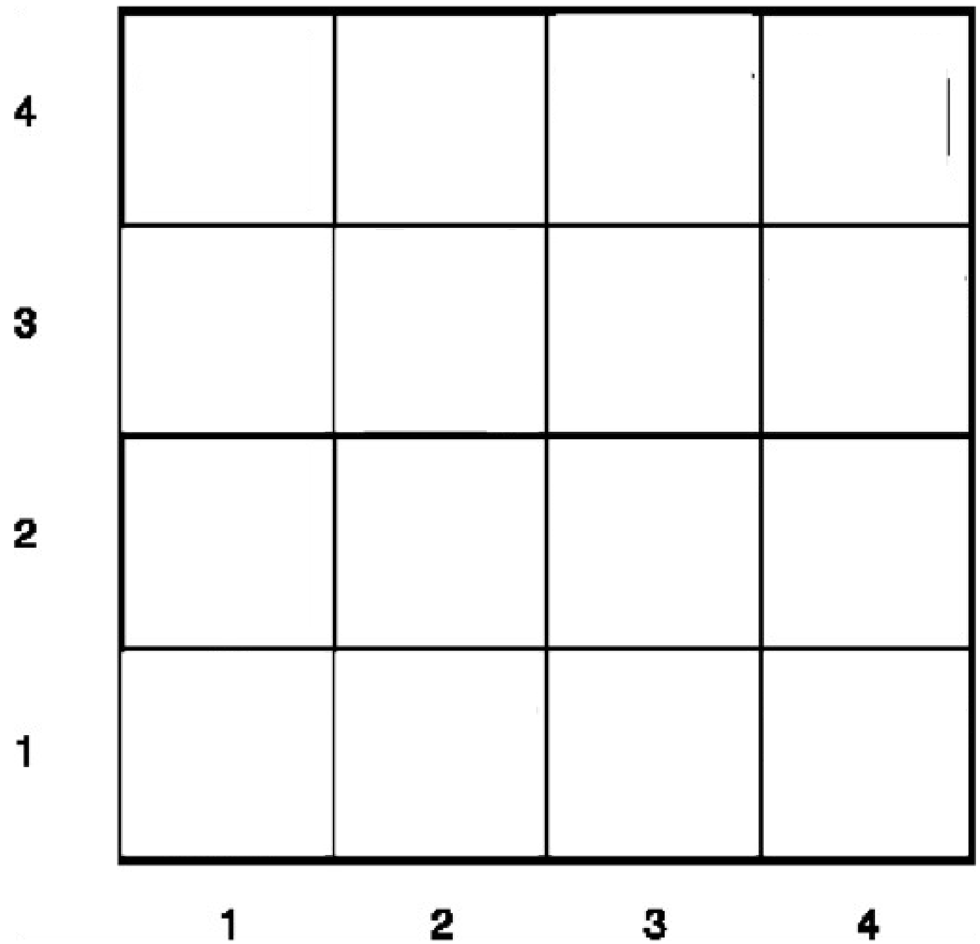
Wumpus 

Agent 

Pit 

Gold 

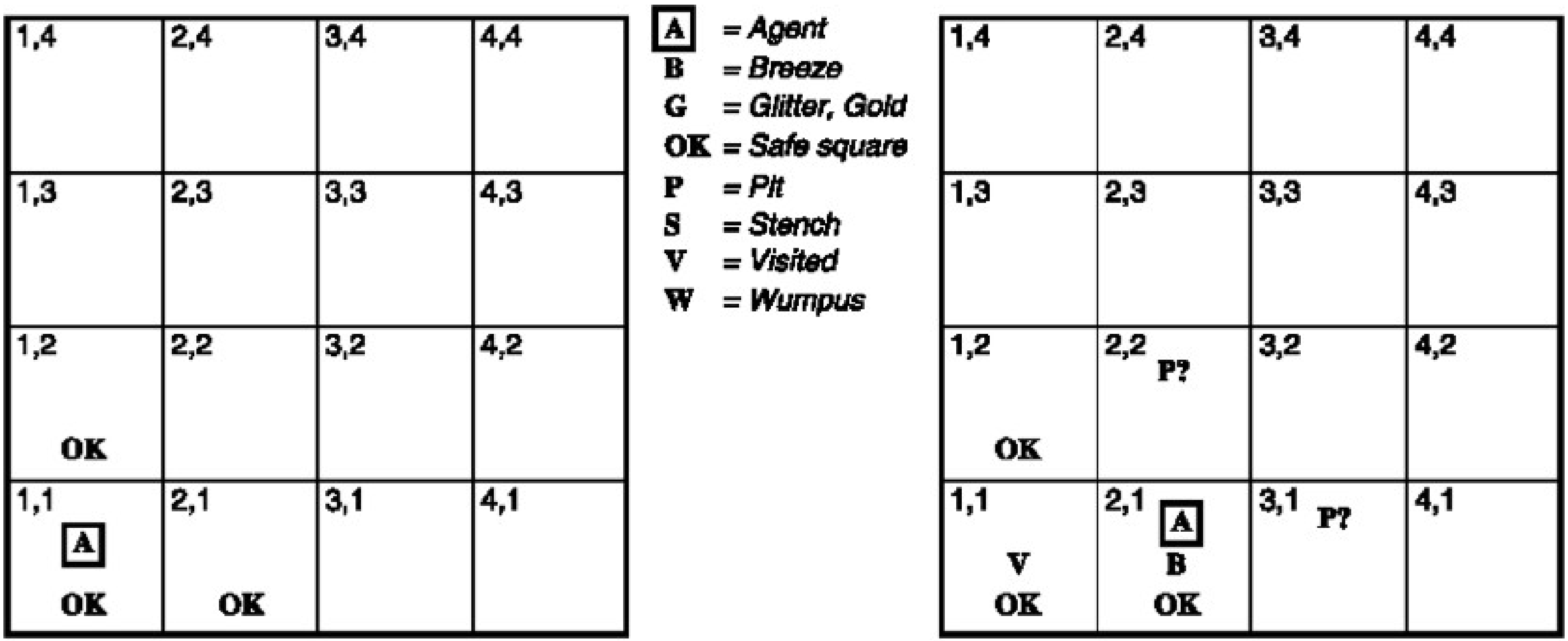
+1000 for gold
-1000 falling into pit
-1 for each action
-10 for using arrow



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



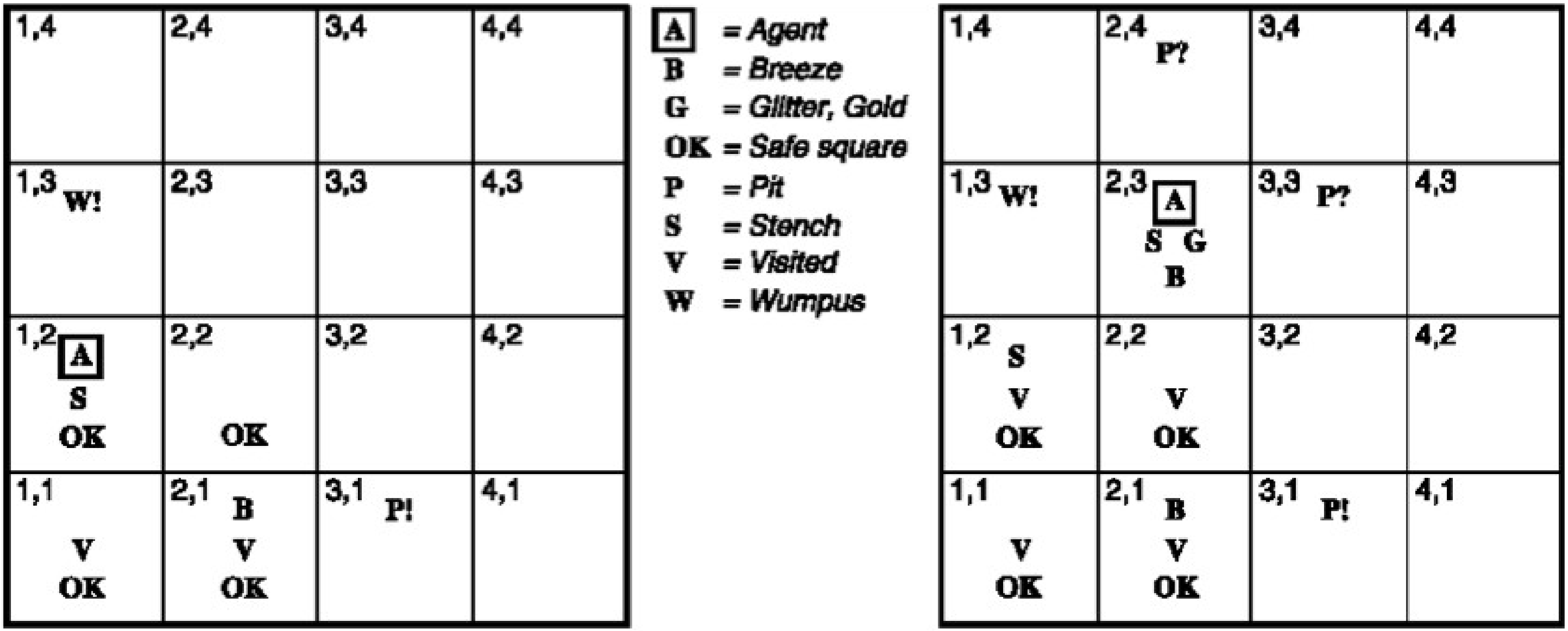
(a) (b)

RUTGERS

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*

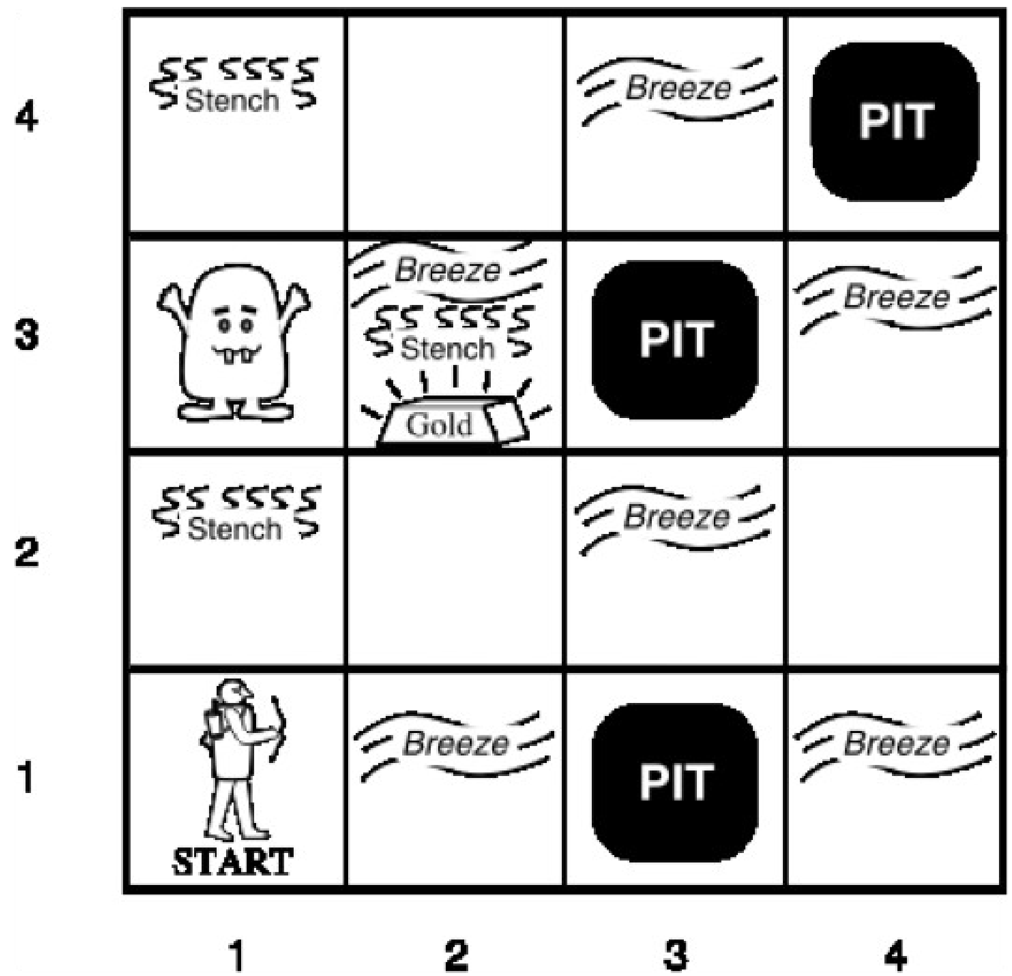
Wumpus World



(a) (b)

Wumpus World

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



CSP Examples: Robotics

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