

AI Approaches

Constraint Satisfaction Problem (CSP)
Computational Logic
Constraint Logic Programming

Learning Outcomes

- After completing this session, you will be able to:
 - Explain the concept of constraint satisfaction problems (CSP)
 - Approaches in solving CSP
 - Knowledge-based agents
 - Mathematical Logic
 - Traditional AI approach of Constraint Logic Programming with Prolog

Constraint Satisfaction Problem (CSP)

- Constraint Satisfaction Problems are described as:
 - a set of variables with values
 - $X = \{x_1, \dots, x_n\}$ where X is a set of variables
 - domains (all the possible values the variables can take)
 - $D = \{d_1, \dots, d_n\}$ where D is a set of values - one for each variable
 - a set of constraints C
 - $C = \{ \langle \text{scope}, \text{rel} \rangle \}$ where C is a set of constraints that specify allowable combinations of values
 - *Scope* = tuple of variables
 - *rel* = relation that defines the values
- Example:
 - $X = \{x_1, x_2\}$
 - $D = \{A, B\}$
 - The constraint “two variables must have different values”
 - $C = \{ \langle (x_1, x_2), [(A, B), (B, A)] \rangle \}$ Or $\{ \langle (x_1, x_2), x_1 \neq x_2 \rangle \}$

The problem is solved when each variable has a value that satisfies all of the constraints on that value.

Constraint Satisfaction Problem (CSP)

- Create a set of values from domain that satisfies the constraint:
 - $X = \{x_1, x_2\}$
 - $D = \{2,3,4,5\}$
 - $C = \{<(x_1, x_2), x_1 < x_2\}$

$[(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)]$

A CSP Example

Colour the map of the Australia with three colours {Red, Blue, Green} in such a way that neighbouring state(s) is/are coloured differently.

Variables: $X = \{WA, NT, Q, NSW, V, SA, T\}$

Domains: $D = \{ \text{Red}, \text{Blue}, \text{Green} \}$

Constraints: States that share a border must be a different colour

$C = \{ SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V \}$

Note: $SA \neq WA$ is a shortcut for $\langle (SA, WA), SA \neq WA \rangle$



Constraint Satisfaction Problems (CSP)

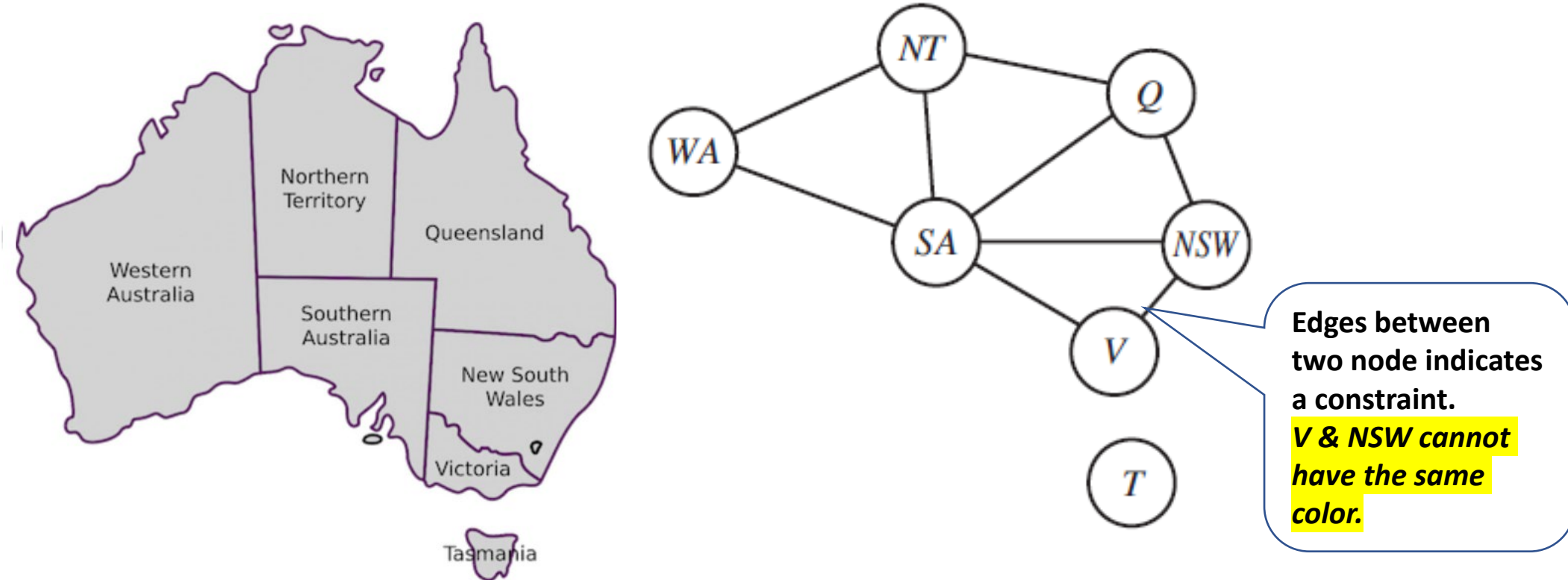
One of the solutions:



Image source: <https://towardsdatascience.com/solving-sudoku-with-ai-d6008993c7de>

{WA=Red ,NT=Green, Q=Red ,NSW=Green, V=Red ,SA=Blue, T=Red }.

Constraint Graph - Map colouring CSP



- Constraint graph: nodes are variables, arcs show constraints.
- Binary CSP: each constraint relates at most two variables.

Why formulate a problem as a CSP?

- CSPs yield a natural representation for a wide variety of problems;
- CSP approach saves time and space compare to brute force search
 - CSP solvers can be faster than state-space searchers because they can quickly eliminate large swatches of the search space.
- **Example:**
- $\{SA=Blue\} \rightarrow \langle \{WA, NT, Q, NSW, V\} \neq Blue \rangle$
- With a brute force search procedure: $3^5 = 243$ (states^{Colour}) assignments for the five neighboring variables
- With constraint propagation : $2^5 = 32$ assignments to look at, a reduction of 87%.

Constraint Satisfaction Problems

Types of Constraints

- **Unary constraints** – restricts a single value *e.g.* $\langle (SA), SA \neq green \rangle$
- **Binary constraints** – relates two variables *e.g.* $SA \neq WA$
- **Global constraints** –
 - Eg: **Alldiff** - all of the variables involved in the constraint must have different values,
Eg: Sudoku
 - *It may not involve all the variables in a problem though*
- **Preference constraints** – assign costs/preference:
 - *Example: red is better than green*

Constraint Satisfaction Problems

Types of Constraints (*cont.*)

- Higher order constraints

- Eg: You are eligible for the Merdeka Generation Package if you:
 - 1.were born from 1 January 1950 to 31 December 1959; **and**
 - 2.became a Singapore citizen on or before 31 December 1996.

-OR-

- 1.were born on or before 31 December 1949; **and**
- 2.became Singapore citizens on or before 31 December 1996; **and**
- 3.do not receive the Pioneer Generation Package.

Constraint Satisfaction Problems

- Domains of CSP
 - Discrete Finite-Domain
 - Variable have values in fixed set of domain.
 - Eg: *map colouring problem, N Queen problem*
 - Discrete Infinite-Domain (Integers, Strings, etc.)
 - Describe using Constraint language. Eg: $Time1 + duration1 \leq Time2$
 - Continuous domain
 - Continuous domain are common in the real world and are widely studied in the field of operations research.
 - Example: The scheduling of experiments on the Hubble Space Telescope requires very precise timing of observations; the start and finish of each observation and maneuver are continuous-valued variables that must obey a variety of astronomical, precedence, and power constraints.

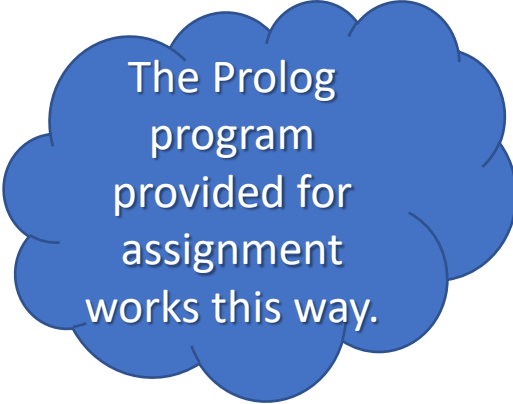
CSP Algorithms

- A CSP Algorithm can do two things in general:
- **Search**: choose a new variable assignment from many possibilities
- **Inference**: *Constraint Propagation*
 - use the constraints to reduce the number of values for a variable which will reduce the legal values of other variables.
 - Map colouring example: {SA=Blue} \rightarrow $\langle \{WA, NT, Q, NSW, V\} \neq \text{Blue} \rangle$
 - Constraint propagation can be intertwined with search
 - Constraint propagation can sometimes solve the problem entirely without search

Backtracking search (BTS) in CSP

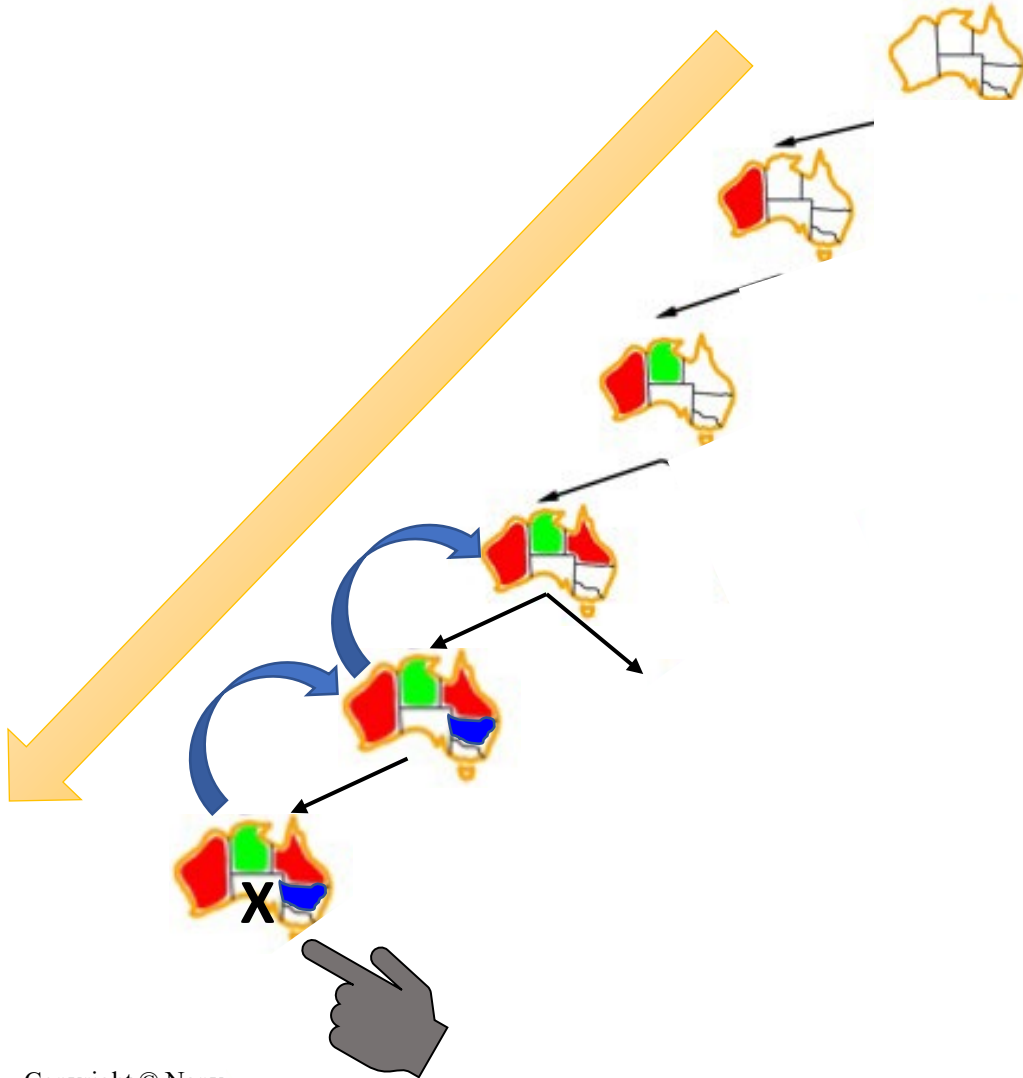
Backtracking search (BTS) is the basic uninformed search for CSPs.

- It's a Depth-First-Search (DFS) s.t.
 1. Assign one variable at a time: *assignments are commutative*. e.g., (WA=red, NT=green) is same as (NT=green, WA=red)
 2. Check constraints on the go: consider values that do not conflict with previous assignments.
- Map colouring example again:
 - **Initial state**: empty assignment {}
 - **States**: are partial assignments
 - **Successor function**: assign a value to an unassigned variable
 - **Goal test**: the current assignment is complete and satisfies all constraints



The Prolog
program
provided for
assignment
works this way.

Back tracking Search (BTS)



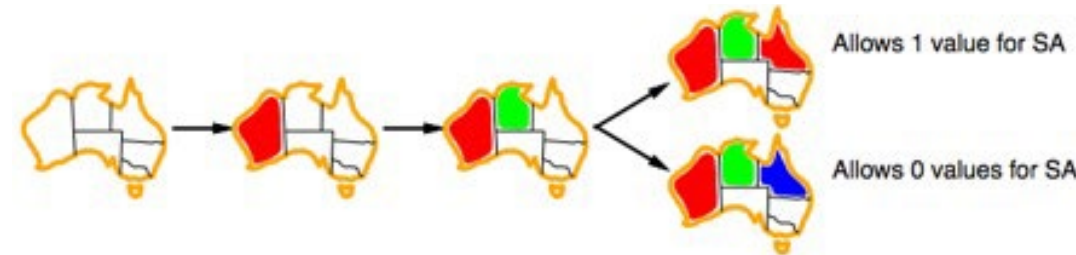
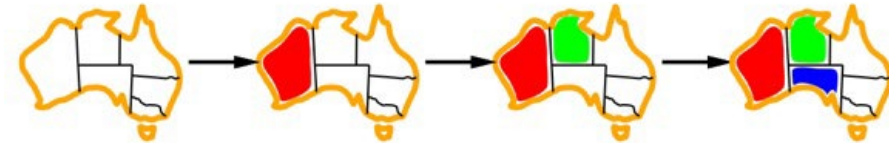
Considerations (Inferences)

- Which state should we colour first?
- What is the best possible order of $\langle \text{state, colour} \rangle$ that does not requiring too much backtracking?
- etc.

Improving BTS



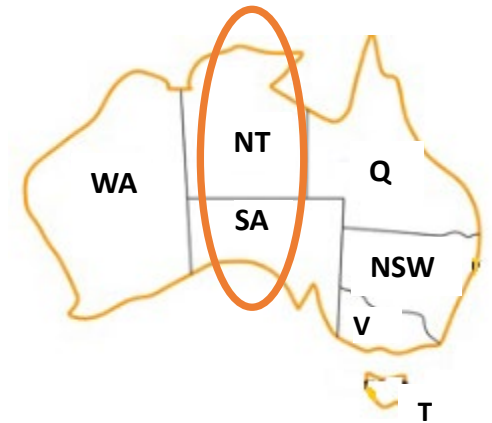
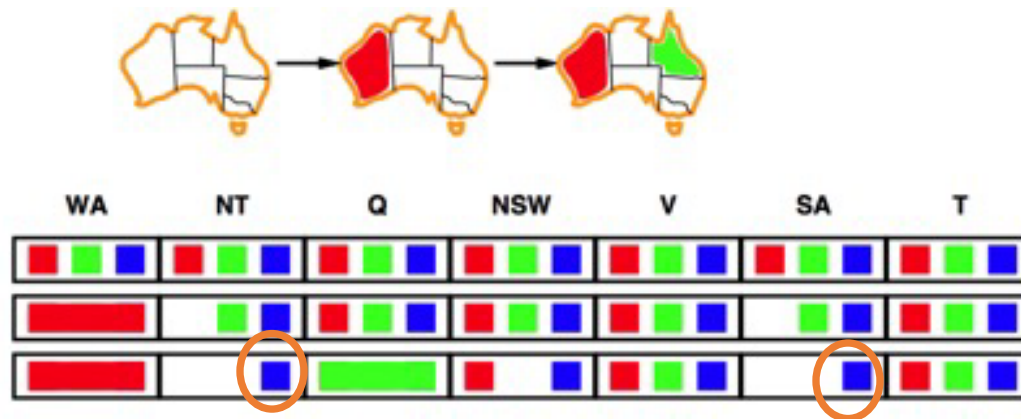
- Which variable should be assigned next?
 - Choose the variable with the fewest legal values in its domain (Pick the hardest)
- In what order should its values be tried?
 - choose the least constraining value: the one that rules out the fewest values in the remaining variables (Pick the ones that are likely to work.)
- Can we detect inevitable failure early?
 - Keep track of remaining legal values for the unassigned variables. Terminate when any variable has no legal values.



WA	NT	Q	NSW	V	SA	T
Red	Green	Blue	Red	Green	Blue	Red
Red	Green	Blue	Red	Green	Blue	Red
Red	Green	Blue	Red	Green	Blue	Red
Red	Green	Blue	Red	Green	Blue	Red

Constraint Propagation

- Forward checking
- It propagates information from **assigned** to **unassigned** variables.
- It does not check interaction between unassigned variables.
 - Example: Both SA and NT become blue, but they can't be in blue!.



- Forward checking improves backtracking search but does not look very far in the future, hence does not detect all failures.

BTS Algorithm

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
  return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment given CONSTRAINTS[csp] then
      add {var = value} to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  return failure
```

Local Consistencies

Local consistencies below are used to reduce the search space and make problems tractable.

- **Node-consistency (unary constraints)**

- A variable X_i is node consistent if all the values of $\text{Domain}(X_i)$ satisfy all unary constraints. (eg: $SA = \text{Green}$)
 - $X \rightarrow Y$ is node consistent if for every value x of X , there is some allowed y .

- **Arc-consistency (binary constraints):**

- $X \rightarrow Y$ is arc-consistent if and only if every value x of X is consistent with some value y of Y . (eg: $SA \neq WA$)
- Arc consistency can be enforced on a CSP by removing all the unsupported values from the domains of variables.

- **Path-consistency (n-ary constraints):**

- Generalizes arc-consistency from binary to multiple constraints.
- *It is always possible to transform all n-ary constraints into binary constraints. Often, CSPs solvers are designed to work with binary constraints.*

Constraint Satisfaction Problem – Example #2

Sudoku rule:

Fill in the numbers 1-9 exactly once in every row, column, and every nine of 3x3 region

- **81 variables: A1-A9 (top row) to I1-I9**
- **Domains {1,2,3,4,5,6,7,8,9}**
- **27 “Alldiff” constraints**

Alldiff (A1,A2,A3,A4,A5,A6,A7,A8,A9)

Alldiff (A1,A2,A3,B1,B2,B3,C1,C2,C3)

Alldiff (A1,B1,C1,D1,E1,F1,G1,H1,I1)

	1	2	3	4	5	6	7	8	9
A		7			4	5			
B	5			8	7				3
C			4	1			9		
D				5		2		6	8
E			2	6				3	
F			1		3		2		
G			5		1	8		9	2
H	1	4				6			7
I	2	9							1

Constraint Satisfaction Problems – Example #2

Use known variables & constraints to eliminate the values available for the remaining variables through inference.

- Example:
 - What is the value of B2 after removing invalid values due to row, column and square constraints
 - B2 = ?

Repeat for all variables and update as new variables are determined.

	1	2	3	4	5	6	7	8	9
A		7			4	5			
B	5			8	7				3
C			4	1			9		
D				5		2		6	8
E			2	6				3	
F			1		3		2		
G			5		1	8		9	2
H	1	4				6			7
I	2	9							1

Some problems can be solved purely by inference

Sudoku Example Solution

8		9	5		1	7	3	6
2		7		6	3			
1	6							
				9		4		7
	9		3		7		2	
7		6		8				
							6	3
			9	3		5		2
5	3	2	6		4	8		9



8	4	9	5	2	1	7	3	6
2	5	7	8	6	3	9	1	4
1	6	3	7	4	9	2	5	8
3	2	5	1	9	6	4	8	7
4	9	8	3	5	7	6	2	1
7	1	6	4	8	2	3	9	5
9	8	4	2	7	5	1	6	3
6	7	1	9	3	8	5	4	2
5	3	2	6	1	4	8	7	9

CSP Summary

- State-space search algorithms - **Just search!**
- CSP Algorithms
 - **Search:** choose a new variable assignment from many possibilities
 - **Inference:** constraint propagation, use the constraints to spread the word: reduce the number of values for a variable which will reduce the legal values of other variables and so on.
- The method focuses on satisfying the constraints rather than on the problem itself, the method is therefore more general and can be applied to different problems.
- Constraint propagation can be intertwined with search (cocktail methods).
- Saves time as can eliminate invalid areas of the search space.
- CSP Algorithm can be configured to give one answer (the first it finds) or all possible answers.
- As a pre-processing step, constraint propagation can sometimes solve the problem entirely without search.
- CSP is domain-independent. Define the problem and then use a solver that implements CSPs mechanisms.

Knowledge-based Agents, Mathematical Logic and Constraint Logic Programming Systems

Knowledge-Based Agents

“Logical AI: The idea is that an agent can represent knowledge of its world, its goals and the current situation by sentences in logic and decide what to do by inferring that a certain action or course of action is appropriate to achieve its goals.”



John McCarthy in Concepts of logical AI, 2000.

Knowledge-based agents

Knowledge-
base



Inference

- Domain-specific content
- A set of representations: “sentences” of things known “facts / truths”
- Expressed in “knowledge representation language”

- Domain-independent algorithms
- A way to “reason”

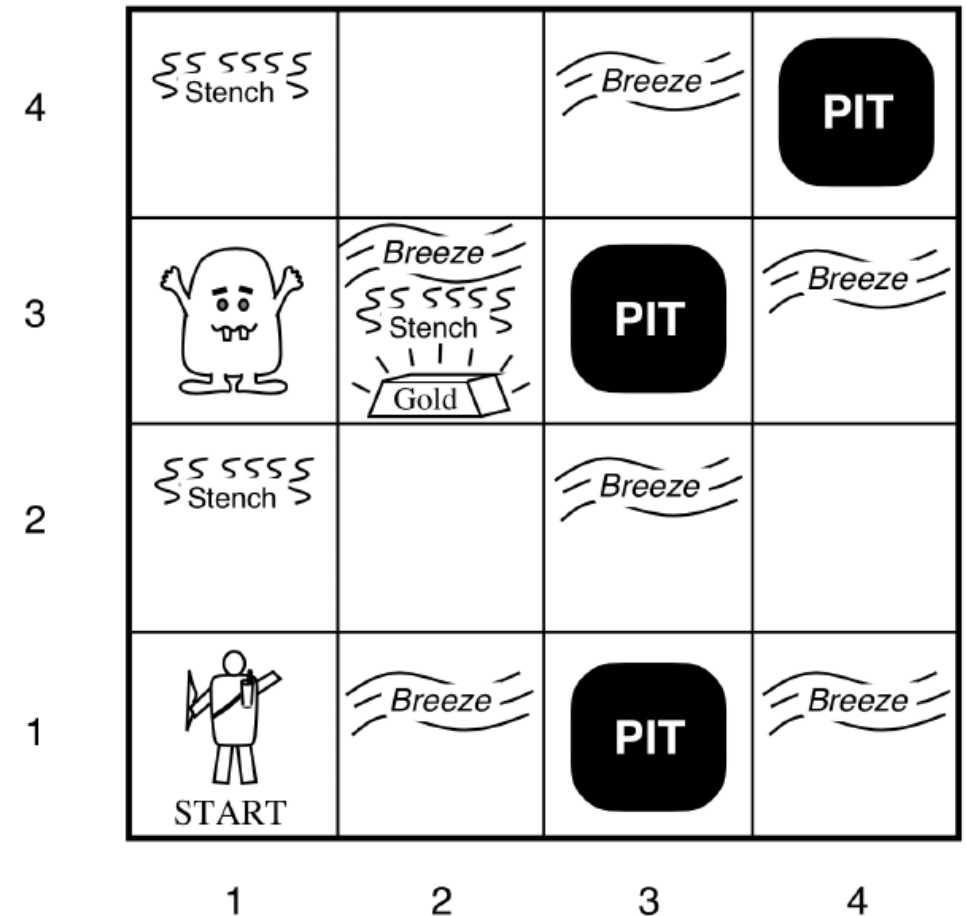
Knowledge Based Agents

- The agent must be able to:
 - Represent states, actions, etc.
 - Incorporate new precepts
 - Update internal representations of the world
 - Deduce hidden properties of the world
 - Deduce appropriate actions

Wumpus World

- 4 X 4 grid of rooms
- Squares adjacent to Wumpus are smelly and squares adjacent to pit are breezy
- Glitter if gold is in the same square
- Shooting kills Wumpus if you are facing it
- Wumpus emits a horrible scream when it is killed that can be heard anywhere
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square

<http://thiagodnf.github.io/wumpus-world-simulator/>



Wumpus World - PEAS

- **Performance measure**

- gold +1000, death (eaten or falling in a pit) -1000, -1 per action taken, -10 for using the arrow.
The game ends either when the agent dies or comes out of the cave.

- **Environment**

- 4 X 4 grid of rooms
 - Agent starts in square [1,1] facing to the right
 - Locations of the gold, and Wumpus are chosen randomly with a uniform distribution from all squares except [1,1]
 - Each square other than the start can be a pit with probability of 0.2

- **Actuators:**

- Left turn, Right turn, Forward, Grab, Release, Shoot

- **Sensors:**

- Stench, Breeze, Glitter, Bump, Scream

Environment

• Wumpus World environment

- Partially observable
 - Breeze, Stench, etc.
- Static
 - Wumpus and Pits are not moving
- Discrete
- Single-agent
- Deterministic
 - the result and outcome of the world are already known
- Sequential
 - the order is important

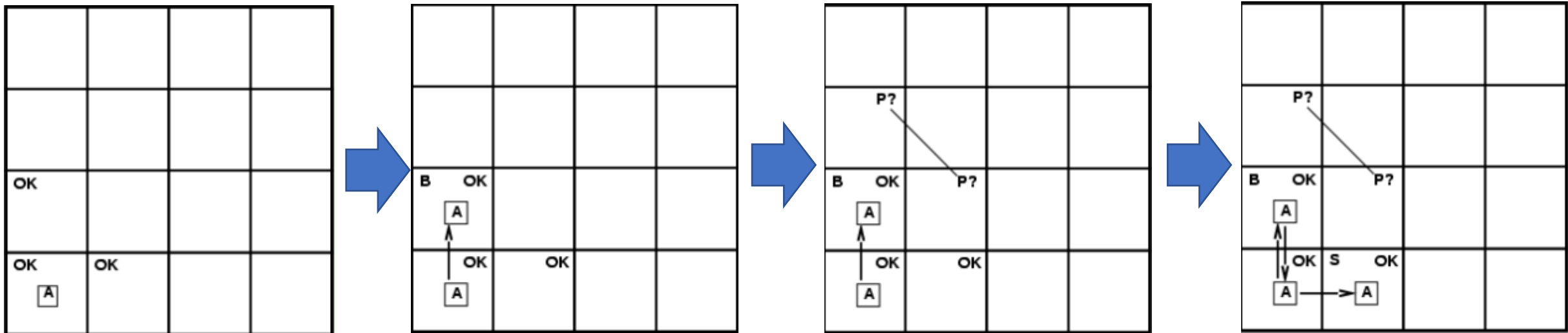
1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

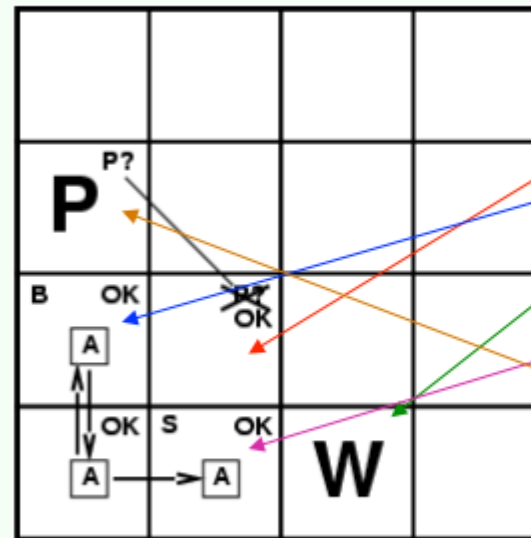
Simulator: <https://thiagodnf.github.io/wumpus-world-simulator/>

(a)

Exploring Wumpus World



Exploring Wumpus World



If the Wumpus were
 here, stench should be
 here. Therefore it is
 here.
 Since, there is no breeze
 here, the pit must be
 there

Knowledge Representation using Logic

Need a method to express knowledge in a way that the KBA agent can read, maintain and build with it.

Knowledge representation language:

- **Syntax** – specifies the structure of sentences and formula
- **Semantics** – defines the truth/facts of each sentence and the links between the facts
- **Truth** – A sentence is “true” if it is satisfied in the model / world
- **Logical entailment** – A sentence that follows logically from another is “entailed”. ie: one thing follows from the other

$$x+y = 4 \text{ entails } 4 = x+y$$

Examples:

Syntax

[Stench, Breeze, Glimmer, Bump, Scream]

@ Start

[None, None, None, None, None]

Entails – no Wumpus in 2,1.

Propositional Logic

- Propositional logic (PL) is the simplest logic.
- A **proposition** is a declarative statement that's either True or False.
 - $2+2=4$ is a true proposition
 - “If there is a stench in $[1,2]$ then there is a Wumpus in $[1,3]$ or $[2,1]$.”
- An **Atomic proposition** is written using single proposition symbol.
 - “13 is a prime number.”
- A **Compound proposition** is constructed from atomic propositions using parentheses and logical connectives.
 - {Born(from 1 January 1950 to 31 December 1959) **and** Singapore citizen on or before (31 December 1996)} **Or** {born on or before (31 December 1949) **and** Singapore citizen on or before (31 December 1996) **and** Do not receive (the Pioneer Generation Package)}

Compound Propositions

Examples of compound/complex propositions:

Let p , p_1 and p_2 be propositions

- **Negation:** $\neg p$ is also a proposition. $\neg W_{1,3}$
- **Conjunction:** $p_1 \wedge p_2$ E.g., $W_{1,3} \wedge P_{3,1}$
- **Disjunction:** $p_1 \vee p_2$ E.g., $W_{1,3} \vee P_{3,1}$
- **Implication:** $p_1 \Rightarrow p_2$ E.g., $W_{1,3} \wedge P_{3,1} \Rightarrow \neg W_{2,2}$
- **If and only if** $p_1 \Leftrightarrow p_2$ E.g., $W_{1,3} \Leftrightarrow \neg W_{2,2}$

Example

- When it is raining, the ground is wet. When the ground is wet, it is slippery
(These are **premises**, an idea or theory on which a statement or action is based).
- It is raining. Prove that it is slippery.

1. Raining \Rightarrow wet **Premise**

2. Wet \Rightarrow slippery **Premise**

3. Raining **Premise**

4. Wet **MP:1,3**

5. Slippery **MP:2,4**

MP stands for Modus Ponens. Eg: "P implies Q and if P is true, Q must be true."

Truth Table

- A truth table defines an operator g on n -tuples by specifying a Boolean value for each tuple.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

Note:

$P \rightarrow Q$ is equivalent to $\neg P \vee Q$

$P \Rightarrow Q$ is equivalent to $P \rightarrow Q \wedge Q \rightarrow P$

Exercise

P	Q	$P \wedge (P \Rightarrow Q)$	$\neg(\neg P \vee \neg Q)$	$(P \wedge (P \Rightarrow Q)) \Leftrightarrow (\neg(\neg P \vee \neg Q))$
<i>false</i>	<i>false</i>			
<i>false</i>	<i>true</i>			
<i>true</i>	<i>false</i>			
<i>true</i>	<i>true</i>			

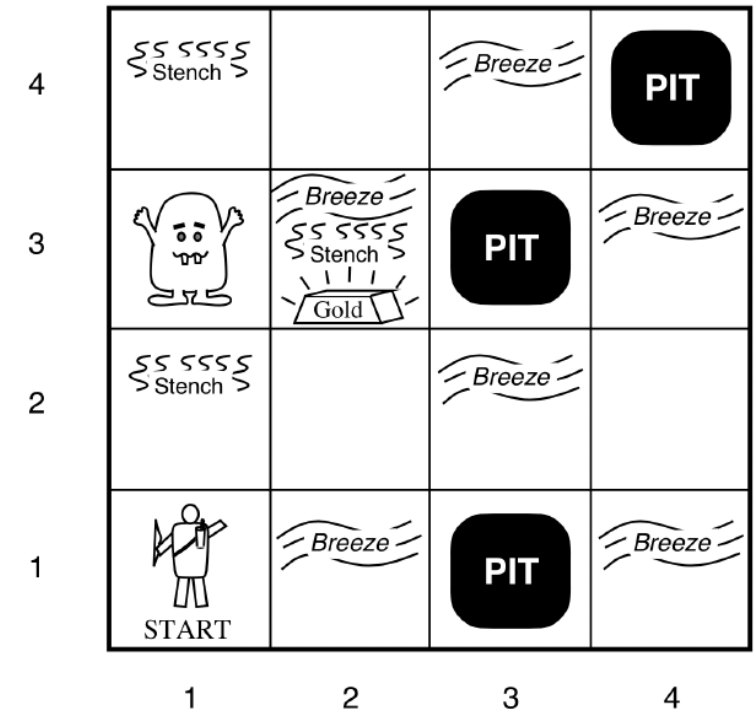
Exercise

P	Q	$P \wedge (P \Rightarrow Q)$	$\neg(\neg P \vee \neg Q)$	$(P \wedge (P \Rightarrow Q)) \Leftrightarrow (\neg(\neg P \vee \neg Q))$
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

Wumpus World KB

The following are true for all models:

- $P_{i,j}$ is true if there is a pit in $[i, j]$
- $B_{i,j}$ is true if agent perceives a breeze is a pit in $[i, j]$
- ...
- There is no pit at $[1,1]$: $\neg P_{1,1}$
- A square is breezy **iff** there is a pit in a neighboring square:
- $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$
- $B_{2,1} \iff (P_{1,2} \vee P_{2,2} \vee P_{3,1})$
- ...
- *The following are true for specific world is in:*
- $\neg B_{1,1}$
- $B_{2,1}$



Limitations of Propositional Logic

- PL is not expressive enough to describe all the world around us. It can't express information about different objects and the relation between objects.
- PL is not compact. It can't express a fact for a set of objects without enumerating all of them which is sometimes impossible.
- Example: We have a vacuum cleaner to clean a 10x10 squares in the classroom. Use PL to express information about the squares.

- **How can we express that all squares in the room are clean?**

$$square_1_is_clean \wedge square_2_is_clean \wedge \cdots \wedge square_{100}_is_clean$$

- **How can we express that some squares in the room are clean?**

$$square_1_is_clean \vee square_2_is_clean \vee \cdots \vee square_{100}_is_clea$$

First Order/Predicate Logic

More expressive and powerful representation language

All squares are clean:

$$\forall x \text{ Square}(x) \Rightarrow \text{Clean}(x)$$

There exists some dirty squares:

$$\exists x \text{ Square}(x) \wedge \neg \text{Clean}(x)$$

- *For all x, y and z , if $x > y$ and $y > z$ then $x > z$.*

$$\forall x \forall y \forall z \text{ greater}(x, z) \Rightarrow \text{greater}(x, y) \wedge \text{greater}(y, z)$$

Logic

Language	World	Beliefs
Propositional logic	Facts	True / false / unknown
First-order logic	Facts, objects, relations	True / false / unknown
Temporal logic	Facts, objects, relations, times	True / false / unknown
Probability theory	Facts	Degree of belief 0 to 1
Fuzzy logic	Degree of truth	Degree of belief 0 to 1

Constraint Logic Programming

- Constraint Logic Programming (CLP) is the merger of constraint satisfaction and logic programming.
- First practical application was the development of efficient programming languages based on Prolog (**P**rogrammation en **L**ogique or **P**rogramming in **L**ogic).
- A quick summary:
 - Invented early seventies by Alain Colmerauer in France and Robert Kowalski in Britain.
 - Prolog is a declarative programming language
 - You specifies a goal to be achieved and the Prolog system works out how to achieve it.
- We will not go into learning how to write a Prolog program. However, we will use one provided to solve a map coloring problems.

A Prolog Example

Facts

- Ann is a female.
- John is father of Ann.
- Joe is a male.

Rules

- A person is considered a senior citizen in Singapore if he/she ages above 60.

female(ann).
father(john, ann).
male(joe).

Must be in small letter to represent a constant or a clause

A Prolog clause must ends with a “.”

Capital letter is used for variable. Once assigned, it cannot be reassigned again.

senior(X) :-
 livein(X, Singapore),
 age(X, Y),
 Y > 60.

“,” represent “^”

head :- body.

A Prolog Example

A and B are sisters if

- A and B are both female **and**
- they have the same father **and**
- they have the same mother **and**
- A is not the same as B.

A (GNU) Prolog program for the above statement is:

```
sister(A,B) :- female(A), female(B), father(A,F), father(B,F),  
mother(A,M),mother(B,M), A \== B.
```

Some Traditional (old) AI Applications of CLP

All constraints – or rules - of the applications must be specified to the system in advance.

- Natural language understanding
- Expert systems
- Specification language
- Machine learning
- Planning (eg: robot)
- Automated reasoning
- Problem solving

Prolog - Constraint Logic Programming

- GNU Prolog is a free Prolog compiler with **constraint solving over finite domains**
 - A free Prolog compiler with constraint solving over finite domains developed by [Daniel Diaz](#).
- Download and install GNU Prolog (<http://www.gprolog.org/>)
 - <http://www.gprolog.org/setup-gprolog-1.5.0-msvc-x64.exe>
 - Download AUMapColouring.pl program from the Blackboard.
- Double click to launch GNU Prolog.
- File->consult and select the downloaded AUMapColouring.pl

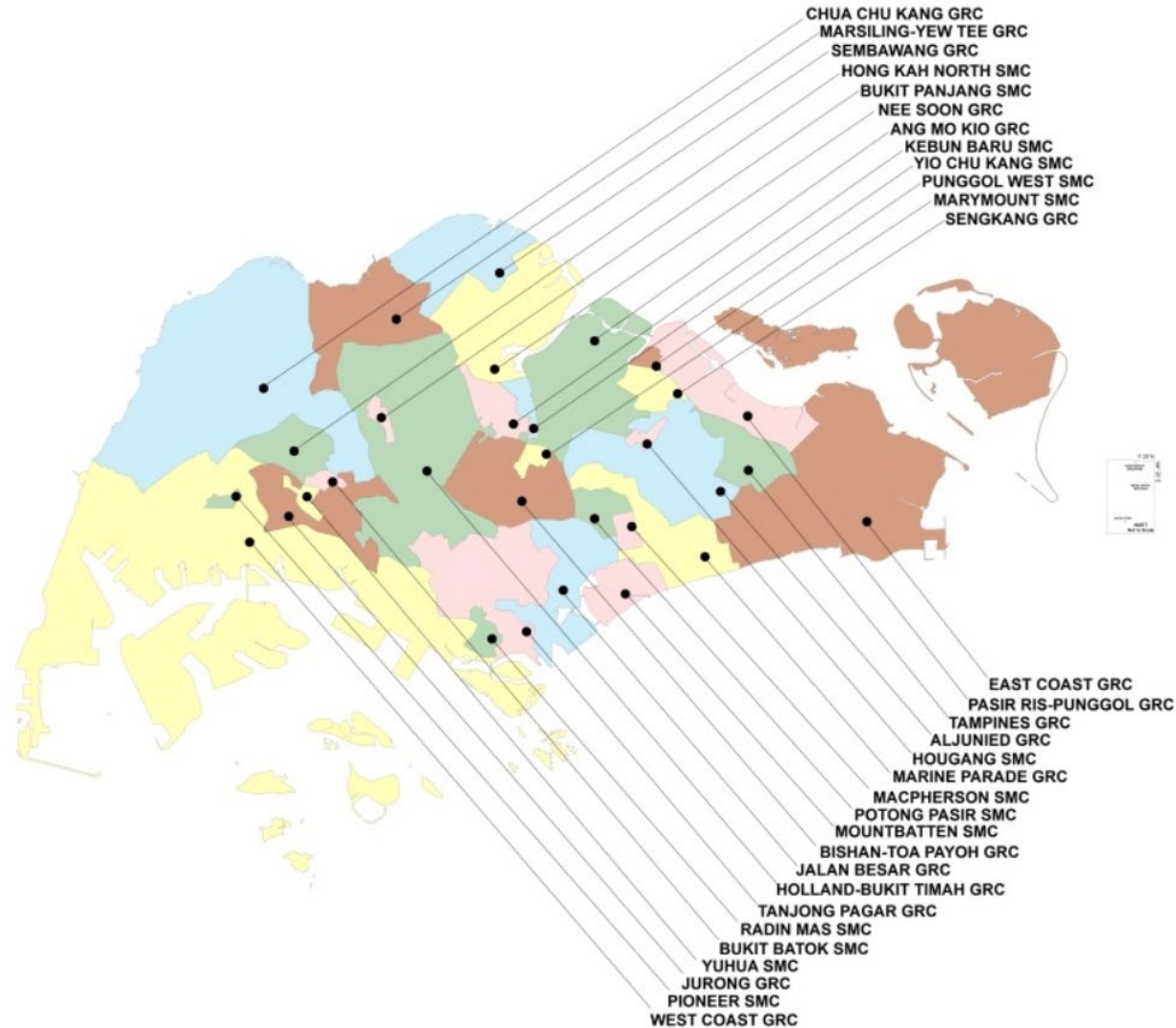
Yet another Map coloring Example

- Let's color bordering region of Romania.



Yet another Map coloring Example

https://www.eld.gov.sg/elections_map_electoral.html



Hands-on Workshop

- Use AUMapcoloring.pl to create USAMapcoloring.pl to color USA map with minimum set of colors.
- Exclude “Alaska” and “Hawaii” as they don’t share borders with other states.
- Include all states (including those with water border).

Summary

- Constraint satisfaction problems (CSPs) represent a state with a set of variable/value pairs and represent the conditions for a solution by a set of constraints on the variables.
- Constraint Satisfaction Problems are described as a set of variables with values, domains (all the possible values the variables can take) and a set of constraints
- Constraint satisfaction solvers provide efficient search and are well used in the various industry domains for scheduling, task prioritization, ordering, and etc.
- First order predicate logic provide definitions of rules and relationships without the ambiguity. Prolog is a declarative programming language based on first order predicate logic.
- Constraint logic solvers, including Prolog, are rule based and blend CSP and Logic well together. Rules and facts are the primary drivers.
- The above approached can be enhanced with current machine learning approaches where data is the key to drive the insight.