Or-Tree-based Search Example: Constraint Satisfaction

CPSC 433: Artificial Intelligence
Fall 2024

Jonathan Hudson, Ph.D.
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

August 8, 2024

Copyright © 2024

UNIVERSITY OF CALGARY

Constraint Satisfaction

UNIVERSITY OF CALGARY

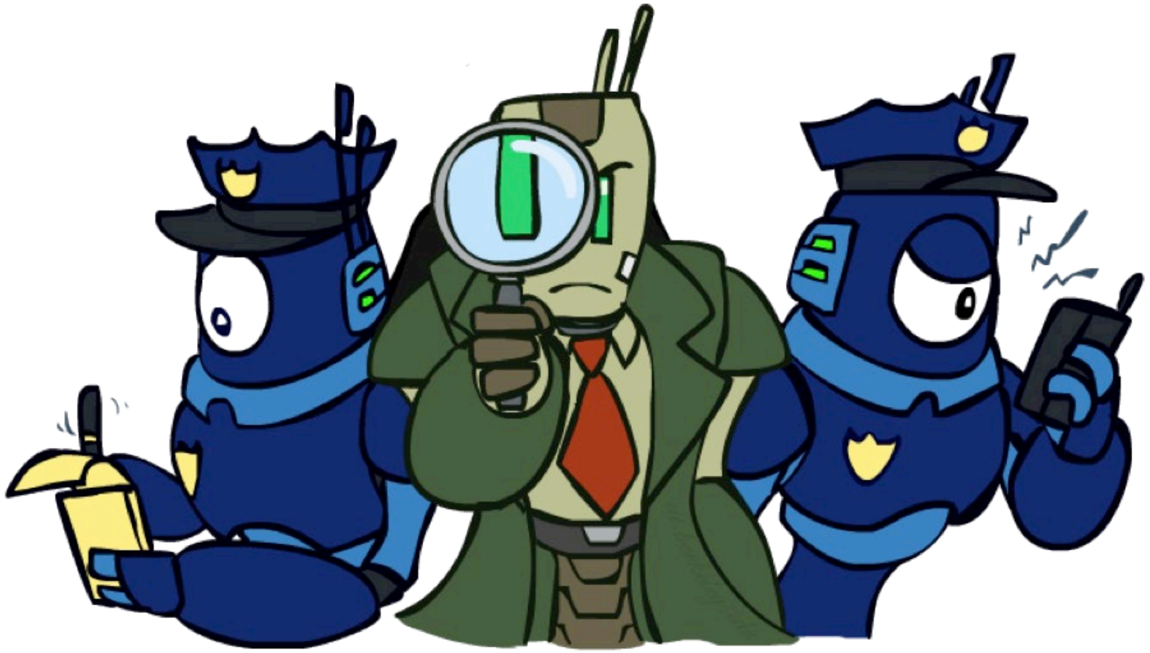
What is Search For?

- Assumptions about the world: a single agent, deterministic actions, fully observed state, discrete state space

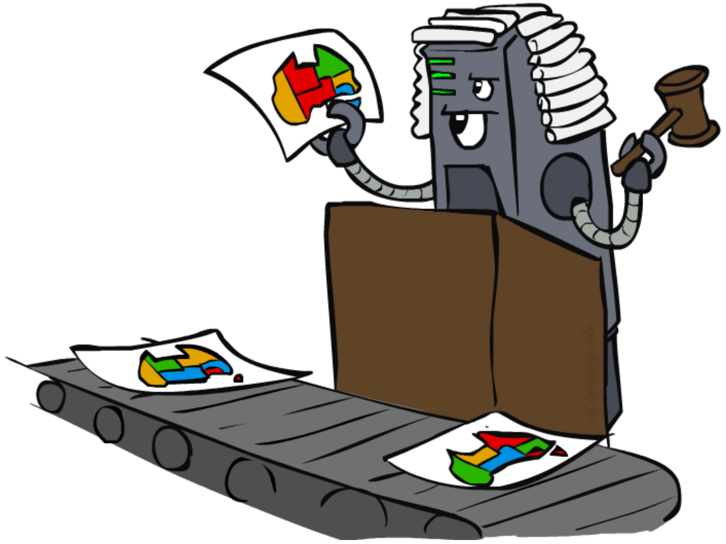
- Planning: sequences of actions
  - The path to the goal is the important thing
  - Paths have various costs, depths
  - Heuristics give problem-specific guidance

- Identification: assignments to variables
  - The goal itself is important, not the path
  - All paths at the same depth (for some formulations)
  - CSPs are specialized for identification problems

Constraint Satisfaction Problems

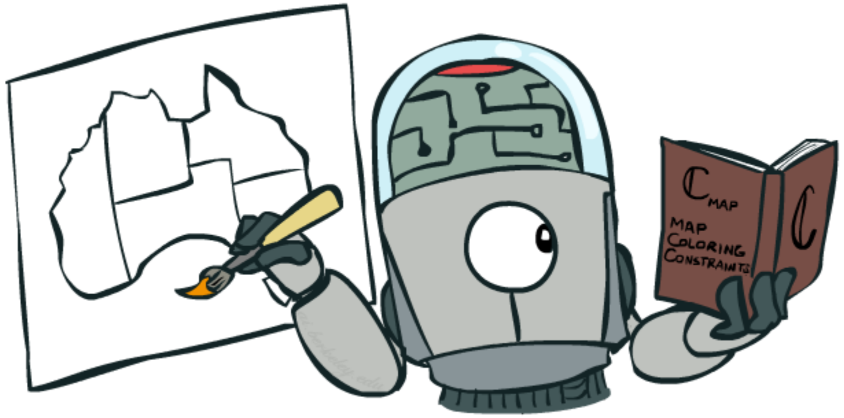


Constraint Satisfaction Problems



- Standard search problems:
  - State is a “black box”: arbitrary data structure
  - Goal test can be any function over states
  - Successor function can also be anything

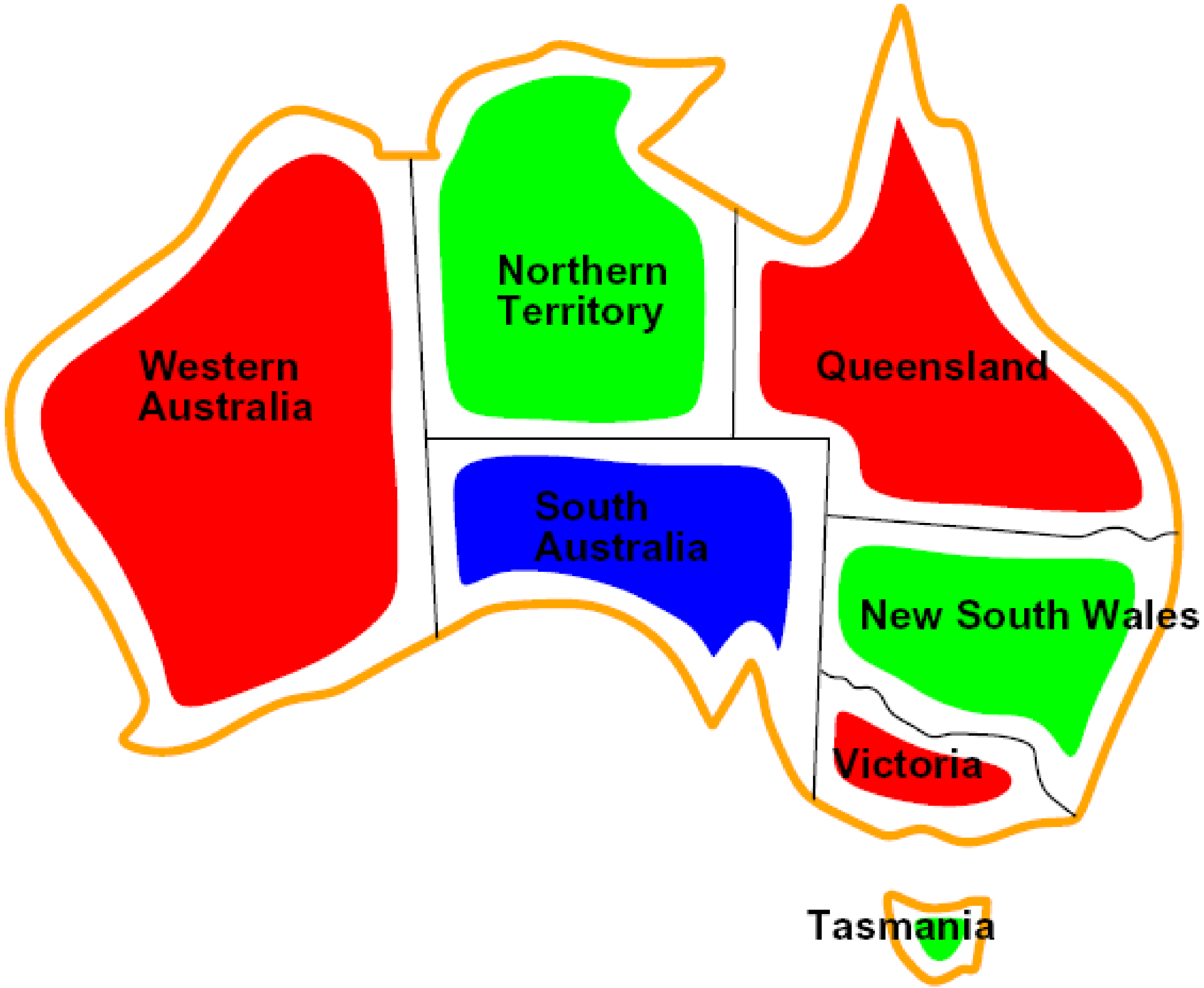
- Constraint satisfaction problems (CSPs):
  - A special subset of search problems
  - State is defined by variables X_i with values from a domain D (sometimes D depends on i)
  - Goal test is a set of constraints specifying allowable combinations of values for subsets of variables



- Allows useful general-purpose algorithms with more power than standard search algorithms

CSP Examples

CSP Examples



UNIVERSITY OF CALGARY

Example: Map Coloring

- Variables: WA, NT, Q, NSW, V, SA, T

- Domains: $D = \{\text{red}, \text{green}, \text{blue}\}$

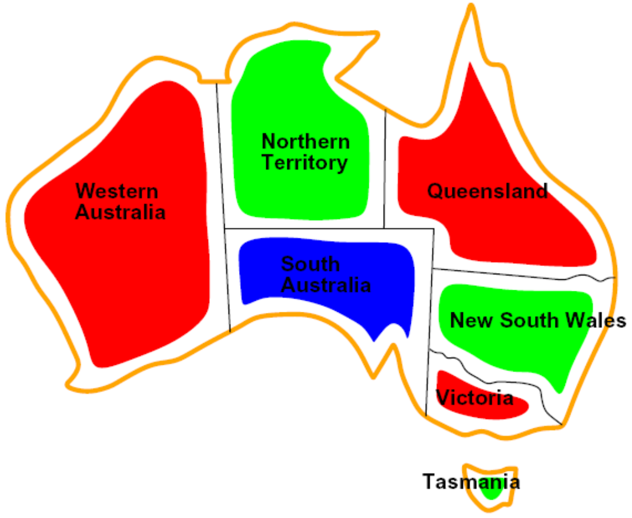
- Constraints: adjacent regions must have different colors

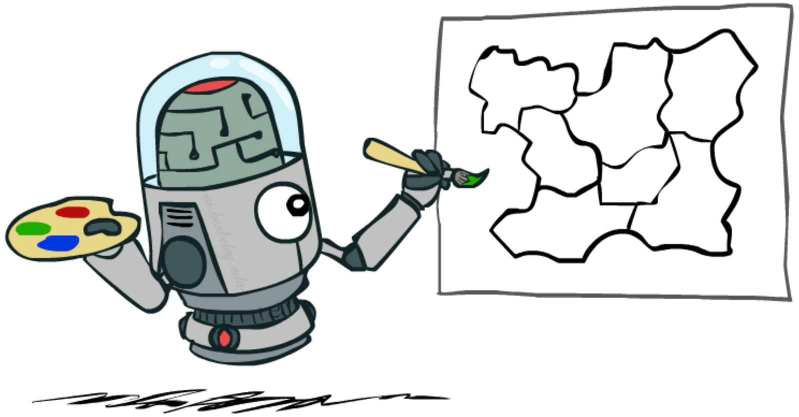
  Implicit: $WA \neq NT$

  Explicit: $(WA, NT) \in \{(\text{red}, \text{green}), (\text{red}, \text{blue}), \ldots\}$

- Solutions are assignments satisfying all constraints, e.g.:

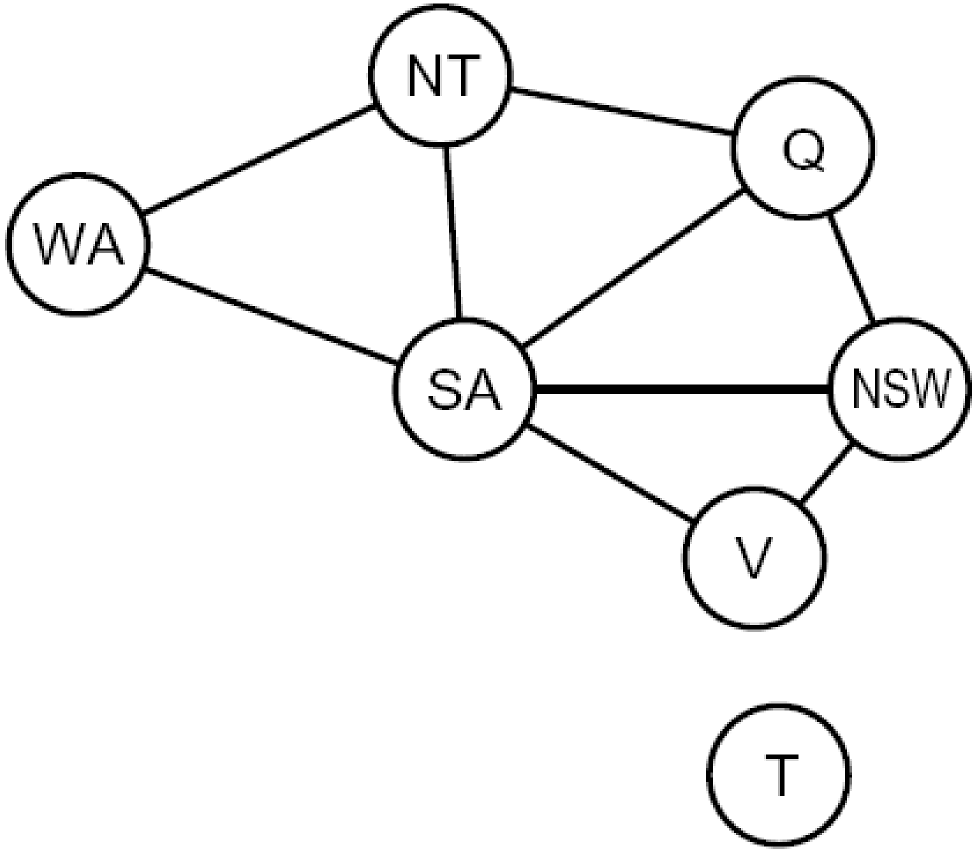
  {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}





UNIVERSITY OF CALGARY

Constraint Graphs



Example: Cryptarithmetic

- Variables:

  $$F\ T\ U\ W\ R\ O\ X_1\ X_2\ X_3$$

- Domains:

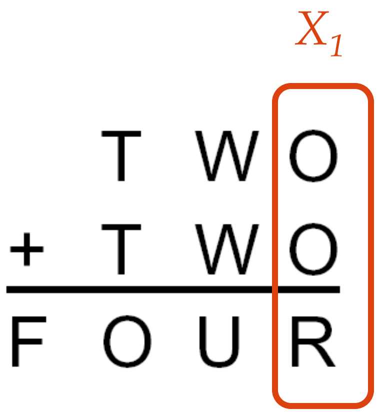
  $$\{0,1,2,3,4,5,6,7,8,9\}$$

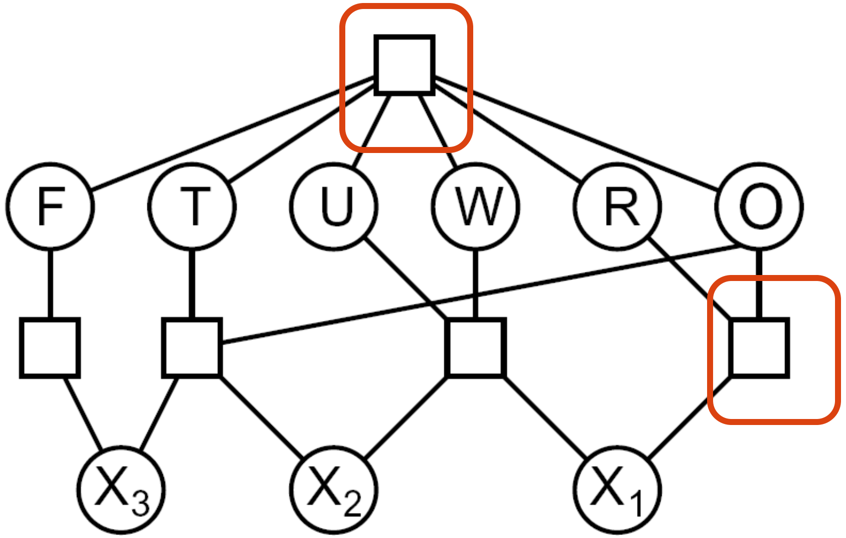
- Constraints:

  $$\text{alldiff}(F,T,U,W,R,O)$$

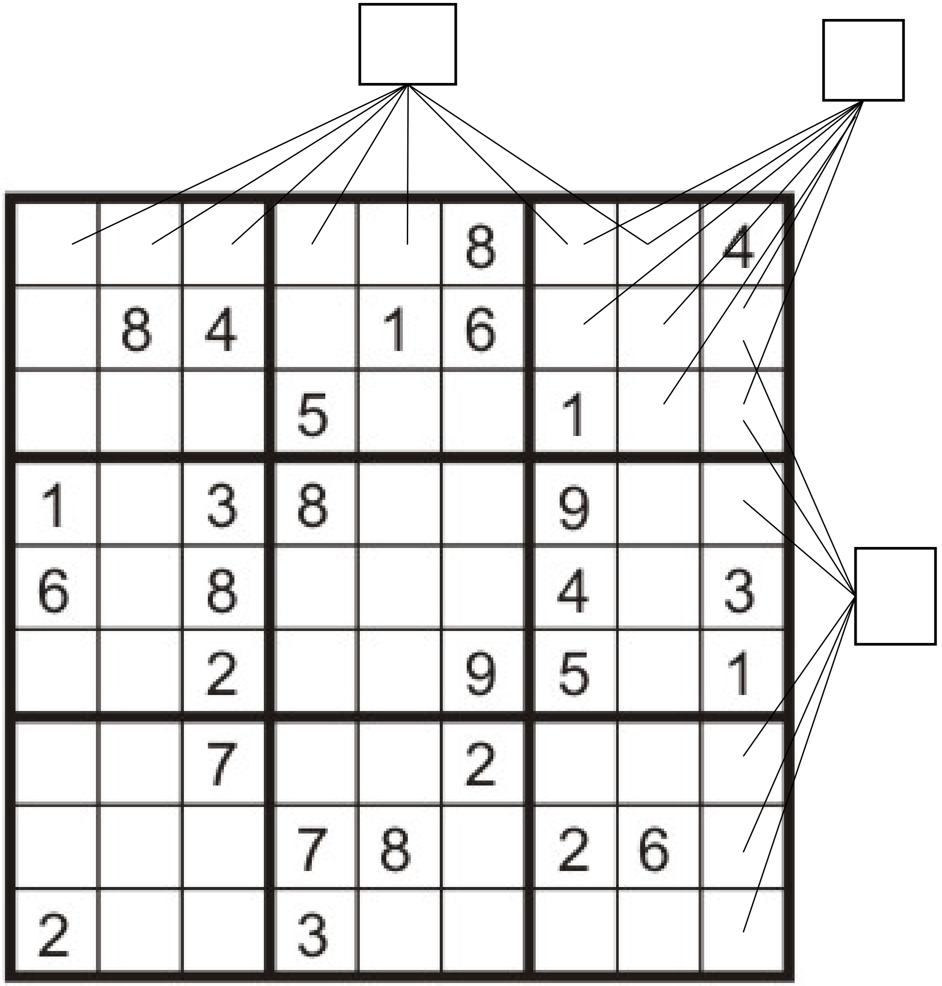
  $$O+O=R+10\cdot X_1$$

  $$\ldots$$





Example: Sudoku



- Variables:
  - Each (open) square
- Domains:
  - {1,2,…,9}
- Constraints:

9-way alldiff for each column

9-way alldiff for each row

9-way alldiff for each region

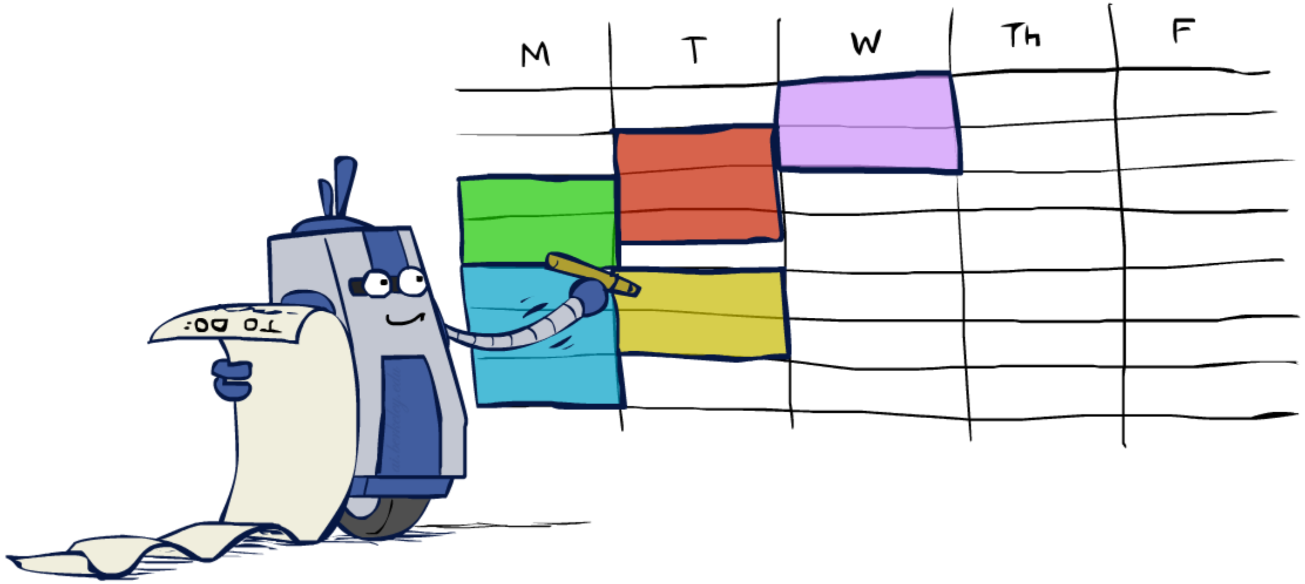
(or can have a bunch of pairwise inequality constraints)

CSP Varieties

UNIVERSITY OF CALGARY

Real-World CSPs

- Assignment problems: e.g., who teaches what class
- Timetabling problems: e.g., which class is offered when and where?
- Hardware configuration
- Transportation scheduling
- Factory scheduling
- Circuit layout
- Fault diagnosis
- … lots more!



- Many real-world problems involve real-valued variables…

UNIVERSITY OF CALGARY

Or-tree-based search applied to Constraint Satisfaction

UNIVERSITY OF CALGARY

Solving CSPs

UNIVERSITY OF CALGARY

Concrete Example: Constraint Satisfaction (I)

- A constraint satisfaction problem (CSP) consists of
  - a set $X = \{X_1, \dots, X_n\}$ of variables over some finite, discrete-valued domains $D = \{D_1, \dots, D_n\}$ and
  - a set of constraints $C = \{C_1, \dots, C_m\}$. Each constraint C_i is a relation over the domains of a subset of the variables, i.e.

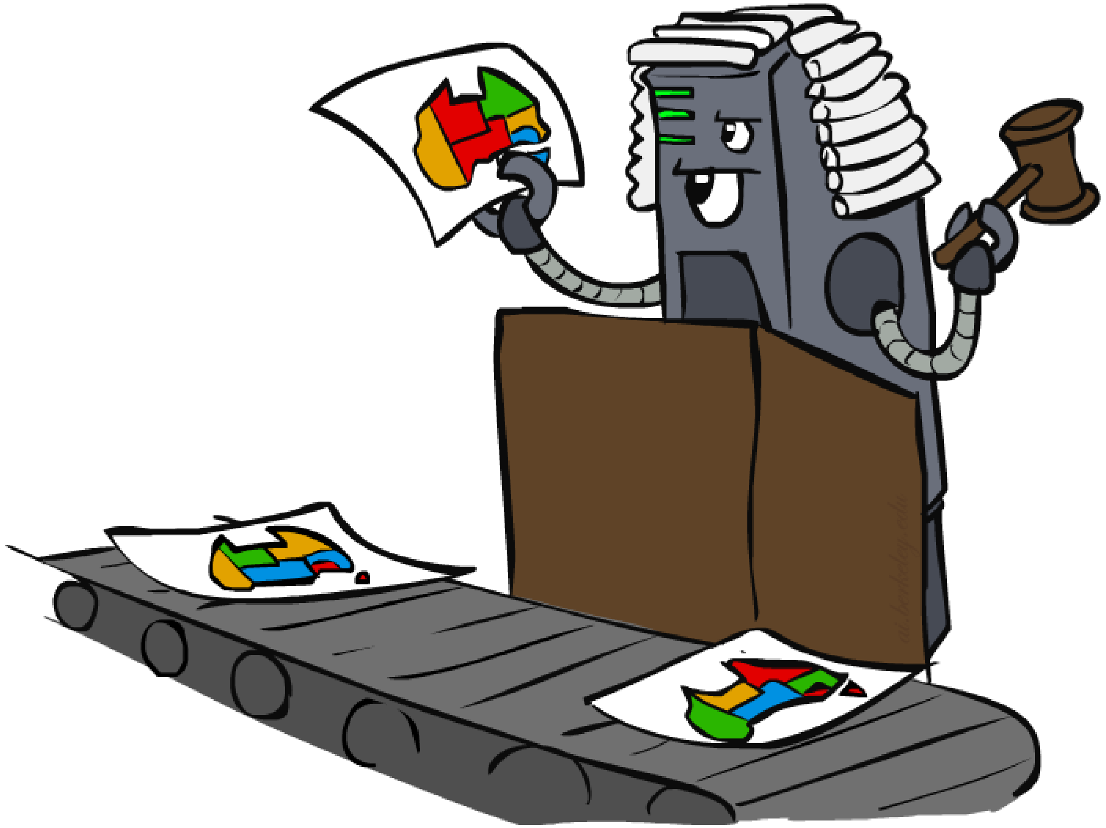
$$C_i = R_i(X_{i,1}, \dots, X_{i,k})$$

where the relation R_i describes every value-tuple in $D_{i,1} \times \cdots \times D_{i,k}$ that fulfills the constraint.

The problem is to

find a value for each X_j (out of its D_j)

that fulfills all C_i.



Constraint Satisfaction: Examples

Constraint Satisfaction (II): Examples

1. $X = \{X_1, X_2\}$
$D_1 = \{1,2,3\}$
$D_2 = \{1,2,3,4\}$

fulfill
$C = \{C_1, C_2, C_3\}$

$C_1: X_1 + X_2 \leq 4$ $C_2: X_1 + X_2 \geq 3$ $C_3: X_1 \geq 2$

2. $X = \{X_1, X_2, X_3\}$
$D_1 = D_2 = D_3 = \{true, false\}$

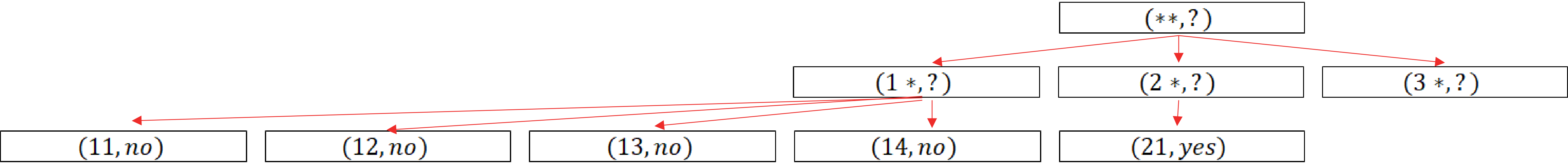
fulfill
$C = \{C_1, C_2, C_3\}$

$C_1: X_1 \vee \neg X_2 \vee X_3$ $C_2: \neg X_1 \vee X_3$ $C_3: \neg X_2 \vee \neg X_3$

Constraint Satisfaction (II): Examples

- $X = \{X_1, X_2\}$
$D_1 = \{1,2,3\}$
$D_2 = \{1,2,3,4\}$
$C = \{C_1, C_2, C_3\}$

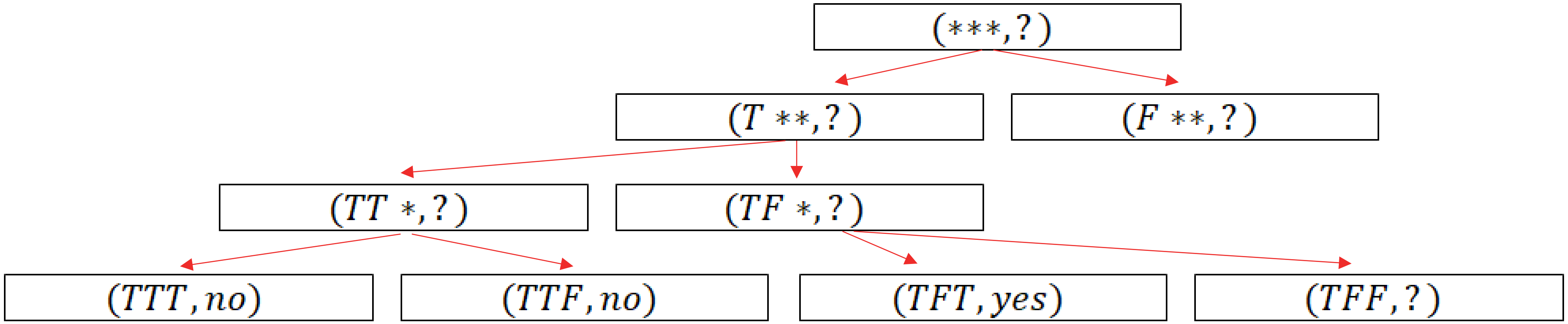
$C_1: X_1 + X_2 \leq 4 \quad C_2: X_1 + X_2 \geq 3 \quad C_3: X_1 \geq 2$



Constraint Satisfaction (II): Examples

- $X = \{X_1, X_2, X_3\}$
$D_1 = D_2 = D_3 = \{true, false\}$
$C = \{C_1, C_2, C_3\}$

$C_1: X_1 \vee \neg X_2 \vee X_3$ $C_2: \neg X_1 \vee X_3$ $C_3: \neg X_2 \vee \neg X_3$



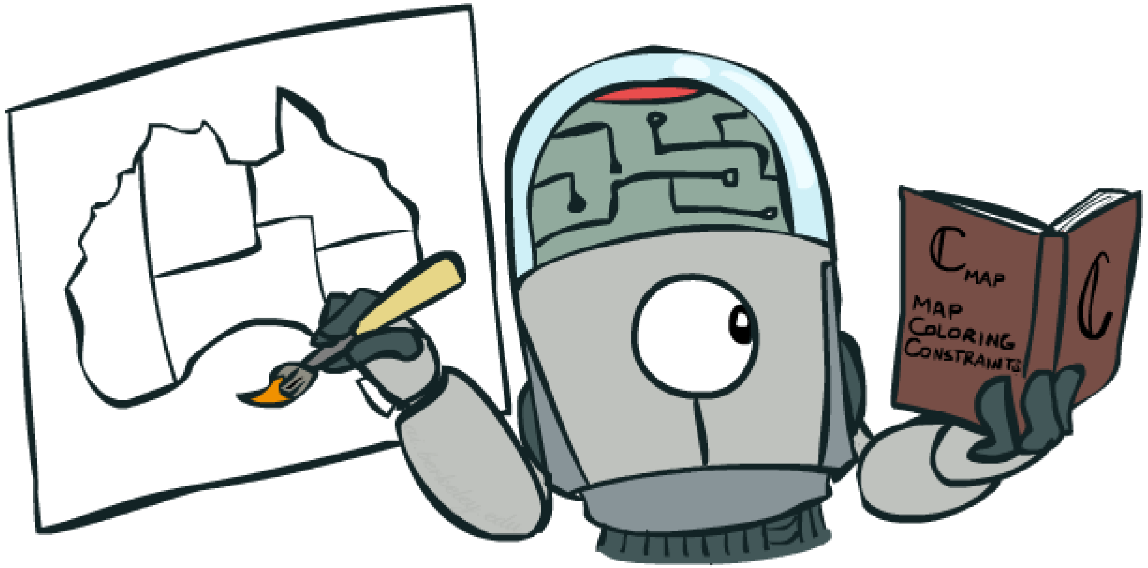
Constraint Satisfaction: Or-Tree-Based

UNIVERSITY OF CALGARY

Constraint Satisfaction (III)

Tasks:

- Describe CSPs as or-tree-based search model
- Describe formally a search control for your model based on the idea of identifying the variable occuring in the most constraints and selecting it and its domain for branching
(combined with a depth-criteria and a tiebreaker, if necessary)
- Solve the problem instances from the last slide



Model?

UNIVERSITY OF CALGARY

Search control for CSP example

Let $(pr_1,?),\ldots,(pr_o,?)$ be the open leafs in the current state and let

$const(X_j) = |\{C_i \mid C_i \in C, C_i = R_i(X_{i,1},\ldots,X_{i,k}), X_j \in \{X_{i,1},\ldots,X_{i,k}\}\}|$

For a problem $pr = (x_1,\ldots,x_n)$ let

$Csolved(pr) = |\{C_i | C_i \in C, x_1,\ldots,x_n \text{ fulfills } C_i\}|$

Then our search control K selects the leaf to work on and the transition to this leaf

UNIVERSITY OF CALGARY

Process?

UNIVERSITY OF CALGARY

Search control for CSP example

If one of the pr_j is solved, perform the transition that changes its sol-entry. If there are several, select one of them randomly.

Else if one of the pr_j is unsolvable, perform the transition that changes its sol-entry. If there are several, again select one of them randomly.

Else

- select the leaf (pr_j,?) such that
  a) $Csolved(pr_j) = max_{prl}(\{Csolved(pr_l)\})$
  b) if there are several, select the deepest leaf in the tree with this property.
  c) if there are still several, select the one the most left in the tree (tiebreaker without knowledge)

UNIVERSITY OF CALGARY

Search control for CSP example

- for the transition select the one with
  Altern(pr_j,pr_{j1},...,pr_{jk}) such that the variable X_i we use to create the element in *Altern* is the one with maximal Const-value.
  If there are several of those, use the one with minimal index i (tiebreaker without knowledge)

UNIVERSITY OF CALGARY

Remarks

Remarks

- And-tree-based and or-tree-based search have a lot in common. The difference from the search problem point of view can be best described as
  or-tree: one solution and done (one-yes)
  and-tree: all solutions and done (all-yes)

- Consequently, the criteria used by search controls differ, due to the different goals.

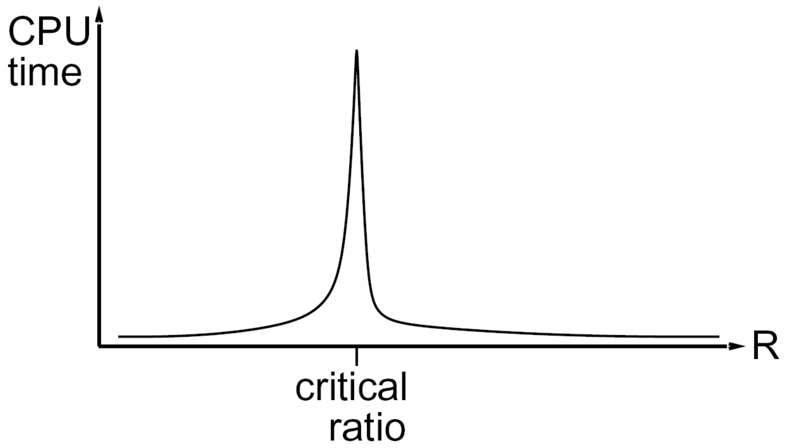
- A lot of problems have transformations into a CSP. Therefore there are a lot of papers on solving CSPs and good controls for it.

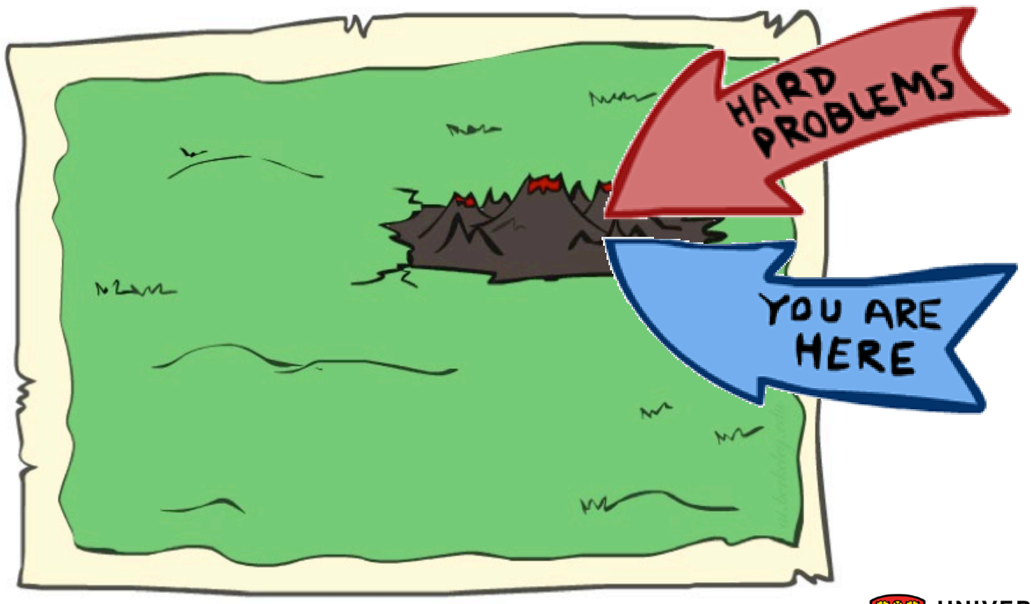
UNIVERSITY OF CALGARY

Performance of Min-Conflicts

- Given random initial state, can solve n-queens (A standard CSP problem) in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)!
- The same appears to be true for any randomly-generated CSP *except* in a narrow range of the ratio

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$





Onward to …
other search models

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY