Artificial Intelligence
Exercise 5

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

7. Januar 2020

1 Programmieraufgabe: Constrained Satisfaction Problems

Pull the current exercise from our server to your local repository.

Task 1: Implement backtracking for the constrained satisfaction problem definition you find in `csp.py`. Make three different versions of it 1) without any heuristic 2) with minimal remaining value as heuristic but without tie-breaker (take the first best solution) 3) with minimal remaining value and the degree heuristic as tie-breaker.

Optional: Implement AC-3 or any approximate form of constraint propagation and activate it if the according parameter is set.

Task 2: Implement a method to convert a Sudoku into a `csp.ConstrainedSatisfactionProblem`, and then use this to solve the sudoku given as a numpy array. Every empty field is set to `0`. The CSP you create should cover all rules of a Sudoku, which are (from `http://en.wikipedia.org/wiki/Sudoku`):

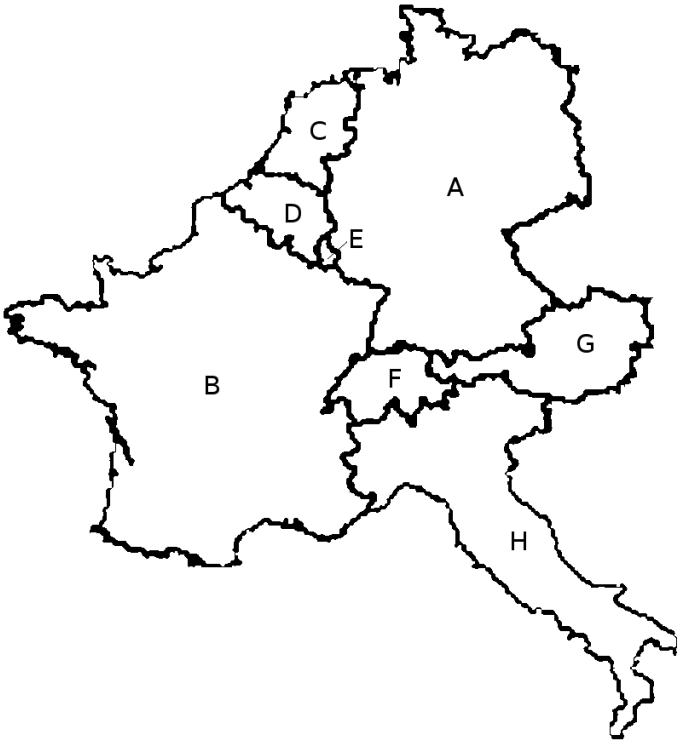
> *Fill a* 9×9 *grid with digits so that each column, each row, and each of the nine* 3×3 *sub-grids that compose the grid (also called 'blocks') contains all of the digits from 1 to 9.*

In the lecture we mentioned the *all_different* constraint for columns, rows, and blocks. As the `csp.ConstrainedSatisfactionProblem` only allows you to represent pair-wise *unequal* constraints (to facilitate constraint propagation) you need to convert this.

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

2 Votieraufgabe: CSP

Betrachten Sie folgenden Kartenausschnitt:



Der Kartenausschnitt soll mit insgesamt 4 Farben so eingefärbt werden, dass je zwei Nachbarländer verschiedene Farben besitzen.

Mit welchem Land würde man am ehesten beginnen?

Färben Sie das erste Land ein und wenden Sie durchgehend Constraint Propagation an.

3 Präsenzaufgabe: Generalized Arc Consistency

We have n variables x_i, each with the (current) domain D_i. Constraint propagation by establishing local constraint consistency ("arc consistency") in general means the following:

For a variable x_i and an adjacent constraint C_k, we delete all values v from D_i for which there exists no tuple $\tau \in D_{I_k}$ with $\tau_i = v$ that satisfies the constraint.

Consider a simple example

$$x_1, x_2 \in \{1, 2\} \ , \quad x_3, x_4 \in \{2, .., 6\} \ , \quad c = \text{AllDiff}(x_1, .., x_4)$$

(a) How does constraint propagation from c to x_3 update the domain D_3?

(b) On `http://norvig.com/sudoku.html` Norvig describes his Sudoku solver, using the following rules for constraint propagation:

(1) If a square has only one possible value, then eliminate that value from the square's peers.

(2) If a unit (block, row or column) has only one possible place for a value, then put the value there.

Is this a general implementation of constraint propagation for the *allDiff* constraint?

Note: The generalized arc consistency is equivalent so-called message passing (or belief propagation) in probabilistic networks, except that the messages are domain sets instead of belief vectors.

See also `www.lirmm.fr/~bessiere/stock/TR06020.pdf`