In some of the exercises below, you are encouraged to try out software demonstrators developed by the authors of the book. The default option here is the Java-based tools found at

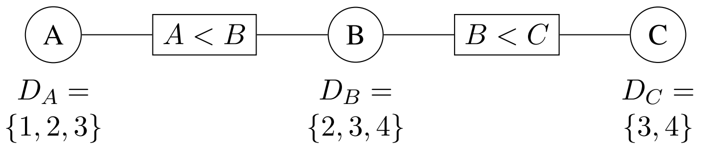
http://aispace.org

but if you are interested in a more programming oriented approach you are also welcome to explore the Python-based library AIPython:

http://artint.info/AIPython/

In case you decide for the latter, I would recommend downloading the pre-packaged AIpython bundle (link found in the optional software exercises for Lecture 2) and follow the setup instructions given there.

Exercice 1 :



- Which arcs in the above constraint network are arc consistent?
- How can the whole network be made arc consistent, i.e., which changes should be made to the domains of the variables making the network arc-consistent without eliminating potential solutions?
- Check your result by implementing the model in the CSP-implementation found at http://aispace.org/constraint/.
- Show how domain splitting can be used to solve the original constraint network.

Exercice 2 :

Consider the following mini-Sudoku:

		4	
2			
		1	
3			

The empty fields have to be filled with numbers 1,2,3, or 4, such that each row, each column, and each of the 2×2 sub-squares contain each of these numbers exactly once.

a. Formalize this problem as a constraint satisfaction problem: define an appropriate set of variables, and a set of constraints that define the solutions of the sudoku (hint: instead of using constraints as on the lecture slide, define a smaller set of constraints, each constraint expressing the full condition for one row, one column, or one sub-square. We have not discussed off-the-shelf formal languages for expressing such constraints, so you should try to express them as formally as possible).

b. Draw the constraint network for this problem. If this gets too large, draw only the part of the network that is sufficient to answer the next question.

c. Apply the generalized arc consistency algorithm to show that the top left square must contain a 1. Does the operation of the algorithm resemble how you would come to that conclusion yourself?

Exercice 3 :

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Constraints: $2|a - c| > 3$; $b^2 - 3d < 9$; $b + 3 < c$.

Run the Generalized Arc Consistency(γ) algorithm, as specified in the lecture. Precisely, for each iteration of the while-loop, give the content of To-do-arcs at the start of the iteration, give the pair (x, c) removed from To-do-arcs, give the domain D_x of x after the call to Revise(γ, x, c), and give the pairs (y, c') added into M.

Note: Initialize To-do-arcs as a lexicographically ordered list (i.e., (a, b) would be before (a, c), both before (b, a) etc., if any of those exist). Furthermore, use to-do-arcs as a FIFO queue, i.e., always remove the element at the front and add new elements at the back.

Exercice 4 :

Solve exercise 4.12 in PM about variable elimination.

Exercice 5 :

Use Variable Elimination to solve the following CSP given by extensional constraints on Boolean variables A, B, C:

A	B
t	f
t	t
f	t

A	C
t	f
f	t

B	C
t	f
f	t

Exercice 6 :

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d, e, f, g\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Constraints: $a \leq c$; $b = c^2 - 2$; $|d - c| \leq 4$; $d < e - 6$; $|3e - f^2| \leq 6$; $|3g^2 - e| < 3$

Run the AcyclicCG(γ) algorithm, as specified in the lecture. Precisely execute its 4 steps as follows:

(a) Draw the constraint graph of γ. Pick a as the root and draw the directed tree obtained by step 1. Give a resulting variable order as can be obtained by step 2.

(b) List the calls to Revise($\gamma, v_{parent(i)}, v_i$) in the order executed by step 3, and for each of them give the resulting domain of $v_{parent(i)}$.

(c) For each recursive call of BacktrackingWithInference during step 4, give the domain D'_{v_i} of the selected variable v_i after Forward Checking, and give the value $d \in D'_{v_i}$ assigned to v_i.

Note: Step 4 runs BacktrackingWithInference with variable order $v_1, \ldots, v_n$. This means that, at the ith recursion level, "select some variable v for which a is not defined" will select v_i.

Exercice 7 :

Solve exercise 4.3 (except d) in PM and using only the network displayed in Figure 4.15(b). For exercise (a), (b), (c) you may use the CSP-implementation found at http://aispace.org/constraint/; observe that the CSP-implementation has a special constraint for the word relations appearing in a crossword.

Exercice 8 :

Consider the cryptarithmetic puzzle

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

Each letter in a cryptarithmetic problem represents a digit; note that F cannot be 0.

- Construct a constraint network representation for the puzzle. *Hint:* For each of the four columns in the table representation above we have a constraint. E.g. for the first column we have the constraint

  $$O + O = R + 10 \cdot C_1,$$

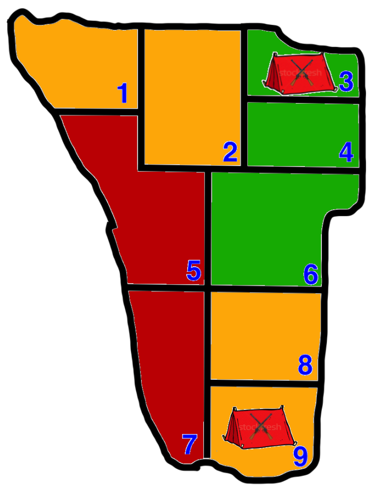
  where C_1 is an auxiliary variable representing what is carried over in the 10 column. Use similar auxiliary variables, say C_2 and C_3, for encoding what is carried over to the 100 and 1000 column, respectively.

- Perform a forward-backward search to find a solution to the puzzle. *Hint:* The ordering of the variables has a big impact on the solution size, so a good strategy would be to chose the variable with the smallest state space. E.g. consider starting with $C_3 = 1$.

Exercice 9 :

(Extra exercise for modelling CSPs)

You are a ranger and own a piece of land, depicted below. You want to make sure that all of your animals survive. This is only possible if no animals that kill each other live in the same or neighboring regions. You own rhinos (R), elephants (E), cheetahs (C), lions (L), jackals (J), springbok (S), zebras (Z), hippos (H) and giraffes (G). Your piece of land contains three different climatic regions: bush (green), desert (red), savannah (orange). Some animals need specific climatic environments. Unfortunately there are also two camps of poachers on the land, who are selling rhino horn and elephant tusk. Poachers treacherously shoot animals in their own and neighboring regions.



- Poachers kill rhinos and elephants
- Jackals eat springbok
- Cheetahs eat springbok and zebras
- Lions eat springbok, zebras and giraffes, and sometimes even kill elephants

- Jackals and cheetahs can live in the desert and in the savannah
- Elephants can only live in the bush
- Lions and zebras can live in the bush and in the savannah
- Springbok and hippos only live in the savannah regions

Formulate the problem as a constraint network. Use the set of variables $V = \{v_1, ..., v_9\}$, where variable v_i models which animal can live in region i.

(i) Specify the domain D_{v_i} of each variable $v_i \in V$. That is, for each region type give the set of animals that can live in that region, and state which variables have this domain.

(ii) Specify the binary constraints $C_{v_i v_j} \subseteq D_{v_i} \times D_{v_j}$ for all pairs of variables. That is, for each combination of regions give the set of animal-pairs that the region pair can inhabit, and state which variable pairs have this constraint.

(iii) Specify the unary constraints $C_{v_i} \subseteq D_{v_i}$ for all relevant variables. That is, for each region affected by the poachers, give the set of animals that can inhabit that region.

Hint: You can specifying constraints with those pairs that are not contained:
$C_{v_i,v_j} = D_{v_i} \times D_{v_j} \setminus \{(d_{v_{i,k}}, d_{v_{j,l}}), ...\}$

Exercice 10 :

(Extra exercise to practice arc consistency (a bit long) and AcyclicCG)

On your piece of land you additionally own a lodge for tourists and a cat sanctuary. You want to show your tourists all your nice animals and everything the region has to offer. You have 5 different time slots: $1, 2, 3, 4, 5$, and want to fit all the tasks (see below) in, such that all of the constraints (see below) are satisfied.

Tasks:

- (GD) go for game drive with tourists
- (DW) go to the desert with tourists
- (HF) make helicopter flight with tourists

- (HB) have braai with tourists
- (PB) prepare braai
- (PC) pet cheetahs
- (FL) feed lions
- (FC) feed cheetahs
- (FZ) feed zebra

Constraints:

- Cheetahs need to be fed immediately before you can pet them ($FC = PC - 1$)
- Lions and cheetahs have to be fed at the same time, otherwise they will get jealous ($FC = FL$)
- The desert walk has to be (at least) two time slots before the game drive, otherwise your tourists will get to exhausted ($DW \leq GD - 2$)
- In order to have braai with your tourists you first have to exhaust them on the game drive and on the desert walk, so they can enjoy their beer ($GD < HB$; $DW < HB$)
- You have to prepare braai before having the braai with your tourists ($PB < HB$)
- You can only find lions on the game drive if they have been fed two time slots earlier ($GD = FL + 2$)
- In order to give your tourists an impression of the area first, the heli flight is done just before the desert walk ($HF = DW - 1$)
- In order to save time you want to feed the zebra on the game drive ($FZ = GD$)

- Variables: $V = \{DW, FC, FL, FZ, GD, HB, HF, PC, PB\}$
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5\}$

(i) Draw the constraint graph of γ. Label each edge with the corresponding constraint.

(ii) Run Generalized Arc consistency. Initialize To-do-arcs with an alphabetical list. For each step: Give the selected pair, the updated domain, and the pairs that are added to To-do-arcs (give all pairs that would be added and mark those that are not already contained). You do not have to give the updated To-do-arcs list in each iteration.

(iii) Run AcyclicCG:

1. Assume that the constraint $DW < HB$ does not exist (remove this edge from the graph). Pick DW as the root and draw the directed tree.
2. Give the resulting variable order obtained by step 2 of the algorithm. If the ordering of variables is not unique, break ties by using the alphabetical order.
3. List the calls to $Revise(\gamma, v_{parent(i)}, v_i)$ in the order executed by step 3 of the algorithm, and for each of them give the resulting domain of $v_{parent(i)}$.
4. Run BacktrackingWithInference with Forward Checking. For each recursive call to Backtracking-WithInference during step 4 of the algorithm, give the domain D'_{v_i} of the selected variable v_i after Forward Checking, and give the value $d \in D'_{v_i}$ assigned to v_i.