Exercise 6

Foundations of Constraint Programming

Prof. Steffen Hölldobler, Sebastian Haufe

International Master Program in Computational Logic — winter term 2010/2011

Date of exercise: 21.01.2011

Exercise 6.1

In the following exercise, always consider A and B to be binary constraints. Moreover, for a binary constraint C, the π_1 and π_2 functions of C (cf. Slide VI/26) are denoted by π_1^C and π_2^C, respectively.

a) Consider two variables x and y with the respective domains $D_x = D_y = \{a, b\}$. Let $A, B \subseteq D_x \times D_y$ such that $A = \emptyset$ and $B = \{(a, a)\}$.

   Show that π_1^A does semi-commute but not commute with π_2^B.

b) Define A and B such that π_1^A and π_2^B do not semi-commute with each other.

c) Define A and B such that π_1^A does semi-commute but not commute with π_1^B.

d) Consider three variables x, y and z with the respective domains D_x, D_y and D_z and let $A \subseteq D_x \times D_y$ and $B \subseteq D_x \times D_z$ arbitrary.

   Prove that π_1^A and π_1^B commute.

Exercise 6.2

Find a CSP for which the ARC algorithm does not terminate.

Exercise 6.3

There is an island, on which there are three kinds of natives: *knaves* always lie, *knights* always tell the truth, and *spies* sometimes lie and sometimes tell the truth. Some local people make statements about themselves and you have to find out what kind of natives they are. For example, three natives make the following statements:

A says: I am a spy.
B says: That is true.
C says: I am not a spy.

We know that A, B and C are of different kinds. Determine who they are by using the constraint handling library `ic`.

Hints:

- Encode knaves by the integer value 0, knights by 1, and spies by 2. Statements have the following syntax:

```
St ---> Ntv is Kind | St and St | St or St| not St
```

  Here `St` denotes a statement, while `Ntv` is a domain variable with domain `[0..2]` representing a native. Kind is an atom from the set {`knave`, `knight`, `spy`} and has to be handled internally via the respective integer encoding.

The atoms `says`, `and`, `or` and `not` are declared operators:

```
:-op(950, xfy, says).
:-op(800, yfx, and).
:-op(900, yfx, or).
:-op(700, fy, not).
```

- Define a predicate `truth(St,Truth)` which recursively processes the syntactic structure of statements `St` and constraints the variables in `St` according to a truth value `Truth` which can itself be a boolean constraint variable (i.e. a constraint variable with domain `[0,1]`).

  For this purpose, you can use the so-called reified constraints `and/3`, `or/3` and `#=/3` of the constraint handling library `ic`. For example,

  ```
  and(C1,C2,Truth)
  ```

  specifies that the two constraints `C1` and `C2` must be valid if and only if `Truth` is valid (that means, has value `1`). `Truth` may itself be a constraint variable. In case `Truth` gets instantiated to `0`, not both constraints can be fulfilled at the same time in a successful derivation.

- The entry point of your program should be the `says/2` predicate. The Prolog goal `X says St` expresses the constraint that native `X` says statement `St`. Use the predicate `truth/2` to translate `St` into constraints on the domain variable `X`.

Examples for testing:

- The example stated at the beginning translates as follows:

  ```
  ?- A says A is spy, B says A is spy, C says not C is spy,
     alldifferent([A,B,C]), labeling([A,B,C]).
  ```

  Your program should deliver the unique answer `A=0, B=2, C=1`.

- `?- A says A is knave, labeling([A]).`

  should give the unique answer `A=2`.

- `?- A says A is spy, labeling([A]).`

  should give the two answers `A=0` and `A=2`.

- `?- A says (A is knave or not A is spy) and not A is knight, labeling([A]).`

  should give the unique answer `A=2`.

- The webpage `http://newheiser.googlepages.com/knightsandknaves` provides a lot of examples for further testing.