

IAIP Excercises Week 9

November 3, 2005

1 Problems

1. Dechter, p.146, 5.7.2, 5.7.3
2. Dechter, p.147, (*) 5.7.7
 - (a) MANDATORY
 - (b) MANDATORY depending on code implementation
 - (c) MANDATORY depending on code implementation
 - (d) MANDATORY depending on code implementation

2 Code implementation

Consider the constraint network instance presented in Fig. 5.4 on page 125, under the variable ordering: $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ and ordering of colors $\{blue, green, red, tal\}$. Your task is to:

1. (*) Develop a simple **backtracking** algorithm (as in Fig. 5.3 on page 124) to solve this instance.
2. Develop a **forward checking** algorithm (figures 5.7 and 5.6 on pages 132 and 133) to solve this instance.

For each implementation you need to deliver:

- Print out the relevant parts of (well commented) code implementation, .
- Number of calls to SELECT-VALUE(-XXX) function, during search.
- Total number of *non-trivial consistency checks*, i.e. how many times you checked whether the relevant (table) constraint $tCons$ is consistent with (asn, x, v) extension of partial assignment asn with new variable assignment $x = v$.

The constraint $tCons$ is relevant with respect to (asn, x, v) iff all the variables in its scope are assigned a value by extended assignment $asn \cup (x, v)$. In case the scope of $tCons$ includes some variables that are not assigned by the extended assignment then the constraint is automatically satisfied (and is not relevant).

2.1 What is mandatory?

You have a choice between the following two "packages":

- Implement only backtracking algorithm and solve completely 5.7.7.
- Implement both backtracking and forward checking algorithm and solve only 5.7.7.a.

2.2 Evaluation criteria

The correctness of your implementation will be evaluated (among others) based on: number of calls to SELECT-VALUE(-XXX) and number of non-trivial consistency checks.

These numbers are unique if you stick to given variable ordering and ordering of color values (this has already been taken care of in the given instance of the code). It also depends on the value selection criteria: it assumes that when selecting a value from domain $List\ dom$ that you are taking the first value available: $Integer\ a = (Integer)\ dom.get(0)$

2.3 What is provided?

The code base for this implementation task is smaller than for the previous ones. After the input from the students we decided to provide you with more "open-ended" code base, i.e. we provided only the basic building blocks that you may choose to use if you want. This allows the desired programming flexibility: many alternative correct implementations. Also, it demands more coding.

You are given implemented function: $CSP::backTrackingSearch()$ that you can use to verify your code implementation.

The files included:

- CSPDemo.java
- MapCSP.java
- CSP.java
- Constraint.java
- TableConstraint.java

- ConstraintsSet.java
- Domains.java
- Assignment.java