CS592a (JAN-MAY 2017): CSP Assignment Problem Statements

Assignments

To be done together: Devise a standard Finite CSP <X, D, C> representation for the extensional form of relations, in consultation with other students. The CSP solvers should be able to read (and generate) the problems in this format. A sample (used last year) is in Appendix A.

1. [TJ] Develop an animation bench that allows a user to invoke demos using the following assignments. For example the user should be able to generate random CSPs with some given parameters, display them on the screen, and execute some of the assignments below. One should also be able to invoke one of the following specific problems.

   Generate explicit CSP representations for the following problems:
   a. n-Queens:
      i. Input: *n* - the number of queens on the *n* x *n* chessboard
   b. map colouring:
      ii. Inputs: *n* - the number of colours;
      iii. A map of countries (choose a suitable representation)
   c. crypto-arithmetic problem:
      iv. Inputs: A set of words to be added;
      v. A word that is the sum of the above set of words

   The program must read the above problems in some input format and output the problems into the format devised by the group.

2. [KTSM] Accept a CSP in explicit form and implement the Min-width and Min-induced-width algorithms to generate an ordering. Implement directional i-consistency on the resulting CSP. Accept a CSP in explicit form with a given ordering and implement adaptive consistency (ADC) using the bucket elimination algorithm. Display the contents (scopes and the relations) of every bucket after each iteration. Show how the solution is constructed.

3. [SJ] Accept a CSP in explicit form with a given ordering and implement the Generalized Lookahead Search algorithm. Implement the following select-value algorithms:
   a. Forward Checking
   b. Partial Lookahead
   c. Full Lookahead
   d. Full AC

   Depict the search graphically by listing the variables in a vertical order with values drawn as circles horizontally. Show the current partial solution with differently colored circles and edges linking the “related” values. Use color codes to distinguish between (a) past values tried (b) current values in partial solution (c) future values in past and future variables and (d) future values deleted by the lookahead algorithm.

4. [AR] Accept a CSP in explicit form with a given ordering and implement
    a. Gashnig's Backjumping.
    b. Graph Based Backjumping.
    c. Conflict-Directed Backjumping.

   Show at each stage the partial solution and the culprit variable identified at a dead-end.

   Extra: Depict the search graphically by listing the variables in a vertical order with values drawn as circles horizontally. Show the current partial solution with differently colored circles and edges linking the "related" values. Use color codes to distinguish between (a) past values tried (b) current values in partial solution (c) future values in past and (d) and the culprint variable the algorithm is jumping back to.

5. [SK] Animation (to fit into Assignment 1)
    a. Accept a CSP in explicit form and draw a matching diagram on the screen. Implement and animate the AC-3 algorithm. The animation must highlight the pairs of variables at each stage, and the values being removed.
    b. Accept a CSP in explicit form and draw a matching diagram on the screen. Implement and animate the AC-4 algorithm. Choose a suitable representation. The algorithm must clearly show how the removal of each value propagates changes into the representation.

6. [DS] Develop an application to produce a crossword puzzle for an Institute magazine. The input to the application are:
        i. A set of dictionary relevant words from the current issue of the magazine. Relevant means that stop words (like 'the', 'an", 'are") and removed.
        ii. A crossword puzzle grid specification (choose a suitable representation).
        iii. A list of crossword puzzles published in the past.

   The output should be a crossword puzzle (grid filled in with words from the given lexicon) for which the crossword setter will set the clues. The application use one of the solution methods developed, else use Backtracking. The application must
        i. avoid repeating words from recent puzzles.
        ii. allow the user to ask for another solution (puzzle).
        iii. allow the user to specify words in specific locations.
        iv. (optionally) provide an interface using which the setter can set the clues, after which the application should print the grid (without the words) and the clues for the words.

7. [NM] Accept a CSP in explicit form with a given ordering and implement Backtracking. Also implement the DVFC algorithm on the same problem. Compare Backtracking with the given ordering with DVFC.

   Depict the search graphically by listing the variables in a vertical order with values drawn as circles horizontally. Show the current partial solution with differently colored circles and edges liking the "related" values. Use color codes to distinguish between (a) past values tried (b) current values in partial solution (c) future values in past and future variables and (d) future values deleted by the lookahead algorithm.

Appendix A

Brief description:
The variableset tag corresponds to the set of all variables.
It contains one or more variable tags, each of which corresponds to an individual variable.
Every variable tag contains one or more domain tags that are the values the variable can take.

Next, the constraintset tag contains a set of constraint tags.
Every constraint tag contains a scope tag and a relation tag.
The scope tag contains an ordered list of variables participating in the constraint.
The relation tag contains zero or more rows that correspond to the values that the variables in the scope can take.

Finally, the ordering tag contains the order in which variables are to be processed.

File: graphcolouring.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<csp>
  <variableset>
    <variable name="X1">
      <domain value="r"/>
      <domain value="b"/>
      <domain value="g"/>
    </variable>
    <variable name="X2">
      <domain value="b"/>
    </variable>
  </variableset>
  <constraintset>
    <constraint>
      <scope>
        <variable name="X1"/>
        <variable name="X2"/>
      </scope>
      <relation>
        <row>
          <entry value="r"/>
          <entry value="b"/>
        </row>
        <row>
          <entry value="g"/>
          <entry value="b"/>
        </row>
      </relation>
    </constraint>
  </constraintset>
  <ordering>
    <variable name="X2"/>
    <variable name="X1"/>
  </ordering>
</csp>
```