Benchmarks

# Essence

Essence is constraint language for specifying combinatorial problems. We will use it as common problem implementation tool. Essence language can be directly translated to Gecode program or Minion input file using Taylor tool. Essence is general language which is easy to write and easy to learn and understand. Essence program consists of three parts. First part is definition of the version of the language, second part defines used variables and third part defines the constraints. Supported data types of variables are Booleans, integers and matrices. For example if we want to declare Boolean variable firstVar and integer variable secondVar whose values is within range from 5 to 9 the corresponding Essence statement would be

find firstVar

# N-queens

N-queens problem is a classical constraint satisfaction task. The task is to place n queens on an n×n chessboard such that none of them is able to capture any other using standard chess queen’s moves. Since there are n columns on the chessboard there have to be one queen in every column. This results in the simplification of the task. We do not have to specify problem as two-dimensional matrix. We can specify it as an n-dimensional vector Q where nth position of the vector represents the row of nth queen. The constraint “one queen does not attack any other” can be fulfilled by two constraints. The queen attacks all fields in row where it stands. Therefore all queens have to be on different rows and we can specify *all values of Q are different*. The queen also attacks all fields on diagonals which contains the field where the queen stands. Field [X ; Y] is on the same diagonal with the field [A ; B] if and only if the distance AX is the same as the distance of BY (see figure 1). This gives us second constraint *for queen Q1 placed on field [A;B] and queen Q2 placed on field [X;Y] must be true following formula: |A-X| != |B - Y|*. There exists symmetric solutions. To avoid most of them we can optionally specify constraint *Q1 < Qn*.

## Formal definition

Let have a vector Q of the length n with values in domain 1...n which satisfies following constraints:

1. [all values of Q are different]
2. [two queens are not on the same diagonal]
3. [optional constraint to avoid symmetry]

Any vector Q which satisfies defined constraints is solution of the n-queens problem.

## Example solutions

For n = 4 we have 4-queens problem. The solutions of 4-queens problem are [2,4,1,3], [3,1,4,2]. The solution are visualised on the chessboard on the figure 2. The highlighted fields means placement of the queens (Qx means *field contains queen number x*), the other fields are empty. We filled them with information which queen attacks which field (Xmn means *the field is attacked by the queen m and n*).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | X13 | X23 | Q3 | X34 | | Q1 | X12 | X13 | X14 | | X14 | X24 | X4 | Q4 | | X12 | Q2 | X23 | X24 |   Solution 1 | |  |  |  |  | | --- | --- | --- | --- | | X12 | Q2 | X23 | X24 | | X14 | X24 | X34 | Q4 | | Q1 | X12 | X13 | X14 | | X13 | X23 | Q3 | X34 |   Solution 2 |
| Figure 2 | |

There are two solutions found but as can be seen the solutions are symmetrical.

## Implementation in Essence

language ESSENCE' 1.b.a

find queens: matrix indexed by [int(1..n)] of int(1..n)

such that

forall i: int(1..n). forall j: int(i+1..n).

| queens[i] - queens[j] | != | i - j |,

alldiff(queens)

# Magic sequence

Magic sequence MS is special sequence of the length n which satisfies the constraint that number x is in MS contained exactly MS(x) times (vector values are numbered from 0). It is easy to estimate that the vector values have to be in domain 0...n-1. The value cannot be greater than n, because in such case it would mean that some number is contained more than n times in sequence of length of n. It cannot be greater than n-1. If there is some x such that MS(x) = n there must be . But since the vector is indexed starting from 0 it means it has maximal index n-1 so index n is out of range.

## Formal definition

Let have a vector MS of length n which satisfies following constraint:

## Implementation in Essence

language ESSENCE' 1.b.a

find s : matrix indexed by [int(0..n-1)] of int(0..n)

such that

forall i : int(0..n-1).

( s[i] = (sum j : int(0..n-1). (s[j] = i)))

# Self Referential Quiz

Self referential quiz is a set of questions whose answers depends one on another. Typical questions in this type of quizzes are “1. First question with answer A is: A) 1, B) 2, C) 3, D) 4, E) 5; 2. Answer to this question is: A) A, B) B, C) C, D) D, E) E.”. Reader can find methods to create such quiz in \cite{bubalo}. We will use the same quiz as Fernandez which contains following ten questions:

1. The first question whose answer is A is: (A) 4 (B) 3 (C) 2 (D) 1 (E) none of above
2. The only two consecutive questions with identical answers are: (A) 3 and 4 (B) 4 and 5 (C) 5 and 6 (D) 6 and 7 (E) 7 and 8
3. The next question with answer A is: (A) 4 (B) 5 (C) 6 (D) 7 (E) 8
4. The first even numbered question with answer B is: (A) 2 (B) 4 (C) 6 (D) 8 (E) 10
5. The only odd numbered question with answer C is: (A) 1 (B) 3 (C) 5 (D) 7 (E) 9
6. A question with answer D: (A) comes before this one, but not after this one (B) comes after this one, but not before this one (C) comes before and after this one (D) does not occur at all (E) none of the above
7. The last question whose answer is E is: (A) 5 (B) 6 (C) 7 (D) 8 (E) 9
8. The number of questions whose answers are consonants is: (A) 7 (B) 6 (C) 5 (D) 4 (E) 3
9. The number of questions whose answers are vowels is: (A) 0 (B) 1 (C) 2 (D) 3 (E) 4
10. The answer to this question is: (A) A (B) B (C) C (D) D (E) E

This quiz can be implemented as table 10 times 5 Boolean variables. Value in the row x and column y is true if and only if answer to question x is y (A=1, B=2, C=3, D=4, E=5). Because there is answer to every question and also every question has only one answer we have to set the constraint that *there is only one true value in a row*.

## Solution

There exists only one solution which is shown in figure x.y.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0 | 1 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 1 | 0 |
| Figure x.y | | | | | |

## Implementation in Essence

language ESSENCE' 1.b.a

find s : matrix indexed by [int(1..10), int(1..5)] of bool

such that

$ the is only one answer to each question and there is not any unanswered question

forall row : int(1..10). ((sum col : int(1..5). s[row,col]) = 1),

$ Question 1

$ A to D

forall col : int(1..4). ( (s[1,col] = 1) <=> ( (s[(4-col+1),1] = 1) /\ ( forall row : int(1..(4-col)). (s[row,1] = 0) ) ) ),

$ E

(s[1,5] = 1) <=> (forall row : int(1..4). (s[row,1] = 0)),

$ Question 2

forall col : int(1..5). ( (s[2,col] = 1) <=> ( forall col2: int(1..5). (s[3+col-1,col2] = s[3+col,col2]) ) ),

$ Question 3

forall col : int(1..5). ( (s[3,col] = 1) <=> ( (s[(4+col-1),1] = 1) /\ ( forall row : int (4..2+col). s[row,1] = 0 ) ) ),

$ Question 4

forall col : int(1..5). ( (s[4,col] = 1) <=> ( (s[col\*2,2] = 1) /\ ( forall row : int(1..(col-1)). s[row\*2,2] = 0 ) ) ),

$ Question 5

forall col : int(1..5). ( (s[5,col] = 1) <=> (s[2\*col-1,3]=1) ),

$ Question 6

(s[6,1] = 1) <=> ( ( exists row : int(1..5). s[row,4] = 1 ) /\ ( forall row : int (7..10). s[row,4] = 0 ) ),

(s[6,2] = 1) <=> ( ( exists row : int(7..10). s[row,4] = 1 ) /\ ( forall row : int (1..5). s[row,4] = 0 ) ),

(s[6,3] = 1) <=> ( ( exists row : int(7..10). s[row,4] = 1 ) /\ ( exists row : int (1..5). s[row,4] = 1 ) ),

(s[6,4] = 1) <=> ( forall row : int (1..10). s[row,4] = 0 ),

(s[6,5] = 1) <=> (s[6,4] = 1),

$ Question 7

forall col : int(1..5). ( (s[7,col] = 1) <=> ( (s[col+4,5] = 1) /\ ( forall row : int (col+4+1..10). s[row,5] = 0 ) ) ),

$ Question 8

forall col: int(1..5). ( (s[8,col] = 1) <=> ( ( sum row: int(1..10). (s[row,2] + s[row,3] + s[row,4]) ) = (7-col+1) ) ),

$ Question 9

forall col: int(1..5). ( (s[9,col] = 1) <=> ( ( sum row: int(1..10). (s[row,1] + s[row,5]) ) = (col-1) ) )

$ Constraints for question 10 are useless