UDC:004
Original scientific paper

MODELING AND SOLVING SELF-REFERENTIAL PUZZLES

Maja Bubalo
Faculty of Pedagogy,Osijek, Croatia
maja@ffos.hr

Mirko Čubrilo
Faculty of Organization and Informatics
Pavlinska 2, Varaždin, Croatia
mirko.cubrilo@foi.hr

Abstract: *The so-called self-referential puzzles are a very interesting kind of logic puzzles, aiming at developing the skill of logical thinking. A self-referential puzzle consists of a sequence of questions about the puzzle itself. In this paper, we shall show some self-referential puzzles, demonstrate how to model and solve them as propositional logic problems, and how to mechanically generate new puzzles. For this, we shall make use of the specific advantages of Mozart/Oz system – the finite domain constraint programming language and environment.*

We shall also show some new puzzles, according to our best knowledge not yet published elsewhere. The program in Mozart/Oz using our method generated these puzzles.

Keywords: *self-referential puzzles, logic programming, finite domain constraint programming, Mozart/Oz.*

1. INTRODUCTION

It is known that it is much easier to learn something by playing games. This is especially true of learning to think in a logical manner, which is generally considered as a difficult skill to learn. In this paper, we consider some interesting logic puzzles, with which one can play and learn herself to reason logically. These puzzles are of a special kind, known as *self-referential puzzles* (also called *quizzes* or *tests*). Our interest in this paper is to show how to solve such kind of logic puzzles by the means of computer program, and especially how to *mechanically generate* such new puzzles.

The first task is to describe the self-referential puzzles. Each puzzle has some number (very often 10) of questions about itself. Every answer is only one of the letters A, B, C, D, and E. The last question is the answer to the puzzle. On Internet self-referential puzzles with up to 20 questions can be found, which are not more difficult to solve then the puzzles we are considering here.

The best introduction to this sort of puzzles is made by examining them. Let us examine a very interesting puzzle dating from 1996 (found on the Internet).

THE PUZZLE 1. (Martin Henz)

1. The first question whose answer is A is the question
   (A) 4 (B) 3 (C) 2 (D) 1 (E) none of the above

2. Identical answers have questions
   (A) 3 and 4 (B) 4 and 5 (C) 5 and 6 (D) 6 and 7 (E) 7 and 8

3. The next question whose answer is A is the question
   (A) 4 (B) 5 (C) 6 (D) 7 (E) 8

4. The first even numbered question whose answer is B is the question
   (A) 2 (B) 4 (C) 6 (D) 8 (E) 10

5. The only odd numbered question whose answer C is the question
   (A) 1 (B) 3 (C) 5 (D) 7 (E) 9

6. The question with answer D
   (A) comes before this one, but not after
   (B) comes after this one, but not before
   (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 the question
   (A) 5 (B) 6 (C) 7 (D) 8 (E) 9

8. The number of questions whose answer are consonants
   (A) 7 (B) 6 (C) 5 (D) 4 (E) 3

9. The number of questions whose answer are vowels
   (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

2. HOW TO SOLVE THE SELF-REFERENTIAL PUZZLE?

One way is to read all the puzzle questions and begin with the first question and the first answer possible, namely the answer A. For all the possibilities explore the answers to other questions and find the solution. It is very useful to write a table and note the answers. When the answer is not true, come back to the place where the last true answer was and go on with the next possible answer.

For example, we solve the puzzle 1 this way.

If we choose the first answer A (1: A) and discuss it, we can see that this answer is impossible. It asserts that the first question with the answer A is the fourth question. It cannot be true anyway.

So we can go to the next possibility for the first question – it is the answer B (1: B). It asserts that the third answer is also A (3: A). If the third answer was A, that asserts the forth answer to be A. Now, look at the second question about identical answers. It asserts that the answer to the second question is also A (2: A), but that is not possible because there are only two questions in the order with the identical answers.

Now, we write a table and start with 1:C. Here is the complete table of the solution we shall talk about in the next paragraph:

1: C						→	1. C
↳	2: A					→	2. A
	I 3:B					→	3. B
	↓	I 4:B				→	4. B
	5:A					→	5. A
	6: B	→ →	→ →	→ →	↓	→	6. B
7:A no	7:C no	7: D no	7 :E		↓	→	7. E
			↓	8: B	↓	→	8. B
			↳	9:E ↑	↓	→	9. E
					10: D !!!	→	10. D

3. THE SELF-REFERENTIAL PUZZLE AS A PROPOSITIONAL PROBLEM

In this section, we discuss the self–referential puzzles as logic systems. For our purpose we suppose that they have only one solution. We shall not discuss puzzles with more solutions or no solutions at all. (It is usual to publish puzzles with just one solution, because the people solving them are mostly interested in this kind of puzzles.)

How can we explain the system? There are 10 questions with only 5 possible answers – A, B, C, D, and E. If we write a logic formula for all the answers, they could either be true or false.

First we must introduce the symbols. For every question we introduce propositional variable Q_i, where $i \in \{1...10\}$ and $Q \in \{A, B, C, D, E\}$. For example, C_2 represents that the answer to question 2 is C. We also use logical connectives: $\wedge$ for conjunction, $\vee$ for disjunction, $\neg$ for negation and $\equiv$ for equivalence.

Now, we can write the first formula for every question Q_j, $j \in \{1...10\}$:

$$\begin{aligned}&A_j \wedge \neg B_j \wedge \neg C_j \wedge \neg D_j \wedge \neg E_j \ \vee \ \neg A_j \wedge B_j \wedge \neg C_j \wedge \neg D_j \wedge \neg E_j \vee \\ &\neg A_j \wedge \neg B_j \wedge C_j \wedge \neg D_j \wedge \neg E_j \ \vee \ \neg A_j \wedge \neg B_j \wedge \neg C_j \wedge D_j \wedge \neg E_j \vee \\ &\neg A_j \wedge \neg B_j \wedge \neg C_j \wedge \neg D_j \wedge E_j\end{aligned} \qquad (1)$$

We know that (1) is true for every j. It is more practical to write the following abbreviation

$$A_j + B_j + C_j + D_j + E_j = 1 \qquad (2)$$

Also, let us introduce the equivalence of i and j

$$i \,\tilde{=}\, j ::= (A_i \equiv A_j) \wedge (B_i \equiv B_j) \wedge (C_i \equiv C_j) \wedge (D_i \equiv D_j) \wedge (E_i \equiv E_j) \qquad (3)$$

The usual notation for the summation is $\sum$.

We will sort the questions in the groups and use this for the purpose of generating puzzles.

The questions in the first group are questions about the first question with answer Q, the last question with answer Q, and the next question with answer Q. This sort of question we shall solve with formulae in which the only logic connectives are conjunction and negation. For example, the formulae $Q_i \equiv D_7 \wedge \neg D_4 \wedge \neg D_5 \wedge \neg D_6$, $Q \in \{A, B, C, D, E\}$ tell us that the answer to the question number i will be Q if the next question with answer D is the 7th question, not the 4th or the 5th or the 6th. In the same way, we can discuss all the questions in the first group. It is also very important to notice that in the course of generating puzzles that sort of questions must be about all the possible answers A, B, C, D and E.

The questions in the second group are about the sums and they are telling us about the sums for single letters, the sums for the vowels, and the sums for the consonants. In every moment we can easily calculate the number of the questions with, for example, letter B. The formula for the sum B is $Q_i \equiv \Sigma_{i \in \{1 \ldots 10\}} B_i = 4$. That means, if there are 4 questions in the puzzle we are solving with the answer B, then the answer to the question i is Q, $Q \in \{A, B, C, D, E\}$.

The third group contains two questions in a row with the same answer. For these questions we use the formula:

$$i \tilde{=} i+1 ::= (A_i \equiv A_{i+1}) \wedge (B_i \equiv B_{i+1}) \wedge (C_i \equiv C_{i+1}) \wedge (D_i \equiv D_{i+1}) \wedge (E_i \equiv E_{i+1})$$

The fourth group consists of different questions not belonging to the groups above.

In the first group we can put questions 1, 3, 4, 5 and 7.

$A_1 \equiv A_4 \wedge \neg A_1 \wedge \neg A_2 \wedge \neg A_3,$

$B_1 \equiv A_3 \wedge \neg A_1 \wedge \neg A_2,$

$C_1 \equiv A_2 \wedge \neg A_1,$

$D_1 \equiv A_1,$

$E_1 \equiv \neg A_1 \wedge \neg A_2 \wedge \neg A_3 \wedge \neg A_4,$

$A_3 \equiv A_4$

$B_3 \equiv A_5 \wedge \neg A_4$

$C_3 \equiv A_6 \wedge \neg A_4 \wedge \neg A_5$

$D_3 \equiv A_7 \wedge \neg A_4 \wedge \neg A_5 \wedge \neg A_6$

$E_3 \equiv A_8 \wedge \neg A_4 \wedge \neg A_5 \wedge \neg A_6 \wedge \neg A_7$

$A_4 \equiv B_2$

$B_4 \equiv B_4 \wedge \neg B_2$

$C_4 \equiv B_6 \wedge \neg B_2 \wedge \neg B_4$

$D_4 \equiv B_8 \wedge \neg B_2 \wedge \neg B_4 \wedge \neg B_6$

$E_4 \equiv B_{10} \wedge \neg B_2 \wedge \neg B_4 \wedge \neg B_6 \wedge \neg B_8$

$A_5 \equiv C_1 \wedge \neg C_3 \wedge \neg C_5 \wedge \neg C_7 \wedge \neg C_9$

$B_5 \equiv \neg C_1 \wedge C_3 \wedge \neg C_5 \wedge \neg C_7 \wedge \neg C_9$

$C_5 \equiv \neg C_1 \wedge \neg C_3 \wedge C_5 \wedge \neg C_7 \wedge \neg C_9$

$D_5 \equiv \neg C_1 \wedge \neg C_3 \wedge \neg C_5 \wedge C_7 \wedge \neg C_9$

$E_5 \equiv \neg C_1 \wedge \neg C_3 \wedge \neg C_5 \wedge \neg C_7 \wedge C_9$

$A_7 \equiv E_5 \wedge \neg E_6 \wedge \neg E_7 \wedge \neg E_8 \wedge \neg E_9 \wedge \neg E_{10}$

$B_7 \equiv E_6 \wedge \neg E_7 \wedge \neg E_8 \wedge \neg E_9 \wedge \neg E_{10}$

$C_7 \equiv E_7 \wedge \neg E_8 \wedge \neg E_9 \wedge \neg E_{10}$

$D_7 \equiv E_8 \wedge \neg E_9 \wedge \neg E_{10}$

$E_7 \equiv E_9 \wedge \neg E_{10}$

The 8th and the 9th question are from the second group.

$A_8 \equiv \Sigma_{i \in \{1...10\}} B_i + C_i + D_i = 7$

$B_8 \equiv \Sigma_{i \in \{1...10\}} B_i + C_i + D_i = 6$

$C_8 \equiv \Sigma_{i \in \{1...10\}} B_i + C_i + D_i = 5$

$D_8 \equiv \Sigma_{i \in \{1...10\}} B_i + C_i + D_i = 4$

$E_8 \equiv \Sigma_{i \in \{1...10\}} B_i + C_i + D_i = 3$

$A_9 \equiv \Sigma_{i \in \{1...10\}} A_i + E_i = 0$

$B_9 \equiv \Sigma_{i \in \{1...10\}} A_i + E_i = 1$

$C_9 \equiv \Sigma_{i \in \{1...10\}} A_i + E_i = 2$

$D_9 \equiv \Sigma_{i \in \{1...10\}} A_i + E_i = 3$

$E_9 \equiv \Sigma_{i \in \{1...10\}} A_i + E_i = 4$

From the third group there is only the question 2.

$A_2 \equiv 3 \tilde{=} 4,$

$B_2 \equiv 4 \tilde{=} 5,$

$C_2 \equiv 5 \tilde{=} 6,$

$D_2 \equiv 6 \tilde{=} 7,$

$E_2 \equiv 7 \tilde{=} 8,$

At last, we also have only the 6th question from the fourth group.

$A_6 \equiv (D_1 \vee D_2 \vee D_3 \vee D_4 \vee D_5) \wedge \neg(D_7 \vee D_8 \vee D_9 \vee D_{10})$

$B_6 \equiv \neg(D_1 \vee D_2 \vee D_3 \vee D_4 \vee D_5) \wedge (D_7 \vee D_8 \vee D_9 \vee D_{10})$

$C_6 \equiv (D_1 \vee D_2 \vee D_3 \vee D_4 \vee D_5) \wedge (D_7 \vee D_8 \vee D_9 \vee D_{10})$

$D_6 \equiv \Sigma_{i \in \{1...10\}} D_i = 0$

$E_6 \equiv D_6$

Finally, we can easily write the formula for the self-referential puzzle in consideration. All answers must be written in a row. The solution is already known, but the problem is how to find it mechanically. Several solvers (tools) can do this in half an hour. The first to suggest the Mozart/Oz system and the finite domain constraint programming as the appropriate environment/language and solution method was M. Henz in the paper [6]. Of course, the system solves the problem in a fraction of a second.

4. SOLUTION OBTAINED WITH THE FINITE DOMAIN PROGRAMMING IN THE MOZART/OZ SYSTEM

Here we give an introduction to the constraint programming in Oz. In this paper we restrict our attention to solving self-referential puzzles. More information about the system can be found at www.mozart-oz.com .

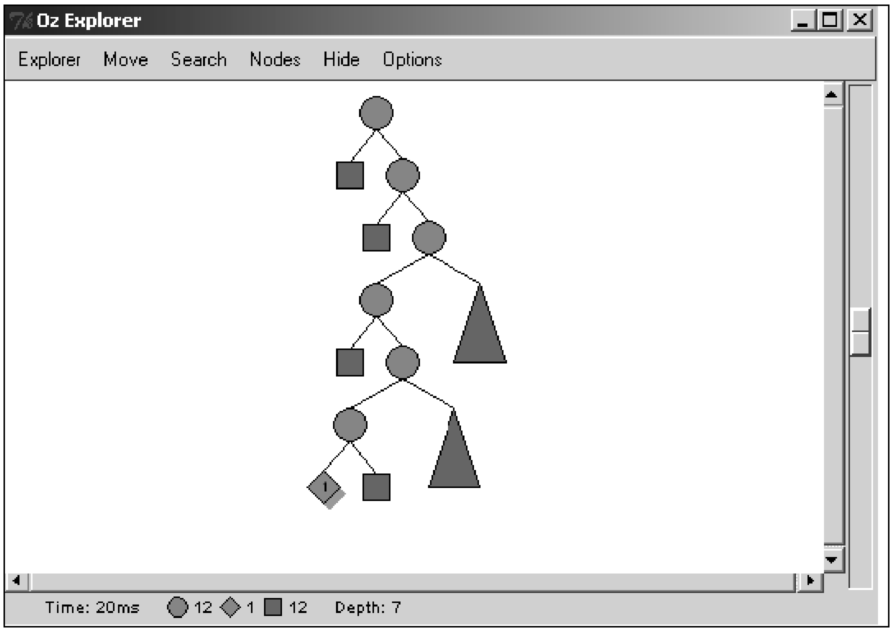
Self-referential puzzle can be stated with variables ranging over finite sets of nonnegative integers.

Two basic techniques of constraint programming are constraint propagation and constraint distribution.

Here is the program:

```
% oz example 1
declare PRIM1 Q in
proc {PRIM1 Q}
  proc {Vector V}  % V is a 0/1-vector of length 10
    {FD.tuple v 10 0#1 V}
  end
  proc {Sum V S}   % S is the sum of the components of vector V
    {FD.decl S} {FD.sum V '=:' S}
  end
  proc {Assert I U V W X Y}
    A.I=U  B.I=V  C.I=W  D.I=X  E.I=Y
  end
  A  = {Vector}    B = {Vector}
  C  = {Vector}    D = {Vector}   E = {Vector}
  SumA   = {Sum A}   SumB = {Sum B}   SumC = {Sum C}
  SumD   = {Sum D}   SumE = {Sum E}
  SumAE = {Sum [SumA SumE]}    SumBCD = {Sum [SumB SumC SumD]}
in
  {FD.tuple q 10 1#5 Q}
  {For 1 10 1
   proc {$ I} {Assert I  Q.I=:1  Q.I=:2  Q.I=:3  Q.I=:4  Q.I=:5} end}
  %% 1 1c
  {Assert 1 {FD.conj A.4 (A.1+A.2+A.3=:0)}
         {FD.conj A.3 (A.2+A.1=:0)}
         {FD.conj A.2 (A.1=:0)}
          A.1
         {FD.conj E.1 (A.1+A.2+A.3+A.4=:0)}}
  %% 2 2a
  {Assert 2 Q.3=:Q.4  Q.4=:Q.5  Q.5=:Q.6  Q.6=:Q.7 Q.7=:Q.8}
  Q.1\=:Q.2 Q.2\=:Q.3  Q.8\=:Q.9  Q.9\=:Q.10
  %% 3 3b
  {Assert 3 A.4
         {FD.conj A.5 (A.4=:0)}
         {FD.conj A.6 (A.4+A.5=:0)}
```

```
        {FD.conj A.7 (A.4+A.5+A.6=:0)}
        {FD.conj A.8 (A.4+A.5+A.6+A.7=:0)}}
  %% 4 4b
  {Assert 4 B.2
        {FD.conj B.4 (B.2=:0)}
        {FD.conj B.6 (B.2+B.4=:0)}
        {FD.conj B.8 (B.2+B.4+B.6=:0)}
        {FD.conj B.10 (B.2+B.4+B.6+B.8=:0)}}
  %% 5 5a
  {Assert 5 {FD.conj C.1 (C.3+C.5+C.7+C.9=:0)}
         {FD.conj C.3 (C.1+C.5+C.7+C.9=:0)}
         {FD.conj C.5 (C.3+C.1+C.7+C.9=:0)}
         {FD.conj C.7 (C.3+C.5+C.1+C.9=:0)}
         {FD.conj C.9 (C.3+C.5+C.7+C.1=:0)}}
  %% 6 6b
  {Assert 6 {FD.conj A.6 ((~(D.1+D.2+D.3+D.4+D.5))+(D.7+D.8+D.9+D.10))=:0}
        {FD.conj B.6 (D.1+D.2+D.3+D.4+D.5+(~(D.7+D.8+D.9+D.10)))=:0}
        {FD.conj C.6 (~(D.1+D.2+D.3+D.4+D.5+D.7+D.8+D.9))=:0}
        SumD=:0 _}
  %% 7 7e
  {Assert 7 {FD.conj E.5 (E.6+E.7+E.8+E.9+E.10=:0)}
        {FD.conj E.6 (E.7+E.8+E.9+E.10=:0)}
        {FD.conj E.7 (E.8+E.9+E.10=:0)}
        {FD.conj E.8 (E.9+E.10=:0)}
        {FD.conj E.9 (E.10=:0)}}
  %% 8 8b
  {FD.element Q.8 [7 6 5 4 3] SumBCD}
  %% 9 9e
  {FD.element Q.9 [0 1 2 3 4] SumAE}
  %% 10 10d
  {FD.distribute ff Q}
end
{ExploreAll PRIM1}
```


Oz Explorer
Explorer Move Search Nodes Hide Options
Time: 20ms ● 12 ◆ 1 ■ 12 Depth: 7

For interested human solvers, we present another one, even more difficult puzzle to solve.

THE SECOND PUZZLE:

1. The first question whose answer is B is the question
   (A) 2 (B) 3 (C) 4 (D) 5 (E) 6

2. The only two consecutive questions with identical answers are the questions
   (A) 2 and 3 (B) 3 and 4 (C) 4 and 5 (D) 5 and 6 (E) 6 and 7

3. The last question with the same answer as this one is the question (A) 10
   (B) 9 (C) 8 (D) 7 (E) 6

4. The number of questions with the answer A is
   (A) 0 (B) 1 (C) 2 (D) 3 (E) 4

5. The answer to this question is the same as the answer to the question
   (A) 10 (B) 9 (C) 8 (D) 7 (E) 6

6. The number of questions with answer A equals the number of questions
   with the answer (A) B (B) C (C) D (D) E (E) none of the above

7. Alphabetically, the answer to this question and the answer to the
   following question are (A) 4 apart (B) 3 apart (C) 2 apart (D) 1 apart
   (E) the same

8. The number of questions whose answers are vowels is
   (A) 2 (B) 3 (C) 4 (D) 5 (E) 6

9. The number of questions whose answer is a consonant is
   (A) a prime (B) a factorial (C) a square (D) a cube (E) divisible by 5

10. The answer to this question is
    (A) A (B) B (C) C (D) D (E) E

5. GENERATING PUZZLES

Now we are ready to discuss one very special point. The most interesting thing is how to generate new puzzles. There are so many possible choices, but which is the best one? There are many different sorts of questions. The idea is to write 10 answers and all facts we know about each of them. Of course, the possible answers are a, b, c, d, and e. For example, we can choose the sequence (the row or the series): B, E, C, D, D, E, C, A, E, and B.

If we look at this sequence, we can see that there is one question with the answer A, two questions with the answers B, C, and D, and three questions with the answer D. We know the sum of the vowels and the sum of the consonants. But, the only two questions with the same answers are the fourth and the fifth. The only odd number with the answer B is the first, the only even number with the answer A is the number 8.

When we see all these facts, we can write a new puzzle. We must collect the right questions which have the answers right in the sequence, and the result should be a new puzzle. For solver to be able to do that, there should be enough information to select the right answers, which means that the questions must be about every possible answer – some information about question with answer A, some information about the question with answer B and so on.

Now, the Oz is helping us to see if the puzzle had the only one solution. We write the program and test it. Sometimes, when there are many solutions, or no solution at all, we have to change some questions.

First, here is the generated puzzle we discussed earlier.

GENERATED PUZZLE 1:

1. The first question whose answer is E is the question
   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

2. The only odd numbered question whose answer is B is the question
   9 (B) 7 (C) 5 (D) 3 (E) 1

3. The only two consecutive questions with identical answers are the questions
   (A) 2 and 3 (B) 3 and 4 (C) 4 and 5 (D) 5 and 6 (E) 6 and 7

4. The only even numbered question whose answer is A is the question
   (A) 2 (B) 4 (C) 6 (D) 8 (E) 10

5. The number of questions with the answer B is
   (A) 5 (B) 4 (C) 3 (D) 2 (E) 1

6. The last odd question with the same answer as this one is the question (A) 1
   (B) 3 (C) 5 (D) 7 (E) 9

7. The answer to this question is the same as the answer to the question
   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

8. Alphabetically, the answer to this question and the answer to the following question are (A) 4 apart (B) 3 apart (C) 2 apart (D) 1 apart (E) the same

9. The number of questions whose answer is a consonant is
   (A) a prime (B) a square (C) a cube (D) divisible by 5 (E) a factorial

10. The answer to this question is (A) A (B) B (C) C (D) D (E) E

Here, of course, we already knew the solution. It consist of the sequence of answers at the beginning of the paragraph 5, where we can also see the program in Oz and the solution tree.

```
% This is a generated quizz
declare GENERI Q in
proc {GENERI Q}
  proc {Vector V}
    {FD.tuple v 10 0#1 V}
  end
  proc {Sum V S}
    {FD.decl S} {FD.sum V '=:' S}
  end
  proc {Assert I U V W X Y}
    A.I=U  B.I=V  C.I=W  D.I=X  E.I=Y
  end
  A = {Vector}   B = {Vector}
  C = {Vector}   D = {Vector}    E = {Vector}
  SumA   = {Sum A}     SumB = {Sum B}   SumC = {Sum C}
  SumD   = {Sum D}      SumE = {Sum E}
  SumAE = {Sum [SumA SumE]}     SumBCD = {Sum [SumB SumC SumD]}
in
  {FD.tuple q 10 1#5 Q}
  {For 1 10 1
   proc {$ I} {Assert I  Q.I=:1  Q.I=:2  Q.I=:3  Q.I=:4  Q.I=:5} end}
     {Assert 1 E.1
         {FD.conj E.2 (E.1=:0)}
         {FD.conj E.3 (E.1+E.2=:0)}
         {FD.conj E.4 (E.1+E.2+E.3=:0)}
         {FD.conj E.5 (E.1+E.2+E.3+E.4=:0)}}
  {Assert 2  {FD.conj B.9 (B.1+B.3+B.5+B.7=:0)}
          {FD.conj B.7 (B.1+B.3+B.5+B.9=:0)}
          {FD.conj B.5 (B.1+B.3+B.7+B.9=:0)}
          {FD.conj B.3 (B.1+B.5+B.7+B.9=:0)}
          {FD.conj B.1 (B.3+B.5+B.7+B.9=:0)}}
    {Assert 3  Q.2=:Q.3  Q.3=:Q.4  Q.4=:Q.5  Q.5=:Q.6  Q.6=:Q.7}
  Q.1\=:Q.2  Q.7\=:Q.8  Q.8\=:Q.9  Q.9\=:Q.10
    {Assert 4  {FD.conj A.2 (A.4+A.6+A.8+A.10=:0)}
          {FD.conj A.4 (A.2+A.6+A.8+A.10=:0)}
```

```
        {FD.conj A.6 (A.4+A.2+A.8+A.10=:0)}
        {FD.conj A.8 (A.4+A.6+A.2+A.10=:0)}
        {FD.conj A.10 (A.4+A.6+A.8+A.2=:0)}}
  %% 5 Sum B  5d
  %% a-4 b-3 c-2 d-1 e-0
  {FD.element Q.5 [5 4 3 2 1] SumB}
  {Assert 6  {FD.conj A.1 (A.3+A.5+A.7+A.9=:0)}
        {FD.conj B.3 (B.5+B.7+B.9=:0)}
        {FD.conj C.5 (C.7+C.9=:0)}
        {FD.conj D.7 (D.9=:0)}
         E.9}
   %% 7 7c
  {Assert 7  Q.1=:Q.7  Q.2=:Q.7  Q.3=:Q.7  Q.4=:Q.7  Q.5=:Q.7}
    {FD.element Q.8 [4 3 2 1 0] {FD.decl}={FD.distance Q.8 Q.9 '=:'}}
    {Assert 9 SumBCD::[2 3 5 7] SumBCD::[0 1 4 9]  SumBCD::[0 1 8]
     SumBCD::[0 5 10]   SumBCD::[1 2 6]   }
  %% 10 10.b
  {FD.distribute ff Q}
end
%{Browse {SearchAll GENERI}}
{ExploreAll GENERI}
```

GENERATED PUZZLE 2:

1. The first question whose answer is D is the question
   (A) 8 (B) 7 (C) 6 (D) 5 (E) 4

2. Identical answers have questions
   (A) 3 and 4 (B) 4 and 5 (C) 5 and 6 (D) 6 and 7 (E) 7 and 8

3. The number of questions with the answer E is
   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

4. The number of questions with the answer A is
   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

5. The number of questions with the answer A equals the number of questions with the answer (A) A (B) B (C) D (D) E (E) none of the above

6. The last question whose answer is B is the question
   (A) 5 (B) 6 (C) 7 (D) 8 (E) 9

7. Alphabetically, the answer to this question and the answer to the following question are (A) 4 apart (B) 3 apart (C) 2 apart (D) 1 apart
   (E) the same

8. The answer to this question is the same as the answer to the question
   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

9. The number of questions whose answers are consonants
   (A) 3 (B) 4 (C) 5 (D) 6 (E) 7

10. The answer to this question is
    (A) A (B) B (C) C (D) D (E) E

We tried to make the puzzle more interesting, so the answer to the tenth question also presents new information for the solver.

6. CONCLUSION

In this paper we demonstrated how one could solve or even generate some kind of self-referential puzzles (quizzes) in the constraint-programming environment Mozart/Oz. Our contribution primarily relates to the development of a kind of heuristics for generating new self-referential puzzles, exploring some properties of the Mozart/Oz environment properties.

REFERENCES

[1] R. Smullyan: *Diagonalization and Self-Reference*, Clarendon press Oxford,1994.

[2] M. Čubrilo: *Mathematical Logic for Expert Systems* (in Croatian), Informator, Zagreb, Croatia,1989.

[3] M. Vuković : *Mathematical logic* (in Croatian), Faculty of Mathematics, Zagreb University,Zagreb, Croatia, 2000.

[4]] www.mozart-oz.org

[5] M. Henz : *Don't Be Puzzeled!*, Workshop on Constraint Programming Aplications at CP 1996.

[6] Yun Fong Lim, Seet Chong Lua, Xiao Ping Shi, J. Paul Walser, Roland H. C. Yap : *Solving Hierarchical Constraints over Finite Domains*, Annals of Mathematics and Artificial Intelligence (AMAI), 40(3), pp 283-302, March 2004.

[7] P.van Roy, P. Brand, D. Duchier, S. Haridi, C. Schulte: *Logic programming in the context of multiparadigm programming: the Oz experience*, Journal of Theory and Practice of Logic Programming (TPLP), 3(6), pp 715-763, November 2003.

[8] Ka Boon Ng, Chiu Wo Choi, Martin Henz: *A Software Engineering Approach to Constraint Programming Systems*, APSEC 2002.