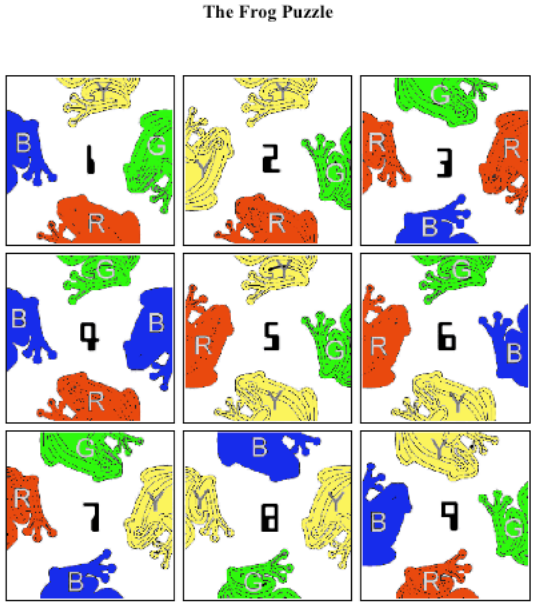
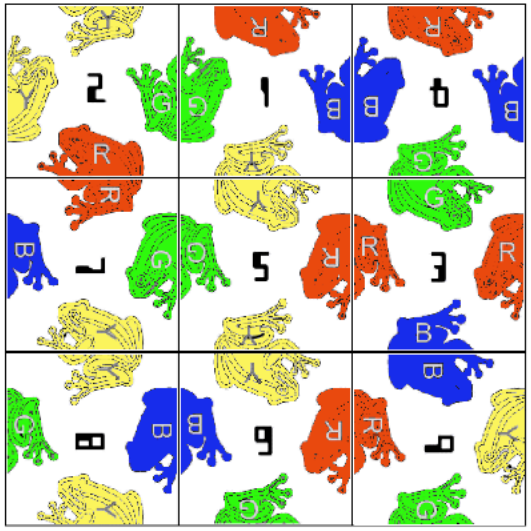
Frog Card Puzzle

Puzzle and solution
from Frank Kriwaczek



A solution to the Frog Puzzle



Brute force trial and error no good

- There are 9*8*7….*1 different ways of placing cards in the 9 slots
- But each card can be in one of 4 different rotations, I.e. 4^9 options
- Hence in total there are 9!* 4^9 diff. placings
  = **95,126,814,720**
- If we first generate a complete placing and then test it, even if we can generate and test 1000 a second, will take 3 years
- Need to have a program which does frog match tests as it incrementally generates a placing, as we would do in trying to solve the puzzle

How to represent a solution

- Each of the slots in solution has a card identifier - a number between 1 and 9 - and a rotation
- Rotation can we a number between 0 and 3 giving number of right angles card has been rotated clockwise
- So a solution can be a list of pairs of numbers:

  [(Slot1,Slot1R), (Slot2,Slot2R), (Slot3,Slot3R), (Slot4,Slot4R), (Slot5,Slot5R), (Slot6,Slot6R), (Slot7,Slot7R), (Slot8,Slot8R), (Slot9,Slot9R)]

  where each Sloti var has value in range 1 to 9
  and each SlotiR var has value in range 0 to 3
- Integer value of Slot1 tells use which number card, say card number 2, goes in slot 1 - the top left of puzzle
- Integer value of SlotR, say 0, tells us by how much it is rotated, if at all.
- Slot1,Slot2,Slot3 are the top row, Slot4, Slot5, Slot6

Describing a card

Have 9 facts of the form:
card(Id, FrogN, FrogE, FrogS, FrogW).

Where Id is an identifying number between 1 and 9

Each Frog argument is (Col,FrogPart) describing the frog picture on that edge of the card, e.g (red,body)

So card number 1 is described by fact:
card(1,(red,head), (blue,body), (yellow,body), (green,head)).

Describing a rotated card

Need to define:

rotated_card(Id,R,FrogN,FrogE,FrogS,FrogW)

which tells gives us the description of the card if it has been rotated by R right angles clockwise.

Need this for our alignment tests.

```
rotated_card(Id,0, FrogN,FrogE,FrogS,FrogW):-
        card(Id, FrogN,FrogE,FrogS,FrogW).
rotated_card(Id,1, FrogW,FrogN,FrogE,FrogS):-
        card(Id, FrogN,FrogE,FrogS,FrogW).
……
```

Describing Frog Image Match

```
match((Col,head),(Col,body)).
match((Col,body), (Col,head)).
```

Frog pictures match along an edge if same colour and complement body part

The CLP(FD) frog/1 program

```
frog([(Slot1,Slot1R), (Slot2,Slot2R), (Slot3,Slot3R), (Slot4,Slot4R), (Slot5,Slot5R),
      (Slot6,Slot6R), (Slot7,Slot7R), (Slot8,Slot8R), (Slot9,Slot9R)]):-
    domian([Slot1,Slot2,Slot3,Slot4,Slot5,Slot6,Slot7,Slot8, Slot9]),
    all_different([Slot1,Slot2,Slot3,Slot4,Slot5,Slot6,Slot7,Slot8, Slot9]),
    rotated_card(Slo1,Slot1R,_,F1E,F1S,_),
    match(F1E,F2W),
    rotated_card(Slot2,Slot2R,_,F2E,F2S,F2W),
    % above ensures that frog image on right edge
    % of card in slot one matches that on left edge of slot 2
    match(F2E,F3W),
    rotated_card(Slot3,Slot3R,_,_,F3S,F3W),
    % above ensures that frog image on right edge
    % of card in slot 2 matches that on left edge of slot 3
    match(F1S,F4N),
    rotated_card(Slot4,Slot4R,F4N,F4E,F4S,_),
    % above ensures that frog image on bottom edge
    % of card in slot 1 matches that on top edge of slot 4
    ……
```