FSA Lab Exercises Week 2

Jan van Eijck
CWI & ILLC, Amsterdam

September 8, 2014

```
module FSAlab2

where

import Data.List
```

You all know Mastermind. If you have to refresh your mind on the rules of the game, please consult en.wikipedia.org/wiki/Mastermind_(board_game).

Generalized mastermind, played with n colours and p pegs, is NP Complete de Bondt [2004].

Eijck and Unger [2010] gives an implementation of Mastermind where the computer picks the secret code, the user is supposed to guess the code on the basis of the feedback about the number of pegs in the correct position and the number of pegs of the right colour but in the wrong positions.

Here are some datatypes for this:

```
data Colour   = Red | Yellow | Blue | Green | Orange
                deriving (Eq,Show,Bounded,Enum)

data Answer   = Black | White deriving (Eq,Show)

type Pattern  = [Colour]
type Feedback = [Answer]
```

Comparing two patterns and compute a reaction:

```
samepos :: Pattern -> Pattern -> Int
samepos _      []                 = 0
samepos []     _                  = 0
samepos (x:xs) (y:ys) | x == y    = samepos xs ys + 1
                      | otherwise = samepos xs ys

occurscount ::  Pattern -> Pattern -> Int
occurscount xs []       = 0
occurscount xs (y:ys)
          | y `elem` xs = occurscount
                            (delete y xs) ys + 1
          | otherwise   = occurscount xs ys

reaction :: Pattern -> Pattern -> [Answer]
reaction secret guess = take n (repeat Black)
                     ++ take m (repeat White)
    where n = samepos secret guess
          m = occurscount secret guess - n
```

Here the exercise is to give an implementation of Mastermind where the user picks the secret code and the computer makes the guesses, on the basis of the feedback of the user.

A naive algorithm for this is: keep a list of all codes that are not ruled out by the feedback to the guesses that were made so far. Make the first item on this list your next guess.

Exercise 1 Implement Mastermind using this algorithm.

In [Knuth, 1976–1977] a different algorithm is proposed.

Let **code** be the set of currently possible codes. For each possible code x, compute the partition of **code** according to the possible answers (n, m) that x gives for the members y of **code** (candidates for the correct code). Pick an x that minimizes the number of elements in the largest V_i for the next guess.

Exercise 2 Implement Mastermind using this algorithm. (If the above description is not clear enough, consult [Knuth, 1976–1977].)

In Kooi [2005] another strategy is proposed. Instead of picking a next guess by minimizing the size of the largest partition block for all the possible answers that

it could generate on the currently possible code store, pick a next guess that maximizes the number of partition blocks.

Exercise 3 Implement Mastermind using this algorithm. (Again, if the above description is not clear enough, consult [Kooi, 2005].)

Kooi [2005] discusses some other strategies, too. One strategy computes, instead of the worst case of the largest partition block, the expected size of the generated partition elements: the probability of getting the answer corresponding to a partition element times the size of that partition element.

Exercise 4 Implement Mastermind using this algorithm. This is a modification of the Knuth worst case algorithm.

Yet another strategy discussed by Kooi [2005] is based on information theory. At each stage in the guessing, compute an answer that maximizes the entropy. However, the author remarks that this algorithm performs badly when compared with all non-naive alternatives.

> A possible explanation is that when one calculates the entropy, the base of the logarithm is important when one compares partitions that have a

> different number of elements. When one compares partitions of the same size, entropy is a good measure, otherwise it is not so good. Perhaps another new strategy could be based on taking entropy where the base of the logarithm depends on the size of the partition. [Kooi, 2005]

Exercise 5 Analyze this, and see if you can come up with a correct version of the 'maximize entropy' strategy.

The Balance puzzle that was discussed in class last week is another example of information feedback game. We discussed the balance problem for 12 coins and one counterfeit coin that could be either lighter or heavier than the normal coins. This can be generalized by allowing n coins, with either d out of n different, or l out of n lighter, or h out of n heavier. Each choice (n, d), (n, l) or (n, h) gives a different game.

The similarities with Mastermind are:

- You start with a secret. In the case of Mastermind: a colour code sequence. In the case of Balance, say with (n, d), a choice of d out of n coins with different weight.

- You perform an experiment. In the case of Mastermind: by making a code guess. In the case of Balance: by deciding on a balancing act. The experiment results in feedback. In the case of Mastermind: a feedback code. In the case of Balance: one of three possible answers `LT`, `GT`, `EQ`.
- There is a challenge. In the case of Mastermind: find the code in a minimum number of guesses. In the case of Balance: find the odd coins in a minimum number of balancing acts.

In order to turn Balance into an interesting game, make some assumptions. The following seem reasonable: normal coins all have the same weight, heavy coins all have the same weight, light coins all have the same weight. Instead of this one could assume that heavy coins all exceed the weight of a normal coin by a different amount, and similarly for light coins. This makes the game much more difficult.

Exercise 6 Fix the rules of Balance in some reasonable way (see above), invent a strategy for Balance and implement it. Next, analyze your strategy. Can you use the rule 'maximize the entropy of each balancing act'? How?

References

M. de Bondt. NP-completeness of Master Mind and Minesweeper. Technical Report 0418, Department of Mathematics, Radboud University Nijmegen, 2004.

J. v. Eijck and C. Unger. Computational Semantics with Functional Programming. Cambridge University Press, 2010.

D. Knuth. The computer as master mind. Journal of Recreational Mathematics, 9 (1):1–6, 1976–1977.

B. Kooi. Yet another Mastermind strategy. ICGA Journal, 28(1):13–20, 2005.