Recently seen solutions
Corrigés récemment consultés

(displayed only for Chapitre4)

- IQ_test_2

Less relevant rules are **more complex** because they require determining a threshold value

- Code_for_strings_1

The program outputs ab. Values stored in memory are not considered for the output. Only the element at position 1 in memory is retrieved and displayed.

- Code_for_strings_2

The correct program is let, ?0, '-', ?0, let.

- Code_for_strings_3

1. let, ?0, '-', ?1, let, mem, 0
2. let, ?0, '-', ?1, let, mem, 0, let, 'abc', let
3. let, ?0, '-', ?1, let, mem, 0, 'abc', 'abd'
4. let, ?0, '-', ?1, let, mem, 0, 'abc'

The instruction let, ?0, '-', ?1, let requires exactly two arguments. Answer 1 and 4 provide 0 and 1 argument only. Answer 2 also provides no argument as well, since the second let instruction does not correspond to a string and does not constitute a valid argument.

- String_generation_1A

Correct

- String_generation_1C

Correct

- String_generation_1D

Correct

- String_generation_1E

Correct, of course

- String_generation_2

The complexity of let,?0,:,?0,m,let,mem,0,rosa is 59. You'll notice that the complexity of the trivial instruction 'rosa:rosam' is significantly larger, equal to 70.

- Analogy_in_Finnish

let,?0,a,:,?0,n,let,mem,0,puhu,::,mem,0,katso

- Analogy_Solving_algorithm

The limitation of the language makes it invalid for analogies which imply knowledge of the order of letters. It won't work for Hofstadter's analogies.

- Analogies

Analogies including conditions on the nature of letters won't work (for instance, "if last letter is a vowel, ...").

- Minimum_description_length_1

Any classification of data as "blue"/"red" corresponds to a function from D into ${0,1}$. There are $2^{|D|}$ such functions (two possible labels for the first object, two for the second, and so on.)

- Minimum_description_length_2

If M_i is perfect, then the complexity of data is compressed from N (null model) down to $C(M_i)$. Compression amounts to: $N - C(M_i)$.

- Minimum_description_length_3

If M_i isn't perfect, then compression amounts to: $N - C(M_i) - C(I(i))$, where $I(i)$ is the list of indices of data that are wrongly classified by M_i. If these errors are randomly distributed, then $C(I(i)) \approx (N-n_i) \times log_2(N)$.

- Minimum_description_length_4

The best model number i is the solution of: $argmin(C(M_i) + (N-n_i) \times log_2(N))$. It offers the best trade-off between its own complexity and the description of exceptions.
We may also write: $argmin(C(i) + C(M_i | i) + (N-n_i) \times log_2(N))$

- Minimum_description_length_5

Worst models assign classes in a random way. They achieve negative compression! Note that a model that would systematically assigns the wrong class is an excellent model (if we ignore its own complexity), as the set of wrong data indices is $I(i) = [0,N]$, which as complexity 0 (knowing N).
The answer ‘half of the data wrong’ is correct as well (assuming the wrong data are randomly distributed).

- Prototype-based_models_1

The distances of data points to their closest prototype are smaller than absolute distances.

- Prototype-based_models_2

Prototypes suffice to decide on labels,
so we can forget about data points and only keep prototypes
(as compared to rote learning in which the decision function must keep all data points).

OK