Heuristic Crypto Problem Solver

Author: Joshua Harkness

12/03/2017

Prof. Graci

The given crypto problem-solving code isn't difficult to understand. A significant reason for this deals with the codes relation to how a human may attempt to solve crypto problems. Similar to humans, the code attempts to use heuristics to systematically solve the problem it's given.

The code looks at a problem and applies the numbers in the problem to situation one first. Situation one will only return true if the numbers apply to the first heuristic. This means the goal must be equal to zero and zero must exist in the numbers. If it does return true, the next line in the solveProblemHeuristically rule will execute. Only when situation one is true will action one be considered. Action one then uses the heuristic to solve the problem and stores it in the knowledge base. In most cases this won't happen though.

It's more likely that the program will fail a few times before finding a heuristic that will apply. The code will print out each heuristic rule number it attempts until it finds a heuristic that works, if one works. Each of the remaining heuristics follow a similar pattern to the how the first was determined as true or false.

Heuristic two takes the numbers from the problem and determines if the goal is amongst the numbers. It also checks if zero is a member of the problem numbers and ensures the goal is not zero. Action two then grabs the goal from the problem and finds the number in the problem that's equal to it. After it knows which number is equal to the goal, it simply multiplies all the other numbers together, since it knows zero is among them. The solution expression is then added to the knowledge base.

The third heuristic checks if a pair exists with the doubleton predicate and that the goal is zero. Once action three is called, it separates the pair from the rest of the numbers. The heuristic is then able to use this pair to create zero, and multiply it through the rest of the numbers to get 0. The solution is then added to the knowledge base.

These three heuristics are just the start of what could lead to a genitive program. Much more research and practice must be done before programs can truly represent a cognitive model.