

# 1. Turing Machines

Table 2

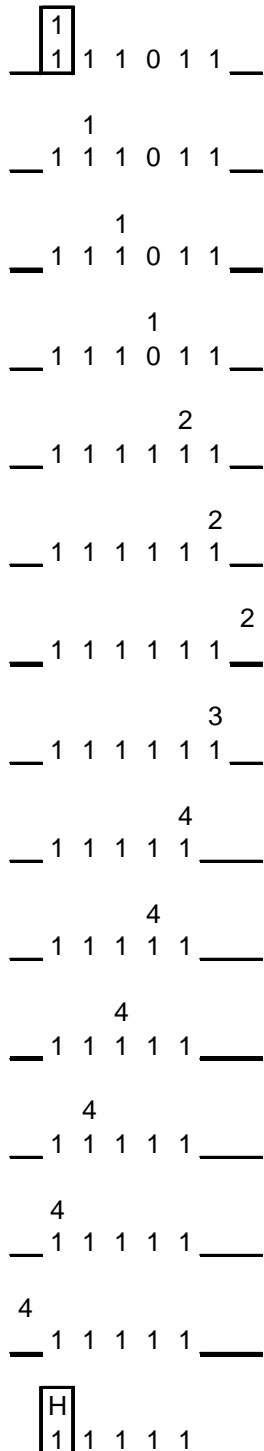
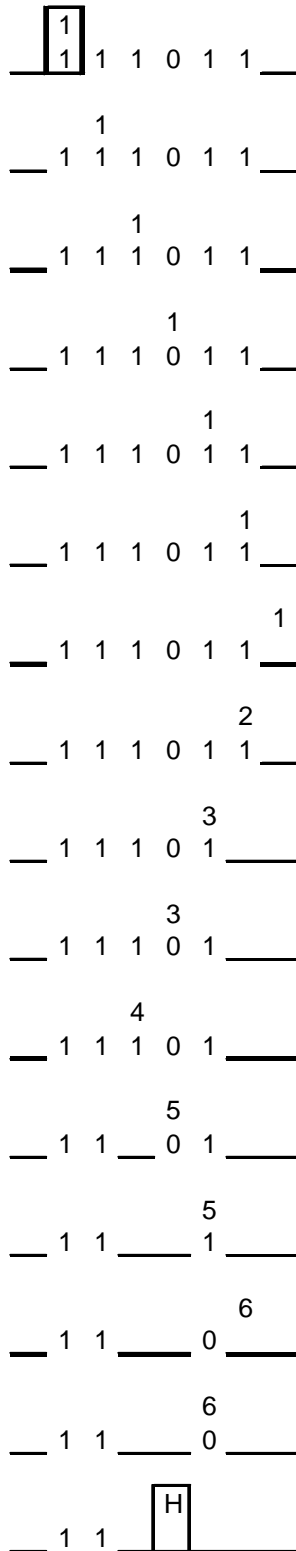


Table 3



The rule set of table 2 is not complete. If there would be another '0' on the tape we would run into a huge problem! – There's no rule defined for a 0 in state 2. → error!

It looks like the only thing what this machine is doing, is, to kick out a 0 and replace it with a 1 if there is a 0 on the tape. It doesn't look quite smart to me!

It does only work with a tape that contains one single 0.

## 1.2. cognition as computation

Cognition as computation has also been called the "cognitivist paradigm" and is the fundamental slogan of classical AI.

The main idea was that we got a deterministic machine that can do everything. Alan Turing formalized the "Turing machine", a theoretical model of a computer. It's an abstract machine that got a deterministic rule-set. Turing machines are meant to be universal. Classical AI believes in the Turing machine as a representation of human intelligence. If humans can solve a problem, then machines can be constructed to solve the same problem. The machine can theoretically carry out any computation.

This is kind of narrow-minded. There must be something else behind human intelligence. It can't be just physical symbol manipulation. Personally I think, there is a lot more behind intelligence, something that we haven't recognized yet. We cannot even imagine how the 4<sup>th</sup> dimension looks like but for sure, there is something above the 3<sup>rd</sup> dimension. I'm pretty sure we're missing something important if we are just trying to generalize intelligence down to a symbol manipulating machine.

What it actually is? – Sorry, I can't tell you!

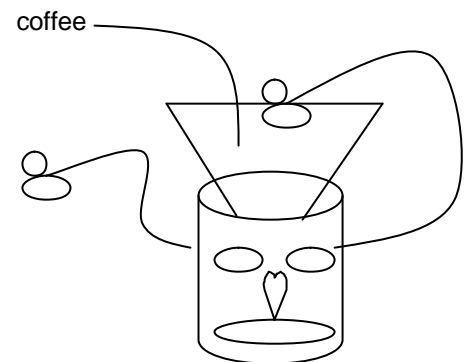
## 2. Expert Systems

### 2.1. set of production rules for making coffee

```

IF is_empty(coffee-box) THEN Flip_go_to_Migros
IF Flip_in_Migros THEN buy_some_good_coffee
IF weight(coffee-box) > 0 THEN Flip_is_ready
IF is_hard(coffee) THEN coffee-beans
IF coffee-beans THEN grind(coffee-beans)
IF coffee_is_ground THEN pour_into_filter(coffee)
IF is_on_place(cup) THEN cup_ok
IF is_not_on_place(cup) THEN call(Flop_the_cup_master)
IF is_cold(water) THEN heat_up(water)
IF is_hot(water) AND cup_ok THEN pour_into_filter(water)
IF is_full(cup) THEN stop_and_beep
IF is_tired(Flip) THEN go_to_bed(Flip)

```



Flip, the coffee-master

### 2.2 problems with Flip, the coffee-master

For the construction of Flip, the coffee-master, a great expert system that one day will conquer the world, we specified a set of 12 production rules.

For a human being this set of rules should be more than enough. Usually a human being would already get along with a set of 3 rules. You would actually just need to tell him to pour the hot water through the filter containing the coffee. Probably you won't even need to tell him to grind the beans, as this is obvious.

If we try to get a robot to apply these 12 rules, then we would probably end up in a self-destruction of the robot. He wouldn't understand any of those rules! There's just too

much implicit knowledge that those rules require. How would the robot be able to go to Migros to buy some coffee? For this we would need to develop a whole new expert system with a separate set of tons of rules. To check whether the coffee is ground or not, we use `is_hard()`. This condition is not clear at all to a robot. If we'd put some stones into the coffee-box, then Flip would try to grind them as well.

When there is no cup on place, Flip performs the following action: `call(Flop_the_cup_master)`. Here we see that Flip is not able to do everything. It needs help from his friend Flop. Flip is no flop though, but he cannot do everything. Flip is pretty limited.

We're not telling Flip how to heat up the water or how to check if the water is cold. We just use the two functions `heat_up()` and `is_cold()` to figure this out. A robot wouldn't be able to interpret those functions. He would need another set of more than a thousand rules to handle those requests.

### 2.3. Situations in which Flip would fail

As I already mentioned, Flip would fail if we would put some stones into the coffee-box instead of coffee beans. He wouldn't be able to distinguish the material and would try to grind the stones. If there is no water around at all, then Flip would fail again as he wouldn't be able to figure out where to get it. In this case he would just stop running.

If Flip doesn't know how to interpret `grind(coffee-beans)` or any other function, he would fail. For all those functions Flip needs implicit knowledge and that exactly is what a robot lacks.

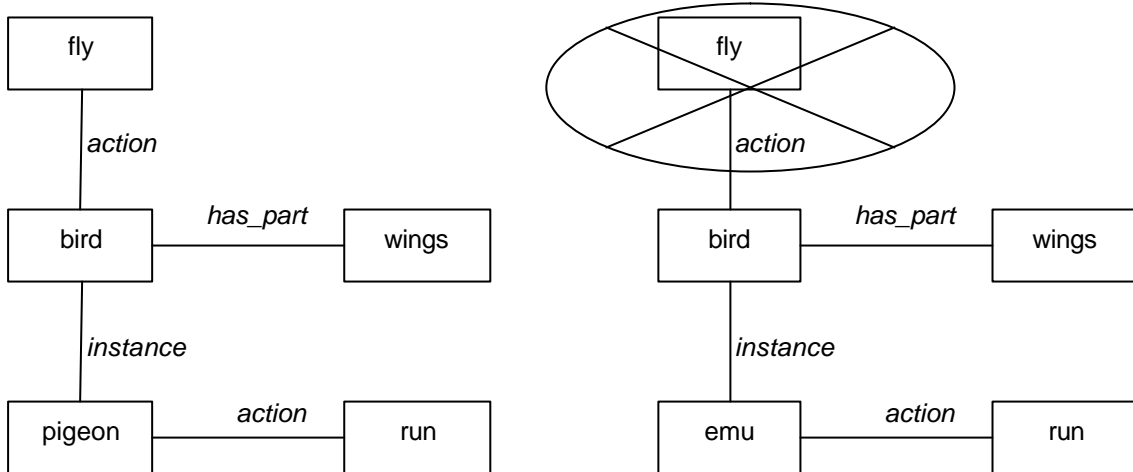
### 2.4. Norwegian coin identification vs. Flip-the-coffee-master

The Norwegian coin identification system just needs to check two conditions, size and color of the coin. Size can be physically measured and is easy to implement. Color-recognition is not such a heavy task either to implement into a robot. The action to be performed is easy for the Norwegian expert system. It just needs to print out the correct value of the coin. This could be solved with a simple LCD-display.

Flip has to handle much more complicated tasks! He has to check a lot of various different conditions that are not such an easy thing to implement. He needs a lot of different sensors and has to be able to process all information chunks and combine them.

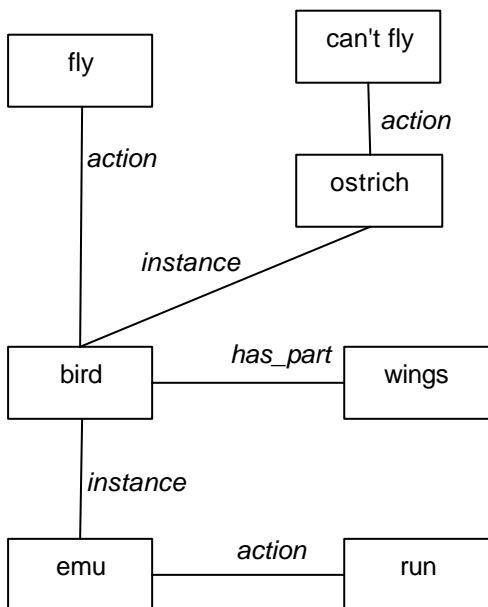
### 3. Semantic Nets

#### 3.1. "Can a pigeon fly?" → "Can an emu fly?"



The conclusion of this semantic net is, that every bird can fly. An emu is a bird but it belongs to the ostriches and an ostrich cannot fly as he's too heavy. An emu even has wings but still we cannot conclude that it can fly.

In a semantic net we need to handle all exceptions. In this case I would suggest:



#### 3.2.

After determine that a pigeon is a bird we need to handle the exception "ostrich". A semantic net that can handle exceptions needs a huge amount of special cases and checks. In this example we just handled the exception "ostrich" but there are a lot more.