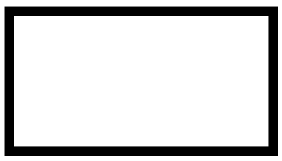
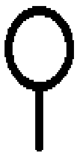
Structural Pattern Recognition
Jeroen Langhorst – s2534657

Animals:

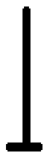
1.
The primitives are defined below:



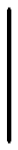
B (body) – The rectangle shape is considered to be the body of the animals.



H (head) – The combination of a stick and a circle is defined as a/the head of an animal.



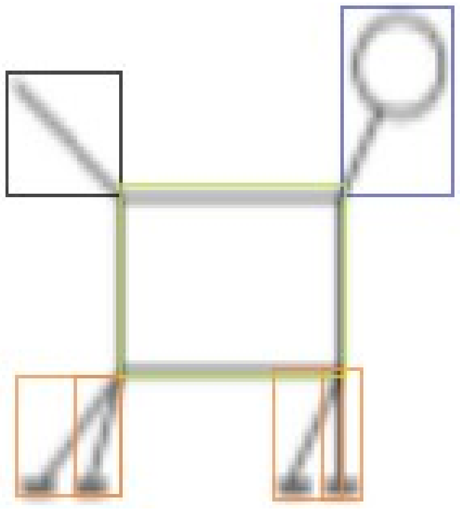
L (limb) – A long stick with a shorter stick at the end is designated as a limb. These can be both feet or hands or even wings.



T (tail) – A single stick without extra features is a tail.

2.
Next follows the procedure with with one can describe the relationship between primitives.

First of, some assumptions are made with respect to primitive recognition. I assume that the primitives discussed earlier are being recognized correctly by the system and some information is about the location of these primitives is known. An example:



This is the first animal of the mammal section. The colored boxes (bounding boxes) represent the different primitives recognized by the system. Of these boxes some information should be obtainable. For example, whether or not the lower left corner pixel of the bounding box of the tail is lower in the image than the lower left corner pixel of the bounding box of the limbs. That the system is able to extract this information is paramount to the further application of the schema.

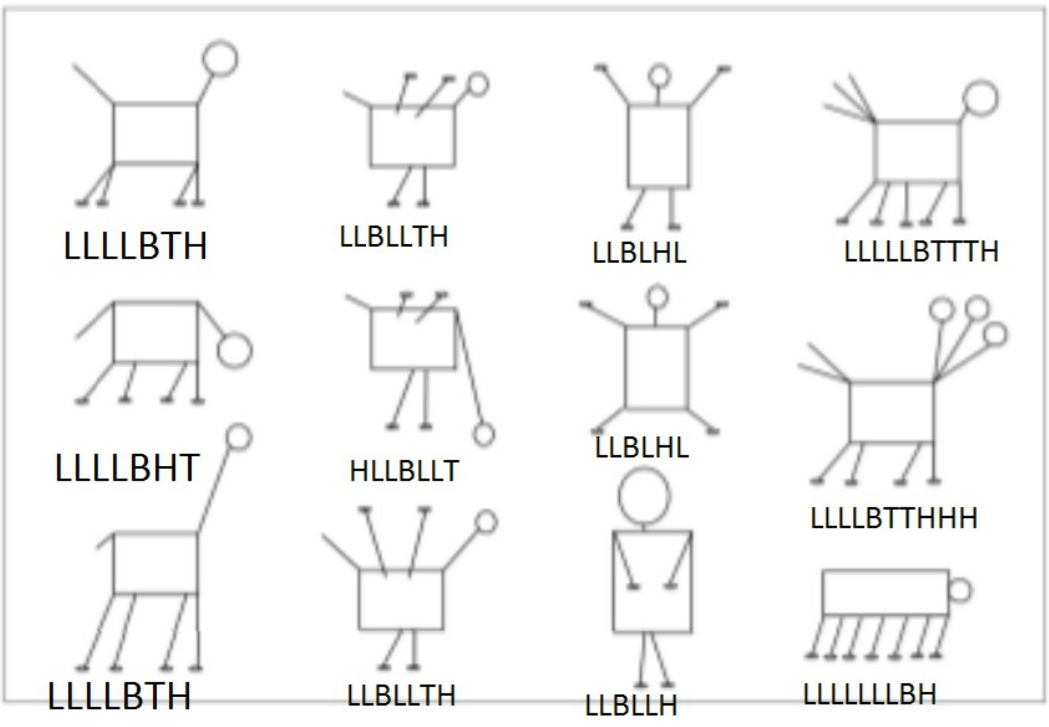
The general procedure first puts together a string based on the character representation of the primitives (i.e. LLBTH is a possible string, but WEHLLH is not). The strings are built up by the following procedure:

1.A. Start at the bottom left of the picture.
2.B. Check if you are at the bottom left pixel of a bounding box.
3.If so, add the character representation of the corresponding primitive to the string.
4.C. Move one pixel to the right. If you cannot go any further to the right, go up one pixel and start all the way back on the left side of the row. Go to C.

By using this method one traverses the entire picture only once, so no pixel is checked twice. This means that each bounding box (and thus each primitive) will only be passed once, therefore no duplicates exist in the string. By using this process the order of the characters creates a relationship based on the location of the primitives.

3.
For the sake of brevity I will fully work out one example and simply give the other string representations. The animal I will be transcribing is the top row mammal. The bounding boxes for this animal are given in the figure on the previous page.
We start at the bottom left corner and check if there is a lower left corner of a bounding box here. There is not one, so we move a pixel to the right and check again. We do this until we reach the end of the line. We now go up one row and back to the left-most pixel and start again moving to the right.

After a few blank lines we come across a line which contains a primitive. The system will now add the character that represents the primitive to the string (which in this case is an "L") and move on to the right. As the next pixel is to the right is not a lower left corner the same primitive will not be added twice. We continue to the right until we get to the second primitive which also translates to an "L". Our string now is "LL". Continuing the line we eventually get "LLLL".
Again, we move up one row and start back at the left side of the picture and repeat the process described above. If we complete the process for every row we eventually come up with the following string: "LLLLBTH".
Below is the full collection of string representations of the animals:



Now that we have these representations, we can generate some rules with which to classify each category. Each colomn represents a different class of animals. The first class is called "mammals" and can be categorized by applying the following rule:

> **If** the string starts with the "LLLL" sub-string **and** is followed by some combination of "B", "T", "H" **and** these last three characters occur exactly once in the string,
> **then** the string represents a mammal.

The second class of animals is "birds". The rule for birds is as follows:

> **If** "LLBLLT" is a sub-string **and** there is only one other character **and** this other character is a "H",
> **then** the string represents a bird.

The last well-defined class is the "human" class:

> **If** the string starts with the "LLB" **and** there is no "T" in the string,
> **then** the string represents a human.

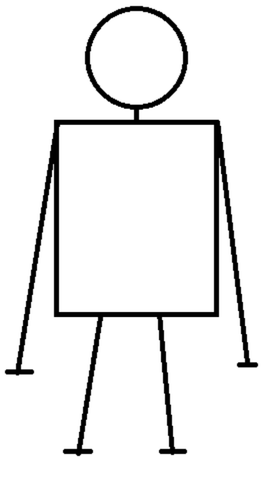
The last class is undefined and is refered to as the "remainder" class. All things that are not classified are put into this class.

4.
Now we have a set of primitives, a procedure to structurally transform pictures into patterns of letters and simple rules can be used on to discern the class of the animal the picture is suppose to represent.

When we now look at the patterns generated in step sub section 3 and apply the rules we can see that all colomns are classified correctly. Let us look at the example again. It is part of the mammal class. This is because we can apply the first rule.

"LLLL" is the start of the string "LLLLBTH" and a combination of the characters "B", "H" and "T" follow that initial sub string. These characters all occur only once in the string. Thus the figure is classified as a mammal.

All animals are classified correctly by the system. We can therefore say that the system works well. However, this is only the case for the animals given. If we would give the system a human character with long arms along its body and the arms would be longer than its body, the system would not classify the animal as human. See the example below:



The system would create the string "LLLLBH". Though it adheres to the "no 'T'- character in the string" part of the rule, the string does not start with "LLB" So the system fails to classify the animal into the correct class.

5.
The error made by the system is due to an ambiguity, as both animals and humans can start with "LLLL". The issue can be resolved as follows:

> **If** the string starts with the "LLB" **or** starts with "LLLLB" **and** there is no "T" in the string,
> **then** the string represents a human.