Math Circle - Robotic Language

Recall that a **formal grammar** is a collection of four things:

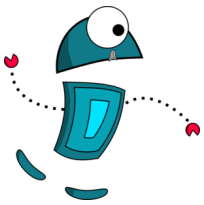
i. A finite alphabet denoted Σ. Elements in Σ are called **terminals**. Usually *terminals* are denoted with lower-case letters. Strings of *terminals* are called **words**.

ii. A finite set of **states** denoted by Ω. Usually *states* are denoted with upper-case letters.

iii. A particular starting state $S \in \Omega$.

iv. A finite set of **rules** for transforming states into words consisting of both *states* and/or *terminals*.

In the future there are only robots. Your group is an extrememly important team of robots whose job it is to program all the new little robots to understand the robot language. You do this by constructing formal grammars which generate the correct language for each robot. If you fail in your duties, the entire next generation of robots will not be able to communicate, and the entire robot civilization will fall. Failure is not an option.



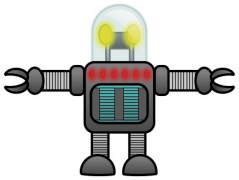
1. Some robots will just be add/multiply machines. These AM machines have knowledge of only the simplest alphabet consisting of 5 terminals: $\Sigma = \{+, *, (,), \mathbf{X}\}$. Here $+$ and $*$ stand for addition and multiplication, the parentheses are used for grouping and order of operations, and $\mathbf{X}$ is a placeholder for any number to be added later.

The language that the AM machines must understand is correct groupings of symbols in Σ to make valid algebraic expressions. For example, $\mathbf{X} + \mathbf{X} * (\mathbf{X} + \mathbf{X})$ is in their language, but $+\mathbf{X}(*\mathbf{X}$ is not. Create a grammar for the AM machines.

2. Another class of robots are the modular robots. Each of these robots understands a language built on the alphabet $\{0, 1, \ldots, 8, 9\}$. Notice that any word using this alphabet is just a nonnegative integer written in base 10.

One subclass of modular robots is the mod-twos. The mod-twos are given any nonnegative integer written in base 10. This number is in their language if it is even. It is not if it is odd. Create a grammar which generates the language of the mod-twos.

3. The mod-threes are another subclass of modular robots. The mod-threes are also given any nonnegative integer written in base 10. The number is in the language of the mod-threes if it is divisible by 3. If it is not, they don't understand that number. Create a grammar which generates the language of the mod-threes.



4. An extremely important class of robots is the comparers. The comparers understand a language built on the alphabet $\{0, 1, \star\}$. Their job is to take two strings consisting of 0s and 1s and see if they have the same length. The two strings are separated by the $\star$ symbol. That is, the language that the comparers need to understand is words that look like $u \star w$, where u and w are both (possibly different) strings of 0s and 1s which have the same length.

Some example words understood by the comparers are $010 \star 110$, $01101 \star 11111$, and $10 \star 10$. Some words which are *not* in the language of the comparers are $11 \star 001$, $0 \star 1001$, and $00 \star 1 \star 01 \star 1$. Create a grammar which generates the language of the comparers.

5. What other types of simple robots can you imagine in the future? These robots should understand a language which is easy to explain (e.g. numbers which are prime, or Fibonacci numbers). Do you think that you can create grammars for all these robots?