BF

The brainfuck language uses a simple machine model consisting of the program and instruction pointer, as well as an array of at least 30,000 byte cells initialized to zero; a movable data pointer (initialized to point to the leftmost byte of the array); and two streams of bytes for input and output (most often connected to a keyboard and a monitor respectively, and using the ASCII character encoding).

MODULE BF-SYNTAX

The syntax of the language consists in eight commands: > < + - . , [ ]

SYNTAX *Ignore* ::= [token, onlyLabel, regex([^\>\<\+\-\.\,\[\]\]+)]

SYNTAX *Instruction* ::= `>`
| `<`
| `+`
| `-`
| `.`
| , [onlyLabel]
| [*Instructions*]
| *Ignore*

A Brainfuck program consists in a list of commands. Brainfuck ignores all characters except the eight commands +-<>[],. so no special syntax for comments is needed. Unfortunately, because of K parsing issues, we assume that programs contain only the language instructions.

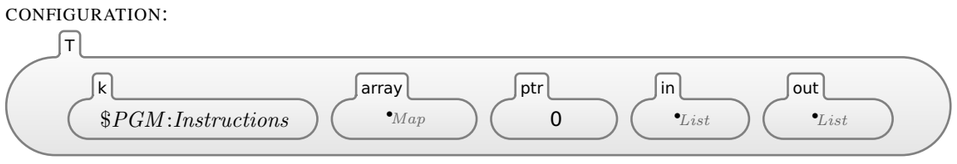
SYNTAX *Instructions* ::= $List\{Instruction, \text{“”}\}$

END MODULE

MODULE BF

The configuration of the language contains the K cell for Brainfuck programs, an array cell containing the byte array, a cell ptr for the instruction pointer and I/O streams.

CONFIGURATION:

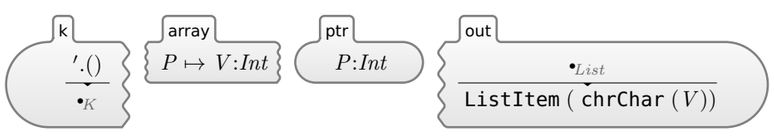


Unroll intructions into KList.

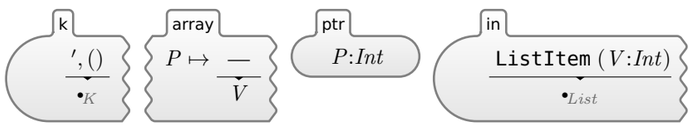
RULE $\dfrac{I{:}Instruction\ Is{:}Instructions}{I \curvearrowright Is}$

RULE $\dfrac{\bullet Instructions}{\bullet K}$

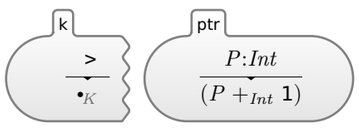
Output the byte at the data pointer.

RULE 

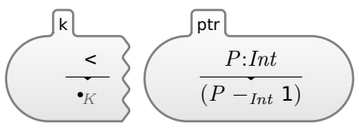
Input the byte at the data pointer.

RULE 

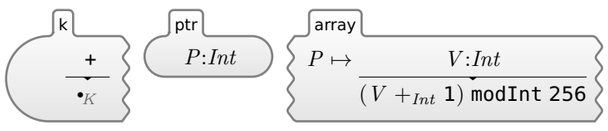
Increment the data pointer.

RULE 

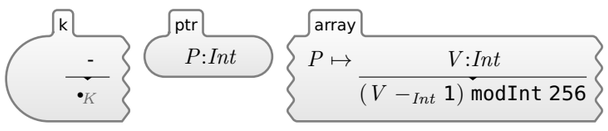
Decrement the data pointer.

RULE 

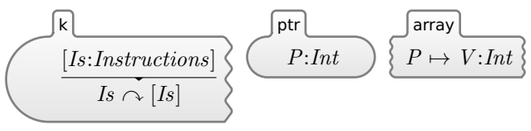
Increment the byte at the data pointer

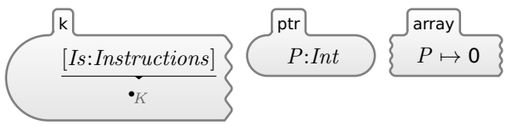
RULE 

Decrement the byte at the data pointer

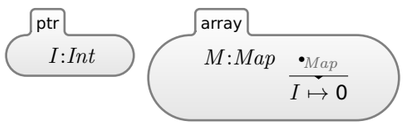
RULE 

Brainfuck jumps (’[’ and ’]’) are considered to be loops. Whenever the byte at the data pointer is not zero, execute the loops instructions.

RULE  requires $V =/=_{Int} 0$

RULE 

RULE $\dfrac{I{:}Ignore}{\bullet K}$

RULE  requires $\neg_{Bool}(I \texttt{ in keys } (M)) \wedge_{Bool} (I \geq_{Int} 0)$ [structural]

END MODULE