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

```
SYNTAX   Instruction ::= >
                       |  <
                       |  +
                       |  -
                       |  .
                       | , [onlyLabel( 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* ::= $List\{Instruction, “”\}$

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:

$$\left\langle \langle \$PGM{:}Instructions \rangle_{\mathsf{k}} \; \langle \texttt{.Map} \rangle_{\mathsf{array}} \; \langle \texttt{0} \rangle_{\mathsf{ptr}} \; \langle \texttt{.List} \rangle_{\mathsf{in}} \; \langle \texttt{.List} \rangle_{\mathsf{out}} \right\rangle_{\mathsf{T}}$$

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 $\left\langle \dfrac{'.()}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle \cdots P \mapsto V{:}Int \cdots \right\rangle_{\mathsf{array}} \; \left\langle P{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle \cdots \dfrac{\texttt{.List}}{\texttt{ListItem}\,(\,\texttt{chrChar}\,(V))} \right\rangle_{\mathsf{out}}$

Input the byte at the data pointer.

RULE $\left\langle \dfrac{',()}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle \cdots P \mapsto \dfrac{\_\!\_}{V} \cdots \right\rangle_{\mathsf{array}} \; \left\langle P{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle \dfrac{\texttt{ListItem}\,(V{:}Int)}{\texttt{.List}} \cdots \right\rangle_{\mathsf{in}}$

Increment the data pointer.

RULE $\left\langle \dfrac{\texttt{>}}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle \dfrac{P{:}Int}{(P +_{Int} \texttt{1})} \right\rangle_{\mathsf{ptr}}$

Decrement the data pointer.

RULE $\left\langle \dfrac{\texttt{<}}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle \dfrac{P{:}Int}{(P -_{Int} \texttt{1})} \right\rangle_{\mathsf{ptr}}$

Increment the byte at the data pointer

RULE $\left\langle \dfrac{\texttt{+}}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle P{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle \cdots P \mapsto \dfrac{V{:}Int}{(V +_{Int} \texttt{1})\ \texttt{modInt}\ \texttt{256}} \cdots \right\rangle_{\mathsf{array}}$

Decrement the byte at the data pointer

RULE $\left\langle \dfrac{\texttt{-}}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle P{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle \cdots P \mapsto \dfrac{V{:}Int}{(V -_{Int} \texttt{1})\ \texttt{modInt}\ \texttt{256}} \cdots \right\rangle_{\mathsf{array}}$

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

RULE $\left\langle \dfrac{[Is{:}Instructions]}{Is \curvearrowright [Is]} \cdots \right\rangle_{\mathsf{k}} \; \left\langle P{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle \cdots P \mapsto V{:}Int \cdots \right\rangle_{\mathsf{array}}$ requires $V =/=_{Int} \texttt{0}$

RULE $\left\langle \dfrac{[Is{:}Instructions]}{\bullet_K} \cdots \right\rangle_{\mathsf{k}} \; \left\langle P{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle \cdots P \mapsto \texttt{0} \cdots \right\rangle_{\mathsf{array}}$

RULE $\dfrac{I{:}Ignore}{\bullet_K}$

RULE $\left\langle I{:}Int \right\rangle_{\mathsf{ptr}} \; \left\langle M{:}Map\ \dfrac{\texttt{.Map}}{I \mapsto \texttt{0}} \right\rangle_{\mathsf{array}}$ requires $\neg_{Bool}(I\ \texttt{in}\ \texttt{keys}\,(M)) \wedge_{Bool} (I \geq_{Int} \texttt{0})$ [structural(structural())]

END MODULE