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([ˆ\>\<\+\-\.\,\[\]\]+)]

```
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:

$$\langle \langle \$PGM{:}Instructions \rangle_{k}\ \langle \cdot_{Map} \rangle_{array}\ \langle 0 \rangle_{ptr}\ \langle \cdot_{List} \rangle_{in}\ \langle \cdot_{List} \rangle_{out} \rangle_{T}$$

Unroll intructions into KList.

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

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

Output the byte at the data pointer.

RULE $\langle \dfrac{'.()}{\cdot_{K}} \cdots \rangle_{k}\ \langle \cdots P \mapsto V{:}Int \cdots \rangle_{array}\ \langle P{:}Int \rangle_{ptr}\ \langle \cdots \dfrac{\cdot_{List}}{\texttt{ListItem}\,(\,\texttt{chrChar}\,(V))} \rangle_{out}$

Input the byte at the data pointer.

RULE $\langle \dfrac{',()}{\cdot_{K}} \cdots \rangle_{k}\ \langle \cdots P \mapsto \dfrac{\_}{V} \cdots \rangle_{array}\ \langle P{:}Int \rangle_{ptr}\ \langle \dfrac{\texttt{ListItem}\,(V{:}Int)}{\cdot_{List}} \cdots \rangle_{in}$

Increment the data pointer.

RULE $\langle \dfrac{>}{\cdot_{K}} \cdots \rangle_{k}\ \langle \dfrac{P{:}Int}{(P +_{Int} 1)} \rangle_{ptr}$

Decrement the data pointer.

RULE $\langle \dfrac{<}{\cdot_{K}} \cdots \rangle_{k}\ \langle \dfrac{P{:}Int}{(P -_{Int} 1)} \rangle_{ptr}$

Increment the byte at the data pointer

RULE $\langle \dfrac{+}{\cdot_{K}} \cdots \rangle_{k}\ \langle P{:}Int \rangle_{ptr}\ \langle \cdots P \mapsto \dfrac{V{:}Int}{(V +_{Int} 1)\ \texttt{modInt}\ 256} \cdots \rangle_{array}$

Decrement the byte at the data pointer

RULE $\langle \dfrac{-}{\cdot_{K}} \cdots \rangle_{k}\ \langle P{:}Int \rangle_{ptr}\ \langle \cdots P \mapsto \dfrac{V{:}Int}{(V -_{Int} 1)\ \texttt{modInt}\ 256} \cdots \rangle_{array}$

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

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

RULE $\langle \dfrac{[Is{:}Instructions]}{\cdot_{K}} \cdots \rangle_{k}\ \langle P{:}Int \rangle_{ptr}\ \langle \cdots P \mapsto 0 \cdots \rangle_{array}$

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

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

END MODULE