Macrolop Specification

Temirkhan Myrzamadi (a.k.a. Hirrolot)

November 16, 2020

Contents

1 EBNF Grammar

```
<eval> ::= "MACROLOP_EVAL(" { <term> }* ")" ;

<term> ::= "call(" <op> "," { <term> }* ")"
         | "v(" <preprocessor-token-list> ")" ;

<op>   ::= <ident> | { <term> }+ ;
```

Figure 1: Grammar rules

A metaprogram in Macrolop consists of a (possibly empty) sequence of terms, each of which is either a macro call or just a value.

Notes:

- The grammar above describes metaprograms already expanded by the C preprocessor, except for `MACROLOP_EVAL`, `call`, and `v`.
- `call` accepts `op` either as an identifier or as a non-empty sequence of terms that reduces to an identifier.
- `call` accepts arguments without a separator.

2 Operational Semantics

We define small-step operational semantics for Macrolop.

$$
\begin{array}{rl}
(v) : \langle k; acc; v(\overline{tok})\ term\ \overline{term'}\rangle & \rightarrow_1 \langle k; acc,\ \overline{tok}; term\ \overline{term'}\rangle \\
(v\text{-}end) : \langle k; acc; v(\overline{tok})\rangle & \rightarrow_1 k(seq\text{-}extract(acc, \overline{tok})) \\
(op) : \langle k; acc; call(\overline{term}, \overline{a})\ \overline{term'}\rangle & \rightarrow_1 \langle\langle k; acc; call(?, \overline{a})\ \overline{term'}\rangle; (); \overline{term}\rangle \\
(args) : \langle k; acc; call(ident, \overline{a})\ \overline{term}\rangle & \rightarrow_1 \langle\langle k; acc; ident(?)\ \overline{term}\rangle; (); \overline{a}\rangle \\
(start) : MACROLOP\_EVAL(\overline{term}) & \rightarrow_1 \langle halt; (); \overline{term}\rangle
\end{array}
$$

Figure 2: Computational rules

Notes:

- A body of a macro called using `call` must follow the grammar of Macrolop, otherwise it might result in a compilation error.