Macrolop Specification

Temirkhan Myrzamadi (a.k.a. Hirrolot)

November 5, 2020

Contents

1 EBNF Grammar

$\langle evaluate \rangle$ ::= `MACROLOP_EVAL(` { $\langle term \rangle$ }* `)`

$\langle term \rangle$ ::= `call(` $\langle macro \rangle$ `,` { $\langle term \rangle$ }* `)`
 | `v(` $\langle tokens \rangle$ `)`

$\langle macro \rangle$ ::= $\langle ident \rangle$

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.

Note that a macro call accepts arguments without a separator. This is intentional: the absence of separators lets you don't care about the position of an argument (if they're generated programmatically), whereas the presence of separators necessitates logic to detect whether a generated argument is the last one or not, and if so, avoid putting a comma after it.

Note that the given syntax holds for metaprograms already expanded by the C preprocessor, except for the macros `MACROLOP_EVAL`, `call`, and `v`. So a syntactically well-formed metaprogram in Macrolop is a C metaprogram that expands to a sequence of preprocessor tokens (again except for the aforementioned cases) matching the given grammar.

2 Operational Semantics

$$\frac{}{v(tokens) \to tokens}\ \text{(final)}$$

$$\frac{arg_1 \to arg'_1 \qquad \ldots \qquad arg_n \to arg'_n}{call(macro, arg_1 \ldots arg_n) \to macro(arg'_1, \ldots, arg'_n)}\ \text{(macro-call)}$$

$$\frac{term_1 \to tokens_1 \qquad \ldots \qquad term_n \to tokens_n}{MACROLOP\_EVAL(term_1 \ldots term_n) \to tokens_1 \ldots tokens_n}\ \text{(eval)}$$

Figure 2: Computational rules

These computational rules say that:

- (final) `v` merely reduces to its arguments.
- (macro-call) `call` implements applicative evaluation strategy, meaning that firstly its arguments are evaluated, then an ordinary call to `macro` is produced, which is then expanded by the C preprocessor resulting in yet another Macrolop metaprogram to be evaluated further.
- (eval) `MACROLOP_EVAL` evaluates a possibly empty sequence of terms according to the given computational rules. A Macrolop metaprogram will have no effect unless it or another Macrolop metaprogram that called it was passed into `MACROLOP_EVAL`.

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