A Simple Yet Effective Method Slicer

Breandan Considine

August 22, 2021

We describe a simple heuristic for extracting method slices in well-formed source code using a Dyck counter. [1] It is common convention in many languages to prefix functions with a keyword, followed by a group of balanced brackets and one or more blank lines. Thus, we accumulate lines until brackets are balanced and a blank line is encountered, then reset. We observe this approach works on a variety of langauges. An implementation in Kotlin is given below, which will output the following source code when run on itself:

```
fun String.sliceIntoMethods(kwds: Set<String> = setOf("fun ")) =
  lines().fold(-1 to List<String>(0)) { (dyckCtr, methods), ln ->
    if (dyckCtr < 0 && kwds.any { it in ln }) {
      ln.countBalancedBrackets() to (methods + ln)
    } else if (dyckCtr == 0) {
      if (ln.isBlank()) -1 to methods else 0 to methods.put(ln)
    } else if (dyckCtr > 0) {
      dyckCtr + ln.countBalancedBrackets() to methods.put(ln)
    } else -1 to methods
  }.second

fun List<String>.put(s: String) = dropLast(1) + (last() + "\n$s")

fun String.countBalancedBrackets() = fold(0) { s, c ->
  val (lbs, rbs) = setOf('(', '{', '[') to setOf(')', '}', ']')
  if (c in lbs) s + 1 else if (c in rbs) s - 1 else s
}

fun main(args: Array<String>) =
  println(args[0].sliceIntoMethods().joinToString("\n\n"))
```

[1] https://en.wikipedia.org/wiki/Dyck_language