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] A common coding convention is to prefix functions with a keyword, followed by a group of balanced brackets and one or more blank lines. While imperfect, we observe this pattern can be used to slice methods in a variety of langauges in practice. A Kotlin implementation 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) { sum, char ->
  val (lbs, rbs) = setOf('(', '{', '[') to setOf(')', '}', ']')
  if (char in lbs) sum + 1 else if (char in rbs) sum - 1 else sum
}

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

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