A Simple Yet Effective Method Slicer

Breandan Considine

August 22, 2021

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

```
val funKeywords = listOf("def ", "function ", "fun ", "void ")
val (lbs, rbs) = listOf('(', '{', '[') to listOf(')', '}', ']')

fun String.sliceIntoMethods(): List<String> =
  lines().fold(-1 to listOf<String>()) { (dyckSum, methods), line ->
    if (dyckSum < 0 && funKeywords.any { it in line }) {
      line.countBalancedBrackets() to (methods + line)
    } else if (dyckSum == 0) {
      if (line.isBlank()) -1 to methods else 0 to methods.put(line)
    } else if (dyckSum > 0) {
      dyckSum + line.countBalancedBrackets() to methods.put(line)
    } else -1 to methods
  }.second

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

fun String.countBalancedBrackets(): Int = fold(0) { s, c ->
  if (c in lbs) s + 1 else if (c in rbs) s - 1 else s
}
```

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