Revision: data flow based coverage

(code taken from 2008 exam, question 2)

Conrad Hughes

March 16, 2009

The Java method `match()` implements a string search algorithm: it returns the (0-based) index of the first occurrence of `String` *needle* in `String` *haystack*, or -1 if *needle* is not a substring of *haystack*.

```
static int match(String haystack, String needle) {
  for(int i = 0; i + needle.length() <= haystack.length(); ++i) {
    int j = 0;
    while(j < needle.length() &&
        haystack.charAt(i + j) == needle.charAt(j))
      ++j;
    if(j == needle.length())
      return i;
  }
  return -1;
}
```

For example, `match("mississippi", "sis")` would return 3.

1. Create a control flow graph for this code.
2. Annotate the graph with defs, c-uses and p-uses.
3. Create a table identifying defs, c-uses and p-uses for each variable.
4. Identify sets of def-use pairs whose coverage by a test suite would achieve each of the following coverage criteria:
   - all-defs
   - all-c-uses
   - all-p-uses
   - all-uses
5. Are any of the above sets the same?
6. Does a suite which satisfies all-defs need to satisfy statement coverage?
7. Create a test suite for each of the above levels of coverage.
8. What extra work would you have to do to achieve all-du-paths?