Seaching through a Trace with Patterns

August 3, 2016

The principal method by which a bug will be located in our system is by the (semi-)automatic edition of the program trace. This can happen as the program is evaluated, or by interactive searching afterward. But we need a way to express the searches.

Text-based approaches (e.g regular expressions) are not much use. Fine for searching for a function name, but no good for tree-based data like functional programs. We want to be able to say, for example:

1. "Find any function application in the trace taking an empty tree as input."
2. "Show me all calls to function `f`"
3. "Show me any time a list begins with a negative number"

Patterns for 1, 2, 3 might be "`_ Lf`", "f _", and "[-_; ...]".

1 Patterns

For example...

`_`	wildcard
`()`	parentheses
`[p; p]`	lists (similarly for records, tuples etc)
`...`	to indicate the tail of a list
`text`	literal text
`remove*`	wildcards as part of text

- We use the same lexical conventions as OCaml, so we can reuse the OCaml lexer. The parser should not be too difficult, just must have a subset of the associativities / precedences of the OCaml one.
- The parser should accept *any* string, just counts as text to match.

2 TODO

1. Define a small example pattern language
2. Choose a subset of `Tinyocaml.t`
3. Find Example program, trace, and patterns
4. Write the pattern parser
5. Write the matcher, which sees if a pattern matches a line of the trace, by matching the tree of the pattern against the tree of that line, rather than text against text. Need a way to indicate the matched part, say by underlining.