```
1  local getHeader = P.define('getHeader', {'which'},
2      P.Sub(P.Ref('which'), P.Var('REQUEST_HEADERS'))
3  )
4
5  local isGet = P.Rx('GET', P.Var('REQUEST_METHOD'));
6  local isPost = P.Rx('POST', P.Var('REQUEST_METHOD'));
7
8  Action("IsGet", "1"):
9    action([[clipp_announce:IsGet]]):
10   predicate(isGet)
11
12 Action("IsPost", "1"):
13   action([[clipp_announce:IsPost]]):
14   predicate(isPost)
15
16 Action("NoHost", "1"):
17     action([[clipp_announce:IsPost]]):
18     predicate((isGet / isPost) + P.Not(P.getHeader('Host')))
19
20 Action("ContentLengthZero", "1"):
21     action([[clipp_announce:IsPost]]):
22     predicate((isGet / isPost) + P.Eq(0, P.getHeader('Content-Length')))
23
24 Action("ThisHasAnError", "1"):
25     predicate(P.Sub(P.Var('a'), 'foo'))
26
```

1: Here is an example of a few Waggle Predicate rules.

2: Note that this expression has an error.

3: Saving this file as `waggle.lua`, we can then generate a report on it by running

```
pp.rb waggle.lua
```

`pp.rb` is located in the predicate subdirectory.

4: Here is the first half of the report (an HTML file). It shows the waggle file with annotations.

5: These show where pp found Predicate expressions. Green means validation passed; orange means warnings; and red means errors.

```
Predicate 1  local getHeader = P.define('getHeader', {'which'},
          2      P.Sub(P.Ref('which'), P.Var('REQUEST_HEADERS'))
          3  )
          4
          5  local isGet = P.Rx('GET', P.Var('REQUEST_METHOD'));
          6  local isPost = P.Rx('POST', P.Var('REQUEST_METHOD'));
          7
Predicate 8  Action("IsGet", "1"):
          9    action([[clipp_announce:IsGet]]):
         10    predicate(isGet)
         11
Predicate 12 Action("IsPost", "1"):
         13    action([[clipp_announce:IsPost]]):
         14    predicate(isPost)
         15
Predicate 16 Action("NoHost", "1"):
         17      action([[clipp_announce:IsPost]]):
         18      predicate((isGet / isPost) + P.Not(P.getHeader('Host')))
         19
Predicate 20 Action("ContentLengthZero", "1"):
         21      action([[clipp_announce:IsPost]]):
         22      predicate((isGet / isPost) + P.Eq(0, P.getHeader('Content-Length')))
         23
Predicate 24 Action("ThisHasAnError", "1"):
```

Note: Define expressions (template bodies) are not fully validated as they cannot be until expanded

6: Clicking on any Predicate will expand/unexpand a details section. Expressions that are not green start expanded.

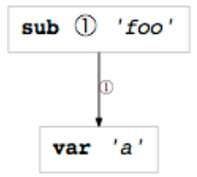
7: Here we see our erroneous expression already expanded.

8: First is the S-Expression.

```
(sub
  (var 'a')
  'foo'
)
```

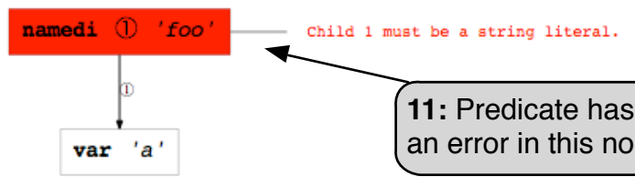
9: Next is the graph, pre-transformations

Pre-Transform Graph



10: Finally is the graph, post-transformations.

Post-Transform Graph

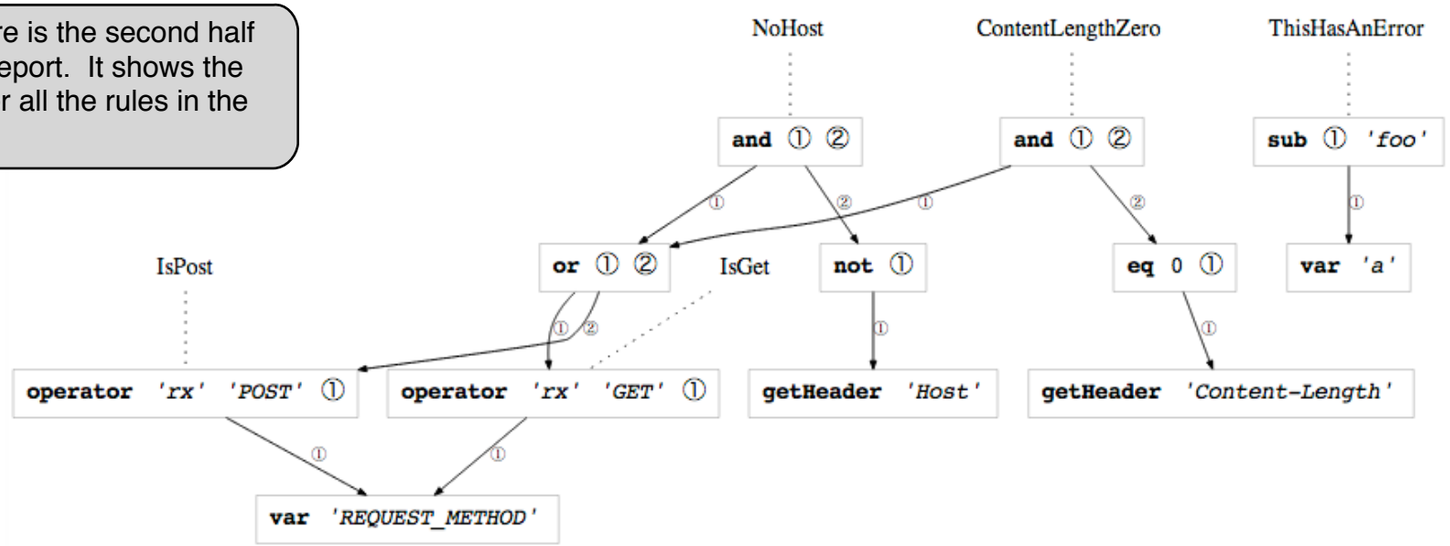


11: Predicate has detected an error in this node.

```
25    predicate(P.Sub(P.Var('a'), 'foo'))
```

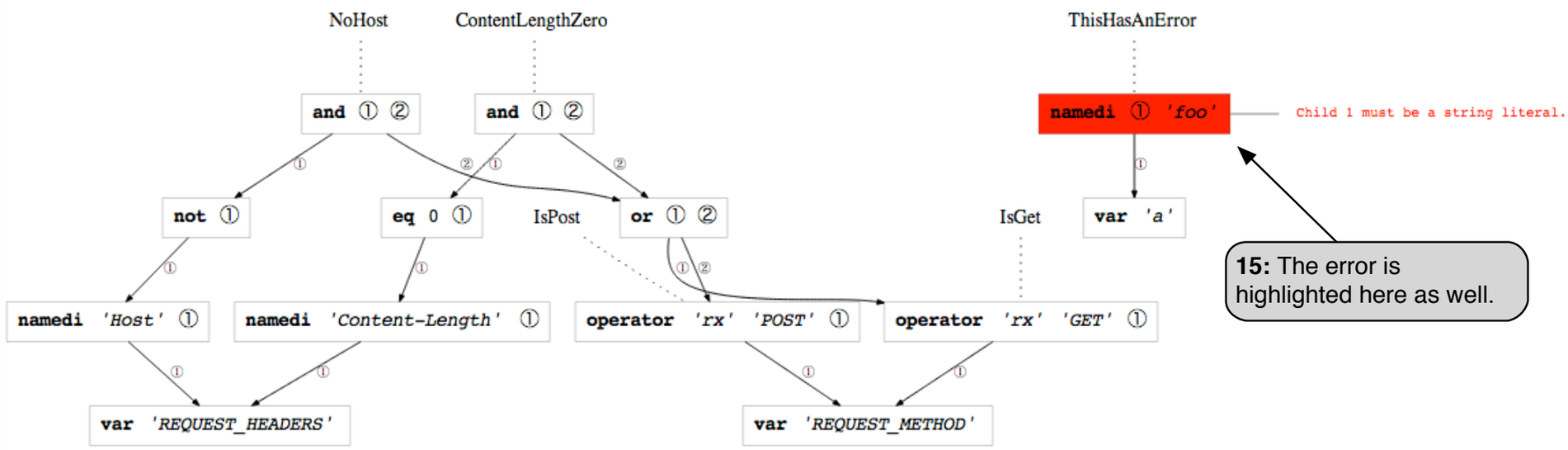
Pre-Transform Graph

12: Here is the second half of the report. It shows the DAG for all the rules in the file.



13: First is the pre-transform graph. Note that root nodes are labelled with the corresponding rule.

Post-Transform Graph



14: Next is the post-transform graph.

15: The error is highlighted here as well.