

1: Here is a clipp test with a couple predicate rules.

```
1 def test_trace
2   clipp(CONFIG.merge(
3     default_site_config: <<-EOS
4     Action id:1 phase:REQUEST_HEADER clipp_announce:a "predicate:(eq 'foo' (sub 'x' (var 'ARGS')))"
5     Action id:2 phase:REQUEST_HEADER clipp_announce:b "predicate:(eq 'bar' (sub 'x' (var 'ARGS')))"
6     Action id:3 clipp_announce:c "predicate:(or (eq 'bar' (sub 'x' (var 'ARGS'))) (eq 'bar' (sub 'y' (var 'ARGS'))))"
7     PredicateTrace "/tmp/ptrace"
8   EOS
9 )) do
10  transaction do |t|
11    t.request(
12      raw: 'POST /a/b/c?x=bar&y=foo HTTP/1.1',
13      headers: {
14        "Content-Type" => "application/x-www-form-urlencoded",
15        "Content-Length" => 5
16      },
17      body: "y=bar"
18    )
19  end
20 end
21 assert_no_issues
22 assert_log_match /PredicateTrace/
23 end
```

2: This directive tells IronBee to write a predicate trace to /tmp/ptrace.

3: You can also pass in "-" to have it output the trace to stderr. This can be useful with Clipp. Finally, you can pass in rule ids as additional arguments to limit the trace to only those rules.

4: There are many caveats:

- Will **append** to file.
- Will behave poorly in multithreaded IronBee.
- Will kill performance.
- Uses lots of memory.
- Does not scale well to large Predicate graphs.

5: But, it can be useful for debugging small sets of Predicate rules; for understanding Predicate; and for understanding IronBee.

6: The PredicateTrace directive is scoped to the current context, so you can trace some contexts but not others.

```
PredicateTrace REQUEST_HEADER context=default:location:/ consider=3 inject=2
```

8: This header indicates a PredicateTrace block. There is a PredicateTrace block for any phase and context that Predicate did something in. It also shows the number of rules (root nodes) considered and the number injected (fires). You can click on it to toggle display of the block.

9: Each block is shown with the portion of the DAG relevant to that block, i.e., all root nodes for that phase and context along with their descendants.

7: To process, run the `predicate/render_pttrace.rb` script, passing in `/tmp/pttrace` as the first argument.

The output is HTML; this is the first part.

site/57f2b6d0-7783-012f-86c6-001f5b320164/3

site/57f2b6d0-7783-012f-86c6-001f5b320164/1

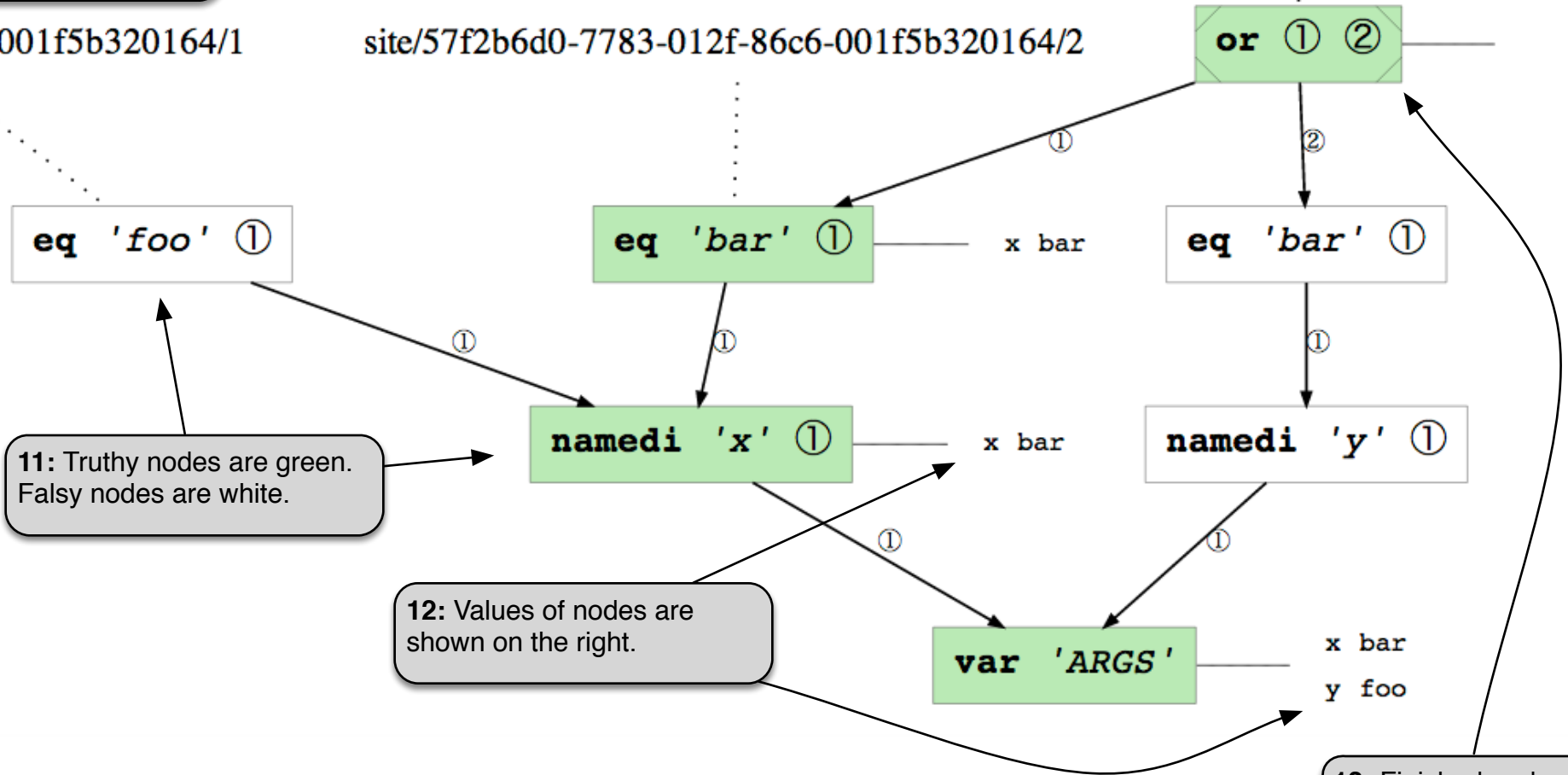
10: This is a root label indicating that this node is the root node for rule 1.

site/57f2b6d0-7783-012f-86c6-001f5b320164/2

11: Truthy nodes are green.
Falsy nodes are white.

12: Values of nodes are shown on the right.

13: Finished nodes have triangles.



15: This root is “out of phase”; it is a root node but for a rule that does not run in this phase.

14: Here is the second part of the trace, the block for REQUEST_BODY.

