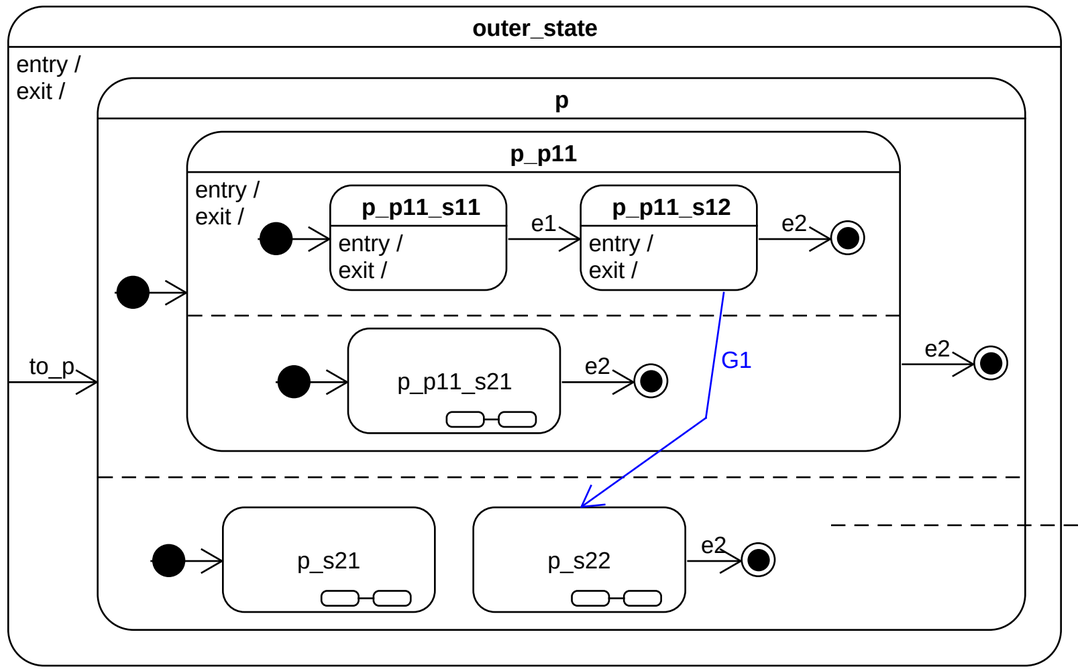

It is p_p11 has cause transitions in p_p11_r1 and p_p11_r2 to their under_hidden_regions, but this happens with an exit

It is p that has cause to cause transitions in p_r1 to under_hidden_region, but this happens with an exit

It is p that has cause to cause a transition in p_r2 to p_s21 via its INIT_SIGNAL

Idea:
* try making an onion from p_s21 to p_p11_s12
* reverse it
* for each item in the onion assign a META_EXIT or META_INIT
* An injector receiving a META_EXIT will only inject an exit_region if the function is in the regions it is injecting to
* If receiving a META_INIT, use peak_meta and see if the function is in the region, if so post_lifo force_region_init

Introducting the second WTF event

[n]::EXIT_META_SIGNAL:p_p11 [n-1]::EXIT_META_SIGNAL:p_r2_region ->
[n]::INIT_META_SIGNAL:p_r2_region [n-1]::EXIT_META_SIGNAL:p_p11 ->
[n]::INIT_META_SIGNAL:p_s22 [n-1]::INIT_META_SIGNAL:p_r2_region ->

Harel Diagram

Mapping onto an orthogonal components

```
payload = namedtuple(
  'META_SIGNAL_PAYLOAD',
  [
    'event',
    'state',
    'previous_state',
    'previous_event'
  ]
)
```

«injectors»

XmlChart

outer_state

p

4

```
INIT_META_SIGNAL as e /
  if self.live_spy and self.instrumented:
    self.live_spy_callback("[p] {}".format(...))
  _e, _state = e.payload.event, e.payload.state
  if _state:
    self.regions['p']._post_fifo(_e)
  self.regions['p'].meta_bounce_across(e)
```

8

5

to_p

p_r1_under_hidden_region

p_r1_region

p_r1_over_hidden_region

p_p11

```
EXIT_META_SIGNAL as e /
  (_e, _state) = r.meta_peel(e)
  if r.has_a_child(p_p11, _state):
    r.outer.post_fifo(_e)
```

3

force_region_init

6

enter_region

exit_region

p_p11_r1_under_hidden_region

p_p11_r1_region

```
EXIT_META_SIGNAL as e /
  (_e, _state) = r.meta_peel(e)
  if r.has_a_child(p_p11_r1_region, _state):
    r.outer.post_fifo(_e)
```

2

p_p11_r1_over_hidden_region

p_p11_s11

entry /
exity /

e1

e2

p_p11_s12

```
G1 as e /
  t, _e = rr.outmost.meta_trans(
    t=p_s21,
    s = p_p11_s12, sig=e)
  post_fifo(_e)
```

1

force_region_init

enter_region

exit_region

[n]::INIT_META_SIGNAL:p_s22 [n-1]::INIT_META_SIGNAL:p_r2_region ->

[n]::INIT_META_SIGNAL:p_r2_region [n-1]::EXIT_META_SIGNAL:p_p11 ->
[n]::INIT_META_SIGNAL:p_s22 [n-1]::INIT_META_SIGNAL:p_r2_region ->

p_r2_under_hidden_region

p_r2_region

p_r2_over_hidden_region

p_s21

p_s22

entry /
exit /

8

7

force_region_init

enter_region

exit_region

```
def meta_bounce_across(self, e):
  trans_init = False
  if e.payload.event.signal != e.payload.previous_signal:
    trans_init = True
  if trans_init:
    for region in self._regions:
      if region.has_state(e.payload.previous_state):
        region.pop_event():
        region.post_lifo(Event(signal=signals.region_exit))
      else:
        region.post_lifo(Event(signal=signals.force_region_init))
    self._complete_circuit()
  else:
    self.post_lifo(Event=signals.force_region_init)
```

6

7

[n]::EXIT_META_SIGNAL:p_p11_r1_region [n-1]::G1:p_p11_s12 ->
[n]::EXIT_META_SIGNAL:p_p11 [n-1]::EXIT_META_SIGNAL:p_r2_region ->
[n]::INIT_META_SIGNAL:p_r2_region [n-1]::EXIT_META_SIGNAL:p_p11 ->
[n]::INIT_META_SIGNAL:p_s22 [n-1]::INIT_META_SIGNAL:p_r2_region ->