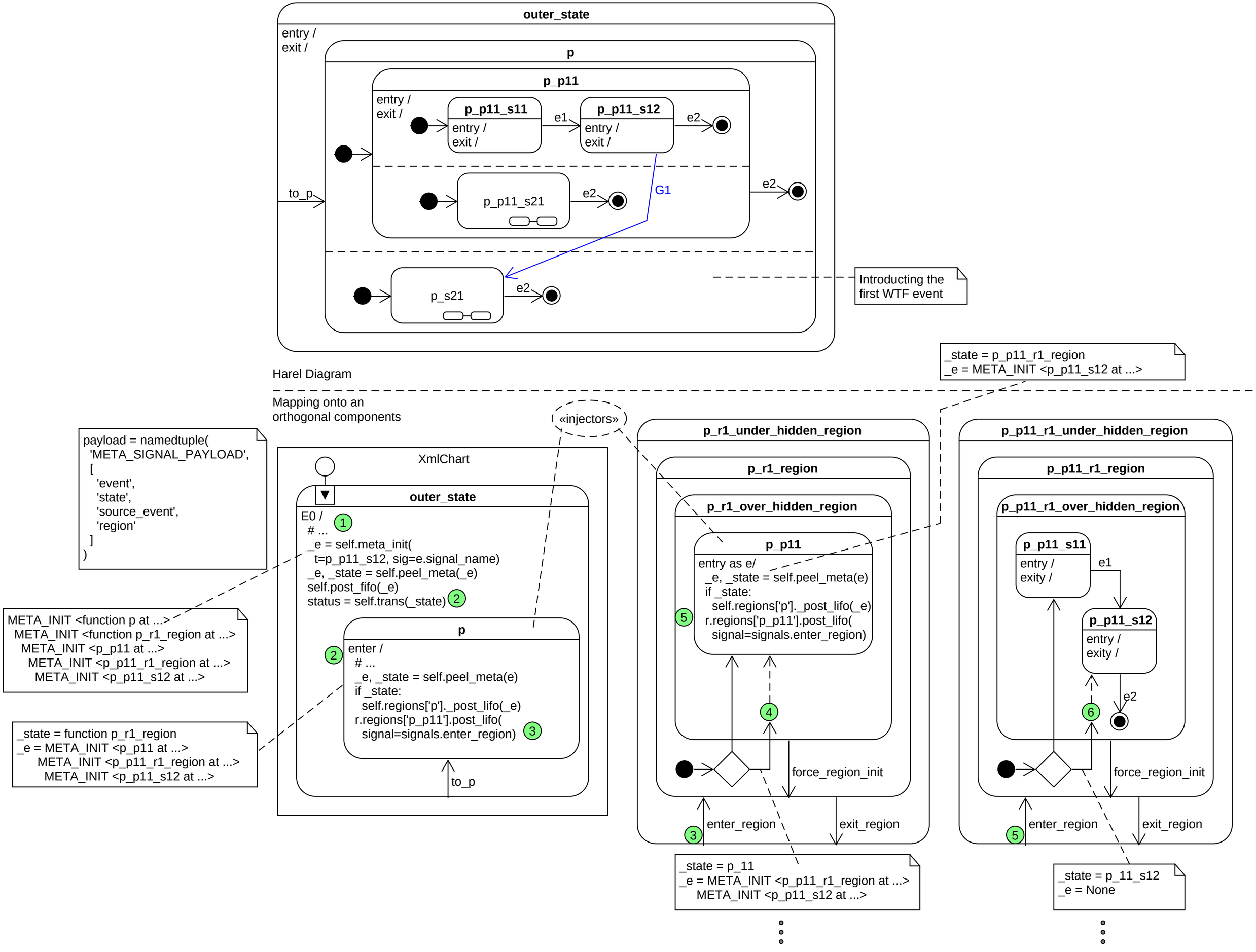
outer_state
entry /
exit /
p
p_p11
entry /
exit /
p_p11_s11
entry /
exit /
e1
p_p11_s12
entry /
exit /
e2
to_p
p_p11_s21
e2
G1
e2
p_s21
e2
Introducting the
first WTF event
Harel Diagram
_state = p_p11_r1_region
_e = META_INIT <p_p11_s12 at ...>
Mapping onto an
orthogonal components
«injectors»
payload = namedtuple(
'META_SIGNAL_PAYLOAD',
[
'event',
'state',
'source_event',
'region'
]
)
XmlChart
outer_state
E0 /
# ...
_e = self.meta_init(
t=p_p11_s12, sig=e.signal_name)
_e, _state = self.peel_meta(_e)
self.post_fifo(_e)
status = self.trans(_state)
1
2
p
enter /
...
_e, _state = self.peel_meta(e)
if _state:
self.regions['p']._post_lifo(_e)
r.regions['p_p11'].post_lifo(
signal=signals.enter_region)
2
3
to_p
META_INIT <function p at ...>
META_INIT <function p_r1_region at ...>
META_INIT <p_p11 at ...>
META_INIT <p_p11_r1_region at ...>
META_INIT <p_p11_s12 at ...>
_state = function p_r1_region
_e = META_INIT <p_p11 at ...>
META_INIT <p_p11_r1_region at ...>
META_INIT <p_p11_s12 at ...>
p_r1_under_hidden_region
p_r1_region
p_r1_over_hidden_region
p_p11
entry as e/
_e, _state = self.peel_meta(e)
if _state:
self.regions['p']._post_lifo(_e)
r.regions['p_p11'].post_lifo(
signal=signals.enter_region)
5
4
force_region_init
3
enter_region
exit_region
_state = p_11
_e = META_INIT <p_p11_r1_region at ...>
META_INIT <p_p11_s12 at ...>
p_p11_r1_under_hidden_region
p_p11_r1_region
p_p11_r1_over_hidden_region
p_p11_s11
entry /
exity /
e1
p_p11_s12
entry /
exity /
e2
6
force_region_init
5
enter_region
exit_region
_state = p_11_s12
_e = None