Orthogonal Regions Diagram

outer_state
entry /

to_p

p
entry /
exit /

S1 Region

S11
entry /
exit /

e4

S12
entry /
exit /

e1

S1Final

S2 Region

S21
entry /
exit /

e1

S22
entry /
exit /

e2

S2Final

p_final

some_other_state
entry /

to_outer

sent when S1Final and S2Final states
have been entered
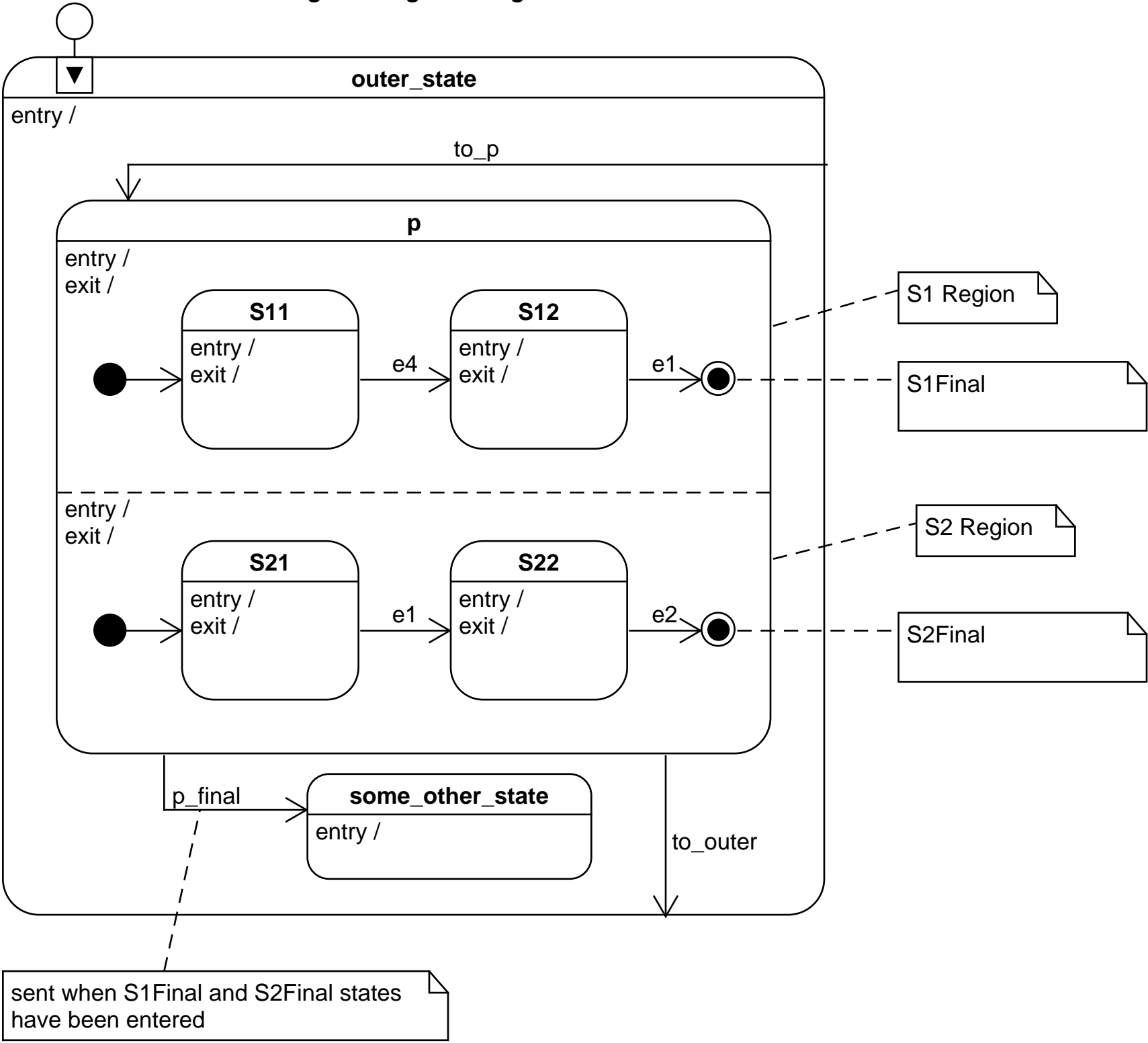
Equivalent Orthogonal Components Diagram

outer_state
entry /
exit /

p
entry /
self.p_dispatcher(Event(signal=signals.to_p))

e1, e2, e4 as e /
self.p_dispatcher(e)

exit /
self.p_dispatcher(
Event(signal=signals.region_exit))

p_final

to_outer

some_other_state

to_p

def p_dispatcher(self, e):
status = return_status.HANDLED
[region.post_fifo(e) for region in self.p_regions]
[region.complete_circuit() for region in self.p_regions]
return status

s1_hidden_state

s1_region
entry /
exit /

s11
entry /
exit /

e4

s12
entry /
exit /

e1

s1_region_final
entry /
r.final = True
r.post_final_to_other_if_ready()

exit /
r.final = False

region_exit

to_p

«pattern»
Othogonal Components

«pattern»
Reminder

s2_hidden_state

s2_region
entry /
exit /

s21
entry /
exit /

e4

s22
entry /
exit /

e1

s2_region_final
entry /
r.final = True
r.post_final_to_other_if_ready()

exit /
r.final = False

region_exit

to_p

def post_p_final_to_outer_if_ready(self):
ready = False if self.regions is None and len(self.regions) < 1 else True
for region in self.regions:
ready &= True if region.final else False
if ready:
self.outer.post_fifo(self.final_event)