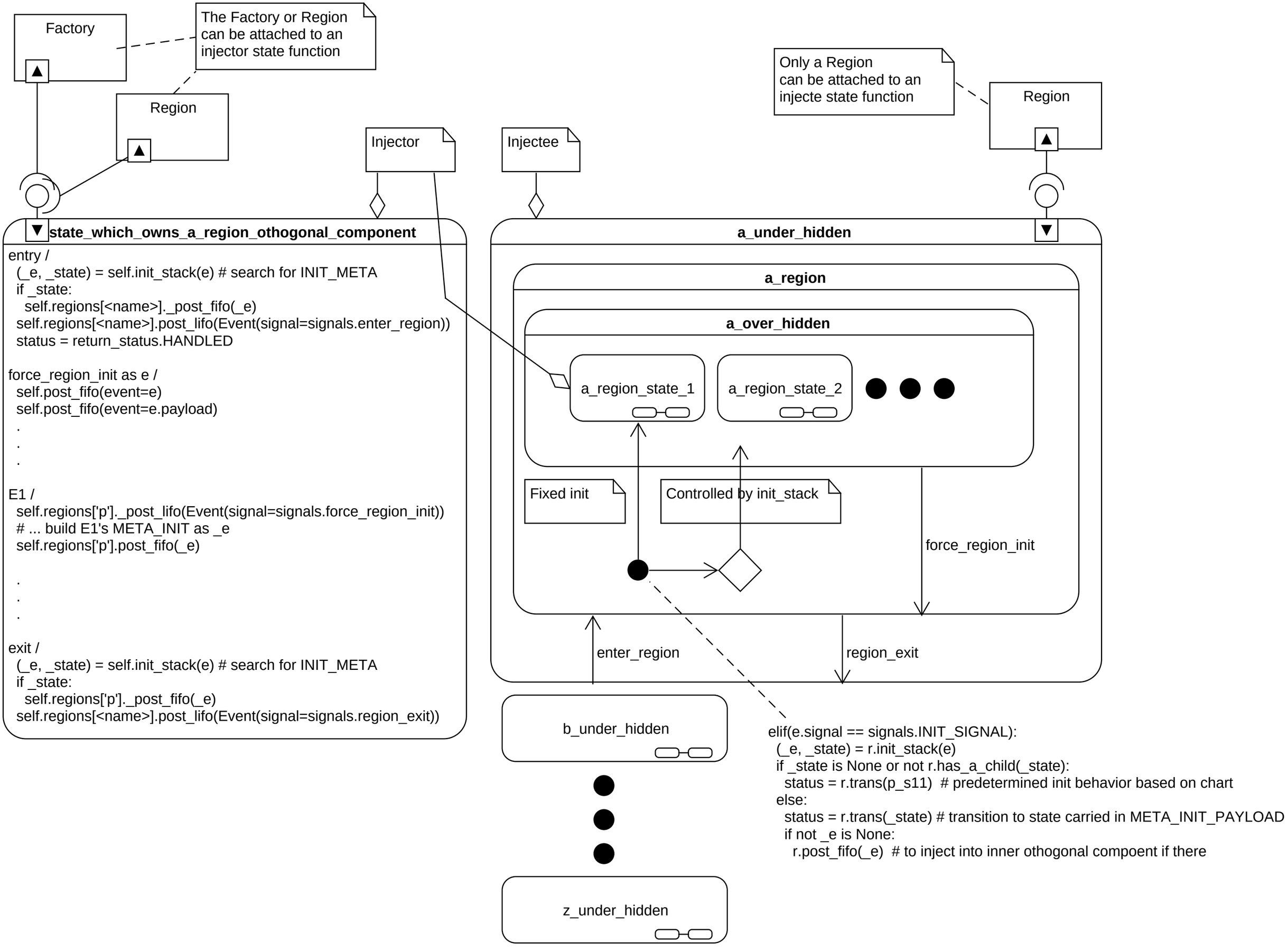
Factory
The Factory or Region can be attached to an injector state function
Region
Only a Region can be attached to an injecte state function
Region
Injector
Injectee
state_which_owns_a_region_othogonal_component
entry /
(_e, _state) = self.init_stack(e) # search for INIT_META
if _state:
self.regions[<name>]._post_fifo(_e)
self.regions[<name>].post_lifo(Event(signal=signals.enter_region))
status = return_status.HANDLED
force_region_init as e /
self.post_fifo(event=e)
self.post_fifo(event=e.payload)
.
.
.
E1 /
self.regions['p']._post_lifo(Event(signal=signals.force_region_init))
... build E1's META_INIT as _e
self.regions['p'].post_fifo(_e)
.
.
.
exit /
(_e, _state) = self.init_stack(e) # search for INIT_META
if _state:
self.regions['p']._post_fifo(_e)
self.regions[<name>].post_lifo(Event(signal=signals.region_exit))
a_under_hidden
a_region
a_over_hidden
a_region_state_1
a_region_state_2
Fixed init
Controlled by init_stack
force_region_init
enter_region
region_exit
b_under_hidden
z_under_hidden
elif(e.signal == signals.INIT_SIGNAL):
(_e, _state) = r.init_stack(e)
if _state is None or not r.has_a_child(_state):
status = r.trans(p_s11) # predetermined init behavior based on chart
else:
status = r.trans(_state) # transition to state carried in META_INIT_PAYLOAD
if not _e is None:
r.post_fifo(_e) # to inject into inner othogonal compoent if there