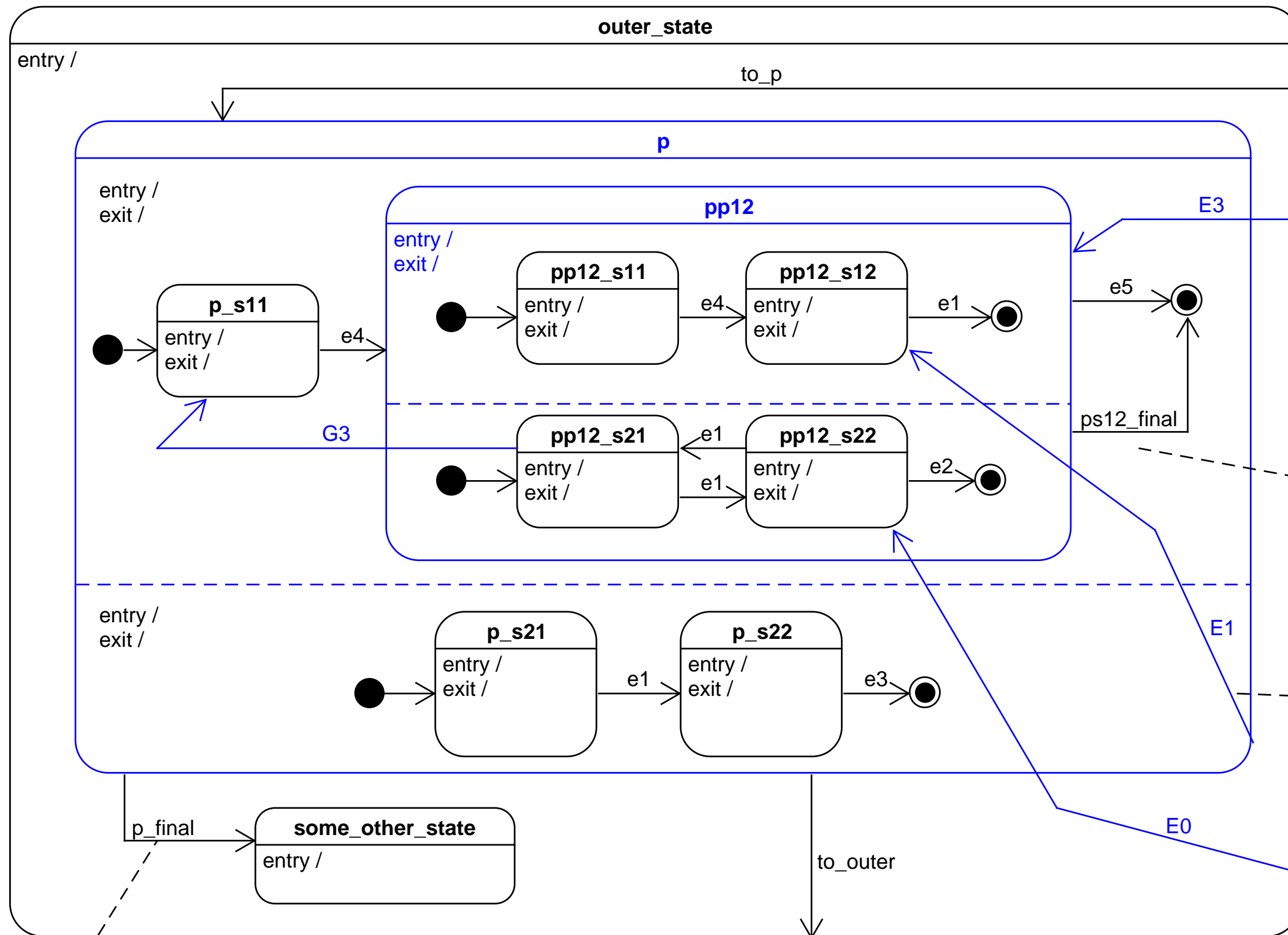


Orthogonal Regions Diagram



sent when s1Final and s2Final states have been entered

sent when ps12_s1Final and ps12_s2Final states have been entered

```
# E3
eeee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=None, state=pp12, source_event=e)
)
eee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=eeee, state="pp12", source_event=e)
)
ee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=eee, state=pp12, source_event=e)
)
_e = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=ee, state="p", source_event=e)
)
```

```
# E1
eee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=None, state=pp12_s12, source_event=e)
)
ee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=eee, state=pp12, source_event=e)
)
_e = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=ee, state=pp12, source_event=e)
)
```

```
# E0
eeee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(
        event=None, state=pp12_s22, source_event=e)
)
eee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=eeee, state=pp12, source_event=e)
)
ee = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=eee, state=pp12, source_event=e)
)
_e = Event(
    signal=signals.INIT_META,
    payload=INIT_META_PAYLOAD(event=ee, state=pp12, source_event=e)
)
```

state just has to return True

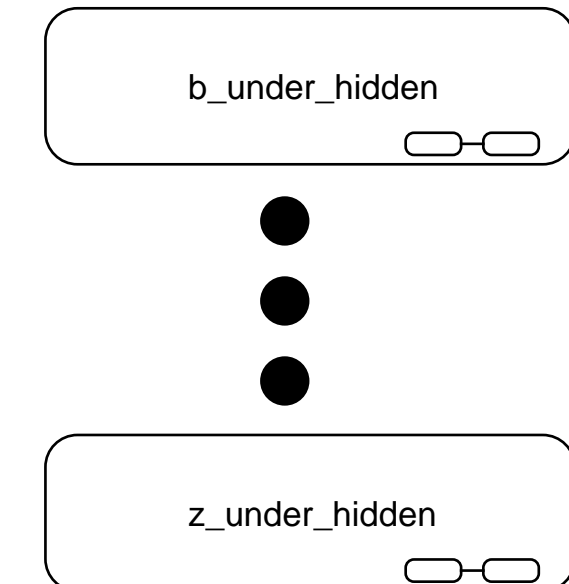
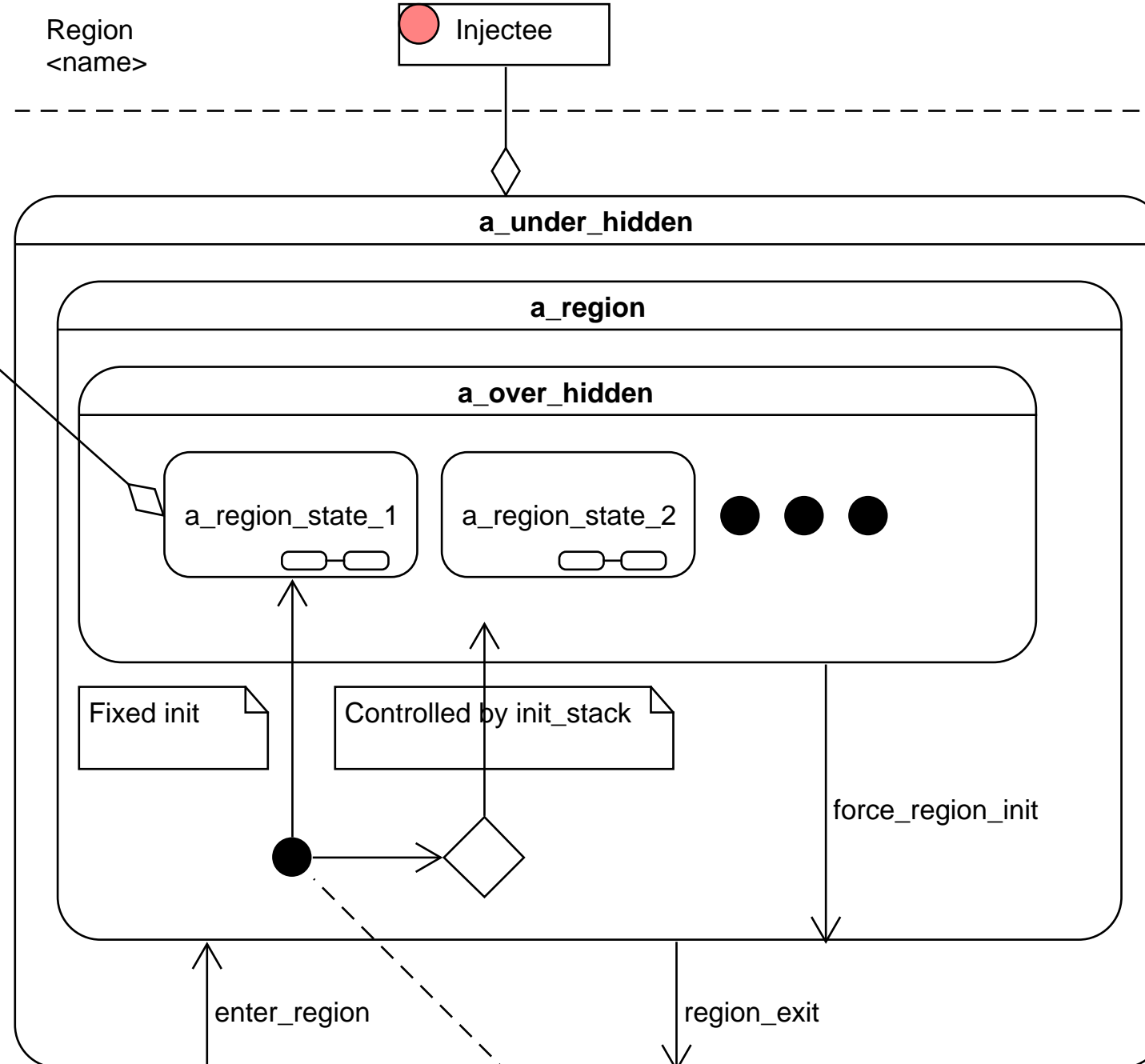
state must be callable

```
state_which_owns_a_region_othogonal_component
entry /
    self.regions[<name>].post_fifo(Event(signal=signals.enter_region))

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))
```



```
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
```