Part I

We can model this system with the tuple $S=\left(Q,\Sigma_1,\Sigma_2,q_0,\vee,\wedge\right)$, where:

$$Q=\{\ dormant,\ exit,\ init,\ idle,\ monitoring,\ error\_diagnosis,\ safe\_shutdown\ \}$$

$$\Sigma_1=\{\ kill,\ start,\ init\_ok,\ begin\_monitoring,\ init\_crash,\ retry\_init,\ shutdown,\ sleep,\ idle\_crash,\ idle\_rescue,\ moni\_rescue\ \}$$

$$\Sigma_2=\{\ retry++,\ broadcast\ init\_err\_msg,\ broadcast\ idle\_err\_msg,\ broadcast\ moni\_err\_msg\ \}$$

$$q_0=dormant$$

$$\vee : retry : \mathrm{N}_0\ ;\ init\_err\_msg : string\ ;\ idle\_err\_msg : string\ ;\ moni\_err\_msg : string$$

$$\wedge : transition\ specification :$$

1. $\xrightarrow{} dormant$

2. $dormant \xrightarrow{kill} exit$

3. $dormant \xrightarrow{start} init$

4. $init \xrightarrow{kill} exit$

5. $init \xrightarrow{init\_ok} idle$

6. $idle \xrightarrow{kill} exit$

7. $idle \xrightarrow{begin\_monitoring} monitoring$

8. $init \xrightarrow{init\_crash / broadcast\, init\_err\_msg} error\_diagnosis$

9. $error\_diagnosis \xrightarrow{retry\_init[retry<3] / retry{+}{+}} init$

10. $error\_diagnosis \xrightarrow{kill} exit$

11. $error\_diagnosis \xrightarrow{shutdown\,[retry>2]} safe\_shutdown$

12. $safe\_shutdown \xrightarrow{kill} exit$

13. $safe\_shutdown \xrightarrow{sleep} dormant$

14. $idle \xrightarrow{idle\_crash / broadcast\, idle\_err\_msg} error\_diagnosis$

15. $error\_diagnosis \xrightarrow{idle\_rescue} idle$

16. $monitoring \xrightarrow{monitor\_crash / broadcast\, moni\_err\_msg} error\_diagnosis$

17. $error\_diagnosis \xrightarrow{moni\_rescue} monitoring$

18. $monitoring \xrightarrow{kill} exit$

Part II

As **init** is a composed state, we define it as the tuple $S=\left(Q,\Sigma_1,\Sigma_2,q_0,\vee,\wedge\right)$, where:

$$Q=\{\ boot\_hw,\ \ senchk,\ \ tchk,\ \ psichk,\ \ ready\ \}$$
$$\Sigma_1=\{\ hw\_ok,\ \ senok,\ \ psi\_ok\ \ t\_ok\}$$
$$\Sigma_2=\{\ \}$$
$$q_0=boot\_hw$$
$$\vee=\{\ \}$$
$$\wedge: transition\ \ specification:$$

$$1. \longrightarrow boot\_hw$$
$$2.\ \ boot\_hw \xrightarrow{hw\_ok} senchk$$
$$3.\ \ senchk \xrightarrow{senok} tchk$$
$$4.\ \ tchk \xrightarrow{t\_ok} psichk$$
$$5.\ \ psichk \xrightarrow{psi\_ok} ready$$

Part III

As **monitoring** is a composed state, we define it as the tuple $S = (Q, \Sigma_1, \Sigma_2, q_0, \vee, \wedge)$, where:

$$Q = \{\ monidle,\ \ regulate\_enironment,\ \ lockdown\ \}$$

$$\Sigma_1 = \{\ no\_contagion,\ \ after\_100ms,\ \ contagion\_alert,\ \ purge\_succ\ \}$$

$$\Sigma_2 = \{\ broadcast\ \ FACILITY\_CRIT\_MESG\ \}$$

$$q_0 = monidle$$

$$\vee : FACILITY\_CRIT\_MESG : string\ \ ;\ \ inlockdown : Boolean$$

$$\wedge : transition\ \ specification :$$

$$1. \longrightarrow monidle$$

$$2.\ \ monidle \xrightarrow{no\_contagion} regulate\_environment$$

$$3.\ \ regulate\_environment \xrightarrow{after\_100\,ms} monidle$$

$$4.\ \ regulate\_environment \xrightarrow{contagion\_alert / broadcast\, FACILITY\_CRIT\_MESG,\ \ inlockdown := true} lockdown$$

$$5.\ \ lockdown \xrightarrow{purge\_succ / inlockdown := false} monidle$$

$$\text{super} :$$

$$6.\ \ monitoring \xrightarrow{monitor\_crash\ [!inlockdown] / broadcast\, moni\_err\_msg} error\_diagnosis$$

$$7.\ \ monitoring \xrightarrow{kill\ [!inlockdown]} exit$$

Part IV

As **lockdown** is a composed state, we define it as the tuple $S = (Q, \Sigma_1, \Sigma_2, q_0, \vee, \wedge)$, where

$$Q = \{\ prep\_vpurge,\ \ alt\_temp,\ \ alt\_psi,\ \ risk\_assess,\ \ safe\_status,\ \ exit\ \}$$

$$\Sigma_1 = \{\ initiate\_purge,\ \ tcyc\_comp,\ \ psicyc\_comp\ \}$$

$$\Sigma_2 = \{\ lock\_doors,\ \ unlock\_doors\ \}$$

$$q_0 = prep\_vpurge$$

$$\vee : risk : \{risk \in \mathrm{Q} \mid 0 \le risk \le 1\}$$

$$\wedge : transition\ specification :$$

1. $\longrightarrow prep\_vpurge$
2. $prep\_vpurge \xrightarrow{initiate\_purge/lock\_doors} alt\_temp$
3. $prep\_vpurge \xrightarrow{initiate\_purge/lock\_doors} alt\_psi$
4. $alt\_temp \xrightarrow{tcyc\_comp} risk\_assess$
5. $alt\_psi \xrightarrow{psicyc\_comp} risk\_assess$
6. $risk\_assess \xrightarrow{[risk>0.01]} prep\_vpurge$
7. $risk\_assess \xrightarrow{[risk<0.01]/unlock\_doors} safe\_status$
8. $safe\_status \longrightarrow exit$

Part V

As **error_diagnosis** is a composed state, we define it as the tuple $S = (Q, \Sigma_1, \Sigma_2, q_0, \vee, \wedge)$, where

$$Q = \{\ error\_rcv,\ \ applicable\_rescue,\ \ exit,\ \ reset\_module\_data\ \}$$
$$\Sigma_1 = \{\ apply\_protocol\_issue,\ \ reset\_to\_stable\ \}$$
$$\Sigma_2 = \{\ \}$$
$$q_0 = error\_rcv$$
$$\vee : err\_protocol\_def\ : Boolean$$
$$\wedge : transition\ \ specification:$$

1. $\longrightarrow error\_rcv$
2. $error\_rcv \xrightarrow{[err\_protocol\_def]} applicable\_rescue$
3. $applicable\_rescue \xrightarrow{apply\_protocol\_rescue} exit$
4. $error\_rcv \xrightarrow{[!err\_protocol\_def]} reset\_module\_data$
5. $reset\_module\_data \xrightarrow{reset\_to\_stable} exit$