Hybrid Planning in Self-adaptive Systems

Ashutosh Pandey

CMU-ISR-20-100-APPENDIX
February 2020

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
David Garlan (Chair)
Jonathan Aldrich
John Dolan
Hausi Müller (University of Victoria, Canada)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Software Engineering.

This document is a supplement to the free-standing thesis document CMU-CS-20-100. This document provides the PRISM planning specifications corresponding to reactive and deliberative planning used for the two case studies used for evaluating the thesis claims in the document CMU-CS-20-100.

Keywords: self-adaptive systems, formal model, automated planning, machine learning, probabilistic model-checking

PRISM Planning Specifications for the Cloud-based System

For a particular situation of the system and environment, this section provides the PRISM planning specifications for non-wait reactive (i.e., deterministic) and deliberative (i.e., MDP) planning used for the cloud-based system. The reactive and deliberative planning specifications have the same initial state.

Deterministic Planning Specification

The listing below is the PRISM specification for deterministic planning ρ_{det}, which ignores uncertainty in the request arrival rate by assuming it to be constant at the current value. In the specification, inter-arrival time (the inverse of average request arrival rate) between two consecutive requests is used for environment modeling. For instance, in the statement "formula stateValue = 0.0163126" the request arrival rate will be 61 (= 1/0.0163126) requests per minute.

```
mdp

const double addServer_LATENCY = 120;
const int HORIZON = 5;
const double PERIOD = 60;
const int DIMMER_LEVELS = 3;
const int ini_dimmer = 1;
const int MAX_SERVERS_A = 1;
const int MAX_SERVERS_B = 1;
const int MAX_SERVERS_C = 1;
const int ini_servers_A = 1;
const int ini_servers_B = 0;
const int ini_servers_C = 0;
const int ini_addServerA_state = 0;
const int ini_addServerB_state = 0;
const int ini_addServerC_state = 0;
const double SERVERA_COST = 1;
const double SERVERB_COST = 0.7;
const double SERVERC_COST = 0.5;
const double MAX_ARRIVALA_CAPACITY = 200;
const double MAX_ARRIVALA_CAPACITY_LOW = 400;
const double MAX_ARRIVALB_CAPACITY = 140;
const double MAX_ARRIVALB_CAPACITY_LOW = 280;
const double MAX_ARRIVALC_CAPACITY = 100;
const double MAX_ARRIVALC_CAPACITY_LOW = 200;
const double penalty = −0.25;
const int ini_traffic_A = 4;
const int ini_traffic_B = 0;
const int ini_traffic_C = 0;
```

```
const double interArrivalScaleFactorForDecision = 1; // 1 has no effect

// The request arrival rate remains constant at the current value
formula stateValue = 0.0163126;


module clk
    time : [0..HORIZON + 1] init 0;
    readyToTick : bool init true;
    [tick] readyToTick & time < HORIZON + 1 -> 1 : (time' = time + 1) & (readyToTick'=false);
    [tack] !readyToTick -> 1 : (readyToTick'=true);
endmodule

label "final" = time = HORIZON + 1;
formula sys_go = readyToTick;

module controller
    active_servers_A : [0..MAX_SERVERS_A] init ini_servers_A;
    active_servers_B : [0..MAX_SERVERS_B] init ini_servers_B;
    active_servers_C : [0..MAX_SERVERS_C] init ini_servers_C;

    dimmer : [1..DIMMER_LEVELS] init ini_dimmer;

    traffic_A : [0..4] init ini_traffic_A;
    traffic_B : [0..4] init ini_traffic_B;
    traffic_C : [0..4] init ini_traffic_C;

    [addServerA_complete] active_servers_A < MAX_SERVERS_A -> 1 : (active_servers_A' = active_servers_A + 1);
    [addServerB_complete] active_servers_B < MAX_SERVERS_B -> 1 : (active_servers_B' = active_servers_B + 1);
    [addServerC_complete] active_servers_C < MAX_SERVERS_C -> 1 : (active_servers_C' = active_servers_C + 1);

    [removeServerA_start] active_servers_A > 0 -> 1 : (active_servers_A' = active_servers_A - 1);
    [removeServerB_start] active_servers_B > 0 -> 1 : (active_servers_B' = active_servers_B - 1);
    [removeServerC_start] active_servers_C > 0 -> 1 : (active_servers_C' = active_servers_C - 1);

    [increaseDimmer_start] dimmer < DIMMER_LEVELS -> 1 : (dimmer' = dimmer + 1);
    [decreaseDimmer_start] dimmer > 1 -> 1 : (dimmer' = dimmer - 1);

    //A-B-C
    //Possible values 0-25-50-75-100

    // 100-0-0
    [divert_100_0_0] active_servers_A > 0
        -> 1 : (traffic_A' = 4) & (traffic_B' = 0) & (traffic_C' = 0);
    // 75-25-0
    [divert_75_25_0] active_servers_A > 0 & active_servers_B > 0
        -> 1 : (traffic_A' = 3) & (traffic_B' = 1) & (traffic_C' = 0);
    // 75-0-25
    [divert_75_0_25] active_servers_A > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 3) & (traffic_B' = 0) & (traffic_C' = 1);
    // 50-50-0
    [divert_50_50_0] active_servers_A > 0 & active_servers_B > 0
        -> 1 : (traffic_A' = 2) & (traffic_B' = 2) & (traffic_C' = 0);
    // 50-0-50
    [divert_50_0_50] active_servers_A > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 2) & (traffic_B' = 0) & (traffic_C' = 2);
    // 50-25-25
    [divert_50_25_25] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 2) & (traffic_B' = 1) & (traffic_C' = 1);
    // 25-75-0
    [divert_25_75_0] active_servers_A > 0 & active_servers_B > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 3) & (traffic_C' = 0);
    // 25-0-75
    [divert_25_0_75] active_servers_A > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 0) & (traffic_C' = 3);
```

```
95      // 25−50−25
96      [divert_25_50_25] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
97          −> 1 : (traffic_A' = 1) & (traffic_B' = 2) & (traffic_C' = 1);
98      // 25−25−50
99      [divert_25_25_50] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
100         −> 1 : (traffic_A' = 1) & (traffic_B' = 1) & (traffic_C' = 2);
101     // 0−100−0
102     [divert_0_100_0] active_servers_B > 0
103         −> 1 : (traffic_A' = 0) & (traffic_B' = 4) & (traffic_C' = 0);
104     // 0−0−100
105     [divert_0_0_100] active_servers_C > 0
106         −> 1 : (traffic_A' = 0) & (traffic_B' = 0) & (traffic_C' = 4);
107     // 0−75−25
108     [divert_0_75_25] active_servers_B > 0 & active_servers_C > 0
109         −> 1 : (traffic_A' = 0) & (traffic_B' = 3) & (traffic_C' = 1);
110     // 0−25−75
111     [divert_0_25_75] active_servers_B > 0 & active_servers_C > 0
112         −> 1 : (traffic_A' = 0) & (traffic_B' = 1) & (traffic_C' = 3);
113     // 0−50−50
114     [divert_0_50_50] active_servers_B > 0 & active_servers_C > 0
115         −> 1 : (traffic_A' = 0) & (traffic_B' = 2) & (traffic_C' = 2);
116 endmodule
117
118
119 formula addServerA_applicable = active_servers_A < MAX_SERVERS_A & !removeServer_used
120                 & addServerB_state = 0 & addServerC_state = 0;
121 formula addServerB_applicable = active_servers_B < MAX_SERVERS_B & !removeServer_used
122                 & addServerA_state = 0 & addServerC_state = 0;
123 formula addServerC_applicable = active_servers_C < MAX_SERVERS_C & !removeServer_used
124                 & addServerA_state = 0 & addServerB_state = 0;
125
126 formula removeServerA_applicable = active_servers_A > 0 & addServerA_state = 0 & active_servers > 1
127                     & addServerB_state = 0 & addServerC_state = 0;
128 formula removeServerB_applicable = active_servers_B > 0 & addServerB_state = 0 & active_servers > 1
129                     & addServerA_state = 0 & addServerC_state = 0;
130 formula removeServerC_applicable = active_servers_C > 0 & addServerC_state = 0 & active_servers > 1
131                     & addServerA_state = 0 & addServerB_state = 0;
132
133 formula increaseDimmer_compatible = !decreaseDimmer_used;
134 formula decreaseDimmer_compatible = !increaseDimmer_used;
135 formula increase_dimmer_applicable = dimmer < DIMMER_LEVELS & increaseDimmer_compatible;
136 formula decrease_dimmer_applicable = dimmer > 1 & decreaseDimmer_compatible;
137
138 const int addServer_LATENCY_PERIODS = ceil(addServer_LATENCY / PERIOD);
139
140 // This remove server constraints that only one server could be removed in one monitoring cycle.
141 module removeServer
142     removeServer_go : bool init true;
143     removeServer_used : bool init false;
144
145     [removeServerA_start] sys_go & removeServer_go
146         & removeServerA_applicable // applicability conditions
147         −> (removeServer_go' = false) & (removeServer_used' = true);
148
149     [removeServerB_start] sys_go & removeServer_go
150         & removeServerB_applicable // applicability conditions
151         −> (removeServer_go' = false) & (removeServer_used' = true);
152
153     [removeServerC_start] sys_go & removeServer_go
154         & removeServerC_applicable // applicability conditions
155         −> (removeServer_go' = false) & (removeServer_used' = true);
156
157     // Case when remove server tactic is applicable but not used
158     [pass_remove_server] sys_go & removeServer_go // can go
159         −> (removeServer_go' = false);
```

```
160
161     [tick] !removeServer_go -> 1 : (removeServer_go' = true) & (removeServer_used' = false);
162 endmodule
163
164 module addServer
165     addServerA_state : [0..addServer_LATENCY_PERIODS] init ini_addServerA_state;
166     addServerB_state : [0..addServer_LATENCY_PERIODS] init ini_addServerB_state;
167     addServerC_state : [0..addServer_LATENCY_PERIODS] init ini_addServerC_state;
168
169     addServer_go : bool init true;
170
171     // tactic applicable, start it
172     [addServerA_start] sys_go & addServer_go // can go
173         & addServerA_state = 0 // tactic has not been started
174         & addServerA_applicable
175         -> (addServerA_state' = 1) & (addServer_go' = false);
176
177     // tactic applicable, start it
178     [addServerB_start] sys_go & addServer_go // can go
179         & addServerB_state = 0 // tactic has not been started
180         & addServerB_applicable
181         -> (addServerB_state' = 1) & (addServer_go' = false);
182
183     // tactic applicable, start it
184     [addServerC_start] sys_go & addServer_go // can go
185         & addServerC_state = 0 // tactic has not been started
186         & addServerC_applicable
187         -> (addServerC_state' = 1) & (addServer_go' = false);
188
189     // tactic applicable, but don't use it
190     [pass_add] sys_go & addServer_go // can go
191         & addServerA_state = 0 // tactic has not been started
192         & addServerB_state = 0 & addServerC_state = 0
193         //& addServerA_applicable
194         -> (addServer_go' = false);
195
196     // progress of the tactic
197     [progressA] sys_go & addServer_go
198         & addServerA_state > 0 & addServerA_state < addServer_LATENCY_PERIODS
199         -> 1 : (addServerA_state' = addServerA_state + 1) & (addServer_go' = false);
200
201     [progressB] sys_go & addServer_go
202         & addServerB_state > 0 & addServerB_state < addServer_LATENCY_PERIODS
203         -> 1 : (addServerB_state' = addServerB_state + 1) & (addServer_go' = false);
204
205     [progressC] sys_go & addServer_go
206         & addServerC_state > 0 & addServerC_state < addServer_LATENCY_PERIODS
207         -> 1 : (addServerC_state' = addServerC_state + 1) & (addServer_go' = false);
208
209     // completion of the tactic
210     [addServerA_complete] sys_go & addServer_go
211         & addServerA_state = addServer_LATENCY_PERIODS // completed
212         -> 1 : (addServerA_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed
213
214     [addServerB_complete] sys_go & addServer_go
215         & addServerB_state = addServer_LATENCY_PERIODS // completed
216         -> 1 : (addServerB_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed
217
218     [addServerC_complete] sys_go & addServer_go
219         & addServerC_state = addServer_LATENCY_PERIODS // completed
220         -> 1 : (addServerC_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed
221
222     [tick] !addServer_go -> 1 : (addServer_go' = true);
223 endmodule
224
```

```
225 // Make sure that divert traffic is executed at the end i.e.after adding or removing the servers.
226 formula divert_traffic_applicable = divert_go & !addServer_go & !removeServer_go & !increaseDimmer_go &
        !decreaseDimmer_go;
227
228 module divert_traffic
229     divert_go : bool init true;
230
231     //A-B-C
232     //Possible values 0-25-50-75-100
233
234     // 100-0-0
235     [divert_100_0_0] sys_go & divert_traffic_applicable
236         -> 1 : (divert_go'=false);
237     // 75-25-0
238     [divert_75_25_0] sys_go & divert_traffic_applicable
239         -> 1 : (divert_go'=false);
240     // 75-0-25
241     [divert_75_0_25] sys_go & divert_traffic_applicable
242         -> 1 : (divert_go'=false);
243     // 50-50-0
244     [divert_50_50_0] sys_go & divert_traffic_applicable
245         -> 1 : (divert_go'=false);
246     // 50-0-50
247     [divert_50_0_50] sys_go & divert_traffic_applicable
248         -> 1 : (divert_go'=false);
249     // 50-25-25
250     [divert_50_25_25] sys_go & divert_traffic_applicable
251         -> 1 : (divert_go'=false);
252     // 25-75-0
253     [divert_25_75_0] sys_go & divert_traffic_applicable
254         -> 1 : (divert_go'=false);
255     // 25-0-75
256     [divert_25_0_75] sys_go & divert_traffic_applicable
257         -> 1 : (divert_go'=false);
258     // 25-50-25
259     [divert_25_50_25] sys_go & divert_traffic_applicable
260         -> 1 : (divert_go'=false);
261     // 25-25-50
262     [divert_25_25_50] sys_go & divert_traffic_applicable
263         -> 1 : (divert_go'=false);
264     // 0-100-0
265     [divert_0_100_0] sys_go & divert_traffic_applicable
266         -> 1 : (divert_go'=false);
267     // 0-0-100
268     [divert_0_0_100] sys_go & divert_traffic_applicable
269         -> 1 : (divert_go'=false);
270     // 0-75-25
271     [divert_0_75_25] sys_go & divert_traffic_applicable
272         -> 1 : (divert_go'=false);
273     // 0-25-75
274     [divert_0_25_75] sys_go & divert_traffic_applicable
275         -> 1 : (divert_go'=false);
276     // 0-50-50
277     [divert_0_50_50] sys_go & divert_traffic_applicable
278         -> 1 : (divert_go'=false);
279
280     [tick] !divert_go -> 1 : (divert_go' = true);
281 endmodule
282
283 module increaseDimmer
284     increaseDimmer_go : bool init true;
285     increaseDimmer_used : bool init false;
286
287     [increaseDimmer_start] sys_go & increaseDimmer_go
288         & increase_dimmer_applicable // applicability conditions
```

```
289             -> (increaseDimmer_go' = false) & (increaseDimmer_used' = true);
290
291         // tactic applicable but not used
292         [pass_inc_dimmer] sys_go & increaseDimmer_go // can go
293             -> (increaseDimmer_go' = false);
294
295         [tick] !increaseDimmer_go -> 1 : (increaseDimmer_go' = true) & (increaseDimmer_used' = false);
296     endmodule
297
298     // tactic
299     module decreaseDimmer
300         decreaseDimmer_go : bool init true;
301         decreaseDimmer_used : bool init false;
302
303         [decreaseDimmer_start] sys_go & decreaseDimmer_go
304             & decrease_dimmer_applicable // applicability conditions
305             -> (decreaseDimmer_go' = false) & (decreaseDimmer_used' = true);
306
307         // tactic applicable but not used
308         [pass_dec_dimmer] sys_go & decreaseDimmer_go // can go
309             -> (decreaseDimmer_go' = false);
310
311         [tick] !decreaseDimmer_go -> 1 : (decreaseDimmer_go' = true) & (decreaseDimmer_used' = false);
312     endmodule
313
314     //*************************************************************************
315     // Queuing network with each server having queueing model of M/G/1/PS
316     //*************************************************************************
317     formula dimmerFactor = (dimmer - 1) / (DIMMER_LEVELS - 1);
318     formula interarrivalMean = stateValue * interArrivalScaleFactorForDecision;
319
320     formula Pa = (traffic_A * 25)/100;
321     formula Pb = (traffic_B * 25)/100;
322     formula Pc = (traffic_C * 25)/100;
323
324     formula loaded_servers = (Pa != 0 ? 1 : 0) + (Pb != 0 ? 1 : 0) + (Pc != 0 ? 1 : 0);
325
326     formula service_rate_A = dimmerFactor * (MAX_ARRIVALA_CAPACITY_LOW)
327                 + (1 - dimmerFactor) * (MAX_ARRIVALA_CAPACITY);
328     formula service_rate_B = dimmerFactor * (MAX_ARRIVALB_CAPACITY_LOW)
329                 + (1 - dimmerFactor) * (MAX_ARRIVALB_CAPACITY);
330     formula service_rate_C = dimmerFactor * (MAX_ARRIVALC_CAPACITY_LOW)
331                 + (1 - dimmerFactor) * (MAX_ARRIVALC_CAPACITY);
332
333     formula rhoA = Pa/(service_rate_A*interarrivalMean);
334     formula rhoB = Pb/(service_rate_B*interarrivalMean);
335     formula rhoC = Pc/(service_rate_C*interarrivalMean);
336     formula overloaded = (rhoA >= 1 | rhoB >= 1 | rhoC >= 1);
337
338     formula rt_A = 1/(service_rate_A - (throughput*Pa));
339     formula rt_B = 1/(service_rate_B - (throughput*Pb));
340     formula rt_C = 1/(service_rate_C - (throughput*Pc));
341
342     // Response time to clients utility function
343     const double RT_THRESHOLD = 1.0;
344
345     formula expected_wait_time = (Pa*rt_A + Pb*rt_B + Pc*rt_C);
346     formula rt = (interarrivalMean = 0 ? 0 : (overloaded ? RT_THRESHOLD + 2 : expected_wait_time));
347     const double NORMAL_A_REVENUE = (SERVERA_COST / MAX_ARRIVALA_CAPACITY) * 10;
348     const double DIMMER_A_REVENUE = (SERVERA_COST / MAX_ARRIVALA_CAPACITY_LOW) * 3 / 2;
349     const double NORMAL_B_REVENUE = (SERVERB_COST / MAX_ARRIVALB_CAPACITY) * 10;
350     const double DIMMER_B_REVENUE = (SERVERB_COST / MAX_ARRIVALB_CAPACITY_LOW) * 3 / 2;
351     const double NORMAL_C_REVENUE = (SERVERC_COST / MAX_ARRIVALC_CAPACITY) * 10;
352     const double DIMMER_C_REVENUE = (SERVERC_COST / MAX_ARRIVALC_CAPACITY_LOW) * 3 / 2;
353
```

```
const double DIMMER_REVENUE = DIMMER_A_REVENUE + DIMMER_B_REVENUE + DIMMER_C_REVENUE;
const double NORMAL_REVENUE = NORMAL_A_REVENUE + NORMAL_B_REVENUE + NORMAL_C_REVENUE;

formula serverA_cost = ((addServerA_state > 0 ? 1 : 0) + active_servers_A) * SERVERA_COST;
formula serverB_cost = ((addServerB_state > 0 ? 1 : 0) + active_servers_B) * SERVERB_COST;
formula serverC_cost = ((addServerC_state > 0 ? 1 : 0) + active_servers_C) * SERVERC_COST;
formula cost = serverA_cost + serverB_cost + serverC_cost;
formula throughput = 1/interarrivalMean;

formula basicUtilityA = throughput * Pa * (dimmerFactor * DIMMER_A_REVENUE + (1 - dimmerFactor) *
    NORMAL_A_REVENUE);
formula basicUtilityB = throughput * Pb * (dimmerFactor * DIMMER_B_REVENUE + (1 - dimmerFactor) *
    NORMAL_B_REVENUE);
formula basicUtilityC = throughput * Pc * (dimmerFactor * DIMMER_C_REVENUE + (1 - dimmerFactor) *
    NORMAL_C_REVENUE);

formula basicUtility = basicUtilityA + basicUtilityB + basicUtilityC;
formula active_servers = active_servers_A + active_servers_B + active_servers_C;
formula poweredServers = (addServerA_state > 0 ? 1 : 0) + (addServerB_state > 0 ? 1 : 0) + (addServerC_state > 0 ? 1 : 0 )
    + active_servers;
formula MAX_SERVERS = MAX_SERVERS_A + MAX_SERVERS_B + MAX_SERVERS_C;

formula MAX_SERVER_COST = MAX_SERVERS_A * SERVERA_COST
            + MAX_SERVERS_B * SERVERB_COST
            + MAX_SERVERS_C * SERVERC_COST;

formula netPenalty = stateValue > 0 ? penalty / stateValue : 0;

formula uTotal = (overloaded & (poweredServers < MAX_SERVERS | dimmer < DIMMER_LEVELS | active_servers !=
    loaded_servers))
                    ? -(1000) // avoid unstable solutions
                : (((rt > RT_THRESHOLD | rt <= 0) ? netPenalty : basicUtility) - cost);

rewards "util"
    // 100000000.0 is added to avoid a negative value during calculation; negative utility is not supported by PRISM.
    [tack] true : 100000000.0 + (PERIOD)*(uTotal);
endrewards
```

Listing 1: PRISM specification for deterministic planning

MDP Planning Specification

In contrast to deterministic planning (i.e., ρ_{det}), MDP planning (i.e., ρ_{mdp}) considers predicted (uncertain) values of request arrival rate. For ρ_{mdp}, we create an environment model using future values of inter-arrival time (the inverse of average request arrival rate) between two consecutive requests. When deliberative planning (i.e., ρ_{mdp}) is triggered, a time-series predictor feeds predicted values as an environment model formulating an MDP, mapping each possible interarrival rate to an outcome of a probabilistic action taken by the environment. The specification has environment modeled as an MDP for the planning horizon (i.e., 5) for MDP planning.

```
mdp

const double addServer_LATENCY = 120;
const int HORIZON = 5;
const double PERIOD = 60;
const int DIMMER_LEVELS = 3;
const int ini_dimmer = 1;
const int MAX_SERVERS_A = 1;
```

```
const int MAX_SERVERS_B = 1;
const int MAX_SERVERS_C = 1;
const int ini_servers_A = 1;
const int ini_servers_B = 0;
const int ini_servers_C = 0;
const int ini_addServerA_state = 0;
const int ini_addServerB_state = 0;
const int ini_addServerC_state = 0;
const double SERVERA_COST = 1;
const double SERVERB_COST = 0.7;
const double SERVERC_COST = 0.5;
const double MAX_ARRIVALA_CAPACITY = 200;
const double MAX_ARRIVALA_CAPACITY_LOW = 400;
const double MAX_ARRIVALB_CAPACITY = 140;
const double MAX_ARRIVALB_CAPACITY_LOW = 280;
const double MAX_ARRIVALC_CAPACITY = 100;
const double MAX_ARRIVALC_CAPACITY_LOW = 200;
const double penalty = −0.25;
const int ini_traffic_A = 4;
const int ini_traffic_B = 0;
const int ini_traffic_C = 0;
const double interArrivalScaleFactorForDecision = 1; // 1 has no effect

\\ Model of the environment as an MDP. Values from the time series predictor
\\ have been used to get the interarrival time.
module environment
s : [0..201] init 0;
[tick] s = 0 −>
    0.185 : (s' = 1)
    + 0.63 : (s' = 2)
    + 0.185 : (s' = 3);
[tick] s = 3 −>
    0.185 : (s' = 4)
    + 0.63 : (s' = 5)
    + 0.185 : (s' = 6);
[tick] s = 6 −>
    0.185 : (s' = 7)
    + 0.63 : (s' = 8)
    + 0.185 : (s' = 9);
[tick] s = 9 −>
    0.185 : (s' = 10)
    + 0.63 : (s' = 11)
    + 0.185 : (s' = 12);
[tick] s = 12 −>
    1 : (s' = 13);
[tick] s = 11 −>
    1 : (s' = 14);
[tick] s = 10 −>
    1 : (s' = 15);
[tick] s = 8 −>
    0.185 : (s' = 16)
    + 0.63 : (s' = 17)
    + 0.185 : (s' = 18);
[tick] s = 18 −>
    1 : (s' = 19);
[tick] s = 17 −>
    1 : (s' = 20);
[tick] s = 16 −>
    1 : (s' = 21);
[tick] s = 7 −>
    0.185 : (s' = 22)
    + 0.63 : (s' = 23)
    + 0.185 : (s' = 24);
[tick] s = 24 −>
    1 : (s' = 25);
```

```
[tick] s = 23 ->
      1 : (s' = 26);
[tick] s = 22 ->
      1 : (s' = 27);
[tick] s = 5 ->
      0.185 : (s' = 28)
      + 0.63 : (s' = 29)
      + 0.185 : (s' = 30);
[tick] s = 30 ->
      0.185 : (s' = 31)
      + 0.63 : (s' = 32)
      + 0.185 : (s' = 33);
[tick] s = 33 ->
      1 : (s' = 34);
[tick] s = 32 ->
      1 : (s' = 35);
[tick] s = 31 ->
      1 : (s' = 36);
[tick] s = 29 ->
      0.185 : (s' = 37)
      + 0.63 : (s' = 38)
      + 0.185 : (s' = 39);
[tick] s = 39 ->
      1 : (s' = 40);
[tick] s = 38 ->
      1 : (s' = 41);
[tick] s = 37 ->
      1 : (s' = 42);
[tick] s = 28 ->
      0.185 : (s' = 43)
      + 0.63 : (s' = 44)
      + 0.185 : (s' = 45);
[tick] s = 45 ->
      1 : (s' = 46);
[tick] s = 44 ->
      1 : (s' = 47);
[tick] s = 43 ->
      1 : (s' = 48);
[tick] s = 4 ->
      0.185 : (s' = 49)
      + 0.63 : (s' = 50)
      + 0.185 : (s' = 51);
[tick] s = 51 ->
      0.185 : (s' = 52)
      + 0.63 : (s' = 53)
      + 0.185 : (s' = 54);
[tick] s = 54 ->
      1 : (s' = 55);
[tick] s = 53 ->
      1 : (s' = 56);
[tick] s = 52 ->
      1 : (s' = 57);
[tick] s = 50 ->
      0.185 : (s' = 58)
      + 0.63 : (s' = 59)
      + 0.185 : (s' = 60);
[tick] s = 60 ->
      1 : (s' = 61);
[tick] s = 59 ->
      1 : (s' = 62);
[tick] s = 58 ->
      1 : (s' = 63);
[tick] s = 49 ->
      0.185 : (s' = 64)
      + 0.63 : (s' = 65)
```

```
139         + 0.185 : (s' = 66);
140     [tick] s = 66 ->
141         1 : (s' = 67);
142     [tick] s = 65 ->
143         1 : (s' = 68);
144     [tick] s = 64 ->
145         1 : (s' = 69);
146     [tick] s = 2 ->
147         0.185 : (s' = 70)
148         + 0.63 : (s' = 71)
149         + 0.185 : (s' = 72);
150     [tick] s = 72 ->
151         0.185 : (s' = 73)
152         + 0.63 : (s' = 74)
153         + 0.185 : (s' = 75);
154     [tick] s = 75 ->
155         0.185 : (s' = 76)
156         + 0.63 : (s' = 77)
157         + 0.185 : (s' = 78);
158     [tick] s = 78 ->
159         1 : (s' = 79);
160     [tick] s = 77 ->
161         1 : (s' = 80);
162     [tick] s = 76 ->
163         1 : (s' = 81);
164     [tick] s = 74 ->
165         0.185 : (s' = 82)
166         + 0.63 : (s' = 83)
167         + 0.185 : (s' = 84);
168     [tick] s = 84 ->
169         1 : (s' = 85);
170     [tick] s = 83 ->
171         1 : (s' = 86);
172     [tick] s = 82 ->
173         1 : (s' = 87);
174     [tick] s = 73 ->
175         0.185 : (s' = 88)
176         + 0.63 : (s' = 89)
177         + 0.185 : (s' = 90);
178     [tick] s = 90 ->
179         1 : (s' = 91);
180     [tick] s = 89 ->
181         1 : (s' = 92);
182     [tick] s = 88 ->
183         1 : (s' = 93);
184     [tick] s = 71 ->
185         0.185 : (s' = 94)
186         + 0.63 : (s' = 95)
187         + 0.185 : (s' = 96);
188     [tick] s = 96 ->
189         0.185 : (s' = 97)
190         + 0.63 : (s' = 98)
191         + 0.185 : (s' = 99);
192     [tick] s = 99 ->
193         1 : (s' = 100);
194     [tick] s = 98 ->
195         1 : (s' = 101);
196     [tick] s = 97 ->
197         1 : (s' = 102);
198     [tick] s = 95 ->
199         0.185 : (s' = 103)
200         + 0.63 : (s' = 104)
201         + 0.185 : (s' = 105);
202     [tick] s = 105 ->
203         1 : (s' = 106);
```

```
[tick] s = 104 ->
      1 : (s' = 107);
[tick] s = 103 ->
      1 : (s' = 108);
[tick] s = 94 ->
      0.185 : (s' = 109)
      + 0.63 : (s' = 110)
      + 0.185 : (s' = 111);
[tick] s = 111 ->
      1 : (s' = 112);
[tick] s = 110 ->
      1 : (s' = 113);
[tick] s = 109 ->
      1 : (s' = 114);
[tick] s = 70 ->
      0.185 : (s' = 115)
      + 0.63 : (s' = 116)
      + 0.185 : (s' = 117);
[tick] s = 117 ->
      0.185 : (s' = 118)
      + 0.63 : (s' = 119)
      + 0.185 : (s' = 120);
[tick] s = 120 ->
      1 : (s' = 121);
[tick] s = 119 ->
      1 : (s' = 122);
[tick] s = 118 ->
      1 : (s' = 123);
[tick] s = 116 ->
      0.185 : (s' = 124)
      + 0.63 : (s' = 125)
      + 0.185 : (s' = 126);
[tick] s = 126 ->
      1 : (s' = 127);
[tick] s = 125 ->
      1 : (s' = 128);
[tick] s = 124 ->
      1 : (s' = 129);
[tick] s = 115 ->
      0.185 : (s' = 130)
      + 0.63 : (s' = 131)
      + 0.185 : (s' = 132);
[tick] s = 132 ->
      1 : (s' = 133);
[tick] s = 131 ->
      1 : (s' = 134);
[tick] s = 130 ->
      1 : (s' = 135);
[tick] s = 1 ->
      0.185 : (s' = 136)
      + 0.63 : (s' = 137)
      + 0.185 : (s' = 138);
[tick] s = 138 ->
      0.185 : (s' = 139)
      + 0.63 : (s' = 140)
      + 0.185 : (s' = 141);
[tick] s = 141 ->
      0.185 : (s' = 142)
      + 0.63 : (s' = 143)
      + 0.185 : (s' = 144);
[tick] s = 144 ->
      1 : (s' = 145);
[tick] s = 143 ->
      1 : (s' = 146);
[tick] s = 142 ->
```

```
        1 : (s' = 147);
[tick] s = 140 ->
        0.185 : (s' = 148)
        + 0.63 : (s' = 149)
        + 0.185 : (s' = 150);
[tick] s = 150 ->
        1 : (s' = 151);
[tick] s = 149 ->
        1 : (s' = 152);
[tick] s = 148 ->
        1 : (s' = 153);
[tick] s = 139 ->
        0.185 : (s' = 154)
        + 0.63 : (s' = 155)
        + 0.185 : (s' = 156);
[tick] s = 156 ->
        1 : (s' = 157);
[tick] s = 155 ->
        1 : (s' = 158);
[tick] s = 154 ->
        1 : (s' = 159);
[tick] s = 137 ->
        0.185 : (s' = 160)
        + 0.63 : (s' = 161)
        + 0.185 : (s' = 162);
[tick] s = 162 ->
        0.185 : (s' = 163)
        + 0.63 : (s' = 164)
        + 0.185 : (s' = 165);
[tick] s = 165 ->
        1 : (s' = 166);
[tick] s = 164 ->
        1 : (s' = 167);
[tick] s = 163 ->
        1 : (s' = 168);
[tick] s = 161 ->
        0.185 : (s' = 169)
        + 0.63 : (s' = 170)
        + 0.185 : (s' = 171);
[tick] s = 171 ->
        1 : (s' = 172);
[tick] s = 170 ->
        1 : (s' = 173);
[tick] s = 169 ->
        1 : (s' = 174);
[tick] s = 160 ->
        0.185 : (s' = 175)
        + 0.63 : (s' = 176)
        + 0.185 : (s' = 177);
[tick] s = 177 ->
        1 : (s' = 178);
[tick] s = 176 ->
        1 : (s' = 179);
[tick] s = 175 ->
        1 : (s' = 180);
[tick] s = 136 ->
        0.185 : (s' = 181)
        + 0.63 : (s' = 182)
        + 0.185 : (s' = 183);
[tick] s = 183 ->
        0.185 : (s' = 184)
        + 0.63 : (s' = 185)
        + 0.185 : (s' = 186);
[tick] s = 186 ->
        1 : (s' = 187);
```

```
[tick] s = 185 ->
    1 : (s' = 188);
[tick] s = 184 ->
    1 : (s' = 189);
[tick] s = 182 ->
    0.185 : (s' = 190)
    + 0.63 : (s' = 191)
    + 0.185 : (s' = 192);
[tick] s = 192 ->
    1 : (s' = 193);
[tick] s = 191 ->
    1 : (s' = 194);
[tick] s = 190 ->
    1 : (s' = 195);
[tick] s = 181 ->
    0.185 : (s' = 196)
    + 0.63 : (s' = 197)
    + 0.185 : (s' = 198);
[tick] s = 198 ->
    1 : (s' = 199);
[tick] s = 197 ->
    1 : (s' = 200);
[tick] s = 196 ->
    1 : (s' = 201);
[tick] (s = 13 | s = 14 | s = 15 | s = 19 | s = 20 | s = 21 | s = 25 | s = 26 | s = 27 | s = 34 | s = 35 | s = 36 | s = 40 | s = 41 | s = 42 |
    s = 46 | s = 47 | s = 48 | s = 55 | s = 56 | s = 57 | s = 61 | s = 62 | s = 63 | s = 67 | s = 68 | s = 69 | s = 79 | s = 80 | s = 81 | s
    = 85 | s = 86 | s = 87 | s = 91 | s = 92 | s = 93 | s = 100 | s = 101 | s = 102 | s = 106 | s = 107 | s = 108 | s = 112 | s = 113 | s
    = 114 | s = 121 | s = 122 | s = 123 | s = 127 | s = 128 | s = 129 | s = 133 | s = 134 | s = 135 | s = 145 | s = 146 | s = 147 | s
    = 151 | s = 152 | s = 153 | s = 157 | s = 158 | s = 159 | s = 166 | s = 167 | s = 168 | s = 172 | s = 173 | s = 174 | s = 178 | s
    = 179 | s = 180 | s = 187 | s = 188 | s = 189 | s = 193 | s = 194 | s = 195 | s = 199 | s = 200 | s = 201) -> 1 : true;
endmodule
formula stateValue = (s = 0 ? 0.0176932 : 0) +
                     (s = 3 ? 0.0214149 : 0) +
                     (s = 6 ? 0.0270787 : 0) +
                     (s = 9 ? 0.0333291 : 0) +
                     (s = 12 ? 0.0407615 : 0) +
                     (s = 13 ? 0.0387364 : 0) +
                     (s = 11 ? 0.0321959 : 0) +
                     (s = 14 ? 0.0311986 : 0) +
                     (s = 10 ? 0.0236303 : 0) +
                     (s = 15 ? 0.0236609 : 0) +
                     (s = 8 ? 0.0266955 : 0) +
                     (s = 18 ? 0.0328878 : 0) +
                     (s = 19 ? 0.0318075 : 0) +
                     (s = 17 ? 0.0263583 : 0) +
                     (s = 20 ? 0.0260615 : 0) +
                     (s = 16 ? 0.0198287 : 0) +
                     (s = 21 ? 0.0203155 : 0) +
                     (s = 7 ? 0.0200618 : 0) +
                     (s = 24 ? 0.0256958 : 0) +
                     (s = 25 ? 0.0254786 : 0) +
                     (s = 23 ? 0.0205206 : 0) +
                     (s = 26 ? 0.0209244 : 0) +
                     (s = 22 ? 0.0153454 : 0) +
                     (s = 27 ? 0.0163702 : 0) +
                     (s = 5 ? 0.0217113 : 0) +
                     (s = 30 ? 0.0273884 : 0) +
                     (s = 33 ? 0.0336873 : 0) +
                     (s = 34 ? 0.0325111 : 0) +
                     (s = 32 ? 0.026968 : 0) +
                     (s = 35 ? 0.0265981 : 0) +
                     (s = 31 ? 0.0202487 : 0) +
                     (s = 36 ? 0.0206851 : 0) +
                     (s = 29 ? 0.0219722 : 0) +
                     (s = 39 ? 0.0276628 : 0) +
```

```
(s = 40 ? 0.0272095 : 0) +
(s = 38 ? 0.0222018 : 0) +
(s = 41 ? 0.0224038 : 0) +
(s = 37 ? 0.0167408 : 0) +
(s = 42 ? 0.0175982 : 0) +
(s = 28 ? 0.0165561 : 0) +
(s = 45 ? 0.0223769 : 0) +
(s = 46 ? 0.0225579 : 0) +
(s = 44 ? 0.0174356 : 0) +
(s = 47 ? 0.0182096 : 0) +
(s = 43 ? 0.0124943 : 0) +
(s = 48 ? 0.0138613 : 0) +
(s = 4 ? 0.016344 : 0) +
(s = 51 ? 0.0221892 : 0) +
(s = 54 ? 0.0278924 : 0) +
(s = 55 ? 0.0274116 : 0) +
(s = 53 ? 0.0223928 : 0) +
(s = 56 ? 0.0225719 : 0) +
(s = 52 ? 0.0168932 : 0) +
(s = 57 ? 0.0177322 : 0) +
(s = 50 ? 0.017249 : 0) +
(s = 60 ? 0.0230006 : 0) +
(s = 61 ? 0.0231068 : 0) +
(s = 59 ? 0.0180454 : 0) +
(s = 62 ? 0.0187462 : 0) +
(s = 58 ? 0.0130901 : 0) +
(s = 63 ? 0.0143856 : 0) +
(s = 49 ? 0.0123087 : 0) +
(s = 66 ? 0.0189053 : 0) +
(s = 67 ? 0.0195029 : 0) +
(s = 65 ? 0.0136979 : 0) +
(s = 68 ? 0.0149204 : 0) +
(s = 64 ? 0.00849053 : 0) +
(s = 69 ? 0.0103379 : 0) +
(s = 2 ? 0.0163126 : 0) +
(s = 72 ? 0.0221616 : 0) +
(s = 75 ? 0.0278631 : 0) +
(s = 78 ? 0.0342389 : 0) +
(s = 79 ? 0.0329965 : 0) +
(s = 77 ? 0.0273858 : 0) +
(s = 80 ? 0.0269657 : 0) +
(s = 76 ? 0.0205326 : 0) +
(s = 81 ? 0.0209349 : 0) +
(s = 74 ? 0.0223684 : 0) +
(s = 84 ? 0.0280829 : 0) +
(s = 85 ? 0.0275792 : 0) +
(s = 83 ? 0.0225505 : 0) +
(s = 86 ? 0.0227107 : 0) +
(s = 82 ? 0.0170181 : 0) +
(s = 87 ? 0.0178422 : 0) +
(s = 73 ? 0.0168738 : 0) +
(s = 90 ? 0.0226608 : 0) +
(s = 91 ? 0.0228078 : 0) +
(s = 89 ? 0.0177152 : 0) +
(s = 92 ? 0.0184556 : 0) +
(s = 88 ? 0.0127696 : 0) +
(s = 93 ? 0.0141035 : 0) +
(s = 71 ? 0.0172213 : 0) +
(s = 96 ? 0.0229754 : 0) +
(s = 99 ? 0.0287337 : 0) +
(s = 100 ? 0.0281519 : 0) +
(s = 98 ? 0.0230846 : 0) +
(s = 101 ? 0.0231807 : 0) +
(s = 97 ? 0.0174355 : 0) +
(s = 102 ? 0.0182095 : 0) +
```

```
(s = 95 ? 0.018021 : 0) +
(s = 105 ? 0.0237147 : 0) +
(s = 106 ? 0.0237352 : 0) +
(s = 104 ? 0.0187248 : 0) +
(s = 107 ? 0.019344 : 0) +
(s = 103 ? 0.0137348 : 0) +
(s = 108 ? 0.0149529 : 0) +
(s = 94 ? 0.0130667 : 0) +
(s = 111 ? 0.0194823 : 0) +
(s = 112 ? 0.0200107 : 0) +
(s = 110 ? 0.0143649 : 0) +
(s = 113 ? 0.0155074 : 0) +
(s = 109 ? 0.00924753 : 0) +
(s = 114 ? 0.0110041 : 0) +
(s = 70 ? 0.0122811 : 0) +
(s = 117 ? 0.0188846 : 0) +
(s = 120 ? 0.0245369 : 0) +
(s = 121 ? 0.0244587 : 0) +
(s = 119 ? 0.0194847 : 0) +
(s = 122 ? 0.0200128 : 0) +
(s = 118 ? 0.0144325 : 0) +
(s = 123 ? 0.0155668 : 0) +
(s = 116 ? 0.0136736 : 0) +
(s = 126 ? 0.0199571 : 0) +
(s = 127 ? 0.0204285 : 0) +
(s = 125 ? 0.014899 : 0) +
(s = 128 ? 0.0159774 : 0) +
(s = 124 ? 0.009841 : 0) +
(s = 129 ? 0.0115263 : 0) +
(s = 115 ? 0.00846263 : 0) +
(s = 132 ? 0.0162147 : 0) +
(s = 133 ? 0.0171352 : 0) +
(s = 131 ? 0.0103134 : 0) +
(s = 134 ? 0.011942 : 0) +
(s = 130 ? 0.00441199 : 0) +
(s = 135 ? 0.0067488 : 0) +
(s = 1 ? 0.0112104 : 0) +
(s = 138 ? 0.0180987 : 0) +
(s = 141 ? 0.0237876 : 0) +
(s = 144 ? 0.0296178 : 0) +
(s = 145 ? 0.0289299 : 0) +
(s = 143 ? 0.0237994 : 0) +
(s = 146 ? 0.0238097 : 0) +
(s = 142 ? 0.0179809 : 0) +
(s = 147 ? 0.0186894 : 0) +
(s = 140 ? 0.0187931 : 0) +
(s = 150 ? 0.0244486 : 0) +
(s = 151 ? 0.024381 : 0) +
(s = 149 ? 0.0194042 : 0) +
(s = 152 ? 0.0199419 : 0) +
(s = 148 ? 0.0143597 : 0) +
(s = 153 ? 0.0155028 : 0) +
(s = 139 ? 0.0137985 : 0) +
(s = 156 ? 0.0200562 : 0) +
(s = 157 ? 0.0205157 : 0) +
(s = 155 ? 0.015009 : 0) +
(s = 158 ? 0.0160741 : 0) +
(s = 154 ? 0.0099617 : 0) +
(s = 159 ? 0.0116325 : 0) +
(s = 137 ? 0.0127314 : 0) +
(s = 162 ? 0.0192249 : 0) +
(s = 165 ? 0.0248675 : 0) +
(s = 166 ? 0.0247496 : 0) +
(s = 164 ? 0.0197842 : 0) +
(s = 167 ? 0.0202763 : 0) +
```

```
                    (s = 163 ? 0.0147008 : 0) +
                    (s = 168 ? 0.015803 : 0) +
                    (s = 161 ? 0.0140698 : 0) +
                    (s = 171 ? 0.0202733 : 0) +
                    (s = 172 ? 0.0207067 : 0) +
                    (s = 170 ? 0.0152477 : 0) +
                    (s = 173 ? 0.0162842 : 0) +
                    (s = 169 ? 0.0102221 : 0) +
                    (s = 174 ? 0.0118617 : 0) +
                    (s = 160 ? 0.00891477 : 0) +
                    (s = 177 ? 0.016513 : 0) +
                    (s = 178 ? 0.0173977 : 0) +
                    (s = 176 ? 0.0107112 : 0) +
                    (s = 179 ? 0.0122921 : 0) +
                    (s = 175 ? 0.00490948 : 0) +
                    (s = 180 ? 0.00718659 : 0) +
                    (s = 136 ? 0.00736404 : 0) +
                    (s = 183 ? 0.0155067 : 0) +
                    (s = 186 ? 0.0214635 : 0) +
                    (s = 187 ? 0.0217541 : 0) +
                    (s = 185 ? 0.0165121 : 0) +
                    (s = 188 ? 0.0173969 : 0) +
                    (s = 184 ? 0.0115608 : 0) +
                    (s = 189 ? 0.0130397 : 0) +
                    (s = 182 ? 0.0093466 : 0) +
                    (s = 192 ? 0.0168019 : 0) +
                    (s = 193 ? 0.0176519 : 0) +
                    (s = 191 ? 0.0110913 : 0) +
                    (s = 194 ? 0.0126265 : 0) +
                    (s = 190 ? 0.0053806 : 0) +
                    (s = 195 ? 0.00760117 : 0) +
                    (s = 181 ? 0.0031865 : 0) +
                    (s = 198 ? 0.0129875 : 0) +
                    (s = 199 ? 0.0142952 : 0) +
                    (s = 197 ? 0.00567036 : 0) +
                    (s = 200 ? 0.00785617 : 0) +
                    (s = 196 ? 0 : 0) +
                    (s = 201 ? 0.00286625 : 0);


module clk
    time : [0..HORIZON + 1] init 0;
    readyToTick : bool init true;
    [tick] readyToTick & time < HORIZON + 1 -> 1 : (time' = time + 1) & (readyToTick'=false);
    [tack] !readyToTick -> 1 : (readyToTick'=true);
endmodule

label "final" = time = HORIZON + 1;
formula sys_go = readyToTick;

module controller
    active_servers_A : [0..MAX_SERVERS_A] init ini_servers_A;
    active_servers_B : [0..MAX_SERVERS_B] init ini_servers_B;
    active_servers_C : [0..MAX_SERVERS_C] init ini_servers_C;

    dimmer : [1..DIMMER_LEVELS] init ini_dimmer;

    traffic_A : [0..4] init ini_traffic_A;
    traffic_B : [0..4] init ini_traffic_B;
    traffic_C : [0..4] init ini_traffic_C;

    [addServerA_complete] active_servers_A < MAX_SERVERS_A -> 1 : (active_servers_A' = active_servers_A + 1);
    [addServerB_complete] active_servers_B < MAX_SERVERS_B -> 1 : (active_servers_B' = active_servers_B + 1);
    [addServerC_complete] active_servers_C < MAX_SERVERS_C -> 1 : (active_servers_C' = active_servers_C + 1);

```

```
    [removeServerA_start] active_servers_A > 0 -> 1 : (active_servers_A' = active_servers_A - 1);
    [removeServerB_start] active_servers_B > 0 -> 1 : (active_servers_B' = active_servers_B - 1);
    [removeServerC_start] active_servers_C > 0 -> 1 : (active_servers_C' = active_servers_C - 1);

    [increaseDimmer_start] dimmer < DIMMER_LEVELS -> 1 : (dimmer' = dimmer + 1);
    [decreaseDimmer_start] dimmer > 1 -> 1 : (dimmer' = dimmer - 1);

    //A-B-C
    //Possible values 0-25-50-75-100

    // 100-0-0
    [divert_100_0_0] active_servers_A > 0
        -> 1 : (traffic_A' = 4) & (traffic_B' = 0) & (traffic_C' = 0);
    // 75-25-0
    [divert_75_25_0] active_servers_A > 0 & active_servers_B > 0
        -> 1 : (traffic_A' = 3) & (traffic_B' = 1) & (traffic_C' = 0);
    // 75-0-25
    [divert_75_0_25] active_servers_A > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 3) & (traffic_B' = 0) & (traffic_C' = 1);
    // 50-50-0
    [divert_50_50_0] active_servers_A > 0 & active_servers_B > 0
        -> 1 : (traffic_A' = 2) & (traffic_B' = 2) & (traffic_C' = 0);
    // 50-0-50
    [divert_50_0_50] active_servers_A > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 2) & (traffic_B' = 0) & (traffic_C' = 2);
    // 50-25-25
    [divert_50_25_25] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 2) & (traffic_B' = 1) & (traffic_C' = 1);
    // 25-75-0
    [divert_25_75_0] active_servers_A > 0 & active_servers_B > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 3) & (traffic_C' = 0);
    // 25-0-75
    [divert_25_0_75] active_servers_A > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 0) & (traffic_C' = 3);
    // 25-50-25
    [divert_25_50_25] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 2) & (traffic_C' = 1);
    // 25-25-50
    [divert_25_25_50] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 1) & (traffic_C' = 2);
    // 0-100-0
    [divert_0_100_0] active_servers_B > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 4) & (traffic_C' = 0);
    // 0-0-100
    [divert_0_0_100] active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 0) & (traffic_C' = 4);
    // 0-75-25
    [divert_0_75_25] active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 3) & (traffic_C' = 1);
    // 0-25-75
    [divert_0_25_75] active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 1) & (traffic_C' = 3);
    // 0-50-50
    [divert_0_50_50] active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 2) & (traffic_C' = 2);
endmodule


formula addServerA_applicable = active_servers_A < MAX_SERVERS_A & !removeServer_used
                & addServerB_state = 0 & addServerC_state = 0;
formula addServerB_applicable = active_servers_B < MAX_SERVERS_B & !removeServer_used
                & addServerA_state = 0 & addServerC_state = 0;
formula addServerC_applicable = active_servers_C < MAX_SERVERS_C & !removeServer_used
                & addServerA_state = 0 & addServerB_state = 0;

```

```
formula removeServerA_applicable = active_servers_A > 0 & addServerA_state = 0 & active_servers > 1
                          & addServerB_state = 0 & addServerC_state = 0;
formula removeServerB_applicable = active_servers_B > 0 & addServerB_state = 0 & active_servers > 1
                         & addServerA_state = 0 & addServerC_state = 0;
formula removeServerC_applicable = active_servers_C > 0 & addServerC_state = 0 & active_servers > 1
                         & addServerA_state = 0 & addServerB_state = 0;

formula increaseDimmer_compatible = !decreaseDimmer_used;
formula decreaseDimmer_compatible = !increaseDimmer_used;
formula increase_dimmer_applicable = dimmer < DIMMER_LEVELS & increaseDimmer_compatible;
formula decrease_dimmer_applicable = dimmer > 1 & decreaseDimmer_compatible;

const int addServer_LATENCY_PERIODS = ceil(addServer_LATENCY / PERIOD);

// This remove server constraints that only one server could be removed in one monitoring cycle.
module removeServer
    removeServer_go : bool init true;
    removeServer_used : bool init false;

    [removeServerA_start] sys_go & removeServer_go
        & removeServerA_applicable // applicability conditions
        -> (removeServer_go' = false) & (removeServer_used' = true);

    [removeServerB_start] sys_go & removeServer_go
        & removeServerB_applicable // applicability conditions
        -> (removeServer_go' = false) & (removeServer_used' = true);

    [removeServerC_start] sys_go & removeServer_go
        & removeServerC_applicable // applicability conditions
        -> (removeServer_go' = false) & (removeServer_used' = true);

    // Case when remove server tactic is applicable but not used
    [pass_remove_server] sys_go & removeServer_go // can go
        -> (removeServer_go' = false);

    [tick] !removeServer_go -> 1 : (removeServer_go' = true) & (removeServer_used' = false);
endmodule

module addServer
    addServerA_state : [0..addServer_LATENCY_PERIODS] init ini_addServerA_state;
    addServerB_state : [0..addServer_LATENCY_PERIODS] init ini_addServerB_state;
    addServerC_state : [0..addServer_LATENCY_PERIODS] init ini_addServerC_state;

    addServer_go : bool init true;

    // tactic applicable, start it
    [addServerA_start] sys_go & addServer_go // can go
        & addServerA_state = 0 // tactic has not been started
        & addServerA_applicable
        -> (addServerA_state' = 1) & (addServer_go' = false);

    // tactic applicable, start it
    [addServerB_start] sys_go & addServer_go // can go
        & addServerB_state = 0 // tactic has not been started
        & addServerB_applicable
        -> (addServerB_state' = 1) & (addServer_go' = false);

    // tactic applicable, start it
    [addServerC_start] sys_go & addServer_go // can go
        & addServerC_state = 0 // tactic has not been started
        & addServerC_applicable
        -> (addServerC_state' = 1) & (addServer_go' = false);

    // tactic applicable, but don't use it
    [pass_add] sys_go & addServer_go // can go
```

```
719             & addServerA_state = 0 // tactic has not been started
720             & addServerB_state = 0 & addServerC_state = 0
721             //& addServerA_applicable
722             -> (addServer_go' = false);
723
724     // progress of the tactic
725     [progressA] sys_go & addServer_go
726             & addServerA_state > 0 & addServerA_state < addServer_LATENCY_PERIODS
727             -> 1 : (addServerA_state' = addServerA_state + 1) & (addServer_go' = false);
728
729     [progressB] sys_go & addServer_go
730             & addServerB_state > 0 & addServerB_state < addServer_LATENCY_PERIODS
731             -> 1 : (addServerB_state' = addServerB_state + 1) & (addServer_go' = false);
732
733     [progressC] sys_go & addServer_go
734             & addServerC_state > 0 & addServerC_state < addServer_LATENCY_PERIODS
735             -> 1 : (addServerC_state' = addServerC_state + 1) & (addServer_go' = false);
736
737     // completion of the tactic
738     [addServerA_complete] sys_go & addServer_go
739             & addServerA_state = addServer_LATENCY_PERIODS // completed
740             -> 1 : (addServerA_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed
741
742     [addServerB_complete] sys_go & addServer_go
743             & addServerB_state = addServer_LATENCY_PERIODS // completed
744             -> 1 : (addServerB_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed
745
746     [addServerC_complete] sys_go & addServer_go
747             & addServerC_state = addServer_LATENCY_PERIODS // completed
748             -> 1 : (addServerC_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed
749
750     [tick] !addServer_go -> 1 : (addServer_go' = true);
751 endmodule
752
753 // Make sure that divert traffic is executed at the end i.e.after adding or removing the servers.
754 formula divert_traffic_applicable = divert_go & !addServer_go & !removeServer_go & !increaseDimmer_go &
        !decreaseDimmer_go;
755
756 module divert_traffic
757     divert_go : bool init true;
758
759     //A-B-C
760     //Possible values 0-25-50-75-100
761
762     // 100-0-0
763     [divert_100_0_0] sys_go & divert_traffic_applicable
764             -> 1 : (divert_go'=false);
765     // 75-25-0
766     [divert_75_25_0] sys_go & divert_traffic_applicable
767             -> 1 : (divert_go'=false);
768     // 75-0-25
769     [divert_75_0_25] sys_go & divert_traffic_applicable
770             -> 1 : (divert_go'=false);
771     // 50-50-0
772     [divert_50_50_0] sys_go & divert_traffic_applicable
773             -> 1 : (divert_go'=false);
774     // 50-0-50
775     [divert_50_0_50] sys_go & divert_traffic_applicable
776             -> 1 : (divert_go'=false);
777     // 50-25-25
778     [divert_50_25_25] sys_go & divert_traffic_applicable
779             -> 1 : (divert_go'=false);
780     // 25-75-0
781     [divert_25_75_0] sys_go & divert_traffic_applicable
782             -> 1 : (divert_go'=false);
```

```
783     // 25-0-75
784     [divert_25_0_75] sys_go & divert_traffic_applicable
785         -> 1 : (divert_go'=false);
786     // 25-50-25
787     [divert_25_50_25] sys_go & divert_traffic_applicable
788         -> 1 : (divert_go'=false);
789     // 25-25-50
790     [divert_25_25_50] sys_go & divert_traffic_applicable
791         -> 1 : (divert_go'=false);
792     // 0-100-0
793     [divert_0_100_0] sys_go & divert_traffic_applicable
794         -> 1 : (divert_go'=false);
795     // 0-0-100
796     [divert_0_0_100] sys_go & divert_traffic_applicable
797         -> 1 : (divert_go'=false);
798     // 0-75-25
799     [divert_0_75_25] sys_go & divert_traffic_applicable
800         -> 1 : (divert_go'=false);
801     // 0-25-75
802     [divert_0_25_75] sys_go & divert_traffic_applicable
803         -> 1 : (divert_go'=false);
804     // 0-50-50
805     [divert_0_50_50] sys_go & divert_traffic_applicable
806         -> 1 : (divert_go'=false);
807
808     [tick] !divert_go -> 1 : (divert_go' = true);
809 endmodule
810
811 module increaseDimmer
812     increaseDimmer_go : bool init true;
813     increaseDimmer_used : bool init false;
814
815     [increaseDimmer_start] sys_go & increaseDimmer_go
816         & increase_dimmer_applicable // applicability conditions
817         -> (increaseDimmer_go' = false) & (increaseDimmer_used' = true);
818
819     // tactic applicable but not used
820     [pass_inc_dimmer] sys_go & increaseDimmer_go // can go
821         -> (increaseDimmer_go' = false);
822
823     [tick] !increaseDimmer_go -> 1 : (increaseDimmer_go' = true) & (increaseDimmer_used' = false);
824 endmodule
825
826 // tactic
827 module decreaseDimmer
828     decreaseDimmer_go : bool init true;
829     decreaseDimmer_used : bool init false;
830
831     [decreaseDimmer_start] sys_go & decreaseDimmer_go
832         & decrease_dimmer_applicable // applicability conditions
833         -> (decreaseDimmer_go' = false) & (decreaseDimmer_used' = true);
834
835     // tactic applicable but not used
836     [pass_dec_dimmer] sys_go & decreaseDimmer_go // can go
837         -> (decreaseDimmer_go' = false);
838
839     [tick] !decreaseDimmer_go -> 1 : (decreaseDimmer_go' = true) & (decreaseDimmer_used' = false);
840 endmodule
841
842 //*************************************************************************
843 // Queuing network with each server having queueing model of M/G/1/PS
844 //*************************************************************************
845 formula dimmerFactor = (dimmer - 1) / (DIMMER_LEVELS - 1);
846 formula interarrivalMean = stateValue * interArrivalScaleFactorForDecision;
847
```

```
848 formula Pa = (traffic_A * 25)/100;
849 formula Pb = (traffic_B * 25)/100;
850 formula Pc = (traffic_C * 25)/100;
851
852 formula loaded_servers = (Pa != 0 ? 1 : 0) + (Pb != 0 ? 1 : 0) + (Pc != 0 ? 1 : 0);
853
854 formula service_rate_A = dimmerFactor * (MAX_ARRIVALA_CAPACITY_LOW)
855             + (1 - dimmerFactor) * (MAX_ARRIVALA_CAPACITY);
856 formula service_rate_B = dimmerFactor * (MAX_ARRIVALB_CAPACITY_LOW)
857             + (1 - dimmerFactor) * (MAX_ARRIVALB_CAPACITY);
858 formula service_rate_C = dimmerFactor * (MAX_ARRIVALC_CAPACITY_LOW)
859             + (1 - dimmerFactor) * (MAX_ARRIVALC_CAPACITY);
860
861 formula rhoA = Pa/(service_rate_A*interarrivalMean);
862 formula rhoB = Pb/(service_rate_B*interarrivalMean);
863 formula rhoC = Pc/(service_rate_C*interarrivalMean);
864
865 formula overloaded = (rhoA >= 1 | rhoB >= 1 | rhoC >= 1);
866
867 formula rt_A = 1/(service_rate_A - (throughput*Pa));
868 formula rt_B = 1/(service_rate_B - (throughput*Pb));
869 formula rt_C = 1/(service_rate_C - (throughput*Pc));
870
871 // Response time to clients utility function
872 const double RT_THRESHOLD = 1.0;
873
874 formula expected_wait_time = (Pa*rt_A + Pb*rt_B + Pc*rt_C);
875 formula rt = (interarrivalMean = 0 ? 0 : (overloaded ? RT_THRESHOLD + 2 : expected_wait_time));
876
877 const double NORMAL_A_REVENUE = (SERVERA_COST / MAX_ARRIVALA_CAPACITY) * 10;
878 const double DIMMER_A_REVENUE = (SERVERA_COST / MAX_ARRIVALA_CAPACITY_LOW) * 3 / 2;
879 const double NORMAL_B_REVENUE = (SERVERB_COST / MAX_ARRIVALB_CAPACITY) * 10;
880 const double DIMMER_B_REVENUE = (SERVERB_COST / MAX_ARRIVALB_CAPACITY_LOW) * 3 / 2;
881 const double NORMAL_C_REVENUE = (SERVERC_COST / MAX_ARRIVALC_CAPACITY) * 10;
882 const double DIMMER_C_REVENUE = (SERVERC_COST / MAX_ARRIVALC_CAPACITY_LOW) * 3 / 2;
883
884 const double DIMMER_REVENUE = DIMMER_A_REVENUE + DIMMER_B_REVENUE + DIMMER_C_REVENUE;
885 const double NORMAL_REVENUE = NORMAL_A_REVENUE + NORMAL_B_REVENUE + NORMAL_C_REVENUE;
886
887 formula serverA_cost = ((addServerA_state > 0 ? 1 : 0) + active_servers_A) * SERVERA_COST;
888 formula serverB_cost = ((addServerB_state > 0 ? 1 : 0) + active_servers_B) * SERVERB_COST;
889 formula serverC_cost = ((addServerC_state > 0 ? 1 : 0) + active_servers_C) * SERVERC_COST;
890 formula cost = serverA_cost + serverB_cost + serverC_cost;
891
892 formula throughput = 1/interarrivalMean;
893
894 formula basicUtilityA = throughput * Pa * (dimmerFactor * DIMMER_A_REVENUE + (1 - dimmerFactor) *
        NORMAL_A_REVENUE);
895 formula basicUtilityB = throughput * Pb * (dimmerFactor * DIMMER_B_REVENUE + (1 - dimmerFactor) *
        NORMAL_B_REVENUE);
896 formula basicUtilityC = throughput * Pc * (dimmerFactor * DIMMER_C_REVENUE + (1 - dimmerFactor) *
        NORMAL_C_REVENUE);
897
898 formula basicUtility = basicUtilityA + basicUtilityB + basicUtilityC;
899 formula active_servers = active_servers_A + active_servers_B + active_servers_C;
900 formula poweredServers = (addServerA_state > 0 ? 1 : 0) + (addServerB_state > 0 ? 1 : 0) + (addServerC_state > 0 ? 1 : 0 )
901       + active_servers;
902 formula MAX_SERVERS = MAX_SERVERS_A + MAX_SERVERS_B + MAX_SERVERS_C;
903
904 formula MAX_SERVER_COST = MAX_SERVERS_A * SERVERA_COST
905             + MAX_SERVERS_B * SERVERB_COST
906             + MAX_SERVERS_C * SERVERC_COST;
907
908 formula netPenalty = stateValue > 0 ? penalty / stateValue : 0;
909
```

```
910 formula uTotal = (overloaded & (poweredServers < MAX_SERVERS | dimmer < DIMMER_LEVELS | active_servers !=
        loaded_servers))
911                         ? −(1000) // avoid unstable solutions
912                     : (((rt > RT_THRESHOLD | rt <= 0) ? netPenalty : basicUtility) − cost);
913
914
915 rewards "util"
916     // 100000000.0 is added to avoid a negative value during calculation; negative utility is not supported by PRISM.
917     [tack] true : 100000000.0 + (PERIOD)*(uTotal);
918 endrewards
```

Listing 2: PRISM specification for MDP planning

PRISM Planning Specifications for the Team of UAVs

For a particular situation of the system and environment, this section provides the PRISM planning specifications for non-wait reactive (i.e., MDP planning with a shorter horizon and a subset of actions compared to MDP planning used for deliberative planning) and deliberative planning used for the team of UAVs. The reactive and deliberative planning specifications have the same initial state.

Short Horizon MDP Planning Specification

Reactive planning ρ_{mdps} plans with a shorter horizon compared to deliberative planning ρ_{mdpl}. Moreover, while planning, ρ_{mdps} do not consider adaptation actions `IncAlt`, `DecAlt`, and `EcmOn`, and `EcmOff`.

```
mdp
const double PERIOD = 60;
const int HORIZON = 2; // Planning horizon for reactive planning
const double IncAlt_LATENCY = 60;
const double DecAlt_LATENCY = 60;
const int MAX_ALT_LEVEL = 3;
const double destructionFormationFactor = 1.5;
const double threatRange = 3;
const double detectionFormationFactor = 1.2;
const double sensorRange = 4;
const init_a = 0;
const init_c = 0;
const init_f = 0;
const bool ECM_ENABLED = false; // ECM is not enabled for reactive planning
const bool ONE_LEVEL_ENABLED = false; // This is not enabled for reactive planning
const bool TWO_LEVEL_ENABLED = true; // Two level increase/decrease altitude enabled
const int ini_IncAlt_state = 0;
const int ini_DecAlt_state = 0;
const int ini_IncAlt2_state = 0;
const int ini_DecAlt2_state = 0;
const double ecm_threat_prob = 0.15;
const double ecm_target_prob = 0.3;
const double survival_reward = 1;


//*************************************************************************
// CLOCK
//*************************************************************************
const int TO_TICK = 0;
const int TO_TICK2 = 1; // intermediate tick for constraint satisf. update
const int TO_TACK = 2;
```

```
32
33 label "final" = time = HORIZON & clockstep=TO_TICK;
34 formula sys_go = clockstep=TO_TICK;
35
36 module clk
37     time : [0..HORIZON] init 0;
38     clockstep : [0..2] init TO_TICK;
39
40     [tick] clockstep=TO_TICK & time < HORIZON -> 1: (time'=time+1) & (clockstep'=TO_TICK2);
41     [tick2] clockstep=TO_TICK2 -> 1 : (clockstep'=TO_TACK);
42     [tack] clockstep=TO_TACK -> 1: (clockstep'=TO_TICK);
43 endmodule
44
45 module env
46 s : [0..45] init 0;
47 [tick] s = 0 ->
48     0.034225 : (s' = 1)
49     + 0.11655 : (s' = 2)
50     + 0.034225 : (s' = 3)
51     + 0.11655 : (s' = 4)
52     + 0.3969 : (s' = 5)
53     + 0.11655 : (s' = 6)
54     + 0.034225 : (s' = 7)
55     + 0.11655 : (s' = 8)
56     + 0.034225 : (s' = 9);
57 [tick] s = 1 ->
58     0.034225 : (s' = 10)
59     + 0.11655 : (s' = 11)
60     + 0.034225 : (s' = 12)
61     + 0.11655 : (s' = 13)
62     + 0.3969 : (s' = 14)
63     + 0.11655 : (s' = 15)
64     + 0.034225 : (s' = 16)
65     + 0.11655 : (s' = 17)
66     + 0.034225 : (s' = 18);
67 [tick] s = 2 ->
68     0.034225 : (s' = 10)
69     + 0.11655 : (s' = 11)
70     + 0.034225 : (s' = 12)
71     + 0.11655 : (s' = 13)
72     + 0.3969 : (s' = 14)
73     + 0.11655 : (s' = 15)
74     + 0.034225 : (s' = 16)
75     + 0.11655 : (s' = 17)
76     + 0.034225 : (s' = 18);
77 [tick] s = 3 ->
78     0.034225 : (s' = 10)
79     + 0.11655 : (s' = 11)
80     + 0.034225 : (s' = 12)
81     + 0.11655 : (s' = 13)
82     + 0.3969 : (s' = 14)
83     + 0.11655 : (s' = 15)
84     + 0.034225 : (s' = 16)
85     + 0.11655 : (s' = 17)
86     + 0.034225 : (s' = 18);
87 [tick] s = 4 ->
88     0.034225 : (s' = 10)
89     + 0.11655 : (s' = 11)
90     + 0.034225 : (s' = 12)
91     + 0.11655 : (s' = 13)
92     + 0.3969 : (s' = 14)
93     + 0.11655 : (s' = 15)
94     + 0.034225 : (s' = 16)
95     + 0.11655 : (s' = 17)
96     + 0.034225 : (s' = 18);
```

```
[tick] s = 5 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 6 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 7 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 8 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 9 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 10 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 11 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
```

```
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 12 ->
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 13 ->
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 14 ->
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 15 ->
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 16 ->
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 17 ->
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
```

```
[tick] s = 18 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 19 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 20 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 21 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 22 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 23 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 24 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
```

```
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 25 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 26 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 27 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 28 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 29 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 30 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
```

```
[tick] s = 31 –>
    0.034225 : (s’ = 37)
    + 0.11655 : (s’ = 38)
    + 0.034225 : (s’ = 39)
    + 0.11655 : (s’ = 40)
    + 0.3969 : (s’ = 41)
    + 0.11655 : (s’ = 42)
    + 0.034225 : (s’ = 43)
    + 0.11655 : (s’ = 44)
    + 0.034225 : (s’ = 45);
[tick] s = 32 –>
    0.034225 : (s’ = 37)
    + 0.11655 : (s’ = 38)
    + 0.034225 : (s’ = 39)
    + 0.11655 : (s’ = 40)
    + 0.3969 : (s’ = 41)
    + 0.11655 : (s’ = 42)
    + 0.034225 : (s’ = 43)
    + 0.11655 : (s’ = 44)
    + 0.034225 : (s’ = 45);
[tick] s = 33 –>
    0.034225 : (s’ = 37)
    + 0.11655 : (s’ = 38)
    + 0.034225 : (s’ = 39)
    + 0.11655 : (s’ = 40)
    + 0.3969 : (s’ = 41)
    + 0.11655 : (s’ = 42)
    + 0.034225 : (s’ = 43)
    + 0.11655 : (s’ = 44)
    + 0.034225 : (s’ = 45);
[tick] s = 34 –>
    0.034225 : (s’ = 37)
    + 0.11655 : (s’ = 38)
    + 0.034225 : (s’ = 39)
    + 0.11655 : (s’ = 40)
    + 0.3969 : (s’ = 41)
    + 0.11655 : (s’ = 42)
    + 0.034225 : (s’ = 43)
    + 0.11655 : (s’ = 44)
    + 0.034225 : (s’ = 45);
[tick] s = 35 –>
    0.034225 : (s’ = 37)
    + 0.11655 : (s’ = 38)
    + 0.034225 : (s’ = 39)
    + 0.11655 : (s’ = 40)
    + 0.3969 : (s’ = 41)
    + 0.11655 : (s’ = 42)
    + 0.034225 : (s’ = 43)
    + 0.11655 : (s’ = 44)
    + 0.034225 : (s’ = 45);
[tick] s = 36 –>
    0.034225 : (s’ = 37)
    + 0.11655 : (s’ = 38)
    + 0.034225 : (s’ = 39)
    + 0.11655 : (s’ = 40)
    + 0.3969 : (s’ = 41)
    + 0.11655 : (s’ = 42)
    + 0.034225 : (s’ = 43)
    + 0.11655 : (s’ = 44)
    + 0.034225 : (s’ = 45);
endmodule

// environment has 2 components. statevalue for threats and statevalue1 is for targets
formula stateValue = (s = 0 ? 0 : 0) +
                     (s = 1 ? 0.00605639 : 0) +
```

```
                    (s = 2 ? 0.00605639 : 0) +
                    (s = 3 ? 0.00605639 : 0) +
                    (s = 4 ? 0.0282836 : 0) +
                    (s = 5 ? 0.0282836 : 0) +
                    (s = 6 ? 0.0282836 : 0) +
                    (s = 7 ? 0.0778979 : 0) +
                    (s = 8 ? 0.0778979 : 0) +
                    (s = 9 ? 0.0778979 : 0) +
                    (s = 10 ? 0 : 0) +
                    (s = 11 ? 0 : 0) +
                    (s = 12 ? 0 : 0) +
                    (s = 13 ? 0 : 0) +
                    (s = 14 ? 0 : 0) +
                    (s = 15 ? 0 : 0) +
                    (s = 16 ? 0 : 0) +
                    (s = 17 ? 0 : 0) +
                    (s = 18 ? 0 : 0) +
                    (s = 19 ? 0.750751 : 0) +
                    (s = 20 ? 0.750751 : 0) +
                    (s = 21 ? 0.750751 : 0) +
                    (s = 22 ? 0.885424 : 0) +
                    (s = 23 ? 0.885424 : 0) +
                    (s = 24 ? 0.885424 : 0) +
                    (s = 25 ? 0.963485 : 0) +
                    (s = 26 ? 0.963485 : 0) +
                    (s = 27 ? 0.963485 : 0) +
                    (s = 28 ? 0.435626 : 0) +
                    (s = 29 ? 0.435626 : 0) +
                    (s = 30 ? 0.435626 : 0) +
                    (s = 31 ? 0.676196 : 0) +
                    (s = 32 ? 0.676196 : 0) +
                    (s = 33 ? 0.676196 : 0) +
                    (s = 34 ? 0.864925 : 0) +
                    (s = 35 ? 0.864925 : 0) +
                    (s = 36 ? 0.864925 : 0) +
                    (s = 37 ? 0.368403 : 0) +
                    (s = 38 ? 0.368403 : 0) +
                    (s = 39 ? 0.368403 : 0) +
                    (s = 40 ? 0.793701 : 0) +
                    (s = 41 ? 0.793701 : 0) +
                    (s = 42 ? 0.793701 : 0) +
                    (s = 43 ? 0.983048 : 0) +
                    (s = 44 ? 0.983048 : 0) +
                    (s = 45 ? 0.983048 : 0);

formula stateValue1 = (s = 0 ? 0 : 0) +
                    (s = 1 ? 0.830037 : 0) +
                    (s = 2 ? 0.904439 : 0) +
                    (s = 3 ? 0.954777 : 0) +
                    (s = 4 ? 0.830037 : 0) +
                    (s = 5 ? 0.904439 : 0) +
                    (s = 6 ? 0.954777 : 0) +
                    (s = 7 ? 0.830037 : 0) +
                    (s = 8 ? 0.904439 : 0) +
                    (s = 9 ? 0.954777 : 0) +
                    (s = 10 ? 0 : 0) +
                    (s = 11 ? 0 : 0) +
                    (s = 12 ? 0 : 0) +
                    (s = 13 ? 0 : 0) +
                    (s = 14 ? 0 : 0) +
                    (s = 15 ? 0 : 0) +
                    (s = 16 ? 0 : 0) +
                    (s = 17 ? 0 : 0) +
                    (s = 18 ? 0 : 0) +
                    (s = 19 ? 0.0156741 : 0) +
```

```
                        (s = 20 ? 0.0719057 : 0) +
                        (s = 21 ? 0.190204 : 0) +
                        (s = 22 ? 0.0156741 : 0) +
                        (s = 23 ? 0.0719057 : 0) +
                        (s = 24 ? 0.190204 : 0) +
                        (s = 25 ? 0.0156741 : 0) +
                        (s = 26 ? 0.0719057 : 0) +
                        (s = 27 ? 0.190204 : 0) +
                        (s = 28 ? 0 : 0) +
                        (s = 29 ? 0 : 0) +
                        (s = 30 ? 0 : 0) +
                        (s = 31 ? 0 : 0) +
                        (s = 32 ? 0 : 0) +
                        (s = 33 ? 0 : 0) +
                        (s = 34 ? 0 : 0) +
                        (s = 35 ? 0 : 0) +
                        (s = 36 ? 0 : 0) +
                        (s = 37 ? 0 : 0) +
                        (s = 38 ? 0 : 0) +
                        (s = 39 ? 0 : 0) +
                        (s = 40 ? 0 : 0) +
                        (s = 41 ? 0 : 0) +
                        (s = 42 ? 0 : 0) +
                        (s = 43 ? 0 : 0) +
                        (s = 44 ? 0 : 0) +
                        (s = 45 ? 0 : 0);


//*************************************************************************
// SYSTEM
//*************************************************************************

// Variable range and initialization
const a_MIN=0; const a_MAX=MAX_ALT_LEVEL; const a_INIT=init_a;
const f_MIN=0; const f_MAX=1; const f_INIT=init_f;
const c_MIN=0; const c_MAX=1; const c_INIT=init_c;

module sys
    a : [a_MIN..a_MAX] init a_INIT;
    f : [f_MIN..f_MAX] init f_INIT;
    c : [c_MIN..c_MAX] init c_INIT;

    [EcmOn_start] c=0 & ECM_ENABLED -> 1: (c'=c_EcmOn_impact);
    [EcmOff_start] c=1 & ECM_ENABLED -> 1: (c'=c_EcmOff_impact);

    [GoTight_start] f=0 -> 1: (a'=a_GoTight_impact)
        & (f'=f_GoTight_impact);
    [GoLoose_start] f=1 -> 1: (a'=a_GoLoose_impact)
        & (f'=f_GoLoose_impact);

    [IncAlt_complete] a < MAX_ALT_LEVEL & ONE_LEVEL_ENABLED -> 1: (a'=a_IncAlt_impact)
        & (f'=f_IncAlt_impact);
    [IncAlt2_complete] a < MAX_ALT_LEVEL-1 & TWO_LEVEL_ENABLED -> 1: (a'=a_IncAlt2_impact);

    [DecAlt_complete] a > 0 & ONE_LEVEL_ENABLED -> 1: (a'=a_DecAlt_impact)
        & (f'=f_DecAlt_impact);
    [DecAlt2_complete] a > 1 & TWO_LEVEL_ENABLED -> 1: (a'=a_DecAlt2_impact);
endmodule


formula c_EcmOn_impact = c + (1) >= c_MIN ? ( c+(1)<=c_MAX? c+(1) : c_MAX) : c_MIN;
formula c_EcmOff_impact = c + (-1) >= c_MIN ? ( c+(-1)<=c_MAX? c+(-1) : c_MAX) : c_MIN;
formula a_GoTight_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) : a_MAX) : a_MIN;
formula f_GoTight_impact = f + (1) >= f_MIN ? ( f+(1)<=f_MAX? f+(1) : f_MAX) : f_MIN;
formula a_GoLoose_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) : a_MAX) : a_MIN;
```

```
552 formula f_GoLoose_impact = f + (−1) >= f_MIN ? ( f+(−1)<=f_MAX? f+(−1) : f_MAX) : f_MIN;
553 formula a_IncAlt_impact = a + (1) >= a_MIN ? ( a+(1)<=a_MAX? a+(1) : a_MAX) : a_MIN;
554 formula f_IncAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) : f_MAX) : f_MIN;
555 formula a_DecAlt_impact = a + (−1) >= a_MIN ? ( a+(−1)<=a_MAX? a+(−1) : a_MAX) : a_MIN;
556 formula f_DecAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) : f_MAX) : f_MIN;
557 formula a_IncAlt2_impact = a + (2) >= a_MIN ? ( a+(2)<=a_MAX? a+(2) : a_MAX) : a_MIN;
558 formula a_DecAlt2_impact = a + (−2) >= a_MIN ? ( a+(−2)<=a_MAX? a+(−2) : a_MAX) : a_MIN;
559
560 // tactic concurrency rules
561 formula IncAlt_used = IncAlt_state != 0;
562 formula DecAlt_used = DecAlt_state != 0;
563 formula IncAlt2_used = IncAlt2_state != 0;
564 formula DecAlt2_used = DecAlt2_state != 0;
565
566 formula EcmOn_compatible = !EcmOn_used;
567 formula EcmOff_compatible = !EcmOff_used;
568 formula GoTight_compatible = !GoLoose_used;
569 formula GoLoose_compatible = !GoTight_used;
570 formula IncAlt_compatible = (!DecAlt_used) & (!IncAlt2_used) & (!DecAlt2_used);
571 formula DecAlt_compatible = (!IncAlt_used) & (!IncAlt2_used) & (!DecAlt2_used);
572 formula IncAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) & (!DecAlt2_used);
573 formula DecAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) & (!IncAlt2_used);
574
575
576 //*************************************************************************
577 // TACTIC: EcmOn
578 //*************************************************************************
579
580 // Applicability conditions
581 formula EcmOn_applicable = EcmOn_compatible & c=0;
582
583 module EcmOn
584     EcmOn_used : bool init false;
585     EcmOn_go : bool init true;
586
587     // Tactic applicable, start it
588     [EcmOn_start] sys_go & EcmOn_go & EcmOn_applicable & ECM_ENABLED −> (EcmOn_used'=true) &
            (EcmOn_go'=false);
589
590     // Tactic applicable, but do not start it
591     [EcmOn_pass] sys_go & EcmOn_go & EcmOn_applicable −> (EcmOn_go'=false);
592
593     // Pass if the tactic is not applicable
594     [EcmOn_invalid] sys_go & EcmOn_go & !EcmOn_applicable −> 1 : (EcmOn_go'=false);
595
596     [tick] !EcmOn_go −> 1: (EcmOn_go'=true) & (EcmOn_used'=false);
597 endmodule
598
599
600 //*************************************************************************
601 // TACTIC: EcmOff
602 //*************************************************************************
603
604 // Applicability conditions
605 formula EcmOff_applicable = EcmOff_compatible & c=1;
606
607 module EcmOff
608     EcmOff_used : bool init false;
609     EcmOff_go : bool init true;
610
611     // Tactic applicable, start it
612     [EcmOff_start] sys_go & EcmOff_go & EcmOff_applicable & ECM_ENABLED −> (EcmOff_used'=true) &
            (EcmOff_go'=false);
613
614     // Tactic applicable, but do not start it
```

```
615     [EcmOff_pass] sys_go & EcmOff_go & EcmOff_applicable -> (EcmOff_go'=false);
616
617     // Pass if the tactic is not applicable
618     [EcmOff_invalid] sys_go & EcmOff_go & !EcmOff_applicable -> 1 : (EcmOff_go'=false);
619
620     [tick] !EcmOff_go -> 1: (EcmOff_go'=true) & (EcmOff_used'=false);
621 endmodule
622
623
624 //*********************************************************************
625 // TACTIC: GoTight
626 //*********************************************************************
627
628 // Applicability conditions
629 formula GoTight_applicable = GoTight_compatible & f=0;
630
631 module GoTight
632     GoTight_used : bool init false;
633     GoTight_go : bool init true;
634
635     // Tactic applicable, start it
636     [GoTight_start] sys_go & GoTight_go & GoTight_applicable -> (GoTight_used'=true) & (GoTight_go'=false);
637
638     // Tactic applicable, but do not start it
639     [GoTight_pass] sys_go & GoTight_go & GoTight_applicable -> (GoTight_go'=false);
640
641     // Pass if the tactic is not applicable
642     [GoTight_invalid] sys_go & GoTight_go & !GoTight_applicable -> 1 : (GoTight_go'=false);
643
644     [tick] !GoTight_go -> 1: (GoTight_go'=true) & (GoTight_used'=false);
645 endmodule
646
647
648 //*********************************************************************
649 // TACTIC: GoLoose
650 //*********************************************************************
651
652 // Applicability conditions
653 formula GoLoose_applicable = GoLoose_compatible & f=1;
654
655 module GoLoose
656     GoLoose_used : bool init false;
657     GoLoose_go : bool init true;
658
659     // Tactic applicable, start it
660     [GoLoose_start] sys_go & GoLoose_go & GoLoose_applicable -> (GoLoose_used'=true) & (GoLoose_go'=false);
661
662     // Tactic applicable, but do not start it
663     [GoLoose_pass] sys_go & GoLoose_go & GoLoose_applicable -> (GoLoose_go'=false);
664
665     // Pass if the tactic is not applicable
666     [GoLoose_invalid] sys_go & GoLoose_go & !GoLoose_applicable -> 1 : (GoLoose_go'=false);
667
668     [tick] !GoLoose_go -> 1: (GoLoose_go'=true) & (GoLoose_used'=false);
669 endmodule
670
671
672 //*********************************************************************
673 // TACTIC: IncAlt
674 //*********************************************************************
675
676 const int IncAlt_LATENCY_PERIODS = ceil(IncAlt_LATENCY/PERIOD);
677
678 // Applicability conditions
679 formula IncAlt_applicable = IncAlt_compatible & a < MAX_ALT_LEVEL;
```

```
680
681 module IncAlt
682     IncAlt_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt_state;
683     IncAlt_go : bool init true;
684
685     // Tactic applicable, start it
686     [IncAlt_start] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable & ONE_LEVEL_ENABLED ->
            (IncAlt_state'=IncAlt_LATENCY_PERIODS) & (IncAlt_go'=false);
687
688     // Tactic applicable, but do not start it
689     [IncAlt_pass] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable -> (IncAlt_go'=false);
690
691     // Pass if the tactic is not applicable
692     [IncAlt_invalid] sys_go & IncAlt_go & IncAlt_state=0 & !IncAlt_applicable -> 1 : (IncAlt_go'=false);
693
694     // Progress of the tactic
695     [IncAlt_progress] sys_go & IncAlt_go & IncAlt_state > 1 -> 1: (IncAlt_state'=IncAlt_state-1) & (IncAlt_go'=false);
696
697     // Completion of the tactic
698     [IncAlt_complete] sys_go & IncAlt_go & IncAlt_state=1 -> 1: (IncAlt_state'=0) & (IncAlt_go'=true);
699
700     [tick] !IncAlt_go -> 1: (IncAlt_go'=true);
701 endmodule
702
703
704 //*********************************************************************
705 // TACTIC: DecAlt
706 //*********************************************************************
707
708 const int DecAlt_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);
709
710 // Applicability conditions
711 formula DecAlt_applicable = DecAlt_compatible & a > 0;
712
713 module DecAlt
714     DecAlt_state : [0..DecAlt_LATENCY_PERIODS] init ini_DecAlt_state;
715     DecAlt_go : bool init true;
716
717     // Tactic applicable, start it
718     [DecAlt_start] sys_go & DecAlt_go & DecAlt_state=0 & DecAlt_applicable & ONE_LEVEL_ENABLED ->
            (DecAlt_state'=DecAlt_LATENCY_PERIODS) & (DecAlt_go'=false);
719
720     // Tactic applicable, but do not start it
721     [DecAlt_pass] sys_go & DecAlt_go & DecAlt_state=0 & DecAlt_applicable -> (DecAlt_go'=false);
722
723     // Pass if the tactic is not applicable
724     [DecAlt_invalid] sys_go & DecAlt_go & DecAlt_state=0 & !DecAlt_applicable -> 1 : (DecAlt_go'=false);
725
726     // Progress of the tactic
727     [DecAlt_progress] sys_go & DecAlt_go & DecAlt_state > 1 -> 1: (DecAlt_state'=DecAlt_state-1) & (DecAlt_go'=false);
728
729     // Completion of the tactic
730     [DecAlt_complete] sys_go & DecAlt_go & DecAlt_state=1 -> 1: (DecAlt_state'=0) & (DecAlt_go'=true);
731
732     [tick] !DecAlt_go -> 1: (DecAlt_go'=true);
733 endmodule
734
735 //*********************************************************************
736 // TACTIC: IncAlt2
737 //*********************************************************************
738
739 // Applicability conditions
740 formula IncAlt2_applicable = IncAlt2_compatible & a < MAX_ALT_LEVEL-1;
741
742 module IncAlt2
```

```
743     IncAlt2_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt2_state;
744     IncAlt2_go : bool init true;
745
746     // Tactic applicable, start it
747     [IncAlt2_start] sys_go & IncAlt2_go & IncAlt2_state=0 & IncAlt2_applicable & TWO_LEVEL_ENABLED ->
            (IncAlt2_state'=IncAlt_LATENCY_PERIODS) & (IncAlt2_go'=false);
748
749     // Tactic applicable, but do not start it
750     [IncAlt2_pass] sys_go & IncAlt2_go & IncAlt2_state=0 & IncAlt2_applicable -> (IncAlt2_go'=false);
751
752     // Pass if the tactic is not applicable
753     [IncAlt2_invalid] sys_go & IncAlt2_go & IncAlt2_state=0 & !IncAlt2_applicable -> 1 : (IncAlt2_go'=false);
754
755     // Progress of the tactic
756     [IncAlt2_progress] sys_go & IncAlt2_go & IncAlt2_state > 1 -> 1: (IncAlt2_state'=IncAlt2_state-1) & (IncAlt2_go'=false);
757
758     // Completion of the tactic
759     [IncAlt2_complete] sys_go & IncAlt2_go & IncAlt2_state=1 -> 1: (IncAlt2_state'=0) & (IncAlt2_go'=true);
760
761     [tick] !IncAlt2_go -> 1: (IncAlt2_go'=true);
762 endmodule
763
764
765 //*********************************************************************
766 // TACTIC: DecAlt2
767 //*********************************************************************
768
769 const int DecAlt2_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);
770
771 // Applicability conditions
772 formula DecAlt2_applicable = DecAlt2_compatible & a > 1;
773
774 module DecAlt2
775     DecAlt2_state : [0..DecAlt2_LATENCY_PERIODS] init ini_DecAlt2_state;
776     DecAlt2_go : bool init true;
777
778     // Tactic applicable, start it
779     [DecAlt2_start] sys_go & DecAlt2_go & DecAlt2_state=0 & DecAlt2_applicable & TWO_LEVEL_ENABLED ->
            (DecAlt2_state'=DecAlt2_LATENCY_PERIODS) & (DecAlt2_go'=false);
780
781     // Tactic applicable, but do not start it
782     [DecAlt2_pass] sys_go & DecAlt2_go & DecAlt2_state=0 & DecAlt2_applicable -> (DecAlt2_go'=false);
783
784     // Pass if the tactic is not applicable
785     [DecAlt2_invalid] sys_go & DecAlt2_go & DecAlt2_state=0 & !DecAlt2_applicable -> 1 : (DecAlt2_go'=false);
786
787     // Progress of the tactic
788     [DecAlt2_progress] sys_go & DecAlt2_go & DecAlt2_state > 1 -> 1: (DecAlt2_state'=DecAlt2_state-1) &
            (DecAlt2_go'=false);
789
790     // Completion of the tactic
791     [DecAlt2_complete] sys_go & DecAlt2_go & DecAlt2_state=1 -> 1: (DecAlt2_state'=0) & (DecAlt2_go'=true);
792
793     [tick] !DecAlt2_go -> 1: (DecAlt2_go'=true);
794 endmodule
795
796
797 //*******************************
798 // Utility Function
799 //*******************************
800 const int LOOSE = 0;
801 const int TIGHT = 1;
802 const int EMC_ON = 1;
803
804 formula probOfThreat = stateValue;
```

```
formula probabilityOfDestruction = probOfThreat
    * ((f = LOOSE) ? 1.0 : (1.0 / destructionFormationFactor))
    * ((c = EMC_ON) ? ecm_threat_prob : 1.0)
        * max(0.0, threatRange − (a + 1)) / threatRange; // +1 because level 0 is one level above ground

module constraint // in this case the constraint is surviving
    satisfied: bool init true;
    [tick2] satisfied −> (1.0 − probabilityOfDestruction): (satisfied'=true)
        + probabilityOfDestruction: (satisfied'=false);
    [tick2] !satisfied −> true;
endmodule


formula probOfTarget= stateValue1;

formula probOfDetection = probOfTarget
    * ((f = LOOSE) ? 1.0 : (1.0 / detectionFormationFactor))
    * ((c = EMC_ON) ? ecm_target_prob : 1.0)
    * max(0.0, sensorRange − (a + 1)) / sensorRange; // +1 because level 0 is one level above ground

module sensor
    targetDetected: bool init false;
    [tick2] true −> probOfDetection: (targetDetected'=true) + (1.0 − probOfDetection): (targetDetected'=false);
endmodule

rewards "util"
    [tack] (time < HORIZON) & satisfied & targetDetected : 1;
    [tack] (time = HORIZON) & satisfied : (targetDetected ? 1 : 0) + survival_reward;

    // give slight preference to not adapting
    [tick] time = 0 & IncAlt_state=ini_IncAlt_state & DecAlt_state=ini_DecAlt_state & a=init_a & f=init_f : 0.000000001;
endrewards
```

Listing 3: PRISM specification for short horizon MDP planning

Long Horizon MDP Planning Specification

To model uncertainty in targets and threats along a route, we adopt the approach suggested by Moreno et al. [35] since they also used the combination of DARTSim and MDP planning to evaluate their ideas. In short, two independent random variables are used in the environment state to represent the probabilities that a segment contains a target and a threat, respectively. Using the target and threat variables, we construct independent environment models for targets and threats, and then join them to produce a joint environment model, which is used by ρ_{mdps} and ρ_{mdpl}.

```
mdp
const double PERIOD = 60;
const int HORIZON = 5; // Planning horizon for deliberative planning
const double IncAlt_LATENCY = 60;
const double DecAlt_LATENCY = 60;
const int MAX_ALT_LEVEL = 3;
const double destructionFormationFactor = 1.5;
const double threatRange = 3;
const double detectionFormationFactor = 1.2;
const double sensorRange = 4;
const init_a = 0;
const init_c = 0;
const init_f = 0;
```

```
const bool ECM_ENABLED = true;
const bool ONE_LEVEL_ENABLED = true; // Unlike reactive planning, one level increase/decrease altitude enabled
const bool TWO_LEVEL_ENABLED = true; // Two level increase/decrease altitude also enabled
const int ini_IncAlt_state = 0;
const int ini_DecAlt_state = 0;
const int ini_IncAlt2_state = 0;
const int ini_DecAlt2_state = 0;
const double ecm_threat_prob = 0.15;
const double ecm_target_prob = 0.3;
const double survival_reward = 1;


//*********************************************************************
// CLOCK
//*********************************************************************
const int TO_TICK = 0;
const int TO_TICK2 = 1; // intermediate tick for constraint satisf. update
const int TO_TACK = 2;

label "final" = time = HORIZON & clockstep=TO_TICK;
formula sys_go = clockstep=TO_TICK;

module clk
    time : [0..HORIZON] init 0;
    clockstep : [0..2] init TO_TICK;

    [tick] clockstep=TO_TICK & time < HORIZON -> 1: (time'=time+1) & (clockstep'=TO_TICK2);
    [tick2] clockstep=TO_TICK2 -> 1 : (clockstep'=TO_TACK);
    [tack] clockstep=TO_TACK -> 1: (clockstep'=TO_TICK);
endmodule

module env
s : [0..45] init 0;
[tick] s = 0 ->
    0.034225 : (s' = 1)
    + 0.11655 : (s' = 2)
    + 0.034225 : (s' = 3)
    + 0.11655 : (s' = 4)
    + 0.3969 : (s' = 5)
    + 0.11655 : (s' = 6)
    + 0.034225 : (s' = 7)
    + 0.11655 : (s' = 8)
    + 0.034225 : (s' = 9);
[tick] s = 1 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 2 ->
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 3 ->
    0.034225 : (s' = 10)
```

```
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
        + 0.034225 : (s' = 16)
        + 0.11655 : (s' = 17)
        + 0.034225 : (s' = 18);
    [tick] s = 4 ->
        0.034225 : (s' = 10)
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
        + 0.034225 : (s' = 16)
        + 0.11655 : (s' = 17)
        + 0.034225 : (s' = 18);
    [tick] s = 5 ->
        0.034225 : (s' = 10)
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
        + 0.034225 : (s' = 16)
        + 0.11655 : (s' = 17)
        + 0.034225 : (s' = 18);
    [tick] s = 6 ->
        0.034225 : (s' = 10)
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
        + 0.034225 : (s' = 16)
        + 0.11655 : (s' = 17)
        + 0.034225 : (s' = 18);
    [tick] s = 7 ->
        0.034225 : (s' = 10)
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
        + 0.034225 : (s' = 16)
        + 0.11655 : (s' = 17)
        + 0.034225 : (s' = 18);
    [tick] s = 8 ->
        0.034225 : (s' = 10)
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
        + 0.034225 : (s' = 16)
        + 0.11655 : (s' = 17)
        + 0.034225 : (s' = 18);
    [tick] s = 9 ->
        0.034225 : (s' = 10)
        + 0.11655 : (s' = 11)
        + 0.034225 : (s' = 12)
        + 0.11655 : (s' = 13)
        + 0.3969 : (s' = 14)
        + 0.11655 : (s' = 15)
```

```
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
    + 0.034225 : (s' = 18);
[tick] s = 10 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 11 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 12 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 13 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 14 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 15 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 16 ->
    0.034225 : (s' = 19)
```

```
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 17 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 18 ->
    0.034225 : (s' = 19)
    + 0.11655 : (s' = 20)
    + 0.034225 : (s' = 21)
    + 0.11655 : (s' = 22)
    + 0.3969 : (s' = 23)
    + 0.11655 : (s' = 24)
    + 0.034225 : (s' = 25)
    + 0.11655 : (s' = 26)
    + 0.034225 : (s' = 27);
[tick] s = 19 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 20 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 21 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
    + 0.034225 : (s' = 34)
    + 0.11655 : (s' = 35)
    + 0.034225 : (s' = 36);
[tick] s = 22 ->
    0.034225 : (s' = 28)
    + 0.11655 : (s' = 29)
    + 0.034225 : (s' = 30)
    + 0.11655 : (s' = 31)
    + 0.3969 : (s' = 32)
    + 0.11655 : (s' = 33)
```

```
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 23 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 24 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 25 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 26 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 27 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 28 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 29 ->
        0.034225 : (s' = 37)
```

```
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 30 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 31 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 32 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 33 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 34 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 35 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
```

```
    + 0.034225 : (s' = 43)
    + 0.11655 : (s' = 44)
    + 0.034225 : (s' = 45);
[tick] s = 36 ->
    0.034225 : (s' = 37)
    + 0.11655 : (s' = 38)
    + 0.034225 : (s' = 39)
    + 0.11655 : (s' = 40)
    + 0.3969 : (s' = 41)
    + 0.11655 : (s' = 42)
    + 0.034225 : (s' = 43)
    + 0.11655 : (s' = 44)
    + 0.034225 : (s' = 45);
endmodule

// environment has 2 components. statevalue for threats and statevalue1 is for targets
formula stateValue = (s = 0 ? 0 : 0) +
                     (s = 1 ? 0.00605639 : 0) +
                     (s = 2 ? 0.00605639 : 0) +
                     (s = 3 ? 0.00605639 : 0) +
                     (s = 4 ? 0.0282836 : 0) +
                     (s = 5 ? 0.0282836 : 0) +
                     (s = 6 ? 0.0282836 : 0) +
                     (s = 7 ? 0.0778979 : 0) +
                     (s = 8 ? 0.0778979 : 0) +
                     (s = 9 ? 0.0778979 : 0) +
                     (s = 10 ? 0 : 0) +
                     (s = 11 ? 0 : 0) +
                     (s = 12 ? 0 : 0) +
                     (s = 13 ? 0 : 0) +
                     (s = 14 ? 0 : 0) +
                     (s = 15 ? 0 : 0) +
                     (s = 16 ? 0 : 0) +
                     (s = 17 ? 0 : 0) +
                     (s = 18 ? 0 : 0) +
                     (s = 19 ? 0.750751 : 0) +
                     (s = 20 ? 0.750751 : 0) +
                     (s = 21 ? 0.750751 : 0) +
                     (s = 22 ? 0.885424 : 0) +
                     (s = 23 ? 0.885424 : 0) +
                     (s = 24 ? 0.885424 : 0) +
                     (s = 25 ? 0.963485 : 0) +
                     (s = 26 ? 0.963485 : 0) +
                     (s = 27 ? 0.963485 : 0) +
                     (s = 28 ? 0.435626 : 0) +
                     (s = 29 ? 0.435626 : 0) +
                     (s = 30 ? 0.435626 : 0) +
                     (s = 31 ? 0.676196 : 0) +
                     (s = 32 ? 0.676196 : 0) +
                     (s = 33 ? 0.676196 : 0) +
                     (s = 34 ? 0.864925 : 0) +
                     (s = 35 ? 0.864925 : 0) +
                     (s = 36 ? 0.864925 : 0) +
                     (s = 37 ? 0.368403 : 0) +
                     (s = 38 ? 0.368403 : 0) +
                     (s = 39 ? 0.368403 : 0) +
                     (s = 40 ? 0.793701 : 0) +
                     (s = 41 ? 0.793701 : 0) +
                     (s = 42 ? 0.793701 : 0) +
                     (s = 43 ? 0.983048 : 0) +
                     (s = 44 ? 0.983048 : 0) +
                     (s = 45 ? 0.983048 : 0);

formula stateValue1 = (s = 0 ? 0 : 0) +
                      (s = 1 ? 0.830037 : 0) +
```

```
                        (s = 2 ? 0.904439 : 0) +
                        (s = 3 ? 0.954777 : 0) +
                        (s = 4 ? 0.830037 : 0) +
                        (s = 5 ? 0.904439 : 0) +
                        (s = 6 ? 0.954777 : 0) +
                        (s = 7 ? 0.830037 : 0) +
                        (s = 8 ? 0.904439 : 0) +
                        (s = 9 ? 0.954777 : 0) +
                        (s = 10 ? 0 : 0) +
                        (s = 11 ? 0 : 0) +
                        (s = 12 ? 0 : 0) +
                        (s = 13 ? 0 : 0) +
                        (s = 14 ? 0 : 0) +
                        (s = 15 ? 0 : 0) +
                        (s = 16 ? 0 : 0) +
                        (s = 17 ? 0 : 0) +
                        (s = 18 ? 0 : 0) +
                        (s = 19 ? 0.0156741 : 0) +
                        (s = 20 ? 0.0719057 : 0) +
                        (s = 21 ? 0.190204 : 0) +
                        (s = 22 ? 0.0156741 : 0) +
                        (s = 23 ? 0.0719057 : 0) +
                        (s = 24 ? 0.190204 : 0) +
                        (s = 25 ? 0.0156741 : 0) +
                        (s = 26 ? 0.0719057 : 0) +
                        (s = 27 ? 0.190204 : 0) +
                        (s = 28 ? 0 : 0) +
                        (s = 29 ? 0 : 0) +
                        (s = 30 ? 0 : 0) +
                        (s = 31 ? 0 : 0) +
                        (s = 32 ? 0 : 0) +
                        (s = 33 ? 0 : 0) +
                        (s = 34 ? 0 : 0) +
                        (s = 35 ? 0 : 0) +
                        (s = 36 ? 0 : 0) +
                        (s = 37 ? 0 : 0) +
                        (s = 38 ? 0 : 0) +
                        (s = 39 ? 0 : 0) +
                        (s = 40 ? 0 : 0) +
                        (s = 41 ? 0 : 0) +
                        (s = 42 ? 0 : 0) +
                        (s = 43 ? 0 : 0) +
                        (s = 44 ? 0 : 0) +
                        (s = 45 ? 0 : 0);
// #ENV ENDS


//*********************************************************************
// SYSTEM
//*********************************************************************

// Variable range and initialization
const a_MIN=0; const a_MAX=MAX_ALT_LEVEL; const a_INIT=init_a;
const f_MIN=0; const f_MAX=1; const f_INIT=init_f;
const c_MIN=0; const c_MAX=1; const c_INIT=init_c;

module sys
    a : [a_MIN..a_MAX] init a_INIT;
    f : [f_MIN..f_MAX] init f_INIT;
    c : [c_MIN..c_MAX] init c_INIT;

    [EcmOn_start] c=0 & ECM_ENABLED -> 1: (c'=c_EcmOn_impact);
    [EcmOff_start] c=1 & ECM_ENABLED -> 1: (c'=c_EcmOff_impact);

    [GoTight_start] f=0 -> 1: (a'=a_GoTight_impact)
```

```
534             & (f'=f_GoTight_impact);
535     [GoLoose_start] f=1 -> 1: (a'=a_GoLoose_impact)
536             & (f'=f_GoLoose_impact);
537
538     [IncAlt_complete] a < MAX_ALT_LEVEL & ONE_LEVEL_ENABLED -> 1: (a'=a_IncAlt_impact)
539         & (f'=f_IncAlt_impact);
540     [IncAlt2_complete] a < MAX_ALT_LEVEL-1 & TWO_LEVEL_ENABLED -> 1: (a'=a_IncAlt2_impact);
541
542     [DecAlt_complete] a > 0 & ONE_LEVEL_ENABLED -> 1: (a'=a_DecAlt_impact)
543             & (f'=f_DecAlt_impact);
544     [DecAlt2_complete] a > 1 & TWO_LEVEL_ENABLED -> 1: (a'=a_DecAlt2_impact);
545 endmodule
546
547
548 formula c_EcmOn_impact = c + (1) >= c_MIN ? ( c+(1)<=c_MAX? c+(1) : c_MAX) : c_MIN;
549 formula c_EcmOff_impact = c + (-1) >= c_MIN ? ( c+(-1)<=c_MAX? c+(-1) : c_MAX) : c_MIN;
550 formula a_GoTight_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) : a_MAX) : a_MIN;
551 formula f_GoTight_impact = f + (1) >= f_MIN ? ( f+(1)<=f_MAX? f+(1) : f_MAX) : f_MIN;
552 formula a_GoLoose_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) : a_MAX) : a_MIN;
553 formula f_GoLoose_impact = f + (-1) >= f_MIN ? ( f+(-1)<=f_MAX? f+(-1) : f_MAX) : f_MIN;
554 formula a_IncAlt_impact = a + (1) >= a_MIN ? ( a+(1)<=a_MAX? a+(1) : a_MAX) : a_MIN;
555 formula f_IncAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) : f_MAX) : f_MIN;
556 formula a_DecAlt_impact = a + (-1) >= a_MIN ? ( a+(-1)<=a_MAX? a+(-1) : a_MAX) : a_MIN;
557 formula f_DecAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) : f_MAX) : f_MIN;
558 formula a_IncAlt2_impact = a + (2) >= a_MIN ? ( a+(2)<=a_MAX? a+(2) : a_MAX) : a_MIN;
559 formula a_DecAlt2_impact = a + (-2) >= a_MIN ? ( a+(-2)<=a_MAX? a+(-2) : a_MAX) : a_MIN;
560
561 // tactic concurrency rules
562 formula IncAlt_used = IncAlt_state != 0;
563 formula DecAlt_used = DecAlt_state != 0;
564 formula IncAlt2_used = IncAlt2_state != 0;
565 formula DecAlt2_used = DecAlt2_state != 0;
566
567 formula EcmOn_compatible = !EcmOn_used;
568 formula EcmOff_compatible = !EcmOff_used;
569 formula GoTight_compatible = !GoLoose_used;
570 formula GoLoose_compatible = !GoTight_used;
571 formula IncAlt_compatible = (!DecAlt_used) & (!IncAlt2_used) & (!DecAlt2_used);
572 formula DecAlt_compatible = (!IncAlt_used) & (!IncAlt2_used) & (!DecAlt2_used);
573 formula IncAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) & (!DecAlt2_used);
574 formula DecAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) & (!IncAlt2_used);
575
576
577 //*************************************************************************
578 // TACTIC: EcmOn
579 //*************************************************************************
580
581 // Applicability conditions
582 formula EcmOn_applicable = EcmOn_compatible & c=0;
583
584 module EcmOn
585     EcmOn_used : bool init false;
586     EcmOn_go : bool init true;
587
588     // Tactic applicable, start it
589     [EcmOn_start] sys_go & EcmOn_go & EcmOn_applicable & ECM_ENABLED -> (EcmOn_used'=true) &
            (EcmOn_go'=false);
590
591     // Tactic applicable, but do not start it
592     [EcmOn_pass] sys_go & EcmOn_go & EcmOn_applicable -> (EcmOn_go'=false);
593
594     // Pass if the tactic is not applicable
595     [EcmOn_invalid] sys_go & EcmOn_go & !EcmOn_applicable -> 1 : (EcmOn_go'=false);
596
597     [tick] !EcmOn_go -> 1: (EcmOn_go'=true) & (EcmOn_used'=false);
```

```
598 endmodule
599
600
601 //*************************************************************************
602 // TACTIC: EcmOff
603 //*************************************************************************
604
605 // Applicability conditions
606 formula EcmOff_applicable = EcmOff_compatible & c=1;
607
608 module EcmOff
609     EcmOff_used : bool init false;
610     EcmOff_go : bool init true;
611
612     // Tactic applicable, start it
613     [EcmOff_start] sys_go & EcmOff_go & EcmOff_applicable & ECM_ENABLED -> (EcmOff_used'=true) &
            (EcmOff_go'=false);
614
615     // Tactic applicable, but do not start it
616     [EcmOff_pass] sys_go & EcmOff_go & EcmOff_applicable -> (EcmOff_go'=false);
617
618     // Pass if the tactic is not applicable
619     [EcmOff_invalid] sys_go & EcmOff_go & !EcmOff_applicable -> 1 : (EcmOff_go'=false);
620
621     [tick] !EcmOff_go -> 1: (EcmOff_go'=true) & (EcmOff_used'=false);
622 endmodule
623
624
625 //*************************************************************************
626 // TACTIC: GoTight
627 //*************************************************************************
628
629 // Applicability conditions
630 formula GoTight_applicable = GoTight_compatible & f=0;
631
632 module GoTight
633     GoTight_used : bool init false;
634     GoTight_go : bool init true;
635
636     // Tactic applicable, start it
637     [GoTight_start] sys_go & GoTight_go & GoTight_applicable -> (GoTight_used'=true) & (GoTight_go'=false);
638
639     // Tactic applicable, but do not start it
640     [GoTight_pass] sys_go & GoTight_go & GoTight_applicable -> (GoTight_go'=false);
641
642     // Pass if the tactic is not applicable
643     [GoTight_invalid] sys_go & GoTight_go & !GoTight_applicable -> 1 : (GoTight_go'=false);
644
645     [tick] !GoTight_go -> 1: (GoTight_go'=true) & (GoTight_used'=false);
646 endmodule
647
648
649 //*************************************************************************
650 // TACTIC: GoLoose
651 //*************************************************************************
652
653 // Applicability conditions
654 formula GoLoose_applicable = GoLoose_compatible & f=1;
655
656 module GoLoose
657     GoLoose_used : bool init false;
658     GoLoose_go : bool init true;
659
660     // Tactic applicable, start it
661     [GoLoose_start] sys_go & GoLoose_go & GoLoose_applicable -> (GoLoose_used'=true) & (GoLoose_go'=false);
```

```
662
663         // Tactic applicable, but do not start it
664         [GoLoose_pass] sys_go & GoLoose_go & GoLoose_applicable -> (GoLoose_go'=false);
665
666         // Pass if the tactic is not applicable
667         [GoLoose_invalid] sys_go & GoLoose_go & !GoLoose_applicable -> 1 : (GoLoose_go'=false);
668
669         [tick] !GoLoose_go -> 1: (GoLoose_go'=true) & (GoLoose_used'=false);
670     endmodule
671
672
673     //*******************************************************************
674     // TACTIC: IncAlt
675     //*******************************************************************
676
677     const int IncAlt_LATENCY_PERIODS = ceil(IncAlt_LATENCY/PERIOD);
678
679     // Applicability conditions
680     formula IncAlt_applicable = IncAlt_compatible & a < MAX_ALT_LEVEL;
681
682     module IncAlt
683         IncAlt_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt_state;
684         IncAlt_go : bool init true;
685
686         // Tactic applicable, start it
687         [IncAlt_start] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable & ONE_LEVEL_ENABLED ->
                (IncAlt_state'=IncAlt_LATENCY_PERIODS) & (IncAlt_go'=false);
688
689         // Tactic applicable, but do not start it
690         [IncAlt_pass] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable -> (IncAlt_go'=false);
691
692         // Pass if the tactic is not applicable
693         [IncAlt_invalid] sys_go & IncAlt_go & IncAlt_state=0 & !IncAlt_applicable -> 1 : (IncAlt_go'=false);
694
695         // Progress of the tactic
696         [IncAlt_progress] sys_go & IncAlt_go & IncAlt_state > 1 -> 1: (IncAlt_state'=IncAlt_state-1) & (IncAlt_go'=false);
697
698         // Completion of the tactic
699         [IncAlt_complete] sys_go & IncAlt_go & IncAlt_state=1 -> 1: (IncAlt_state'=0) & (IncAlt_go'=true);
700
701         [tick] !IncAlt_go -> 1: (IncAlt_go'=true);
702     endmodule
703
704
705     //*******************************************************************
706     // TACTIC: DecAlt
707     //*******************************************************************
708
709     const int DecAlt_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);
710
711     // Applicability conditions
712     formula DecAlt_applicable = DecAlt_compatible & a > 0;
713
714     module DecAlt
715         DecAlt_state : [0..DecAlt_LATENCY_PERIODS] init ini_DecAlt_state;
716         DecAlt_go : bool init true;
717
718         // Tactic applicable, start it
719         [DecAlt_start] sys_go & DecAlt_go & DecAlt_state=0 & DecAlt_applicable & ONE_LEVEL_ENABLED ->
                (DecAlt_state'=DecAlt_LATENCY_PERIODS) & (DecAlt_go'=false);
720
721         // Tactic applicable, but do not start it
722         [DecAlt_pass] sys_go & DecAlt_go & DecAlt_state=0 & DecAlt_applicable -> (DecAlt_go'=false);
723
724         // Pass if the tactic is not applicable
```

```
725     [DecAlt_invalid] sys_go & DecAlt_go & DecAlt_state=0 & !DecAlt_applicable -> 1 : (DecAlt_go'=false);
726
727     // Progress of the tactic
728     [DecAlt_progress] sys_go & DecAlt_go & DecAlt_state > 1 -> 1: (DecAlt_state'=DecAlt_state-1) & (DecAlt_go'=false);
729
730     // Completion of the tactic
731     [DecAlt_complete] sys_go & DecAlt_go & DecAlt_state=1 -> 1: (DecAlt_state'=0) & (DecAlt_go'=true);
732
733     [tick] !DecAlt_go -> 1: (DecAlt_go'=true);
734 endmodule
735
736 //*********************************************************************
737 // TACTIC: IncAlt2
738 //*********************************************************************
739
740 // Applicability conditions
741 formula IncAlt2_applicable = IncAlt2_compatible & a < MAX_ALT_LEVEL-1;
742
743 module IncAlt2
744     IncAlt2_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt2_state;
745     IncAlt2_go : bool init true;
746
747     // Tactic applicable, start it
748     [IncAlt2_start] sys_go & IncAlt2_go & IncAlt2_state=0 & IncAlt2_applicable & TWO_LEVEL_ENABLED ->
            (IncAlt2_state'=IncAlt_LATENCY_PERIODS) & (IncAlt2_go'=false);
749
750     // Tactic applicable, but do not start it
751     [IncAlt2_pass] sys_go & IncAlt2_go & IncAlt2_state=0 & IncAlt2_applicable -> (IncAlt2_go'=false);
752
753     // Pass if the tactic is not applicable
754     [IncAlt2_invalid] sys_go & IncAlt2_go & IncAlt2_state=0 & !IncAlt2_applicable -> 1 : (IncAlt2_go'=false);
755
756     // Progress of the tactic
757     [IncAlt2_progress] sys_go & IncAlt2_go & IncAlt2_state > 1 -> 1: (IncAlt2_state'=IncAlt2_state-1) & (IncAlt2_go'=false);
758
759     // Completion of the tactic
760     [IncAlt2_complete] sys_go & IncAlt2_go & IncAlt2_state=1 -> 1: (IncAlt2_state'=0) & (IncAlt2_go'=true);
761
762     [tick] !IncAlt2_go -> 1: (IncAlt2_go'=true);
763 endmodule
764
765
766 //*********************************************************************
767 // TACTIC: DecAlt2
768 //*********************************************************************
769
770 const int DecAlt2_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);
771
772 // Applicability conditions
773 formula DecAlt2_applicable = DecAlt2_compatible & a > 1;
774
775 module DecAlt2
776     DecAlt2_state : [0..DecAlt2_LATENCY_PERIODS] init ini_DecAlt2_state;
777     DecAlt2_go : bool init true;
778
779     // Tactic applicable, start it
780     [DecAlt2_start] sys_go & DecAlt2_go & DecAlt2_state=0 & DecAlt2_applicable & TWO_LEVEL_ENABLED ->
            (DecAlt2_state'=DecAlt2_LATENCY_PERIODS) & (DecAlt2_go'=false);
781
782     // Tactic applicable, but do not start it
783     [DecAlt2_pass] sys_go & DecAlt2_go & DecAlt2_state=0 & DecAlt2_applicable -> (DecAlt2_go'=false);
784
785     // Pass if the tactic is not applicable
786     [DecAlt2_invalid] sys_go & DecAlt2_go & DecAlt2_state=0 & !DecAlt2_applicable -> 1 : (DecAlt2_go'=false);
787
```

```
    // Progress of the tactic
    [DecAlt2_progress] sys_go & DecAlt2_go & DecAlt2_state > 1 -> 1: (DecAlt2_state'=DecAlt2_state-1) &
        (DecAlt2_go'=false);

    // Completion of the tactic
    [DecAlt2_complete] sys_go & DecAlt2_go & DecAlt2_state=1 -> 1: (DecAlt2_state'=0) & (DecAlt2_go'=true);

    [tick] !DecAlt2_go -> 1: (DecAlt2_go'=true);
endmodule


//*******************************
// Utility Function
//*******************************
const int LOOSE = 0;
const int TIGHT = 1;
const int EMC_ON = 1;

formula probOfThreat = stateValue;

formula probabilityOfDestruction = probOfThreat
    * ((f = LOOSE) ? 1.0 : (1.0 / destructionFormationFactor))
    * ((c = EMC_ON) ? ecm_threat_prob : 1.0)
        * max(0.0, threatRange - (a + 1)) / threatRange; // +1 because level 0 is one level above ground

module constraint // in this case the constraint is surviving
    satisfied: bool init true;
    [tick2] satisfied -> (1.0 - probabilityOfDestruction): (satisfied'=true)
        + probabilityOfDestruction: (satisfied'=false);
    [tick2] !satisfied -> true;
endmodule


formula probOfTarget= stateValue1;

formula probOfDetection = probOfTarget
    * ((f = LOOSE) ? 1.0 : (1.0 / detectionFormationFactor))
    * ((c = EMC_ON) ? ecm_target_prob : 1.0)
    * max(0.0, sensorRange - (a + 1)) / sensorRange; // +1 because level 0 is one level above ground

module sensor
    targetDetected: bool init false;
    [tick2] true -> probOfDetection: (targetDetected'=true) + (1.0 - probOfDetection): (targetDetected'=false);
endmodule

rewards "util"
    //[tack] satisfied & targetDetected : 1;

    [tack] (time < HORIZON) & satisfied & targetDetected : 1;
    [tack] (time = HORIZON) & satisfied : (targetDetected ? 1 : 0) + survival_reward;

    // give slight preference to not adapting
    [tick] time = 0 & IncAlt_state=ini_IncAlt_state & DecAlt_state=ini_DecAlt_state & a=init_a & f=init_f : 0.000000001;
endrewards
```

Listing 4: PRISM specification for long horizon MDP planning