Appendix D

Code

D.1 ASP Encoding of the Igniting the Burner Example

```
%% ------------------------------------------------------
%% ignite.lp
%% ------------------------------------------------------

section(s1).
section(s2).
section(s3).

valve(v1).
valve(v2).

connected_to_tank(s1).
connected(s1,v1,s2).
connected(s2,v2,s3).
connected_to_burner(s3).

fluent(inertial, burner_on).
fluent(inertial, opened(V)) :- valve(V).
fluent(defined, pressurized(S)) :- section(S).

action(open(V)) :- valve(V).
action(close(V)) :- valve(V).
action(ignite).

#const n = 4.
step(0..n).

%% ----------------------
```

```
%% AL System Description:
%% ----------------------

%% pressurized(S) if connected_to_tank(S).
holds(pressurized(S),I) :- step(I),
                           connected_to_tank(S).

%% pressurized(S2) if connected(S1,V,S2),
%%                    opened(V),
%%                    pressurized(S1).
holds(pressurized(S2), I) :- connected(S1,V,S2),
                             holds(opened(V), I),
                             holds(pressurized(S1), I).

%% -burner_on if connected_to_burner(S),
%%               -pressurized(S).
-holds(burner_on, I) :- connected_to_burner(S),
                        -holds(pressurized(S),I).

%% open(V) causes opened(V).
holds(opened(V), I+1) :- occurs(open(V),I),
                         I < n.

%% impossible open(V) if opened(V).
-occurs(open(V),I) :- holds(opened(V),I).

%% impossible open(V1) if connected(S1,V1,S2),
%%                        connected(S2,V2,S3),
%%                        opened(V2).
-occurs(open(V1),I) :- connected(S1,V1,S2),
                       connected(S2,V2,S3),
                       holds(opened(V2),I).

%% close(V) causes -opened(V).
-holds(opened(V), I+1) :- occurs(close(V), I),
                          I < n.

%% impossible close(V) if -opened(V).
-occurs(close(V), I) :- -holds(opened(V), I).
```

```
68
69 %% ignite causes burner_on.
70 holds(burner_on, I+1) :- occurs(ignite, I),
71                           I < n.
72
73 %% impossible ignite if connected_to_burner(S),
74 %%                       -pressurized(S).
75 -occurs(ignite, I) :- connected_to_burner(S),
76                       -holds(pressurized(S), I).
77
78 %% CWA for Defined Fluents:
79 -holds(F,I) :- fluent(defined,F),
80                step(I),
81                not holds(F,I).
82
83 %% General Inertia Axiom
84 holds(F,I+1) :- fluent(inertial,F),
85                 holds(F,I),
86                 not -holds(F,I+1),
87                 I < n.
88 -holds(F,I+1) :- fluent(inertial,F),
89                  -holds(F,I),
90                  not holds(F,I+1),
91                  I < n.
92
93 %% CWA for Actions
94 -occurs(A,I) :- action(A), step(I),
95                 not occurs(A,I).
96
97 %% ------------------------
98 %% Simple Planning Module:
99 %% ------------------------
100 success :- goal(I),
101            I <= n.
102 :- not success.
103
104 1{occurs(A,I): action(A)}1 :- step(I),
105                               not goal(I),
106                               I < n.
```

```
%% ------------------
%% Initial Situation:
%% ------------------
-holds(burner_on, 0).
-holds(opened(v1),0).
holds(opened(v2),0).

%% -----
%% Goal:
%% -----
goal(I) :- holds(burner_on,I).

%% ------------------
%% Output formatting:
%% ------------------

#show occurs/2.
```

D.2 ASP Encoding of the Missionaries and Cannibals Example

```
%% ------------------------------------------------------
%% crossing.lp
%% ------------------------------------------------------

%% ----------
%% Signature:
%% ----------

%% Steps:
#const n = 11.
step(0..n).

location(bank1).
location(bank2).

%% Number of cannibals/missionaries:
num(0..3).

%% Number of Boats:
```

```
num_boats(0..1).

%% --------
%% Statics:
%% --------

%% opposite bank:
opposite(bank1,bank2).
opposite(bank2,bank1).

%% --------
%% Fluents:
%% --------

%% number of missionaries at location Loc is N:
fluent(inertial, m(Loc, N)) :- location(Loc), num(N).

%% number of cannibals at location Loc is N:
fluent(inertial, c(Loc, N)) :- location(Loc), num(N).

%% number of boats at location Loc is NB:
fluent(inertial, b(Loc, NB)) :- location(Loc), num(NB).

%% true if cannibals outnumber missionaries on the same
%% bank:
fluent(inertial, casualties).

%% --------
%% Actions:
%% --------

%% move NC (a given number of cannibals) and NM
%% (a given number of missionaries) to Dest
%% (a destination):
action(move(NC, NM, Dest)) :- num(NC), num(NM),
                              location(Dest).

%%-----------------------------------
%% Encoding of AL System Description:
```

```
%%-----------------------------------

%% Moving objects increases the number of objects
%% at the destination by the amount moved.

holds(m(Dest, N+NM), I+1) :- holds(m(Dest,N),I),
                             occurs(move(NC,NM,Dest),I),
                             I < n.

holds(c(Dest, N+NC), I+1) :- holds(c(Dest,N),I),
                             occurs(move(NC,NM,Dest),I),
                             I < n.

holds(b(Dest, 1), I+1) :- occurs(move(NC, NM, Dest),I),
                          I < n.

%% The number of missionaries/cannibals at the opposite
%% bank is 3 - number_on_this_bank. The number of boats
%% at the opposite bank is
%% 1-number_of_boats_on_this_bank.

holds(m(Source, 3-N),I) :- holds(m(Dest, N),I),
                           opposite(Source,Dest).

holds(c(Source, 3-N),I) :- holds(c(Dest, N),I),
                           opposite(Source,Dest).

holds(b(Source, 1-NB), I) :- holds(b(Dest,NB),I),
                             opposite(Source,Dest).

%% There cannot be different numbers of the same type
%% of person at the same location.
-holds(m(Loc, N1), I) :- num(N1),
                         holds(m(Loc, N2),I),
                         N1 != N2.

-holds(c(Loc, N1), I) :- num(N1),
                         holds(c(Loc, N2),I),
                         N1 != N2.
```

```
98
99  %% A boat can't be in and not in a location
100 -holds(b(Loc, NB1), I) :- num(NB1),
101                          holds(b(Loc, NB2), I),
102                          NB1 != NB2.
103
104 %% A boat can't be in two places at once.
105 -holds(b(Loc1, N), I) :- location(Loc1),
106                         holds(b(Loc2, N),I),
107                         Loc1 != Loc2.
108
109 %% There will be casualties if cannibals outnumber
    %% missionaries:
110 holds(casualties,I) :- holds(m(Loc, NM),I),
111                        holds(c(Loc, NC),I),
112                        NM > 0, NM < NC.
113
114 %% It is impossible to move more than two people at the
115 %% same time; it is also impossible to move less than
    %% 1 person.
116 -occurs(move(NC,NM,Dest),I) :- num(NC), num(NM),
117                                location(Dest), step(I),
118                                (NC+NM) > 2.
119 -occurs(move(NC,NM,Dest),I) :- num(NC), num(NM),
120                                location(Dest), step(I),
121                                (NM+NC) < 1.
122
123 %% It is impossible to move objects without a boat at
    %% the source.
124 -occurs(move(NC,NM,Dest), I) :- num(NC), num(NM),
125                                 opposite(Source,Dest),
126                                 holds(b(Source,0),I).
127
128 %% It is impossible to move N objects from a source if
129 %% there aren't at least N objects at the source in the
    %% first place.
130 -occurs(move(NC,NM,Dest), I) :- num(NC), num(NM),
131                                 opposite(Source,Dest),
132                                 holds(m(Source,NMSource),I),
```

```
                              NMSource < NM.
-occurs(move(NC,NM,Dest), I) :- num(NC), num(NM),
                              opposite(Source,Dest),
                              holds(c(Source,NCSource),I),
                              NCSource < NC.

%%---------------
%% Inertia Axiom:
%%---------------

holds(F, I+1) :- fluent(inertial, F),
                 holds(F,I),
                 not -holds(F, I+1),
                 I < n.

-holds(F, I+1) :- fluent(inertial, F),
                  -holds(F,I),
                  not holds(F, I+1),
                  I < n.

%%-----------------
%% CWA for Actions:
%%-----------------

-occurs(A,I) :- action(A), step(I),
                not occurs(A,I).

%% ------------------
%% Initial Situation:
%% ------------------

holds(m(bank1,3), 0).
holds(c(bank1,3), 0).
holds(b(bank1,1), 0).
-holds(casualties,0).

%% -----
%% Goal:
%% -----
```

```
goal(I) :-
   -holds(casualties,I),
    holds(m(bank2,3),I),
    holds(c(bank2,3),I).

%% ----------------
%% Planning Module:
%% ----------------

success :- goal(I),
           I <= n.
:- not success.

1{occurs(A,I): action(A)}1 :- step(I),
                              not goal(I),
                              I < n.

#show occurs/2.
```

D.3 ASP Encoding of the Circuit Diagnostic Example

```
%% ------------------------------------------------------
%% circuit.lp
%% ------------------------------------------------------

%% ----------
%% Signature:
%% ----------

%% Components:
comp(r).  %% relay
comp(b).  %% bulb

%% Switches:
switch(s1).
switch(s2).

%% Fluents:

```

```
fluent(inertial, prot(b)).
fluent(inertial, closed(SW)) :- switch(SW).
fluent(inertial, ab(C)) :- comp(C).
fluent(defined, active(r)).
fluent(defined, on(b)).

%% Actions:

action(agent, close(s1)).
action(exogenous, break).
action(exogenous, surge).

action(X) :- action(agent, X).
action(X) :- action(exogenous,X).
#domain action(A).

%% Steps:

#const n = 1.
step(0..n).
#domain step(I).

%% --------------------
%% System Description:
%% --------------------

%% Causal laws:

%% close(s1) causes closed(s1)
holds(closed(s1),I+1) :- occurs(close(s1),I),
                         I < n.

%% break causes ab(b)
holds(ab(b),I+1) :- occurs(break,I),
                    I < n.

%% surge causes ab(r)
holds(ab(r),I+1) :- occurs(surge,I),
                    I < n.
```

```
58
59 %% surge causes ab(b) if -prot(b)
60 holds(ab(b),I+1) :- occurs(surge,I),
61                      -holds(prot(b),I),
62                      I < n.
63
64
65 %% State constraints:
66
67 %% active(r) if closed(s1), -ab(r)
68 holds(active(r),I) :- holds(closed(s1),I),
69                        -holds(ab(r),I).
70
71 %% closed(s2) if active(r)
72 holds(closed(s2),I) :- holds(active(r),I).
73
74 %% on(b) if closed(s2), -ab(b)
75 holds(on(b),I) :- holds(closed(s2),I),
76                   -holds(ab(b),I).
77
78
79 %% Executability conditions:
80
81 %% impossible close(s1) if closed(s1)
82 -occurs(close(s1), I) :- holds(closed(s1),I).
83
84
85 %% CWA for Defined Fluents:
86
87 -holds(F,I) :- fluent(defined,F),
88                not holds(F,I).
89
90 %% General Inertia Axiom:
91
92 holds(F,I+1) :- fluent(inertial,F),
93                 holds(F,I),
94                 not -holds(F,I+1),
95                 I < n.
96
```

```
-holds(F,I+1) :- fluent(inertial,F),
                 -holds(F,I),
                 not holds(F,I+1),
                 I < n.

%% CWA for Actions:

-occurs(A,I) :- not occurs(A,I).

%% --------
%% History:
%% --------

obs(closed(s1),false,0).
obs(closed(s2),false,0).
obs(ab(b),false,0).
obs(ab(r),false,0).
obs(prot(b),true,0).

hpd(close(s1),0).

obs(on(b),false,1).

%% -------
%% Axioms:
%% -------

%% Full Awareness Axiom:
holds(F,0) | -holds(F,0) :- fluent(inertial, F).

%% Take what actually happened into account:
occurs(A,I) :- hpd(A,I).

%% Reality Check:
:- obs(F,true,I), -holds(F,I).
:- obs(F,false,I), holds(F,I).


%% ------------------------
```

```
%% Explanation Generation:
%% ------------------------

{occurs(A,K) : action(exogenous,A)} :- step(K),
                                       K >= 0, K < n.


expl(A,I) :- action(exogenous,A),
             occurs(A,I),
             not hpd(A,I).

#show expl/2.
```