Appendix A Encoding generated by our CNL tool for reald-world use cases

A.1 Nurse Scheduling Problem

CNL specifications reported in Section 5.1 are automatically converted as ASP rules by our tool. The generated encoding is the following:

```
nurse(1..numberOfNurses).
day(1..365).
shift(1,"morning",7).
shift(2,"afternoon",7).
shift(3,"night",10).
shift(4,"specrest",0).
shift(5,"rest",0).
shift(6,"vacation",0).
1 <= {work_in(NURSE,DAY,SHIFT):shift(_,SHIFT,_)} <= 1 :- nurse(NURSE), day(DAY).
:- day(DAY), #count{NURSE: work_in(NURSE,DAY,"morning")} > maxNurseMorning.
:- day(DAY), #count{NURSE: work_in(NURSE,DAY,"afternoon")} > maxNurseAfternoon.
:- day(DAY), #count{NURSE: work_in(NURSE,DAY,"night")} > maxNurseNight.
:- day(DAY), #count{NURSE: work_in(NURSE,DAY,"morning")} < minNurseMorning.
:- day(DAY), #count{NURSE: work_in(NURSE,DAY,"afternoon")} < minNurseAfternoon.
:- day(DAY), #count{NURSE: work_in(NURSE,DAY,"night")} < minNurseNight.
:- nurse(NURSE), #sum{HOURS,DAY: work_in(NURSE,DAY,SHIFT), shift(_,SHIFT,HOURS)}
    > 1692.
:- nurse(NURSE), #sum{HOURS,DAY: work_in(NURSE,DAY,SHIFT), shift(_,SHIFT,HOURS)}
    < 1687.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"vacation")} != 30.
:- shift(SO1,SHIFT1,_), shift(SO2,SHIFT2,_), work_in(NURSE,DAY,SHIFT2),
    work_in(NURSE,DAY+1,SHIFT1), SO1 < SO2, SO2 >= 1, SO2 <= 3, SO1 >= 1, SO1 <=
    6.
:- nurse(NURSE), day(D1), D1 <= 352, #count{D2: work_in(NURSE,D2,"rest"), D2 >=
    D1, D2 <= D1+13} < 2.
:- day(D1), #count{D2: work_in(NURSE,D2,"night"), D2 >= D1, D2 <= D1+1} = 2, not
    work_in(NURSE,D1+2,"specrest"), nurse(NURSE).
:- work_in(NURSE,DAY,"specrest"), day(DAY), #count{SHIFT:
    work_in(NURSE,SHIFT,"night"), SHIFT >= DAY-2, SHIFT <= DAY-1} != 2.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"morning")} > 82.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"afternoon")} > 82.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"night")} > 61.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"morning")} < 74.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"afternoon")} < 74.
:- nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"night")} < 58.
:~ nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"morning")} = TOTDAYS, TOTDAYS
    >= 74, TOTDAYS <= 82, RES = |78 - TOTDAYS|. [RES@3, NURSE]
:~ nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"afternoon")} = TOTDAYS, TOTDAYS
    >= 74, TOTDAYS <= 82, RES = |78 - TOTDAYS|. [RES@3, NURSE]
:~ nurse(NURSE), #count{DAY: work_in(NURSE,DAY,"night")} = TOTDAYS, TOTDAYS >=
    58, TOTDAYS <= 61, RES = |60 - TOTDAYS|. [RES@3, NURSE]
```

A.2 Manipulation of Articulated Objects Using Dual-Arm Robots

CNL specifications reported in Section 5.2 are automatically converted as ASP rules by our tool. The generated encoding is the following:

```
time(1). time(2). time(3). time(4). time(5). time(6). time(7). time(8). time(9). time(10).
link(J2,J1) :- link(J1,J2).
0 <= {rotation(J1,J2,A,AI,time(T)):joint(J1), joint(J2), angle(A), link(J1,J2), position(joint(J1),angle(AI),time(T))} <= 1 :- time(T), T > 0.
:- rotation(_,_,_,_,time(T)), T >= timemax.
:- rotation(J1,J2,_,_,time(_)), J1 <= J2.
:- rotation(_,_,A,AI,time(_)), (A)\360 == (AI)\360.
:- rotation(_,_,A,AI,time(_)), (A)\360 > 0, (AI)\360 > (A)\360, (AI)\360 != (A + granularity)\360.
:- rotation(_,_,A,AI,time(_)), (AI)\360 > 0, (A)\360 > (AI)\360, (A)\360 != (AI + granularity)\360.
:- rotation(_,_,A,0,time(_)), (A)\360 != 360 - granularity.
:- rotation(_,_,0,AI,time(_)), (AI)\360 != 360 - granularity.
1 <= {position(joint(J),angle(A),time(T)):angle(A)} <= 1 :- joint(J), time(T).
:- position(joint(J),angle(A1),time(T)), position(joint(J),angle(A2),time(T+1)), not rotation(_,_,_,_,time(T)), T <= timemax, (A1)\360 != (A2)\360.
:- position(joint(J1),angle(A1),time(T)), rotation(J1,_,A2,_,time(T-1)), (A1)\360 != (A2)\360.
:- time(T), position(joint(J1),angle(AN),time(T+1)), rotation(J2,_,A,AP,time(T)), position(joint(J1),angle(AC),time(T)), J1 > J2, (AN)\360 != (|AC+(A-AP)+360|)\360.
:- position(joint(J1),angle(A1),time(T)), position(joint(J1),angle(A2),time(T+1)), rotation(J2,_,_,_,time(T)), J2 > J1, T <= timemax, (A1)\360 != (A2)\360.
:- goal(joint(J),angle(A1)), position(joint(J),angle(A2),time(timemax)), (A1)\360 != (A2)\360.
```

We want to emphasize here that each angle is automatically followed by `\ 360` which represents the module operation to ensure that an angle is always between 0 and 359 degrees.

A.3 Chemotherapy Treatment Scheduling Problem

CNL specifications reported in Section 5.3 are automatically converted as ASP rules by our tool. The generated encoding is the following:

```
timeslot(1,"07:30"). timeslot(2,"07:40"). timeslot(3,"07:50").
timeslot(4,"08:00"). timeslot(5,"08:10"). timeslot(6,"08:20").
timeslot(7,"08:30"). timeslot(8,"08:40"). timeslot(9,"08:50").
timeslot(10,"09:00"). timeslot(11,"09:10"). timeslot(12,"09:20").
timeslot(13,"09:30"). timeslot(14,"09:40"). timeslot(15,"09:50").
timeslot(16,"10:00"). timeslot(17,"10:10"). timeslot(18,"10:20").
timeslot(19,"10:30"). timeslot(20,"10:40"). timeslot(21,"10:50").
timeslot(22,"11:00"). timeslot(23,"11:10"). timeslot(24,"11:20").
timeslot(25,"11:30"). timeslot(26,"11:40"). timeslot(27,"11:50").
timeslot(28,"12:00"). timeslot(29,"12:10"). timeslot(30,"12:20").
timeslot(31,"12:30"). timeslot(32,"12:40"). timeslot(33,"12:50").
timeslot(34,"13:00"). timeslot(35,"13:10"). timeslot(36,"13:20").
day(1,"01/01/2022").
day(2,"02/01/2022").
day(3,"03/01/2022").
day(4,"04/01/2022").
day(5,"05/01/2022").
```

```
18 day(6,"06/01/2022").
19 day(7,"07/01/2022").
20 1 <= {assignment(registration(patient(R),0), day(D), timeslot(TS)) : day(D,_),
       timeslot(TS,_) } <= 1 :- registration(patient(R),0,_,_,_,_,_).
21 1 <= {assignment(registration(patient(P),OR),day(D+W),timeslot(T)) :
       timeslot(T,_)} <= 1 :- registration(patient(P),OR,W,_,_,_,_),
       assignment(registration(patient(P),OR-1), day(D), timeslot(_)), day(D+W,_).
22 1 <= {position_in(patient(P),S) : seat(S,_)} <= 1 :- patient(P,_),
       assignment(registration(patient(P),_), day(D), timeslot(T)),
       registration(patient(P),_,_,_,_,_,_,PH4), PH4 > 0.
23 position_in(patient(P),S,timeslot(T), day(D)) :-
       _generated_support(patient(P),S,timeslot(T),day(D)),
       position_in(patient(P),S).
24 _generated_support(patient(P),S,timeslot(T..T+PH4-1),day(D)) :-
       position_in(patient(P),S), patient(P,_),
       assignment(registration(patient(P),_), day(D), timeslot(T)),
       registration(patient(P),_,_,_,_,_,_,PH4).
25 :- registration(patient(P),OR,_,PH1,PH2,PH3,_),
       assignment(registration(patient(P),OR),day(_),timeslot(T)), T <= PH1 + PH2 +
       PH3.
26 :- day(D,_), timeslot(TS,_), seat(S,_), #count{P:
       position_in(patient(P),S,timeslot(TS),day(D))} >= 2.
27 :- registration(patient(P),OR,_,_,_,_,PH4),
       assignment(registration(patient(P),OR),day(_),timeslot(TS)), PH4 > 50, TS <
       24.
28 :~ seat(S,T), patient(P,T), position_in(patient(P),S,timeslot(_),day(_)).
       [-1@3, P]
```