```
/* Files: model/model4.pl, model/heart4.pl, model/possible4.pl

   kardio_dm.pl   deep model of the heart, 4th level of detail,
                  standard Prolog, suitable for simulation

   Copyright (c) 1989, Igor Mozetic. All rights reserved.

   Author : Dr. Igor Mozetic
            Austrian Research Institute for Artificial Intelligence
            Schottengasse 3
            A-1010 Vienna, Austria
            Phone: (+43 1) 533-6112
            Email: igor@ai.univie.ac.at

   Updated: 6/29/1990

The user agrees that this program and any derivative works are to be used solely
for academic purposes and are not to be sold, distributed, or commercially
exploited in any manner. As experimental, research software, this program is
provided free of charge on an "as is" basis without warranty of any kind. All
title, ownership and rights to this program and any copies remain with the
author. When publishing any results using this program, this will be properly
acknowledged and the following publications (whichever is relevant) will be
referenced:

   Bratko, I., Mozetic, I., Lavrac, N. KARDIO: A Study in Deep and Qualitative
   Knowledge for Expert Systems. The MIT Press, Cambridge, MA, 1989.

The following is a rational reconstruction of a major part of the heart model
described in the KARDIO book. A comparison to other representations, and
(some) relations to the original KARDIO model are in:

   Mozetic, I. Diagnostic efficiency of deep and surface knowledge in KARDIO.
   Artificial Intelligence in Medicine 2 (2), pp. 67-83, 1990.

   Mozetic, I., Pfahringer, B. Improving diagnostic efficiency in KARDIO:
   abstractions, constraint propagation, and model compilation.
   To appear in Deep Models for Medical Knowledge Engineering
   (E. Keravnou, Ed.), Elsevier, Amsterdam, 1992.


The following are domain specifications (domains are sorted):

domains_arr(arr(SA, Atr, AV, Jun, BB, Reg, Ect)) :-
   domain( SA,  [quiet,sa,sad,sb,sr,st] ),
   domain( Atr, [aeb,af,afl,at,mat,quiet,wp] ),
   domain( AV,  [avb1,avb3,lgl,mob2,normal,wen,wpw] ),
   domain( Jun, [jb,jeb,jr,jt,quiet] ),
   domain( BB,  [lbbb,normal,rbbb] ),
   domain( Reg, [avr,quiet,vf,vfl,vr,vt] ),
   domain( Ect, [quiet,veb] ).

domains_ecg(ecg(Rhythm, P, RateP, P_QRS, PR, QRS, Rate, EctP, EctPR, EctQRS)) :-
   domain( Rhythm,[irregular,regular] ),
   domain( P,     [abnormal,absent,changing,normal] ),
   domain( RateP, [between_100_250,between_250_350,between_60_100,
                   over_350,under_60,zero] ),
   domain( P_QRS, [after_P_always_QRS,after_P_some_QRS_miss,
                   independent_P_QRS,meaningless] ),
   domain( PR,    [after_QRS_is_P,changing,meaningless,normal,
                   prolonged,shortened] ),
   domain( QRS,   [absent,delta_LBBB,delta_RBBB,normal,wide_LBBB,
                   wide_LBBB_RBBB,wide_RBBB] ),
   domain( Rate,  [between_100_250,between_250_350,between_60_100,
                   over_350,under_60] ),
   domain( EctP,  [abnormal,absent] ),
   domain( EctPR, [after_QRS_is_P,meaningless,normal,prolonged,shortened] ),
   domain( EctQRS,[absent,delta_LBBB,delta_RBBB,normal,wide_LBBB,
                   wide_LBBB_RBBB,wide_RBBB] ).

********************************************************************************
*/

%%
%% File: model4.pl
%%
%  Model of the heart at the most detailed, 4th level.
%  Fast for simulation, slower for diagnosis,
%  can be used to generate a complete Arr-ECG relational table.
%  Non-redundant specification, relates 943 Arr to 3096 ECG, 5240 pairs.
/*
?- sample_arr(4, Arr ), simulate( Arr, ECG ).
ECG = ecg(regular,abnormal,between_100_250,after_P_always_QRS,shortened,
          normal,between_100_250,abnormal,shortened,normal) ;
ECG = ecg(regular,abnormal,between_100_250,after_P_always_QRS,shortened,
          normal,between_100_250,abnormal,after_QRS_is_P,normal) ;
ECG = ecg(regular,abnormal,between_100_250,after_P_always_QRS,shortened,
          normal,between_100_250,absent,meaningless,normal) ;
no

?- sample_ecg(4, ECG ), simulate( Arr, ECG ).
Arr = arr(quiet,at,normal,jeb,normal,quiet,quiet) ;
Arr = arr(quiet,at,lgl,jeb,normal,quiet,quiet) ;
no

?- simulate( Arr, ECG ).
Arr = arr(sa,aeb,avb1,quiet,normal,quiet,quiet),
ECG = ecg(irregular,normal,between_60_100,after_P_always_QRS,prolonged,
          normal,between_60_100,abnormal,prolonged,normal) ;
Arr = arr(sa,aeb,avb1,quiet,lbbb,quiet,quiet),
ECG = ecg(irregular,normal,between_60_100,after_P_always_QRS,prolonged,
          wide_LBBB,between_60_100,abnormal,prolonged,wide_LBBB) ;
   ...
*/

sample_ecg(4, ecg(regular, abnormal, between_100_250, after_P_always_QRS,
   shortened, normal, between_100_250, abnormal, after_QRS_is_P, normal)).

sample_arr(4, arr(quiet,at,lgl,jeb,normal,quiet,quiet)).

mem( X, [X|_] ).
mem( X, [_|L] ) :- mem( X, L ).


simulate( Arr, ECG ) :-
   possible( Arr ),      % possible4.pl
   heart( Arr, ECG ).    % heart4.pl


%%
%% File: heart4.pl
%%
%  Deep model of the heart, 4th level of detail.

heart( arr(SA, AF, AV, JUN, BB, VF, VEF),
       ecg(Rhythm, P, RateP, P_QRS, PR, QRS, Rate, EctP, EctPR, EctQRS) ) :-
   gen_sa_4( SA, ImpSA ),
   gen_atr_4( AF, ImpAF ),
   summ_4( ImpSA, ImpAF, ImpATR ),
   cond_ant_4( AV, ImpATR, ImpAV ),
   gen_jun_4( JUN, ImpJUN ),
   gen_reg_4( VF, ImpVF ),
   gen_ect_4( VEF, ImpVF ),
   summ_4( ImpJUN, ImpVF, ImpIV ),
   cond_ret_4( AV, ImpIV, ImpRET ),
   summ_4( ImpAV, ImpIV, ImpINV ),
   summ_4( ImpRET, ImpATR, ImpSV ),
   summ_4( ImpAV, ImpJUN, ImpHIS ),
   cond_bb_4( BB, ImpHIS, ImpBB ),
   summ_4( ImpBB, ImpVF, ImpVENT ),
   atr_ecg_4( ImpSV, P, RateP, EctP ),
   av_ecg_4( ImpSV, ImpINV, P_QRS, PR, EctPR ),
   vent_ecg_4( ImpVENT, Rhythm, QRS, Rate, EctQRS ).


%  Impulse generation.

gen_sa_4( quiet, form(reg(sa_node, none,      zero),
                      ect(sa_node, no)) ).
gen_sa_4( sr,    form(reg(sa_node, regular,   between_60_100),
                      ect(sa_node, no)) ).
gen_sa_4( sb,    form(reg(sa_node, regular,   under_60),
                      ect(sa_node, no)) ).
gen_sa_4( st,    form(reg(sa_node, regular,   between_100_250),
                      ect(sa_node, no)) ).
gen_sa_4( sad,   form(reg(sa_node, irregular, between_60_100),
                      ect(sa_node, no)) ).
gen_sa_4( sad,   form(reg(sa_node, irregular, under_60),
                      ect(sa_node, no)) ).
gen_sa_4( sa,    form(reg(sa_node, irregular, between_60_100),
                      ect(sa_node, no)) ).
gen_sa_4( sa,    form(reg(sa_node, irregular, under_60),
                      ect(sa_node, no)) ).
```

```
gen_atr_4( quiet, form(reg(atr_focus, none,      zero),
                       ect(atr_focus, no)) ).
gen_atr_4( wp,    form(reg(wandering, irregular, between_60_100),
                       ect(atr_focus, no)) ).
gen_atr_4( at,    form(reg(atr_focus, regular,   between_100_250),
                       ect(atr_focus, no)) ).
gen_atr_4( mat,   form(reg(wandering, irregular, between_100_250),
                       ect(atr_focus, no)) ).
gen_atr_4( afl,   form(reg(flutter,   regular,   between_250_350),
                       ect(atr_focus, no)) ).
gen_atr_4( af,    form(reg(fibflut,   irregular, over_350),
                       ect(atr_focus, no)) ).
gen_atr_4( af,    form(reg(fibrill,   irregular, over_350),
                       ect(atr_focus, no)) ).
gen_atr_4( aeb,   form(reg(atr_focus, none,      zero),
                       ect(atr_focus, uni)) ).

gen_jun_4( quiet, form(reg(junction, none,    zero),
                       ect(junction, no)) ).
gen_jun_4( jb,    form(reg(junction, regular, under_60),
                       ect(junction, no)) ).
gen_jun_4( jr,    form(reg(junction, regular, between_60_100),
                       ect(junction, no)) ).
gen_jun_4( jt,    form(reg(junction, regular, between_100_250),
                       ect(junction, no)) ).
gen_jun_4( jeb,   form(reg(junction, none,    zero),
                       ect(junction, uni)) ).


gen_reg_4( quiet, form( reg(ventric, none,      zero), _) ).
gen_reg_4( vr,    form( reg(ventric, regular,   under_60), _) ).
gen_reg_4( avr,   form( reg(ventric, regular,   between_60_100), _) ).
gen_reg_4( vt,    form( reg(ventric, regular,   between_100_250), _) ).
gen_reg_4( vfl,   form( reg(ventric, regular,   between_250_350), _) ).
gen_reg_4( vf,    form( reg(ventric, irregular, over_350), _) ).

gen_ect_4( quiet, form( _, ect(ventric, no)) ).
gen_ect_4( veb,   form( _, ect(ventric, uni)) ).


%  AV impulse conduction.

cond_ant_4( AV, form(RegATR, EctATR), form(RegAV, EctAV) ) :-
   ant_reg_4( AV, RegATR, RegAV ),
   ant_ect_4( AV, EctATR, EctAV ).

ant_ect_4( AV,   ect(Loc, no),  ect(Loc,  no) ).
ant_ect_4( avb3, ect(Loc, uni), ect(Loc,  no) ).
ant_ect_4( AV,   ect(_,   uni), ect(Cond, uni) ) :-
   ante_4( AV, Cond ).

   ante_4( normal,normal ).
   ante_4( avb1,  delayed ).
   ante_4( wen,   prog_delay ).
   ante_4( mob2,  part_block ).
   ante_4( wpw,   kent ).
   ante_4( lgl,   james ).

ant_reg_4( AV, reg(_,    Rhythm1, Rate1), reg(Cond, Rhythm2, Rate2) ) :-
   anterograd_4( AV, Cond,       Rhythm1,   Rhythm2,   Rate1,     Rate2 ).

anterograd_4( normal,normal,     Rhythm,    Rhythm,    Rate,      Rate ) :-
   mem( Rate, [zero, under_60, between_60_100, between_100_250] ).
anterograd_4( avb1,  delayed,    Rhythm,    Rhythm,    Rate,      Rate ) :-
   mem( Rate, [under_60, between_60_100, between_100_250] ).
anterograd_4( wen,   prog_delay, regular,   irregular, Rate0,     Rate ) :-
   reduced_4( Rate0, Rate ).
anterograd_4( wen,   prog_delay, irregular, irregular, Rate0,     Rate ) :-
   reduced_4( Rate0, Rate ).
anterograd_4( mob2,  part_block, Rhythm,    Rhythm,    Rate0,     Rate ) :-
   reduced_4( Rate0, Rate ).
anterograd_4( avb3,  _,          _,         none,      Rate,      zero ) :-
   non_zero_4( Rate ).
anterograd_4( wpw,   kent,       Rhythm,    Rhythm,    Rate,      Rate ) :-
   non_zero_4( Rate ).
anterograd_4( lgl,   james,      Rhythm,    Rhythm,    Rate,      Rate ) :-
   mem( Rate, [under_60, between_60_100, between_100_250] ).
anterograd_4( normal,normal,     irregular, irregular, over_350,  Rate ) :-
   mem( Rate, [under_60, between_60_100, between_100_250] ).
anterograd_4( normal,normal,     regular,   irregular, between_250_350, Rate ) :-
   mem( Rate, [under_60, between_60_100, between_100_250] ).
anterograd_4( normal,normal,     regular,   regular,   between_250_350, Rate ) :-
   mem( Rate, [under_60, between_60_100, between_100_250] ).


%  Retrograd AV impulse conduction.

cond_ret_4( AV, form(RegIV, EctIV), form(RegRET, EctRET) ) :-
   ret_reg_4( AV, RegIV, RegRET ),
   ret_ect_4( AV, EctIV, EctRET ).

ret_reg_4( avb3, reg(Loc, _, _    ),        reg(Loc,   none, zero)  ).
ret_reg_4( _   , reg(Loc, _, Rate),         reg(Loc,   none, zero)  ) :-
   mem( Rate, [zero, between_250_350, over_350] ).
ret_reg_4( AV,   reg(Loc,   Rhythm, Rate), reg(Retro, Rhythm, Rate) ) :-
   mem( AV, [normal, avb1, wen, mob2, wpw, lgl] ),
   mem( Rate, [under_60, between_60_100, between_100_250] ),
   retrograd_4( Loc,      Retro ).

ret_ect_4( AV,   ect(Loc, no),  ect(Loc,   no) ).
ret_ect_4( avb3, ect(Loc, uni), ect(Loc,   no) ).
ret_ect_4( AV,   ect(Loc, uni), ect(Retro, uni) ) :-
   mem( AV, [normal, avb1, wen, mob2, wpw, lgl] ),
   retrograd_4( Loc,      Retro ).

   retrograd_4( junction, faster ).
   retrograd_4( junction, slower ).
   retrograd_4( junction, equal  ).
   retrograd_4( ventric,  equal  ).
   retrograd_4( ventric,  slower ).


%  Impulse conduction through bundle branches.

cond_bb_4( BB, form(RegHIS, EctHIS), form(RegBB, EctBB) ) :-
   bb_reg_4( BB, RegHIS, RegBB ),
   bb_ect_4( BB, EctHIS, EctBB ).

bb_reg_4( BB, reg(Loc,      none,   zero), reg(Loc,      none,   zero) ).
bb_reg_4( BB, reg(His_cond, Rhythm, Rate), reg(BB_cond,  Rhythm, Rate) ) :-
   non_zero_4( Rate ),
   bb_conduct_4( BB,     His_cond, BB_cond ).

bb_ect_4( BB, ect(Loc,      no),  ect(Loc,     no) ).
bb_ect_4( BB, ect(His_cond, uni), ect(BB_cond, uni) ) :-
   bb_conduct_4( BB,     His_cond, BB_cond ).

   bb_conduct_4( _,      kent,     kent       ).
   bb_conduct_4( normal, His_cond, normal     ) :-
      mem( His_cond, [normal, james, delayed, prog_delay, part_block, junction] ).
   bb_conduct_4( lbbb,   His_cond, lbb_block  ) :-
      mem( His_cond, [normal, james, delayed, prog_delay, part_block, junction] ).
   bb_conduct_4( rbbb,   His_cond, rbb_block  ) :-
      mem( His_cond, [normal, james, delayed, prog_delay, part_block, junction] ).


%  SV impulse projection on ECG (P).

atr_ecg_4( form(Reg, Ect), P, Prate, EctP ) :-
   atr_reg_4( Reg, P, Prate ),
   atr_ect_4( Ect, EctP ).

atr_reg_4( reg(_,  none, zero), absent, zero ).
atr_reg_4( reg(SV, _,    Rate), P,      Prate ) :-
   non_zero_4( Rate ),
   supra_vent_4( SV,        P,        Rate, Prate ).

atr_ect_4( ect(_,  no),  absent ).
atr_ect_4( ect(SV, uni), P ) :-
   supra_vent_4( SV,        P,        _,    _ ).

   supra_vent_4( sa_node,   normal,   Rate, Rate   ).
   supra_vent_4( atr_focus, abnormal, Rate, Rate   ).
   supra_vent_4( wandering, changing, Rate, Rate   ).
   supra_vent_4( flutter,   abnormal, Rate, Rate   ).
   supra_vent_4( fibflut,   abnormal, Rate, Rate   ).
   supra_vent_4( fibrill,   absent,   _,    zero   ).
   supra_vent_4( faster,    abnormal, Rate, Rate   ).
   supra_vent_4( equal,     absent,   _,    zero   ).
   supra_vent_4( slower,    abnormal, Rate, Rate   ).


% SV and AV impulse projection on ECG (P_QRS, PR).

```

```
av_ecg_4( form(RegSV, EctSV), form(RegAV, EctAV), P_QRS, PR, EctPR ) :-
   av_reg_4( RegSV, RegAV, P_QRS, PR ),
   av_ect_4( EctSV, EctAV, EctPR ).

av_reg_4( reg(_, _,  zero), _,             meaningless, meaningless ).
av_reg_4( reg(SV, _, Rate), reg(AV, _, _), P_QRS,       PR ) :-
   non_zero_4( Rate ),
   his_bundle_4( AV,      SV,        PR,          P_QRS      ).

av_ect_4( ect(_,  no),  _,           meaningless ).
av_ect_4( ect(_,  uni), ect(_,  no),  meaningless ).
av_ect_4( ect(SV, uni), ect(AV, uni), PR ) :-
   his_bundle_4( AV,      SV,        PR,          _ ).

his_bundle_4( delayed,         _,    prolonged,   after_P_always_QRS ).
his_bundle_4( prog_delay,      _,    prolonged,   after_P_some_QRS_miss   ).
his_bundle_4( part_block,      _,    normal,      after_P_some_QRS_miss   ).
his_bundle_4( james,           _,    shortened,   after_P_always_QRS ).
his_bundle_4( kent,       sa_node,   shortened,   after_P_always_QRS ).
his_bundle_4( kent,       atr_focus, shortened,   after_P_always_QRS ).
his_bundle_4( kent,       wandering, shortened,   after_P_always_QRS ).
his_bundle_4( kent,       flutter,   meaningless, meaningless    ).
his_bundle_4( kent,       fibflut,   meaningless, meaningless    ).
his_bundle_4( kent,       fibrill,   meaningless, meaningless    ).
his_bundle_4( normal,     sa_node,   normal,      after_P_always_QRS ).
his_bundle_4( normal,     fibrill,   meaningless, meaningless    ).
his_bundle_4( normal,     fibflut,   meaningless, meaningless    ).
his_bundle_4( normal,     fibflut,   meaningless, independent_P_QRS   ).
his_bundle_4( normal,     fibflut,   meaningless, after_P_some_QRS_miss   ).
his_bundle_4( normal,     flutter,   meaningless, after_P_some_QRS_miss   ).
his_bundle_4( normal,     wandering, normal,      after_P_always_QRS ).
his_bundle_4( normal,     wandering, changing,    after_P_always_QRS ).
his_bundle_4( normal,     atr_focus, normal,      after_P_always_QRS ).
his_bundle_4( normal,     atr_focus, shortened,   after_P_always_QRS ).
his_bundle_4( normal,     atr_focus, prolonged,   after_P_always_QRS ).
his_bundle_4( junction,   faster,    shortened,   after_P_always_QRS ).
his_bundle_4( AV,         equal,     meaningless, meaningless ) :-
   mem( AV, [junction, ventric] ).
his_bundle_4( AV,         slower,    after_QRS_is_P, after_P_always_QRS ) :-
   mem( AV, [junction, ventric] ).
his_bundle_4( AV,         SV,        meaningless, independent_P_QRS ) :-
   mem( AV, [junction, ventric] ),
   mem( SV, [sa_node, atr_focus, wandering, flutter, fibflut, fibrill] ).


%  VENT impulse projection on ECG (Rhythm, Rate, QRS).

vent_ecg_4( form(Reg, Ect), Rhythm, QRS, Rate, EctQRS ) :-
   vent_reg_4( Reg, Rhythm, QRS, Rate ),
   vent_ect_4( Ect, EctQRS ).

vent_reg_4( reg(_,    Rhythm, over_350), Rhythm, absent, over_350 ).
vent_reg_4( reg(VENT, Rhythm, Rate),     Rhythm, QRS,    Rate ) :-
   mem( Rate, [under_60, between_60_100, between_100_250, between_250_350] ),
   ventricles_4( VENT,       QRS ).

vent_ect_4( ect(_,    no),   absent ).
vent_ect_4( ect(VENT, uni),  QRS ) :-
   ventricles_4( VENT,       QRS ).

   ventricles_4( normal,     normal  ).
   ventricles_4( lbb_block,  wide_LBBB  ).
   ventricles_4( rbb_block,  wide_RBBB  ).
   ventricles_4( ventric,    wide_RBBB  ).
   ventricles_4( ventric,    wide_LBBB  ).
   ventricles_4( ventric,    wide_LBBB_RBBB ).
   ventricles_4( kent,       delta_LBBB ).
   ventricles_4( kent,       delta_RBBB ).


%  Impulse summation.

summ_4( form(Reg1,Ect1), form(Reg2,Ect2), form(Reg,Ect) ) :-
   summ_reg_4( Reg1, Reg2, Reg ),
   summ_ect_4( Ect1, Ect2, Ect ).

summ_reg_4( reg(_,   none,   zero), Impulse, Impulse ).
summ_reg_4( reg(Loc, Rhythm, Rate), reg(_, none, zero),
            reg(Loc, Rhythm, Rate) ) :-
   non_zero_4( Rate ).

summ_ect_4( ect(Loc, no),  ect(_,   no),  ect(Loc, no) ).
summ_ect_4( ect(Loc, uni), ect(_,   no),  ect(Loc, uni) ).
summ_ect_4( ect(_,   no),  ect(Loc, uni), ect(Loc, uni) ).


%  Utilities

reduced_4( Rate0,           Rate0 ) :-
   mem( Rate0, [under_60, between_60_100, between_100_250] ).
reduced_4( between_60_100,  under_60 ).
reduced_4( between_100_250, between_60_100 ).

non_zero_4( Rate ) :-
   mem( Rate, [under_60,between_60_100,between_100_250,between_250_350,over_350]).

%%
%% File: possible4.pl
%%
%  Constraints which eliminate physiologically impossible
%  and medically uninteresting arrhythmias at level 4.
%  Non-redundant (943 Arr), nonground (BB,Ect) specification.

possible(arr(SA,   aeb,  AV,    quiet,BB,    quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(AV, [avb1,lgl,mob2,wen]).
possible(arr(SA,   aeb,  avb3,  Jun,  BB,    quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(Jun,[jb,jr,jt]).
possible(arr(SA,   aeb,  avb3,  quiet,normal,Reg,  quiet)) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(Reg,[avr,vr,vt]).
possible(arr(SA,   aeb,  normal,quiet,BB,    quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]).
possible(arr(SA,   aeb,  wpw,   quiet,normal,quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]).
possible(arr(SA,   quiet,AV,    jeb,  BB,    quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(AV, [avb1,lgl,mob2,wen]).
possible(arr(SA,   quiet,AV,    quiet,BB,    quiet,Ect)  ) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(AV, [avb1,lgl,mob2,wen]).
possible(arr(SA,   quiet,avb3,  Jun,  BB,    quiet,Ect)  ) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(Jun,[jb,jr,jt]).
possible(arr(SA,   quiet,avb3,  jeb,  BB,    Reg,  quiet)) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(Reg,[avr,vr,vt]).
possible(arr(SA,   quiet,avb3,  quiet,normal,Reg,  Ect)  ) :- mem(SA, [sa,sad,sb,sr,st]),
                                                              mem(Reg,[avr,vr,vt]).
possible(arr(SA,   quiet,normal,jeb,  BB,    quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]).
possible(arr(SA,   quiet,normal,quiet,BB,    quiet,Ect)  ) :- mem(SA, [sa,sad,sb,sr,st]).
possible(arr(SA,   quiet,wpw,   jeb,  BB,    quiet,quiet)) :- mem(SA, [sa,sad,sb,sr,st]).
possible(arr(SA,   quiet,wpw,   quiet,normal,quiet,Ect)  ) :- mem(SA, [sa,sad,sb,sr,st]).
possible(arr(quiet,Atr,  avb3,  Jun,  BB,    quiet,Ect)  ) :- mem(Atr,[af,afl,at,mat,wp]),
                                                              mem(Jun,[jb,jr,jt]).
possible(arr(quiet,Atr,  avb3,  jeb,  BB,    Reg,  quiet)) :- mem(Atr,[af,afl,at,mat,wp]),
                                                              mem(Reg,[avr,vr,vt]).
possible(arr(quiet,Atr,  avb3,  quiet,normal,Reg,  Ect)  ) :- mem(Atr,[af,afl,at,mat,wp]),
                                                              mem(Reg,[avr,vr,vt]).
possible(arr(quiet,Atr,  normal,jeb,  BB,    quiet,quiet)) :- mem(Atr,[af,afl,at,mat,wp]).
possible(arr(quiet,Atr,  normal,quiet,BB,    quiet,Ect)  ) :- mem(Atr,[af,afl,at,mat,wp]).
possible(arr(quiet,Atr,  wpw,   quiet,normal,quiet,quiet)) :- mem(Atr,[af,afl]).
possible(arr(quiet,Atr,  AV,    jeb,  BB,    quiet,quiet)) :- mem(Atr,[at,mat,wp]),
                                                              mem(AV, [avb1,lgl,mob2,wen]).
possible(arr(quiet,Atr,  AV,    quiet,BB,    quiet,Ect)  ) :- mem(Atr,[at,mat,wp]),
                                                              mem(AV, [avb1,lgl,mob2,wen]).
possible(arr(quiet,Atr,  wpw,   jeb,  BB,    quiet,quiet)) :- mem(Atr,[at,mat,wp]).
possible(arr(quiet,Atr,  wpw,   quiet,normal,quiet,Ect)  ) :- mem(Atr,[at,mat,wp]).
possible(arr(quiet,aeb,  normal,Jun,  BB,    quiet,quiet)) :- mem(Jun,[jb,jr,jt]).
possible(arr(quiet,aeb,  normal,quiet,BB,    Reg,  quiet)) :- mem(Reg,[avr,vr,vt]).
possible(arr(quiet,quiet,normal,Jun,  BB,    quiet,Ect)  ) :- mem(Jun,[jb,jr,jt]).
possible(arr(quiet,quiet,normal,jeb,  BB,    Reg,  quiet)) :- mem(Reg,[avr,vr,vt]).
possible(arr(quiet,quiet,normal,quiet,normal,Reg,  Ect)  ) :- mem(Reg,[avr,vr,vt]).
possible(arr(quiet,quiet,normal,quiet,normal,Reg,  quiet)) :- mem(Reg,[vf,vfl]).
```