FOL Burden

Luis

December 2025

0.1 Burden of Proof and discretionary evaluation formulation

Eligibility function. The main premise is whether the person qualifies under legal rules. First, the supporting information needs to meet Burden of Production and then Burden of Persuasion. Therefore, the first step would be modeling ”enough evidence to have the issue decided on merits”. This means based on the substance on of the claim, not a threshold defect. Some examples of not meeting are didn't submit the documents, the form was incomplete, there is simply no evidence for an essential element (too thin to analyze). In other words, they need enough relevant evidence that the AAO can reach a reach a conclusion. The second Burden evaluates that based on evidence, is it more likely than not that the element is satisfied.

Let X denote the set of cases. For each eligibility element i, let

$$p_i : X \to [0, 1],$$
$$p_i(x) := \Pr(E_i(x) \mid \mathrm{record}(x)),$$

be a subjective probability that element i is satisfied in case x given its evidentiary record.

Define the threshold predicate

$$T_i(x) :\Leftrightarrow p_i(x) > 0.5,$$

expressing that it is more likely than not that element i holds in case x.

Let $k_i(x)$ be the burden-of-production predicate for element i, which is true iff there is sufficient probative evidence in the record for element i to be decided on the merits. The element-level eligibility predicate is then

$$E_i(x) :\Leftrightarrow k_i(x) \wedge T_i(x).$$

Let $\{E_1, \ldots, E_m\}$ be the set of element-level predicates as defined above. For each intermediate legal goal j, we define a predicate $R_j(x)$ by a rule whose body is a logical combination of atoms $E_i(x)$ and possibly other intermediate predicates $R_{j'}(x)$.

Finally, the legal-eligibility-level predicate $L(x)$, representing that case x satisfies all non-discretionary legal conditions under the applicable standard of proof, is defined by a propositional formula ψ:

$$L(x) :\Leftrightarrow \psi\big(E_1(x), \ldots, E_m(x), R_1(x), \ldots, R_r(x)\big),$$

where ψ encodes the structure of the applicable legal rules.

0.1.1 Discretionary modeling and exceptions to preponderance of evidence

There are cases where preponderance of evidence is not required. I'm still trying to find the cases where other standard is used. For now, other cases are modeled using $L^*(x)$. I'm defining this as a rule and a exception to adjust to proleg notation.

$$\begin{aligned} S(x) :\Leftrightarrow &\ \big(L(x) \wedge \neg AException(x)\big) & (1)\\ \vee &\ \ \big(L^*(x) \wedge AException(x)\big). & (2) \end{aligned}$$

$$AException(x) := \text{“for case } x,$$

the law specifies a standard different from preponderance of evidence.”

Important note: $AException(x)$ is a wrapper for $OtherStandardApplies(x)$, when a different standard is specified at a benefit or issue level; we will consider it an attribute of the case, for the size of the statutory and benefit sets.

Finally, the case could be decided as a matter of discretion; which depends on the requested benefit. If $B_b(x)$ is a benefit-level predicate meaning ”case x qualifies for benefit b”, then:

$$\begin{aligned} B_b(x) :\Leftrightarrow & & (3)\\ \big(S(x) \wedge \neg DException(x, b) \wedge D(x, b)\big) & & (4)\\ \vee \big(S(x) \wedge DException(x, b)\big). & & (5) \end{aligned}$$

Here, $D(x, b)$ is a predicate indicating that, for case x, the discretionary factors relevant to benefit b favor granting it (positive factors outweigh the negative ones). $DException(x, b)$ is a predicate indicating that, for case x, there is an exception to the discretionary requirement for benefit b. In such cases, the appellant is not required to show that positive factors outweigh negative ones, and no discretionary balancing is performed. For instance, when $DException(x, b)$ is false, the benefit is discretionary and favorable discretionary assessment $D(x, b)$ is required.

The notation b could be removed if the benefit is part of the description of the case x and there is only one benefit per case (check this).

We can simplify the last formula to:

$$\begin{aligned} B_b(x) :\Leftrightarrow & & (6)\\ S(x, b) \wedge \big(D(x, b) \vee DException(x, b)\big). & & (7) \end{aligned}$$

0.1.2 Discretionary term formulation

The discretionary predicate $D(x,b)$ weights "discretionary factors" $D(x,b) \in [-1,1]$ and compares it to 0 to find the truth value such that:

$$D(x,b) :\Leftrightarrow W(x,b) > 0,$$

where

$$W(x,b) = \sum_{j=1}^{k} w_j^+ s_x(F_j^+) \tag{8}$$

$$-\sum_{l=1}^{m} w_l^- s_x(F_l^-) + w_0 C_{eligibility}, \tag{9}$$

$$\text{and confidence factor} \quad C_{elegibility} = \min_i p_i$$

Weight terms $w \in [0,1]$ could be functions of the factor or merely a constant product of interpretation, calculation, or expert opinion. The function $s_x \in [0,1]$ over the factor F is a subjective evaluation for each case regarding how strong the factor is.

0.2 Language model evaluation function

A language model with evaluation function $M_{func}(x)$ for a case x, can produce the following terms.

$$M_{func}(x) \in \{p_i, s_x\}$$

A predicate $M_{pred}(x)$ evaluated by a language model can be operationally (but not perfectly) defined as follows.

$$k_i(x) :\Leftrightarrow M_{pred}(x),$$

$$DException(x,b) :\Leftrightarrow M_{pred}(x), \text{and}$$

$$AException(x) :\Leftrightarrow M_{pred}(x)$$

Finally, the language model will generate a clauses from the rules that are analyzed in the case x, if they are not already modeled.

0.3 Proleg formulation

Simple case example (not real)

```
threshold(preponderance, 0.5).

eligibility_psi(benefit_approved, [higher_education,
↪  interest_for_country(chile)]).

interest_for_country(chile) <=
    from_country_approved(chile),
    selected_interest.

fact(higher_education).
probative(higher_education).
prob(higher_education, 0.6).

fact(from_country_approved(chile)).
probative(from_country_approved(chile)).
prob(from_country_approved(chile), 1).

fact(selected_interest).
probative(selected_interest).
prob(selected_interest, 0.9).

discretionary_evaluation_necessary.
discretionary_factors(
    factors{
        hardship: points{w:[1.0], s:[2.0]},
        merit:    points{w:[1.0], s:[0.5]}
    }).
```

Adapted proleg code

```
:- encoding(utf8).
:- discontiguous (<=)/2.
:- discontiguous fact/1.
:- discontiguous exception/2.
:- discontiguous probative/1.
:- discontiguous prob/2.

:- op(1100, xfx, user:(<=)).
:- dynamic (<=)/2.
:- multifile (<=)/2.
:- dynamic fact/1.
:- dynamic exception/2.
:- dynamic message/1.
:- dynamic prob/2.
:- dynamic probative/1.
:- dynamic threshold/2.
:- use_module(library(prolog_code)).  % comma_list/2
:- dynamic eligibility_psi/2.
:- use_module(library(apply)).        % foldl/4
:- dynamic discretionary_evaluation_necessary/0.
:- dynamic discretionary_factors/1.   % discretionary_factors
:- initialization(init_defaults).
:- set_prolog_flag(print_write_options, [portray(true), quoted(false),
↪  numbervars(true)]).

r :- reconsult(proleg).

answer(Query):-
       init_case_standard,
       solve(0,Query).

messageon  :- retractall(message(_)), assertz(message(on)).
messageoff :- retractall(message(_)), assertz(message(off)).

message(on).

%% All print definitions

print_message(_,_,_):-message(off),!.
print_message(I,(P<=Q),M):-message(on),!,
    print_rule(I,(P<=Q)),print(' '),
    print(M),nl.
print_message(I,no_counter_argument(Rel),_):-message(on),!,
    tab(I),print('No Exception:'), print(Rel),nl.
print_message(I,Q,M):-message(on),!,
    tab(I),print(Q),print(' '),print(M),nl.

```

```
print_rule(I,(P<=Q)):-!,
    tab(I),
    print(P),print('<='),nl,
    I2 is I + 2,
    print_body(I2,Q).

print_body(I,(B1,B)):-!,
    tab(I),print(B1),print(','),nl,
    print_body(I,B).
print_body(I,B1):-!,
    tab(I),print(B1).


%%%%%% Solve predicates

solve(I,(Goal,RestGoal)):-!,
    solve(I,Goal), solve(I,RestGoal).

solve(I, call(P)) :-
    !,
    I2 is I + 2,
    nb_setval(current_indent, I2),
    print_message(I2, call(P), 'calling...'),
    (   call(P)
    ->  print_message(I2, call(P), 'succeeded.')
    ;   print_message(I2, call(P), 'FAILED.'), fail
    ).

solve(I, Goal) :-
    eligibility_psi(Goal, Goals),
    !,
    I2 is I + 2,
    print_message(I2, eligibility_psi(Goal), 'expanding eligibility.'),
    comma_list(Body, Goals),
    solve(I2, Body),
    print_message(I2, Goal, 'eligibility succeeded.'),
    print_message(I2, 'Exception check for', Goal),
    \+ succesful_exception(I2, Goal),
    print_message(I2, 'No Exception for', Goal),
    print_message(I2, no_counter_argument(Goal), 'succeeded.').

solve(I,Goal):-
    I2 is I + 2,
    is_fact(Goal),!,
    fact_check(I2,Goal).
solve(I,Goal):-
    I2 is I + 2,
    print_message(I2,'Starting to prove:',Goal),fail.
```

```
solve(I,Goal):-
%     print_message(I,(Goal <= Body),'is now checked.'),
    (Goal <= Body),
    I2 is I + 2,
    print_message(I2,(Goal <= Body),'found.'),
    solve(I2,Body),
    print_message(I2,Goal,'succeeded.'),
    print_message(I2,'Exception check for',Goal),
    \+ succesful_exception(I2,Goal),
    print_message(I2,'No Exception for',Goal),
    print_message(I2,no_counter_argument(Goal),'succeeded.').
solve(I,Goal):-
    I2 is I + 2,
    print_message(I2,'Failed to prove',Goal),!,fail.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Threshold initialization
%%%%%%%%%%%%%%%%%%

init_case_standard :-
    nb_setval(case_standard, preponderance).

current_threshold(Std, T) :-
    nb_getval(case_standard, Std), %nb_current/2 quiet fail test later
    threshold(Std, T).

%%%%%%%%%%%%%%%%%

fact_check(I2, Goal) :-
    (   fact(Goal)
    ->  print_message(I2, fact(Goal), 'found.')
    ;   print_message(I2, fact(Goal), 'FAILED: no such fact.'), !, fail
    ),

    (   production_met(Goal)
    ->  print_message(I2, production_met(Goal), 'burden of production met.')
    ;   print_message(I2, production_met(Goal), 'FAILED: burden of production not
    ↪   met.'), !, fail
    ),

    current_threshold(Std, T),
    print_message(I2, standard(Std), threshold(T)),

    (   persuasion_met(Goal, P)
    ->  print_message(I2, persuasion_met(Std, Goal, P), 'burden of persuasion
    ↪   met.')
    ;   report_persuasion_failure(I2, Goal), !, fail
    ).

report_persuasion_failure(I, Goal) :-
    current_threshold(Std, T),
    (   prob(Goal, P), number(P)
    ->  print_message(I, prob(Goal, P), failed_threshold(Std, T))
    ;   prob(Goal, P)
    ->  print_message(I, prob(Goal, P), 'FAILED: probability is not numeric.')
    ;   print_message(I, prob(Goal, _), 'FAILED: no probability in record.')
    ).

succesful_exception(I,Goal):-
    exception(Goal,Exc), %Pick exception
    print_message(I,'Try to deny',Exc),
    solve(I,Exc),!, %Prove exception and cut
    print_message(I,'Failed to deny',Exc).

%%%%%%%%%BURDEN PREDICATES%%%%%%%%%

% For production I can model lack of substance for: didnt submit docs,
% imcomplete form, no enough evidence for an element.
production_met(F) :-
    probative(F).

persuasion_met(F, P) :-
    prob(F, P),
    number(P),
    current_threshold(_Std, T),
    P > T.

plausible(F):-
    production_met(F),
    persuasion_met(F,_).

%%%%%%%%%%
%Discretionary factors evaluation
%%%%%%%%%%
%

%To void calling it in the case
init_defaults :-
    % ensure the default discretionary rule exists
    (   (discretionary_denial <= _)
    ->  true
    ;   assertz((discretionary_denial <= call(discretionary_denial_calc)))
    ).

```

```
discretionary_denial_calc :-
    discretionary_evaluation_necessary,
    discretionary_factors(Dict),
    discretionary_score(Dict, Score),
    Score < 0.

discretionary_score(Dict, Score) :-
    is_dict(Dict),
    dict_pairs(Dict, _Tag, Pairs),
    foldl(add_contrib, Pairs, 0.0, Score).

add_contrib(_Factor-Points, Acc0, Acc) :-
    is_dict(Points),
    get_dict(w, Points, W0),
    get_dict(s, Points, S0),
    scalar_number(W0, W),
    scalar_number(S0, S),
    Acc is Acc0 + W*S.

% accept either N or [N]
scalar_number([X], N) :- !, scalar_number(X, N).
scalar_number(N, N) :- number(N).

is_fact(P):-
   \+ (P<=_). %Is fact if the clause has no atoms in the body
```