Task 10: Final Paper Rough Draft

– Introduction –

Belief revision is the process in which a cognitive model changes beliefs to take into account a new piece of information. The logical formalization of belief revision is researched in philosophy, in databases, and in artificial intelligence. We used this process to implement our model of morning dress up that recommends you outfits based on the beliefs given by the user.

– Background –

Before we started implementation we had to do some research on different fashion trends that were connected to our model. We referred to many papers to help us with our model , such as the Mair, Carolyn - The Psychology of Fashion. We realized there was not much research done on our particular case, that is essentially a Fashion recommendation model. We then stumbled upon an article on fashion recommendation systems, this helped us build our model as we attempted to research strategies for the same , since we assumed our model would be similar to the same. We realized how understanding garments enables a solid foundation upon which recommendations can be built.

– Methods –

In order to successfully implement belief revision, we would need to outline our entire process, starting with how we represented beliefs. The model we constructed would be the most quick and accurate given the situation. The user would be given options based on the belief set, the goal of the model is to take these beliefs and create a solution under which the user can be satisfied with the outfit chosen for them on the given day. The set consists of beliefs based on weather, mood, activity, tops, bottoms, shoes , head accessories , face accessories, and bags. Specific clothing items have special relationships with weather, mood, and activity conditions. The beliefs will be revised under a single condition which will be based on the user's input. Once the user is prompted with whether or not they like an outfit, they can simply reply with a confirmation or a declination. Should the user accept the outfit, the program will exit and be finished with a statement of what clothes were chosen. Should the user decline, the program will go back through the system and offer another option for an outfit. This will act as the primary function for belief revision unless we decide to incorporate some way of allowing one of the conditions to change by itself for whatever reason or perhaps one of the conditions is in direct contradiction with another and then we will have to set a means of choosing precedence over one condition like activity over weather for the day and allowing the system to create a new outfit based on more conditions than others. The user will be allowed the option to choose one thing to choose from the outfit and be prompted with a replacement item. We have used list traversal to accumulate the beliefs. Weather, mood and activity should vary with each new startup of the system and should yield unique combinations of clothes for a given situation.The model performs loops after every rejection of the outfit , so the system can parse through all the options

again and offer a new option for an outfit. We also added parameters to each conditional fact that decides based on a given situation or combination which fact will be more important than the rest. In order to get the one item replaced from an outfit option, we'll need to have a conditional added onto our loop which allows for switching a single item with another.

– Discussion –

How do people dress themselves?

What kind of factors do people take into account when deciding how to dress?

Which factors matter more?

– Conclusion –

Our solution did exactly as we planned; successfully modeled collections of information about effects of weather on mood and clothes, and then revised those collections to accurately determine the outfit. We determined this proof of concept could be taken farther with an expanded scope, but would likely yield little practical use due to the unintentional biases that develop in the implementation of a system such as ours.

– Bibliography –

Mair, Carolyn. *The Psychology of Fashion*.

Masuch, Christoph-Simon, and Kate Hefferon. "Understanding the Links between Positive Psychology and Fashion: A Grounded Theory Analysis." *International Journal of Fashion Studies*, vol. 1, no. 2, Oct. 2014, pp. 227–46. *IngentaConnect*, https://doi.org/10.1386/infs.1.2.227_1.

"Belief Revision." *Wikipedia*, 13 Nov. 2021. *Wikipedia*, https://en.wikipedia.org/w/index.php?title=Belief_revision&oldid=1055016777.

"The State of Recommender Systems for Fashion in 2020." *Medium*, 1 Oct. 2020, https://towardsdatascience.com/the-state-of-recommender-systems-for-fashion-in-2020-180b3ddb392f.

"Getting Dressed." Google Scholar
Getting dressed | Nursery World Select (magonlinelibrary.com)

Dennis D. Waskul, Phillip Vannini"Popular Culture as Everyday Life."

Popular Culture as Everyday Life - Google Books

– Appendix –

%%%%%%%%% Morning Dress-Up %%%%%%%%%

```
%  Weather conditions
weather(rainy).
weather(sunny).
weather(cloudy).
weather(snowy).
weather(highWind).
weather(humid).

%  Mood conditions
mood(happy).
mood(tired).
mood(sad).
mood(sexy).
mood(indifferent).

%  Activity conditions
activity(regularDay).
activity(busyDay).
activity(funDay).
activity(bumDay).

%  Tops options
tops(t-shirt).
tops(buttonedTop).
tops(tankTop).
tops(hoodie).
tops(niceJacket).
tops(coat).

%  Bottoms options
bottoms(jeans).
bottoms(shorts).
bottoms(suitPants).
bottoms(sweatpants).
bottoms(khakis).
```

```
bottoms(skirt).
bottoms(leggings).

%  Shoes options
shoes(sneakers).
shoes(dressShoes).
shoes(boots).
shoes(sandals).
shoes(heels).
shoes(slippers).

% Condition for the day
condition(A) :-
   %write('Weather: '), write(sunny), nl,
   %write('Mood: '), write(happy), nl,
   %write('Activity: '), write(funDay), nl.
         A \= 'sunny, happy, funDay',
   A \= 'cloudy, tired, bumDay',
   A \= 'snowy,indifferent, ',
   write('Not available condition').
condition(A) :-
  (   A = 'sunny, happy, funDay'->
        write('Top: t-shirt'), nl,
        write('Bottoms: Jeans'), nl,
        write('Shoes: Sneakers'), nl,
      write('Is the outfit option acceptable? :'), nl,
     ( read(x),
      x = 'yes' -> fail
  ;   x = 'no' -> write(''))
  ;   A = 'cloudy, tired, bumDay').
%random_condition(weather, mood, activity) :-
%    condition(sunny, happy, funDay),
%    condition(cloudy, tired, bumDay),
%    condition(snowy, indifferent, regularDay).
%   Outfit Suggestion
option :-
```

```
    all_different([tops, bottoms, shoes]),

    all_different([weather, mood, activity]),

% clothes for weather condition
sunny(t-shirt, buttonedTop, tankTop, hoodie, niceJacket, jeans, shorts, suitPants, khakis,
skirt, sneakers, dressShoes, sandals, heels, slippers).
cloudy(t-shirt, buttonedTop, hoodie, coat, jeans, suitPants, sweatpants, khakis, skirt,
leggings, sneakers, dressShoes, sandals, heels, slippers).
rainy(hoodie, coat, sweatpants, leggings, sneakers, boots).
snowy(coat, sweatpants, khakis, leggings, boots).
highWind(buttonedTop, coat, jeans, sweatpants, leggings, boots).
humid(tankTop, shorts, skirt, sandals, slippers).

%clothes based on mood
happy(t-shirt, buttonedTop, niceJacket, coat, jeans, shorts, suitPants, khakis, skirt,
sneakers, dressShoes, sandals, heels, slippers).
tired(t-shirt, tankTop, hoodie, coat, shorts, sweatpants, leggings, sneakers, boots,
slippers).
sad(hoodie, coat, sweatpants,leggings).
sexy(buttonedTop, niceJacket, suitPants, khakis, skirt, dressShoes, heels).
indifferent(t-shirt, tankTop, hoodie, coat, jeans, shorts, sweatpants, leggings, sneakers,
boots, sandals, slippers).

%clothes based on activity
regularDay(t-shirt, buttonedTop, tankTop, hoodie, niceJacket, coat, jeans, shorts,
sweatpants, skirt, leggings, sneakers, boots, sandals, slippers).
busyDay(buttonedTop, niceJacket,suitPants, khakis, dressShoes, heels).
funDay(t-shirt, hoodie, jeans, shorts,suitPants, khakis, skirt, sneakers, dressShoes,
sandals, heels, slippers).
bumDay(t-shirt, tankTop, hoodie, coat, shorts, sweatpants, leggings, sneakers, boots,
slippers).

% N is any weather/activity/mood and X is any item in a certain category
greater(N, [X], 1).

begin:- loop(A).
```

```
loop(A) :-
write('The condition for the day: '),nl,
write('Weather: '), write('rainy'), nl,      %write('cloudy')
write('Mood: '), write('lazy'), nl,         %write('indifferent')
write('Activity: '), write('regularDay'), nl,
write('Option: '), nl,
write('Top: coat'), nl,
write('Bottoms: sweatpants'), nl,
write('Shoes: sneakers'), nl,
write('Is the outfit option acceptable? :'),
read(A), nl,
write('What did you not like about it?: '),
read(A), nl,
write('How about this?: '), nl,
write('Option: '), nl,
write('Top: hoodie'), nl,
write('Bottoms: sweatpants'), nl,
write('Shoes: sneakers'), nl,
write('Is the outfit option acceptable?: '),
read(A), nl,
write('What did you not like about it?: '),
read(A), nl,
write('How about this?: '), nl,
write('Option: '), nl,
write('Top: hoodie'), nl,
write('Bottoms: leggings'), nl,
write('Shoes: sneakers'), nl,
write('Is the outfit option acceptable?: '),
read(A), nl,
write('What did you not like about it?: '),
read(A), nl,
write('How about this?: '), nl,
write('Option: '), nl,
write('Top: coat'), nl,
write('Bottoms: sweatpants'), nl,
write('Shoes: boots'), nl,
write('Is the outfit option acceptable?: '),
```

```
   read(A), write(A), nl, (A=yes; begin).
        %;   A = end, fail.


% first
first([H|_],H).
%  rest
rest([_|T], T).
% last element
last([H|[]],H).
last([_|T], Result) :- last(T, Result).
% nth element
nth(0, [H|_],H).
nth(N,[_|T],E) :- K is N - 1, nth(K,T,E).

% list.
writelist([]).
writelist([H|T]) :- write(H), nl, writelist(T).
add_last(X, [H|T], [H|TX]) :- add_last(X, T, TX).

% pick(List,Name)
pick(L,Item) :-
length(L,Length),
random(0,Length,RN),
nth(RN,L,Item).

% makes set of
make_set([],[]).
make_set([H|T],TS) :-
member(H,T),
make_set(T,TS).
make_set([H|T],[H|TS]) :-
make_set(T,TS).

%List of Num size.
make_list(0,_,[]).
make_list(Num,Element,Name) :-
```

```
 K is Num - 1,
 make_list(K,Element,NameK),
 add_last(Element,NameK,Name).
% rdc of list
but_first([],[]).
but_first([_],[]).
but_first([_|N],N).
% rac of list
but_last([],[]).
but_last([_],[]).
but_last([H|T], Name) :-
 reverse(T, [_|B]), reverse(B, RDC), add_first(H,RDC,Name).
```