**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Definitions in explanation files:**

**Level of detail: 1,2**

**Tone: a,b**

**Format: i,ii,iii**

1 a i (High Detail + Formal Tone + Step-by-Step)

1 a ii (High Detail + Formal Tone + Paragraph)

1 a iii (High Detail + Formal Tone + Bullet Points)

1 b i (High Detail + Casual Tone + Step-by-Step)

2 a i (Concise Summary + Formal Tone + Step-by-Step)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Map to POMDP polidy. Initial prompt selected based on POMDP policy:**

For non\_expert policy:

a2 = 2

b2 = b

c1 = i

For planning\_expert policy: ‘’’AI’expert”

a1 = 1

b1 = a

c2 = ii

For case\_study\_expert policy: ‘’domain expert

a1 = 1

b1 = a

c3 = iii

**This is, explanation files selected for the initial explanation for each profile:**

Non-experts: a-AgriNonExpert-2bi

AI experts: a-AgriAIExpert-1aii

Domain experts (finance): a-AgriDomainExpert-1aiii

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Property:**

**R{"acceptance"}max=? [ F done ]**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PRISM Model:**

pomdp

//------------Set variables--------

const int selectedprofile;// = non\_expert;

//-----------Done Formula----------

formula done = prompt\_input = max\_prompt\_inputs;

//-----------Observables

observables a,b,c,profile,prompt\_input,step,patiencePred,underPred endobservables

//-----------Variables----------

//--Patience/Understanding levels

const int low = 0;

const int high = 1;

//--User Profile

const int non\_expert = 1;

const int planning\_expert = 2;

const int case\_study\_expert = 3;

//--Prompt blank spaces

const int max\_prompt\_inputs=3;

//-----------Probability Values----------

// For "patience" (Low or High) and "predicted patience" (Low or High) combinations.

// Similar for "understanding".

//-------------Non\_expert

//patience

const double pHH=0.4; //non-expert has about half prob. of been patient (0.4+0.1)

const double pHL=0.1;

const double pLH=0.05;

const double pLL=1-pHH-pHL-pLH;

//understanding

const double pHH\_u=0.2; //non-expert has lower prob. of initially understanding (0.2+0.05)

const double pHL\_u=0.05;

const double pLH\_u=0.15;

const double pLL\_u=1-pHH\_u-pHL\_u-pLH\_u;

//-------------Planning\_expert

//patience

const double pHH\_pe=0.6;//planning expert has more patience (0.6+0.24)

const double pHL\_pe=0.24;

const double pLH\_pe=0.05;

const double pLL\_pe=1-pHH\_pe-pHL\_pe-pLH\_pe;

//understanding

const double pHH\_u\_pe=0.79;//planning expert has more understanding (0.79+0.023)

const double pHL\_u\_pe=0.023;

const double pLH\_u\_pe=0.012;

const double pLL\_u\_pe=1-pHH\_u\_pe-pHL\_u\_pe-pLH\_u\_pe;

//-------------Case\_study\_expert

//patience

const double pHH\_ce=0.16;//case study expert has low patience (0.16+0.03)

const double pHL\_ce=0.03;

const double pLH\_ce=0.29;

const double pLL\_ce=1-pHH\_ce-pHL\_ce-pLH\_ce;

//understanding

const double pHH\_u\_ce=0.56;//case study expert has medium understanding (0.56+0.02)

const double pHL\_u\_ce=0.02;

const double pLH\_u\_ce=0.2;

const double pLL\_u\_ce=1-pHH\_u\_ce-pHL\_u\_ce-pLH\_u\_ce;

// Mechanism in the real world it woul de possible to obtain this as... --describe credibly how to obtain these

//Predicted: from the time taken to read the instruction

//Real: Asking patient for patient from high or low

// Brainstorming: How to get these?

// a) the time taken to read the instructions is translated to a probability of the user been patient or not.

// b) pre-questioneers assessing the patient of profile groups

// c) available data (e.g., 20-30 less patience)

// d) the more interactions with the feedback form (asking the user if they like the explanation)

//-----------Model divided into turns------

module Turn

step:[0..4];

//step 0-2: guess human demeanor

//step 2: obtain prompt

[observeUserPatience] step=0 -> 1:(step'=1);

[observeUserUnderstand] step=1 -> 1:(step'=2);

[end] true->1:(step'=3);

endmodule

//-----------Human demeanor model----------

module HumanBehavioralModel

profile:[0..3] init selectedprofile;

patience:[0..1] init high;

patiencePred:[0..1] init high;

under:[0..1] init high;

underPred:[0..1] init high;

//based on historical data, assign value of non-observable

//(e.g., toss in https://github.com/prismmodelchecker/prism/blob/master/prism-examples/pomdps/simple/guess-multi.prism)

//---------------Non\_expert----

//--patience

[observeUserPatience] profile=non\_expert & step=0 -> pHH:(patience'=high)&(patiencePred'=high)+

pHL:(patience'=high)&(patiencePred'=low)+

pLH:(patience'=low)&(patiencePred'=high)+

pLL:(patience'=low)&(patiencePred'=low);

//--understanding

[observeUserUnderstand] profile=non\_expert & step=1 -> pHH\_u:(under'=high)&(underPred'=high)+

pHL\_u:(under'=high)&(underPred'=low)+

pLH\_u:(under'=low)&(underPred'=high)+

pLL\_u:(under'=low)&(underPred'=low);

//---------------Planning\_expert----

//--patience

[observeUserPatience] profile=planning\_expert & step=0 -> pHH\_pe:(patience'=high)&(patiencePred'=high)+

pHL\_pe:(patience'=high)&(patiencePred'=low)+

pLH\_pe:(patience'=low)&(patiencePred'=high)+

pLL\_pe:(patience'=low)&(patiencePred'=low);

//--understanding

[observeUserUnderstand] profile=planning\_expert & step=1 -> pHH\_u\_pe:(under'=high)&(underPred'=high)+

pHL\_u\_pe:(under'=high)&(underPred'=low)+

pLH\_u\_pe:(under'=low)&(underPred'=high)+

pLL\_u\_pe:(under'=low)&(underPred'=low);

//---------------Case\_study\_expert----

//--patience

[observeUserPatience] profile=case\_study\_expert & step=0 -> pHH\_ce:(patience'=high)&(patiencePred'=high)+

pHL\_ce:(patience'=high)&(patiencePred'=low)+

pLH\_ce:(patience'=low)&(patiencePred'=high)+

pLL\_ce:(patience'=low)&(patiencePred'=low);

//--understanding

[observeUserUnderstand] profile=case\_study\_expert & step=1 -> pHH\_u\_ce:(under'=high)&(underPred'=high)+

pHL\_u\_ce:(under'=high)&(underPred'=low)+

pLH\_u\_ce:(under'=low)&(underPred'=high)+

pLL\_u\_ce:(under'=low)&(underPred'=low);

endmodule

//-----------Prompt selection model----------

// Take decisions based on the non-observable patience of the user

module PromptModel

prompt\_input: [0..1+max\_prompt\_inputs] init 0;

a:[0..2];

b:[0..2];

c:[0..3];

// Select prompt input - Rewards are synchronised based on prompt,profile

// a action = level of detail

[a1] a=0 & prompt\_input=0 & step=2 -> 1:(prompt\_input'=1) & (a'=1); //a1=in high-detail (long answer)

[a2] a=0 & prompt\_input=0 & step=2 -> 1:(prompt\_input'=1) & (a'=2); //a2=in a concise summary with minimum detail (short answer)

// b action = tone

[b1] b=0 & prompt\_input=1 & step=2 -> 1:(prompt\_input'=2) & (b'=1) ; //b1=technical and precise tone (avoid examples, straight to the point)

[b2] b=0 & prompt\_input=1 & step=2 -> 1:(prompt\_input'=2) & (b'=2) ; //b2=casual, conversational, simple tone (use examples if needed)

// c action = output\_format

[c1] c=0 & prompt\_input=2 & step=2 -> 1:(prompt\_input'=3) & (c'=1) ; //c1=in a step-by-step list

[c2] c=0 & prompt\_input=2 & step=2 -> 1:(prompt\_input'=3) & (c'=2) ; //c2=casual, as a summary, no bullet points nor list

[c3] c=0 & prompt\_input=2 & step=2 -> 1:(prompt\_input'=3) & (c'=3) ; //c3=as a series of bullet points only highlighting key items

[end] prompt\_input=3 -> 1:(prompt\_input'= 1+max\_prompt\_inputs); //required to count final transition reward

endmodule

// [retry] step=4 -> pPermissionToRetryGranted:(step'=3) + (1-pPermissionToRetryGranted):(step'=6);

// ?

// Do after getting Google Form!

//Human behaviour - conditional on actions -- how the user is "potentially" responding to the answer

// [] a1=1 & step=3 -> p:() + 1-p:(); //think or retry

// [] b1=2 //

//"sub-DTMC" end in:

//retry

//done

//https://www.prismmodelchecker.org/casestudies/power\_ctmc4.php

//-----------Reward Values----------

//--Human input based on specific policies. The reward values to reflect human behavior.

//------------Non\_expert

// a action = level of detail;

const int prompt\_accepted\_a1 = 5; // Lowered acceptance of detailed answer

const int prompt\_rejected\_a1 = 4;

const int prompt\_accepted\_a2 = 10; // Higher preference for concise summary

const int prompt\_rejected\_a2 = 1;

// b action = tone

const int prompt\_accepted\_b1 = 2; // Technical tone less accepted

const int prompt\_rejected\_b1 = 7;

const int prompt\_accepted\_b2 = 15; // Conversational tone more preferred

const int prompt\_rejected\_b2 = 3;

// c action = output\_format

const int prompt\_accepted\_c1 = 23; // Step-by-step lists preferred

const int prompt\_rejected\_c1 = 6;

const int prompt\_accepted\_c2 = 1;

const int prompt\_rejected\_c2 = 12;

const int prompt\_accepted\_c3 = 14;

const int prompt\_rejected\_c3 = 13;

//a1=in high-detail (long answer) //a2=in a concise summary with minimum detail (short answer)

//b1=technical and precise tone (avoid examples, straight to the point) //b2=casual, conversational, simple tone (use examples if needed)

//c1=in a step-by-step list //c2=casual, as a summary, no bullet points nor list //c3=as a series of bullet points only highlighting key items

//------------Planning\_expert

// a action = level of detail

const int prompt\_accepted\_a1\_pe = 8; // Prefers detailed answers

const int prompt\_rejected\_a1\_pe = 1;

const int prompt\_accepted\_a2\_pe = 2;

const int prompt\_rejected\_a2\_pe = 4;

// b action = tone

const int prompt\_accepted\_b1\_pe = 25; // Technical tone preferred

const int prompt\_rejected\_b1\_pe = 2;

const int prompt\_accepted\_b2\_pe = 5;

const int prompt\_rejected\_b2\_pe = 16;

// c action = output\_format

const int prompt\_accepted\_c1\_pe = 11;

const int prompt\_rejected\_c1\_pe = 12;

const int prompt\_accepted\_c2\_pe = 29; //summary prefered

const int prompt\_rejected\_c2\_pe = 8;

const int prompt\_accepted\_c3\_pe = 5;

const int prompt\_rejected\_c3\_pe = 4;

//------------Case\_study\_expert

// a action = level of detail

const int prompt\_accepted\_a1\_ce = 36;

const int prompt\_rejected\_a1\_ce = 1;

const int prompt\_accepted\_a2\_ce = 9;

const int prompt\_rejected\_a2\_ce = 14;

// b action = tone

const int prompt\_accepted\_b1\_ce = 44; // More okay with technical tone

const int prompt\_rejected\_b1\_ce = 3;

const int prompt\_accepted\_b2\_ce = 15;

const int prompt\_rejected\_b2\_ce = 14;

// c action = output\_format

const int prompt\_accepted\_c1\_ce = 4;

const int prompt\_rejected\_c1\_ce = 2;

const int prompt\_accepted\_c2\_ce = 3;

const int prompt\_rejected\_c2\_ce = 3;

const int prompt\_accepted\_c3\_ce = 50; // Bullet points preferred for case summaries

const int prompt\_rejected\_c3\_ce = 2;

//-----------Prospect Theory Parameters----------

// These values are based on Kahneman & Tversky's original paper.

const double alpha = 0.88; // Exponent for gains (diminishing sensitivity)

//Negative reward bounded by 0, so don't use loss aversion:

//const double beta = 0.88; // Exponent for losses (diminishing sensitivity)

//const double lambda = 2.25; // Loss aversion coefficient (losses feel ~2.25x more impactful)

rewards "acceptance"

// -- Non-Expert Rewards --

// If (Accepted - Rejected) is a GAIN (>= 0), value = (gain)^alpha

[a1] profile=non\_expert & (prompt\_accepted\_a1 - prompt\_rejected\_a1 >= 0) : pow(prompt\_accepted\_a1 - prompt\_rejected\_a1, alpha);

[a2] profile=non\_expert & (prompt\_accepted\_a2 - prompt\_rejected\_a2 >= 0) : pow(prompt\_accepted\_a2 - prompt\_rejected\_a2, alpha);

[b1] profile=non\_expert & (prompt\_accepted\_b1 - prompt\_rejected\_b1 >= 0) : pow(prompt\_accepted\_b1 - prompt\_rejected\_b1, alpha);

[b2] profile=non\_expert & (prompt\_accepted\_b2 - prompt\_rejected\_b2 >= 0) : pow(prompt\_accepted\_b2 - prompt\_rejected\_b2, alpha);

[c1] profile=non\_expert & (prompt\_accepted\_c1 - prompt\_rejected\_c1 >= 0) : pow(prompt\_accepted\_c1 - prompt\_rejected\_c1, alpha);

[c2] profile=non\_expert & (prompt\_accepted\_c2 - prompt\_rejected\_c2 >= 0) : pow(prompt\_accepted\_c2 - prompt\_rejected\_c2, alpha);

[c3] profile=non\_expert & (prompt\_accepted\_c3 - prompt\_rejected\_c3 >= 0) : pow(prompt\_accepted\_c3 - prompt\_rejected\_c3, alpha);

// -- Planning Expert Rewards --

[a1] profile=planning\_expert & (prompt\_accepted\_a1\_pe - prompt\_rejected\_a1\_pe >= 0) : pow(prompt\_accepted\_a1\_pe - prompt\_rejected\_a1\_pe, alpha);

[a2] profile=planning\_expert & (prompt\_accepted\_a2\_pe - prompt\_rejected\_a2\_pe >= 0) : pow(prompt\_accepted\_a2\_pe - prompt\_rejected\_a2\_pe, alpha);

[b1] profile=planning\_expert & (prompt\_accepted\_b1\_pe - prompt\_rejected\_b1\_pe >= 0) : pow(prompt\_accepted\_b1\_pe - prompt\_rejected\_b1\_pe, alpha);

[b2] profile=planning\_expert & (prompt\_accepted\_b2\_pe - prompt\_rejected\_b2\_pe >= 0) : pow(prompt\_accepted\_b2\_pe - prompt\_rejected\_b2\_pe, alpha);

[c1] profile=planning\_expert & (prompt\_accepted\_c1\_pe - prompt\_rejected\_c1\_pe >= 0) : pow(prompt\_accepted\_c1\_pe - prompt\_rejected\_c1\_pe, alpha);

[c2] profile=planning\_expert & (prompt\_accepted\_c2\_pe - prompt\_rejected\_c2\_pe >= 0) : pow(prompt\_accepted\_c2\_pe - prompt\_rejected\_c2\_pe, alpha);

[c3] profile=planning\_expert & (prompt\_accepted\_c3\_pe - prompt\_rejected\_c3\_pe >= 0) : pow(prompt\_accepted\_c3\_pe - prompt\_rejected\_c3\_pe, alpha);

// -- Case Study Expert Rewards --

[a1] profile=case\_study\_expert & (prompt\_accepted\_a1\_ce - prompt\_rejected\_a1\_ce >= 0) : pow(prompt\_accepted\_a1\_ce - prompt\_rejected\_a1\_ce, alpha);

[a2] profile=case\_study\_expert & (prompt\_accepted\_a2\_ce - prompt\_rejected\_a2\_ce >= 0) : pow(prompt\_accepted\_a2\_ce - prompt\_rejected\_a2\_ce, alpha);

[b1] profile=case\_study\_expert & (prompt\_accepted\_b1\_ce - prompt\_rejected\_b1\_ce >= 0) : pow(prompt\_accepted\_b1\_ce - prompt\_rejected\_b1\_ce, alpha);

[b2] profile=case\_study\_expert & (prompt\_accepted\_b2\_ce - prompt\_rejected\_b2\_ce >= 0) : pow(prompt\_accepted\_b2\_ce - prompt\_rejected\_b2\_ce, alpha);

[c1] profile=case\_study\_expert & (prompt\_accepted\_c1\_ce - prompt\_rejected\_c1\_ce >= 0) : pow(prompt\_accepted\_c1\_ce - prompt\_rejected\_c1\_ce, alpha);

[c2] profile=case\_study\_expert & (prompt\_accepted\_c2\_ce - prompt\_rejected\_c2\_ce >= 0) : pow(prompt\_accepted\_c2\_ce - prompt\_rejected\_c2\_ce, alpha);

[c3] profile=case\_study\_expert & (prompt\_accepted\_c3\_ce - prompt\_rejected\_c3\_ce >= 0) : pow(prompt\_accepted\_c3\_ce - prompt\_rejected\_c3\_ce, alpha);

endrewards