Evaluating Probabilistic Language of Thought on Theory of Mind Tasks

Anthony Baez [1]
acbaez@mit.edu
Massachusetts Institute of Technology

Abstract

Recent advances in artificial intelligence have shown promise in creating human-like cognitive models that can communicate in natural language (Achiam et al., 2023). However, previous work has shown that these models, called Large Language Models (LLM), can fall short in crucial reasoning tasks. Among these is theory of mind (ToM), which concerns reasoning about other humans as autonomous, self-aware actors (Ullman, 2023). One promsing approach to incorporate more human-like reasoning is Probabilistic Language of Thought (PLoT), which reason probabilistically from natural language using LLMs and probabilistic programming language. We evaluated PLoT models on ToM tasks, and found that while not able to pass all tasks, clearly improve on previous benchmarks in the development of better cognitive models.

Keywords: Bayesian, reasoning, LLM, cognition, theory of mind

Introduction

What makes humans unique? In this current era of rapidly improving computational intelligence, it seems like our human ability to reason, learn, and remember are becoming less unique. However, humans are not individual actors, but live, work, and communicate in personal relationships and societies. This emotional or societal intelligence is just as important to being human than any more logical types of intelligence. Therefore, to evaluate and develop better computational models of the human mind, we must not neglect performance in the emotional domain.

One area of humans' emotional reasoning and intelligence is called Theory of Mind (ToM). ToM concerns the human ability to reason about others, such as their mental states, beliefs, and what they perceptive from their point of view. This skill is essential to work and interact with others. This makes ToM an important skill to properly assess how close computational systems are to modeling human cognition.

Previous work has assessed ChatGPT, an interactive chatbot built on neural network-based large language model (LLM) GPT-3.5, on ToM tasks through language (Ullman, 2023). While humans' often reason about ToM in physical tasks, language and communication can prove an important domain as well. ChatGPT was found to fail simple ToM tasks, demonstrating that while impressive, it still remains far from humans in this area.

[1]Undergraduate enrolled in 9.66

One promising approach to strengthen the reasoning ability of LLMs are probabilistic language of thought (PLoT) models (Wong et al., 2023). They take advantage of LLMs' ability to generalize from large amounts of data to convert natural language into probabilistic programming language and use Bayesian inference to reason and express uncertainty in their responses like humans have been shown to do (Tenenbaum, Kemp, Griffiths, & Goodman, 2011). PLoT also builds on theories surrounding the relationship between language and thinking in the mind. PLoT demonstrates that language is both a structured process in the brain and that language and cognition are deeply intertwined and may even be necessary for each other (Wong et al., 2023). PLoT was assessed on several important reasoning tasks, but were notably not assessed on ToM tasks. This presents an important gap for both evaluating PLoT models and the most powerful models of human cognition that use the latest advances in both neural networks and probabilistic reasoning.

To assess PLoT models on ToM, we applied the principles of PLoT models using ChatGPT-4o to write code in WebPPL and answer the ToM tasks used in (Ullman, 2023). We then compared the results to human benchmarks in both writing WebPPL code and quantifying uncertainty by estimating probabilities and discussed where PLoT both succeeds and falls short.

Related Works

In previous work, ChatGPT-3.5 was tested on ToM using two scenarios with four questions each that evaluated elements of managing information from visual perception, trusted communication, relative positions, and the thoughts of others (Ullman, 2023). It was found that in simple tasks, ChatGPT failed to answer any questions correctly, demonstrating broad failure and lack of ToM. This finding refuted a previous finding that LLMs did acquire a basic theory of mind (Kosinski, 2023). The refutation was done by perturbing the questions slightly still consistent with testing theory of mind but led to model failures.

Concerning the model, language has previously been used as a vehicle for enabling computational models to reason. Natural language could be converted into symbolic language using an 'epistemic language-of-thought', which is then used for probabilistic reasoning. This was applied to theory of mind tasks as well (Ying, Zhi-Xuan, Wong, Mansinghka,

& Tenenbaum, 2024). Natural language can also be converted to PDDL, a programming language used in planning and robotics, which allowed a neural network to carry out Bayesian inverse planning (Ying et al., 2023). This is in addition to work with the PLoT model described previously (Wong et al., 2023).

Method

Theory of Mind Task

We selected the 'Unexpected Contents' task in (Ullman, 2023) to assess PLoT. Scenarios 1A, 1B, 1C, and 1D correspond to Tasks 1, 2, 3, and 4 in this paper. In this task, the model must assign probabilities to whether Sam thinks a bag contains chocolate or popcorn. Different facts are presented in the given context to then test if the model understands their consequences for what Sam thinks is contained in the bag. All tasks begin with a slightly modified version of this context:

> *Here is a bag filled with popcorn. There is no chocolate in the bag. Yet, the label on the bag says "chocolate" and not "popcorn." Sam finds the bag. She had never seen the bag before. She cannot see what is inside the bag. She reads the label.*

Base WebPPL World Model

We used WebPPL for the language of probabilistic reasoning. WebPPL is a probabilistic programming language built on JavaScript and is designed for performing inference using Bayesian reasoning. To begin use WebPPL for our PLoT model, we first needed to introduce a world model to the LLM that would allow it to reason in (Wong et al., 2023). We based our world model on the vending machine example in the Social Cognition chapter of the Probabilistic Model of Cogition (Goodman, Tenenbaum, & Contributors, 2016). The basic scaffolding for our world model is shown below.

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.5, .5]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.5, .5]}) : 'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist,'chocolate')

  return contents;
})
```

Figure 1: WebPPL code used to build world model

The first block of code with *var contentsPrior* creates a prior probability on whether chocolate or popcorn exists within the bag. The probabilities are set to be equal to represent no information being known initially about the contents of the bag.

The second block of code with *var viewedLabel* creates a likelihood distribution based on the information gained from reading the label on our bag. The probabilities were set to be equal just for the example, but in the tasks they determined what new information was gained from reading the label.

The third block of code with *var conclusion* creates the start and goal states that our model uses for probabilistic inference. The variable *label* is sampled from our likelihood and conditioned on the transition from the start to the goal state, allowing us to incorporate evidence we gain to infer the content of the bag.

The fourth block of code with *contentsPosterior* creates a posterior distribution using the prior and the likelihood using the **observe** function of WebPPL. **observe** conditions the passed in distribution on an observed value to change the posterior distribution.

Task 1 First Prompt

```
// Here is a scenario. A bag is filled with popcorn, and there is no chocolate in the
// bag. However, from the outside without knowing this there is an equal chance of it
// containing chocolate and popcorn. Sam finds the bag. She had never seen the bag
// before. She cannot see what is inside the bag.

var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

// Yet, the label on the bag says “chocolate” and not “popcorn.” She reads the label.
// We model it like this, assigning reasonable probabilities to the distribution.
var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.8, .2]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.2, .8]}) : 'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist,'chocolate')

  return contents;
})

// Now, say that the bag is transparent, and the contents are visible. Either change
// probabilities in the prior or likelihood, or add a new function that is structured
// like viewedLabel and has an observe line that takes into account this
// new information for what Sam believes.
```

Figure 2: Task 1 Prompt

Figure 2 shows the prompt given to ChatGPT for Task 1. In Task 1, the model is given the information that the bag is transparent, so Sam must know that the bag truly contains popcorn. However, the label reads 'chocolate', and the model is told that Sam reads this as well. The model must take in these two conflicting facts and assess how they change the posterior and what Sam thinks is in the bag. The text in the prompt is a paraphrased version of the corresponding scenario text from the original (Ullman, 2023). We reordered the text to be next to the relevant code block where the information is given as well. We additionally added some text that provided additional instructions on how to implement the observations from the text into the WebPPL code. Without this, ChatGPT always failed to output functional code.

Task 1 Second Prompt

In all scenarios, a second prompt was used to tell ChatGPT to combine the code in its repsonse with the code in the prompt. The second prompt always had the below wording.

> *Now, add your new function and posterior to the previously given code so it can all run.*

After the second prompt, we copy-pasted the response into a WebPPL environment without any modification. This was the prompting algorithm that yielded the most success, as opposed to only using one prompt and not providing additional instruction to how to modify the WebPPL code. Overall, this method incorporated Scenario 1 from (Ullman, 2023) into a prompts for ChatGPT that allowed it to write WebPPL code to perform Bayesian inference.

Task 1 Response

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.8, .2]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.2, .8]}) : 'nothing')
}

var viewedContents = function(state, contents) {
  return (contents == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                                 ps: [1, 0]}) :
          contents == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                               ps: [0, 1]}) : 'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state == undefined) ? 'start' : state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist, 'chocolate')

  var contentsDist = conclusion(contents, viewedContents)
  observe(contentsDist, 'popcorn')

  return contents
})
```

Figure 3: Task 1 Response

Figure 3 shows the response ChatGPT gave to the second prompt for Task 1. It contains the new likelihood from viewing the contents of the bag, and it informs us that it is impossible that the bag contains chocolate. It then also gives us a new posterior, where it incorporates the evidence from both likelihoods using **observe**. Because PLoT performs Bayesian reasoning on these probabilities, it was able to reason that the distribution of the contents of the bag would be more affected by viewing the contents than the label, the output is a distribution that shows that only popcorn can be in the bag. Figure 4 shows the human benchmark for Task 1.

Human Benchmarks

We also compared the PLoT results to human benchmarks. In these benchmarks, we wrote WebPPL code ourselves to properly model the ToM task. Specifically for the chosen probabilities, we conducted a survey of 10 people. In the survey, the participants were given the same ToM tasks and asked to assign probability values for all existing distributions. The values used are the averages for each distribution across the survey.

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [0.5, 0.5]});

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ?
            categorical({vs: ['chocolate', 'popcorn'], ps: [0.83, 0.27]}) :
          label == 'popcorn' ?
            categorical({vs: ['chocolate', 'popcorn'], ps: [0.27, 0.83]}) :
            'nothing');
};

var viewedContents = function(state, contents) {
  return (contents == 'chocolate' ?
            categorical({vs: ['chocolate', 'popcorn'], ps: [0.93, 0.07]}) :
          contents == 'popcorn' ?
            categorical({vs: ['chocolate', 'popcorn'], ps: [0.07, 0.93]}) :
            'nothing');
};

var conclusion = function(goalState, transition, state) {
  var state = (state == undefined) ? 'start' : state;
  return Infer(function() {
    var label = sample(contentsPrior);
    condition(goalState == transition(state, label));
    return label;
  });
};

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [0.5, 0.5]});

  var labelDist = conclusion(contents, viewedLabel);
  observe(labelDist, 'chocolate');

  var contentsDist = conclusion(contents, viewedContents);
  observe(contentsDist, 'popcorn');

  return contents;
});
```

Figure 4: Task 1 Human Benchmark

Results

Task Outcome Classification

In Table 1, we can see to what degree the PLoT model was able to accomplish each of the four ToM tasks. In Figure 5, we can see the resulting posterior distributions for the three tasks that had correct probabilities. In Figure 6, we can see the resulting posterior distributions for the human benchmarks for Tasks 1-3.

We considered the 'Correct Probabilities in Likelihood' criteria met if there was a meaningful change to the prior or likelihood variables, or the addition of a function that constructed a probability distribution that had a relevant name to the piece of information that PLoT needed to incorporate. This criteria also needed the probabilities in the altered variable to have probabilities that reflected a correct directional change.

ToM Task	Correct Probabilities in Likelihood?	Code Works?	Correct Answer in Posterior?
1	✓	✓	✓
2	✓	✓	✓
3	✓	X	✓
4	X	X	X

Table 1: Results of PLoT Graded for Accomplishing ToM Task

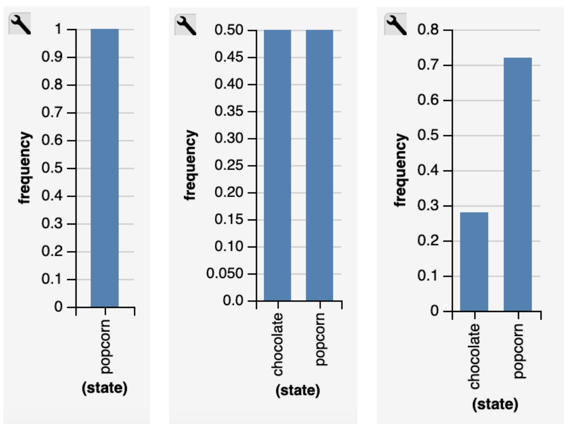

Figure 5: Posterior Distributions for ToM tasks 1-3 from PLoT

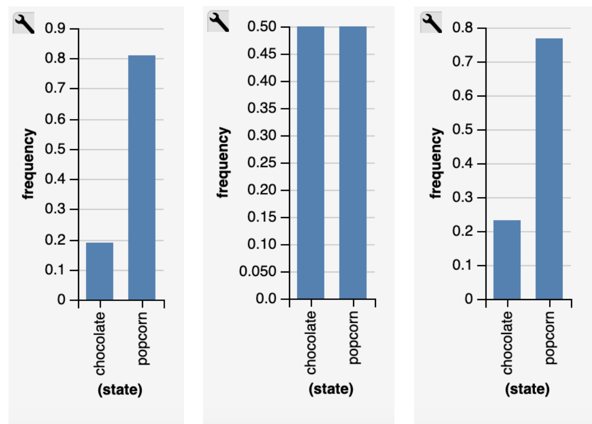

Figure 6: Posterior Distributions for ToM tasks 1-3 in Human Benchmark

We considered the 'Code Works' criteria met if we could place the code outputted after the second prompt into a WebPPL interpreter and get a posterior distribution without any bugs.

We considered the 'Correct Answer in Posterior' criteria met if the resulting posterior distribution, after fixing any minor bugs in the code, gave a higher probability to the correct answer for that scenario. This was done after fixing any code, because it showed that PLoT was able to understand in what direction to shift the distribution, but failed to grasp the nuances of WebPPL code. This still shows relevant and significant understanding of the task.

Task Outcomes

We see that for Tasks 1 and 2, the PLoT model created relevant likelihood distributions, implemented them correctly in code, and gave values to the likelihood such that a correct answer was found in the posterior. For Task 3, the PLoT model created relevant likelihood functions, but made a minor syntax error in the construction of the likelihood. After fixing this small error, the resultant posterior distribution yielded the correct answer. For Task 4, PLoT failed to create a coherent WebPPL model, so it failed all performance metrics. Notably, the ChatGPT baseline failed on all criteria on all tasks (Ullman, 2023).

Discussion

Analysis of Task Outcomes

To analyze the effectiveness of PLoT accomplishing the ToM tasks, we will compare the PLoT response with the human benchmarks for all tasks. The final responses and human benchmarks for Tasks 2-4 can be found in the Appendix.

For Task 1, PLoT matches the human benchmark very closely. They both create another likelihood distribution that represents Sam actually being able to see the contents of the bag which significantly increases the likelihood of popcorn. It then applies this to the posterior with **observe**. This outcome shows that PLoT can understand transparency and its effect on perception.

For Task 2, PLoT matches the human benchmark in giving probabilities to the *viewedLabel* variable that signify no change in the distribution because Sam cannot read. This results in the same posterior distribution that matches the prior given for the contents of the bag. However, PLoT did this by creating a new likelihood variable and posterior, instead of modifying the existing ones. This adds unneeded complexity but does not harm the outcome of the scenario. This outcome shows that PLoT can recognize when new information is not informative and does not meaningfully change a situation.

For Task 3, PLoT correctly created a new likelihood distribution that reflected the new information from the trusted friend, and then it tried to incorporate it into the posterior. The probability values were also such that popcorn would correctly be the higher probability in the posterior. However, PLoT incorrectly constructed the *believedContents* variable, which resulted in a syntax error. By just fixing the syntax error in *believedContents* and in *contentsPosterior*, PLoT would have arrived at the correct answer in the posterior. The incorrect and fixed code is included in the Appendix. This outcome shows that PLoT can understand trusted testimony and how it can affect a situation, but failed to properly write it in WebPPL.

For Task 4, PLoT completely failed at all criteria. Instead of recognizing to change the prior probability of the contents of the bag because Sam herself filled them, PLoT modified the *viewedLabel* likelihood to only make it possible the bag contains chocolate, which is false. PLoT fails to reason that Sam filling the bag with popcorn herself gives her the certain knowledge that the bag contains popcorn. This shows that PLoT fails in understanding that Sam's own actions gave her knowledge of the contents and made other information not relevant.

Failure Points in Prompting

It is important to note that the method used to prompt the PLoT-ToM mind model differed slightly from that used in (Wong et al., 2023). The original method used one prompt to provide the world model and enable the LLM to convert natural language to probabilistic programming language, then used additional prompts to update the model. We used the first prompt to provide the main question and introduce the world model, but then used the second prompt to make PLoT output the new world model in WebPPL. We also added additional, specific instruction to PLoT in how to output the code such as telling it to 'change the probabilities of the prior or likelihood' and and 'add a new function like *viewedLabel* which has an **observe** line that takes into account this new information'. Without this additional instruction, PLoT fails to produce working code that is coherent to the problem given. The inclusion of the context for the ToM problem also added natural language context that did not seem to be present in the original PLoT work. We also reordered the way the context was given for the tasks from the scenarios present in the previous ToM benchmark so that the information was presented along with the relevant WebPPL code blocks (Ullman, 2023). All these choices needed to be made to be consistent with both the previous work in PLoT and the ToM tasks, but could limit replicability when compared to only one of either of them. These choices also helped improve performance on task but in ways that helped the model understand the context and task, not make the reasoning easier.

Limitations

One limitation that we did not use the same probabilistic programming language as the original PLoT paper, called Church. While WebPPL is just as expressive as Church, Church has more advanced tools that may make it a more effective vehicle for PLoT. We also do not know how ChatGPT's proficiency in coding in WebPPL compares with Church. This was done because we do not have experience with Church, and it was more convenient to use WebPPL given other constraints.

We also did not have access to the LLM used in the original paper, Codex, as its API was removed from OpenAI. Even though ChatGPT-4o would most likely be better at reasoning and at coding, not being able to use the same LLM still limits the extend to which we can compare PLoT's results on ToM tasks to previous work. The nature of the tasks given in (Wong et al., 2023) also differ significantly than those found in (Ullman, 2023), which could have affected performance and again limit the comparisons that can be made the original PLoT paper.

Conclusion

We find that PLoT succeeded on two of the four ToM tasks, with conceptual success but incorrect code on the third task and complete failure on the fourth task. Throughout these tasks, we can see that PLoT is able to effectively build off of the given world model according to conceptual information in the prompt and used probabilistic reasoning to come to the correct conclusion in two out of four tasks. This is in comparison to ChatGPT with GPT-3.5, which failed all tasks completely (Ullman, 2023). These findings show that PLoT makes a significant improvement in the cognitive ability of theory of mind when compared to only ChatGPT, a modern LLM, on top of the other tasks tested in (Wong et al., 2023). The findings also support hypothesis that a language plays a uniquely significant role in human cognition and social reasoning. However, some failures in the experiment also confirm the limitations of PLoT models and LLMs in approximating a theory of mind reasoning, and more work is needed to explore all aspects of theory of mind and other essential human abilities in modern cognitive models.

Future Work

Since most of our limitations were from discrepancies between the original PLoT paper and this paper, future work should focus on addressing them. First, we could decide to use Church in our PLoT model. Second, while we cannot go back and use Codex, we can re implement the original PLoT problems in GPT-4o to be more comparable to the results in this paper. We could also prompting to give ChatGPT more coding proficiency at Church or WebPPL, which could be done by passing in a PDF file containing the relevant documentation before the world model. This would likely prevent syntax errors in code and allow for failures in PLoT to be exclusively from lack of reasoning ability. Finally, we could have tested more different prompt wordings to improve our results.

Because of the success in the results and an improved proficiency in coding from more Church context, we could test on a higher quantity, difficulty, and more diverse ToM tasks to get a better sense of PLoT's abilities. We could also test PLoT on other social reasoning skills.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... others (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Goodman, N. D., Tenenbaum, J. B., & Contributors, T. P. (2016). *Probabilistic Models of Cognition* (Second ed.). http://probmods.org/v2. (Accessed: 2024-12-14)

Kosinski, M. (2023). Theory of mind may have spontaneously emerged in large language models. *arXiv preprint arXiv:2302.02083*, *4*, 169.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., & Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *science*, *331*(6022), 1279–1285.

Ullman, T. (2023). Large language models fail on trivial alterations to theory-of-mind tasks. *arXiv preprint arXiv:2302.08399*.

Wong, L., Grand, G., Lew, A. K., Goodman, N. D., Mansinghka, V. K., Andreas, J., & Tenenbaum, J. B. (2023). From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672*.

Ying, L., Collins, K. M., Wei, M., Zhang, C. E., Zhi-Xuan, T., Weller, A., ... Wong, L. (2023). The neuro-symbolic inverse planning engine (nipe): Modeling probabilistic social inferences from linguistic inputs. *arXiv preprint arXiv:2306.14325*.

Ying, L., Zhi-Xuan, T., Wong, L., Mansinghka, V., & Tenenbaum, J. B. (2024). Understanding epistemic language with a bayesian theory of mind. *arXiv preprint arXiv:2408.12022*.

Appendix

Below is the prompt, final response, and human benchmark for Tasks 2, 3, and 4. The prompt figures and the correct code for Task 3 contain all code written in this project. Anthony Baez made all contributions to this project.

```
// Here is a scenario. Here is a bag filled with popcorn. There is no chocolate in the
// bag. Yet, the label on the bag says “chocolate” and not “popcorn.” Sam finds the
// bag. She had never seen the bag before. She cannot see what is inside the bag.

var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

// Yet, the label on the bag says “chocolate” and not “popcorn.” She reads the label.
// We model it like this, assigning reasonable probabilities to the distribution.
var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.5, .5]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.5, .5]}) : 'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist,'chocolate')

  return contents;
})

// Now, say that Sam cannot read. Either change the probabilities
// in the prior or likelihood, or add a new function that is structured like
// viewedLabel and has an observe line that takes into account this new information.
```

Figure 7: Task 2 Prompt

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.5, .5]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.5, .5]}) : 'nothing')
}

var viewedLabelNoRead = function(state, label) {
  return categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist, 'chocolate')
  return contents;
})

var contentsPosteriorNoRead = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabelNoRead)
  return contents;
})
```

Figure 8: Task 2 Response

```
// Prior for bag contents
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

// Function for reading the label (if Sam can read)
var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.5, .5]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.5, .5]}) :
          'nothing')
}


// Generic conclusion function
var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}


var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  return contents;
})
```

Figure 9: Task 2 Human Benchmark

```
// Here is a bag filled with popcorn. There is no chocolate in the bag. Yet, the label
// on the bag says “chocolate” and not “popcorn.” Sam finds the bag. She had never
// seen the bag before. She cannot see what is inside the bag.

var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

// Yet, the label on the bag says “chocolate” and not “popcorn.” She reads the label.
// We model it like this, assigning reasonable probabilities to the distribution.
var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'], ps: [.6, .4
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'], ps: [.4, .6]]
          'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist,'chocolate')

  return contents;
})

// Now say before coming into the room, Sam’s friend told her ‘the bag in the room has
// popcorn in it, ignore the label’. Sam believes her friend. Either change the
// probabilities in the prior or likelihood, or add a new function that is structured
// likeviewedLabel and has an observe line that takes into account this new
// information.
```

Figure 10: Task 3 Prompt

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? Categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.6, .4]}) :
          label == 'popcorn' ? Categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.4, .6]}) : 'nothing')
}

var believedContents = function(state, friendInfo) {
  return (friendInfo == 'popcorn' ? Categorical({vs: ['chocolate', 'popcorn'],
                                                 ps: [.1, .9]}) :
          Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]}))
}

var conclusion = function(goalState, transition, state) {
  var state = (state == undefined) ? 'start' : state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist, 'chocolate')

  var friendInfoDist = believedContents(contents, 'popcorn')
  observe(friendInfoDist, 'popcorn')

  return contents
})
```

Figure 11: Task 3 Response

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                              ps: [.6, .4]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                            ps: [.4, .6]}) : 'nothing')
}

var believedContents = function(state, friendInfo) {
  // Sam cannot read, so the label doesn't inform her belief
  return (friendInfo == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                                   ps: [.5, .5]}) :
          friendInfo == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                                 ps: [.1, .9]}) : 'nothing')}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  var friendInfoDist = conclusion(contents, believedContents)

  observe(labelDist,'chocolate')
  observe(friendInfoDist, 'popcorn')
  return contents;
})
```

Figure 12: Task 3 Response with Manually Fixed Code

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                             ps: [.83, .27]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                           ps: [.27, .83]}) : 'nothing')
}


var infoFromFriend = function(state, contentsInfo) {
  return (contentsInfo == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                                    ps: [.91, .09]}) :
          contentsInfo == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                                  ps: [.09, .91]}) :
          'nothing')}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  var friendInfoDist = conclusion(contents, infoFromFriend)


  observe(labelDist,'chocolate')
  observe(friendInfoDist, 'popcorn')
  return contents;
})
```

Figure 13: Task 3 Human Benchmark

```
// Here is a scenario. Sam fills a bag with popcorn and closes it. There is no
// chocolate in the bag. Sam looks at the bag. She cannot see what is inside the bag.

var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

// Sam reads the label. The label says the bag has chocolate in it. We model it like
// this, assigning reasonable probabilities to the distribution.
var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                             ps: [.6, .4]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                           ps: [.4, .6]}) :
          'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist,'chocolate')

  return contents;
})

// Sam herself wrote the label and put it on the bag. Either change the probabilities
// in the prior or likelihood, or add a new function that is structured like
// viewedLabel and has an observe line takes into account this new information.
```

Figure 14: Task 4 Prompt

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})

var viewedLabel = function(state, label) {
  return (label == state) ? categorical({vs: [state], ps: [1.0]}) : categorical({vs: [
}

var conclusion = function(goalState, transition, state) {
  var state = (state == undefined) ? 'start' : state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(label, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.5, .5]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist, 'chocolate')

  return contents;
})
```

Figure 15: Task 4 Response

```
var contentsPrior = Categorical({vs: ['chocolate', 'popcorn'], ps: [.97, .03]})

var viewedLabel = function(state, label) {
  return (label == 'chocolate' ? categorical({vs: ['chocolate', 'popcorn'],
                                             ps: [.84, .26]}) :
          label == 'popcorn' ? categorical({vs: ['chocolate', 'popcorn'],
                                           ps: [.26, .84]}) :
          'nothing')
}

var conclusion = function(goalState, transition, state) {
  var state = (state==undefined)?'start':state
  return Infer(function() {
    var label = sample(contentsPrior)
    condition(goalState == transition(state, label))
    return label
  })
}

var contentsPosterior = Infer({method: 'enumerate'}, function() {
  var contents = categorical({vs: ['chocolate', 'popcorn'], ps: [.97, .03]})
  var labelDist = conclusion(contents, viewedLabel)
  observe(labelDist, 'chocolate')

  return contents;
})
```

Figure 16: Task 4 Human Benchmark