# A Domain-Specific Probabilistic Programming Language for Reasoning about Reasoning (or: a memo on memo)

KARTIK CHANDRA and TONY CHEN, MIT, USA
JOSHUA B. TENENBAUM and JONATHAN RAGAN-KELLEY, MIT, USA

The human ability to think about thinking ("theory of mind") is a fundamental object of study in many disciplines. In recent decades, researchers across these disciplines have converged on a rich computational paradigm for modeling theory of mind, grounded in *recursive probabilistic reasoning*. However, practitioners often find programming in this paradigm extremely challenging: first, because thinking-about-thinking is confusing for programmers, and second, because models are extremely slow to run. This paper presents **memo**, a new *domain-specific* probabilistic programming language that overcomes these challenges: first, by providing specialized syntax and semantics for theory of mind, and second, by taking a unique approach to inference that scales well on modern hardware via array programming. memo enables practitioners to write dramatically faster models with much less code, and has already been adopted by several research groups.

## 1 INTRODUCTION

*The vitality of language lies in its ability to limn the actual, imagined and possible lives of its speakers, readers, writers.*

Toni Morrison, Nobel Lecture for Prize Awarded in Literature, 1993

*The sentence "Mary thought she saw a ghost" has a main clause that is true ("Mary thought") and an embedded clause that is false ("she saw a ghost"). Such syntax may give learners the logical tools for understanding the false beliefs of others.*

Pyers and Senghas [108]

Human beings have the capacity not only to *think*, but to think *about* thinking: our minds can reason recursively about other minds. This remarkable ability, called **"theory of mind,"** underlies nearly everything we do: from playing a game, to explaining a concept, persuading a jury, initiating a collaboration, consoling a friend, recognizing a misconception, or waging a war.

Because theory of mind is so central to the human experience, many disciplines have sought to understand it through computational models. These disciplines include cognitive science [12, 13, 128], social psychology [2, 3, 15, 35, 77, 80, 138], linguistics [76, 87, 118, 122, 139], ethics [91, 111, 112], and behavioral economics [8, 27, 86, 123]. Such models are often applied to engineer real-world systems in robotics [45, 50, 101], education [29, 113, 120], human-computer interaction [31, 38], program synthesis [1, 106, 130], and computer graphics [28, 33].

Interestingly, models across these many disciplines are often more alike than they are different. They are united by recurring motifs like Bayesian inference (because minds hold beliefs under uncertainty), decision theory (because minds act rationally to pursue their goals), and recursion (because minds think about other minds). Textbooks from various disciplines introduce in similar ways how these motifs emerge and operate when modeling minds [47, 67, 69, 90, 96, 118]. We venture, then, that there is a whole programming paradigm lurking here—a *probabilistic* programming paradigm, which we call the **"Recursive Rationality" (RR) paradigm**.

RR models can be implemented elegantly in probabilistic programming languages (PPLs) by using nested inference queries to model recursive Bayesian reasoning [125]. But—as we have concluded from many years of collaborations with practitioners—programming in the RR paradigm remains extremely challenging, even with the best of today's PPLs. As is often the case with programming, the challenges lie broadly along two axes: correctness and efficiency. The **problem with correctness** is a matter of abstractions. Theory of mind comes with a rich vocabulary of concepts (verbs like *believe, think, see, do, want, imagine*), but most PPLs only come with a small set of primitives (verbs like *sample, infer, condition*). Programmers must translate from the former to the latter, but because there is no way to check if semantics are preserved during the translation, they are often bitten by subtle bugs, like accidentally having one agent "mind-read" another agent by mixing variables that "belong" to different minds. The **problem with efficiency** is that thinking about thinking leads to a combinatorial explosion in possible world states (not only in a single mind, but in possible minds imagined by other minds...), which in turn leads to extremely slow inference. We believe these two issues are serious obstructions to scientific progress: across disciplines, scientists typically need *weeks* of implementation work and *days* of compute to test their ideas.

In this paper, we address these issues with a new *domain-specific* PPL specialized for the RR paradigm, which we call **memo** ("mental models"). In particular, we make the following contributions:

(1) We recognize a **distinct class of bugs** faced by RR practitioners (§1.1.1), and present a language whose syntax and semantics protect users from such bugs (§3.1).
(2) We characterize the scalability challenges faced in practice by RR practitioners (§1.1.2), and address them by noticing that RR models are particularly well-suited for acceleration on modern hardware (§1.2.2). We thus take a **unique approach to inference**: we lower models to array programs that can be compiled to fast parallel code (§3.2). Our inference algorithm is **end-to-end differentiable**, enabling rapid parameter fitting by gradient descent.
(3) Through several case studies of classic RR models, we show that memo's design decisions enable practitioners to write **dramatically faster models with far less code** compared to expert-written implementations in existing general-purpose PPLs (§4.1).
(4) We make possible **new, exotic kinds of models** that RR practitioners are increasingly interested in: models that efficiently integrate with the broader deep learning ecosystem (§4.3.1), and models that reflect on the computational cost of performing inference (§4.3.2).
(5) We release memo as open-source software (https://github.com/kach/memo). memo has **already been adopted by several research groups**, delivering speedups of several orders of magnitude, and enabling ambitious new extensions to their models (§4.2).

In the rest of this introduction, we discuss the two key challenges that motivated us to design memo (§1.1), and preview the two key insights behind memo's design (§1.2). We then introduce memo by example (§2), study its implementation (§3), and evaluate it through a series of case studies (§4).

## 1.1 Motivating memo: the two challenges

*1.1.1 Correctness.* Correctly implementing RR models is difficult for many reasons. Perhaps most obviously, situations involving many agents and recursive layers can easily get muddled and confusing (see for instance this clip from the sitcom *Friends*: youtube.com/watch?v=a4CS2tCjAgk).

Yet even for simple models that researchers fully understand, implementing them in code can be tricky and fraught. One simple class of common bugs is caused by errors in implementing Bayesian calculations: many researchers implement inference by hand in R or MATLAB, where it is easy to (e.g.) forget to normalize a probability distribution. Such issues can be addressed by using PPLs to automate nested inference [125], so we do not say more about them here.

The more interesting class of bugs occurs even when using PPLs. While existing PPLs guarantee that probabilistic computations will be *mathematically correct* (according to the PPL's semantics), they do not guarantee that the computations will be *meaningful* with respect to our basic intuitions about agency. As we discussed earlier, when programming in the RR paradigm using a PPL we have to translate our intuitive "mental language" of agency (*believe, think, see, do, want, imagine, …*) to the probabilistic programming language (*sample, condition, infer, …*). This translation may introduce subtle bugs that cause models to violate our basic intuitions about agency.

Let us take a motivating example. A common pattern in RR models is to have an agent make a *choice* based on some *utility function*. Concretely, suppose one morning Alice chooses a restaurant for a dinner reservation, weighing the quality and cost of a meal there. Traditionally, such decision-theoretic situations are encoded in PPLs by *sampling* the choice uniformly at random, and then *factoring* on the agent's utility function [47]. For example, in WebPPL [66], we might write:

```
var r = sample(Uniform(Restaurants));
factor(quality(r) - cost(r));
```

The **factor** statement causes inference to favor execution paths with high utility (in the spirit of "planning as inference" [25]). Hence, inferring a distribution over $r$ predicts Alice's behavior: she prefers better, cheaper restaurants, and is indifferent among comparable ones.

Unfortunately, this "spooky action at a distance" between **sample** and **factor** can lead to subtle bugs. For example, suppose Alice slightly prefers outdoor restaurants over indoor ones (say, quality 11 vs. 10), but if it were to storm that evening, she would experience a very high cost for being outdoors (say, a penalty of $-100$). We might expect to encode this additional desideratum by factoring in a new random variable $s$ for the storm:

```
var r = sample(Uniform(Restaurants));
var s = sample(Bernoulli(0.5));  // suppose 50% chance of storm
factor((outdoor(r) ? 11 : 10) + (s && outdoor(r) ? -100 : 0));
```

However, this seemingly innocent encoding of the modified scenario is actually incorrect. It inadvertently gives "control" of environmental randomness to Alice, erroneously predicting that Alice will make risky, overconfident reservations under the illusion that she can influence the weather! In this case, even though our intuition says that Alice should play it safe and pick an indoor restaurant, and even though picking an outdoor restaurant has dramatically lower expected utility ($-39 \ll 10$), WebPPL's inference predicts that Alice is likely to pick an outdoor restaurant 58% of the time. (See Levine [92, §2.3] for more on this issue in the context of reinforcement learning.)

By analogy to "perturbation confusion" in automatic differentiation [121], we might call such bugs "*perpetration* confusion": confusion about which agent is responsible for making which choices. As we will see below, our proposed solution to perpetration confusion is analogous to Siskind and Pearlmutter's solution to perturbation confusion: "tagging" choices with agents. We will show a real-world example of perpetration confusion, and how memo addresses it, in Section 4.1.2.

#### 1.1.2 *Efficiency.*

RR models are notoriously slow to evaluate, largely because of the combinatorial explosion of possible world states in different agents' mental models (including in their mental models of each other!). This severely limits the scope and scale of problems that researchers consider.

Nested inference also poses its own challenges. When nesting approximate inference methods, the "inner" inference algorithm must produce sufficiently precise results for the "outer" inference algorithm to use. Typically this requires increasing the number of samples taken, which dramatically slows down critical inner loops (see a real-world example of this in §4.2.1). Selecting inference algorithms for each recursive level is also a serious burden on programmers, especially as the number of hyperparameters to tune increases.

Finally, in practice empirical scientists not only *run* models, but also *fit* models' free parameters to experimentally-collected data. This is typically done by grid search, which can require millions of model evaluations to fit even very simple models (exponential in the number of parameters). Researchers may additionally wish to fit the model for multiple different experimental conditions ("treatment groups"), fit multiple different ablations of the model, and cross-validate their results by independently fitting the model to multiple different subsets of the data. These needs dramatically amplify the cost of a single model evaluation, making even relatively simple models quite costly to work with. Researchers we talked to often reported that inference alone takes on the order of minutes for their models, and model fitting and cross-validation adds up to *days* of computation.

These issues make efficiency a fundamental bottleneck to productivity and progress: the cost of model fitting makes it difficult to tinker with models, test intuitions, and iterate on ideas. It also forces researchers to scale down models, or to perform a less thorough hyperparameter sweep than they would like. Of course, grid search and cross-validation are "embarrassingly parallelizable" jobs and can be accelerated dramatically on modern hardware. But cognitive scientists typically lack the expertise needed to recognize and take advantage of even very-low-hanging opportunities for performance optimization. Similarly, gradient-based optimization would undoubtedly be more efficient than grid search, but very few PPLs support automatic differentiation through inference.

Beyond these practical issues, there is also a scientific question at stake. RR models may predict human intuitions very well, but they are sometimes critiqued for being too computationally expensive to be a plausible model of how the human mind produces those intuitions [97, 134]. The possibility of faster implementations gives scientists a way to resolve this tension.

## 1.2 memo's design: the two insights

*1.2.1 On correctness: respecting our "metatheory of mind".* As hinted above, our insight about correctness is to **provide dedicated syntax for associating random choices with agents**. In memo, we would implement the restaurant-choice problem by writing that *Alice* chooses the restaurant, and the *world* chooses the weather (see supplementary materials for full implementation). This specialized syntax allows us to statically catch and prevent bugs like perpetration confusion.

More generally, by making agency explicit in the syntax and semantics of the language, memo can statically enforce a wide variety of conditions in our intuitive "metatheory of mind," ensuring that models are consistent with basic principles of agency. For example, memo enforces that there is no inadvertent "mind control" or "mind reading" in models. We will discuss memo's static semantics, and how they reify key principles of agency, in Section 3.1.

*1.2.2 On efficiency: "PPL meets APL".* Our insight about efficiency is that the vast majority of real-world RR models are amenable to exact enumerative inference: structurally, they require only a finite, statically-known sequence of samples from discrete distributions with modest cardinality. Conveniently, recent work has shown that exact enumerative inference in such models can often be compiled to efficient *array programs*, where the arrays represent joint distributions over discrete domains [103, 104]. This suggests a possible opportunity to dramatically accelerate RR models.

We can see a hint that we are on the right track by looking at how researchers describe RR models. Many classic models are discussed in the literature in two complementary ways: first, in the language of recursive trees, and second, in the language of arrays. For example, the famous value iteration algorithm is sometimes described as memoized graph search [16], and other times as an iterated operation on a value table [19]. Similarly, the Rational Speech Acts model (discussed in-depth below) is sometimes described as nested inference [52], and other times as operations on a "truth table" [106]. Economists too speak in terms of both "game trees" and "payoff matrices."

With this in mind, memo is designed to **compile nested enumerative inference to array programs**. This offers users a whole host of immediate benefits: (1) inference is exact and has **no hyperparameters**; (2) array programs with simple loop structures are easy to compile to **fast, optimized code**; (3) inference can be **parallelized for free**, both within a single evaluation (over possible world states) and when fitting parameters (over batches of data); (4) existing array program compilers can accelerate inference on **GPUs and other hardware platforms**; (5) inference natively supports **automatic differentiation**, which dramatically accelerates parameter fitting; and (6) models can seamlessly interoperate with the host array language's **library ecosystem**, including packages for deep learning, data visualization, and more. We will discuss how memo compiles models to array programs in Section 3.2.

## 2 A DEMO OF MEMO

We will introduce memo with the “drosophila” or “hello, world” of the RR paradigm: the highly-influential Rational Speech Acts (RSA) framework from cognitive linguistics [43, 52, 54, 64]. RSA models how people make *pragmatic inferences* that go beyond the literal semantics of language [68]. We will consider a classic experiment that demonstrates this, and model the results using RSA.

(To be clear, our goal is not to promote RSA over competing theories—RSA does have its critics [55, 58, 140]. Here, we are only using RSA as a paradigmatic example to introduce memo.)

### 2.1 Introducing the Rational Speech Acts framework, our running example

Consider the following game, first proposed by philosopher Lewis [93]. Imagine three objects: a green square 🟩, a green circle 🟢, and a pink circle 🔴. Your friend is tasked with asking you to pick one of them, but can only use the words “green,” “pink,” “square,” or “round.” Suppose your friend says “pink” — which object would you pick? Most people pick the pink circle 🔴. Similarly, if your friend says “square,” most people pick the green square 🟩. So far, so good. But now, suppose your friend says “green.” This is trickier, because both the green square 🟩 and green circle 🟢 are consistent with the literal denotation of the word “green.” Yet a surprising, replicable finding from cognitive linguistics is that most people have a *pragmatic* preference for the green circle 🟢 over the green square 🟩 — for example, in one study 64% of people preferred the former [109]. What is going on here? It seems that people reason that *if* the friend wanted 🟩, they could have said “square” (unambiguously) rather than “green” (ambiguously). Hence, given that they chose the risky, ambiguous utterance “green,” they probably meant to distinguish the two circles 🟢 and 🔴.

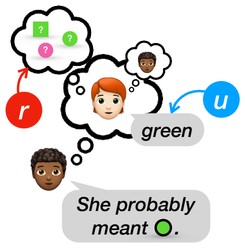


Fig. 1. The listener infers the speaker’s referent $r$ from their utterance $u$.

The RSA framework seeks to formalize this intuition in precise computational terms. It analyzes such situations as speakers ($S$) and listeners ($L$) reasoning recursively about each other under uncertainty. Let us say that initially, the listener has some prior belief about the speaker’s object $p_{\text{prior}}(r)$, where $r$ stands for *referent* (the object the speaker intended to refer to). Given the speaker’s *utterance* $u$, the listener can apply Bayes’ rule to infer $p_L(r \mid u) \propto p_S(u \mid r) \cdot p_{\text{prior}}(r)$, where $p_S(u \mid r)$ is the likelihood of the speaker saying $u$ to refer to $r$. Critically, a listener might think that a rational, strategic speaker would choose $u$ to maximize the chances that the listener infers $r$ correctly. Hence, $p_S$ may itself recursively depend on $p_L$. (This is the essence of the “recursive rationality” paradigm.)

### 2.2 First steps: expressing the “base case” of a naïve listener in memo

To express this model in memo, we will start by implementing the “base case” of a naïve listener who thinks the speaker simply chooses a true utterance uniformly at random. Let `U = ["green", "pink", "square", "round"]` be the space of possible *utterances*, `R = [`🟩, 🟢, 🔴`]` be the space of possible *referents*, and let the Boolean function `denotes(u, r)` be true iff $u \in U$ describes $r \in R$. Our naïve listener will assume that $p_S(u \mid r) \propto [\text{denotes}(u, r)]$; that is, the speaker chooses an utterance uniformly at random among those that describe the referent. Finally, for the sake of simplicity, let $p_{\text{prior}}(r) \propto 1$, i.e. assume that the speaker is equally likely to refer to each of the three referents.

memo is at heart a language for computing arrays of numbers. In this case, we wish to compute the probability table of $p_L(r \mid u)$, which is a $|U| \times |R|$ array. We thus begin our model by stating these dimensions. This notation is by analogy to array indexing: `L` is a table with dimensions $u$ and $r$, and memo will populate this table by running the model for each $u$ and $r$.

```
@memo
def L[u: U, r: R]():
```

This model involves two agents: the listener and speaker, where the listener has a mental model of the speaker. We express this using the compound statement `listener: thinks[ speaker: ... ]`. As discussed earlier, the naïve listener thinks that the imagined speaker `chooses` the true referent $r_{\text{true}}$ based on $p_{\text{prior}}(r)$, and then $u$ uniformly among denotationally consistent referents. In memo, we write this as follows. (Here, the keyword "wpp" stands for "with probability proportional to," analogous to the symbol "$\propto$" in expressions like $p_S(u \mid r) \propto [\text{denotes}(u, r)]$.)

```
listener: thinks[
  speaker: chooses($r_{\text{true}}$ in R, wpp=1), # uniform prior
  speaker: chooses($u_{\text{said}}$ in U, wpp=1 if denotes($u_{\text{said}}$, $r_{\text{true}}$) else 0)
]
```

Notice how—unlike a traditional PPL—memo associates each random choice with the agent making that choice. The variable $r_{\text{true}}$ does not exist in the listener's scope—only in the scope of the listener's model of the speaker. It is thus a type error for the listener to reference $r_{\text{true}}$ directly in an expression. However, the listener *can* reason about computation involving $r_{\text{true}}$ in the *speaker's* mind. For example, the expression `speaker[` $r_{\text{true}}$ `]` (or its syntactic sugar `speaker.`$r_{\text{true}}$) gives a random variable representing the speaker's (unknown-to-listener) referent, and `speaker[` $r_{\text{true}}$ `==` 🟩 `]` gives a binary random variable representing whether or not the referent is 🟩.

Now that we have established the listener's mental model of the speaker, we can move on to the main events. First, the listener `observes` the speaker's utterance to be $u$, and updates their belief accordingly. Then, the listener picks $r_{\text{pick}}$ proportional to their confidence that $r_{\text{pick}}$ is the speaker's true referent $r_{\text{true}}$; that is, with probability proportional to **Pr**`[speaker.`$r_{\text{true}}$ `==` $r_{\text{pick}}$`]`.

```
listener: observes [speaker.$u_{\text{said}}$] is $u$
listener: chooses($r_{\text{pick}}$ in R, wpp=Pr[speaker.$r_{\text{true}}$ == $r_{\text{pick}}$])
```

It is worth pausing here to consider what would happen if we omitted **Pr**`[...]` in the line above. From the listener's perspective, the expression `speaker.`$r_{\text{true}}$ `==` $r_{\text{pick}}$ is a binary random variable because `speaker.`$r_{\text{true}}$ is unknown. It does not make sense for an agent to choose based on a probability they cannot even evaluate! So, without **Pr**`[...]`, memo statically raises an error at that line. However, even though that expression's *actual boolean value* is unknown to the listener, its *probability of being true* is known. Hence, with **Pr**`[...]`, memo accepts the statement. In this way, memo statically reasons about knowledge and uncertainty, ensuring that expressions that span multiple agents' frames of minds are well-formed and meaningful. We discuss memo's static reasoning, and how it prevents bugs like perpetration confusion (§1.1.1), in Section 3.1.

Finally, to compute the probability table for $p_L$, we return the probability that the listener's $r_{\text{pick}}$ is equal to $r$. (Recall from above that "$r$" was defined as the second axis of the returned array.)

```
return Pr[listener.$r_{\text{pick}}$ == $r$]
```

We can run this model in Python by calling `L()` like an ordinary function. As expected, we get a $4 \times 3$ array representing the probability table of $p_L(r \mid u)$ (as a sanity-check, all rows sum to 1).

```
>>> L()
#  🟩    🟢    ○
[[0.5, 0.5, 0. ],  # "green"
 [0. , 0. , 1. ],  # "pink"
 [1. , 0. , 0. ],  # "square"
 [0. , 0.5, 0.5]]  # "round"
```

For the unambiguous utterances ("pink" and "square"), the model correctly picks out ○ and 🟩 respectively, whereas for the ambiguous utterances ("green" and "round") it has no preference between the two denotationally-consistent possibilities. This is contrary to our intuitions, and to Qing and Franke's data, suggesting that we are not done yet...

## 2.3 Completing the model: adding recursive reasoning

Now, let us see what happens if the listener thinks the speaker is *themselves* thinking about the listener (who, indeed, might recursively be thinking about the speaker, and so on). We extend our code to model this in three steps, summarized in Figure 2.

First, we parametrize the listener by $\ell$, which denotes the recursive level of the current listener, such that $\ell = 0$ is equivalent to the naïve listener. (Notice that the *parameter* $\ell$ is syntactically distinct from the *axes* $u$ and $r$, because $\ell$ does not affect the shape of the array to be computed.) Second, we modify the speaker to recursively refer to the listener at level $\ell - 1$ when $\ell > 0$. This is as easy as referring to $L$ with the appropriate array indexing notation: we can write `L[u, r](ℓ - 1)`. Third, following standard practice in decision theory [53, 95], we model the speaker as choosing $u$ from a softmax over the listener's predicted posterior belief. We thus introduce a second parameter, $\beta$, to modulate the temperature of the softmax. For high $\beta$, the speaker approaches deterministically choosing the optimal $u$, whereas for low $\beta$ the speaker approaches a uniform choice.

Running the updated model with $\ell = 1$ and $\beta = 1.0$ shows the pattern we expect: a slight preference for the green *circle* 🟢 over the green square 🟩 for utterance "green."

```
>>> L(ℓ=1, β=1.0)
#  🟩    🟢    ●
[[0.43, 0.57, 0.  ],  # "green"
 [0.  , 0.  , 1.  ],  # "pink"
 [1.  , 0.  , 0.  ],  # "square"
 [0.  , 0.57, 0.43]]  # "round"
```

Compared to an idiomatic textbook implementation [118] of this model in WebPPL, our memo implementation is 2× shorter (10 vs. 20 lines, excluding helpers like `denotes`) and 7× faster (80µs vs. 550µs). It is also easier to read: for example, the WebPPL version confusingly uses `Infer` and `uniformDraw` to model the speaker's choice of $u_{\text{said}}$, even though the speaker is not really performing inference, and certainly not drawing $u_{\text{said}}$ uniformly.

## 2.4 Using our model: a peek into the scientist's workflow

As we discussed in Section 1.1.2, an RR practitioner's work does not end with expressing a model: they typically go on to fit it to data and validate it. Let us see what this looks like with memo.

```
@memo
def L[u: U, r: R](ℓ, β):
  listener: thinks[
    speaker: chooses(r_true in R, wpp=1),
    speaker: chooses(u_said in U, wpp=
      denotes(u_said, r_true) * (1 if ℓ==0 else exp(β*L[u_said, r_true](ℓ-1, β))))
  ]
  listener: observes [speaker.u_said] is u
  listener: chooses(r_pick in R, wpp=Pr[speaker.r_true == r_pick])
  return Pr[listener.r_pick == r]
```

Fig. 2. A complete implementation of RSA in memo. Highlighted fragments represent changes required to go from a naïve listener (§2.2) to a full recursively-reasoning listener (§2.3).

Recall that Qing and Franke found that 64% of people choose the circle for utterance "green." At $\beta = 1.0$ our model predicts 57%: close but not perfect. Can we do better by optimizing $\beta$? Let us define a mean-square-error loss between the model predictions `L(1, β)` and the dataset `Y` of human responses: `def loss(β): return np.mean((L(1, β) - Y)**2)`. Typically, researchers optimize such losses by slow, expensive grid search. Fortunately, because memo is built on JAX, we can use `jax.vmap(loss)` to search a large space of candidate $\beta$-values *in parallel*. We can even use `jax.grad(loss)` to differentiate through our loss function, and recruit existing gradient-based optimizers (e.g. Adam) to rapidly find the best $\beta$. Finally, because arrays are natively supported across the scientific Python ecosystem, we can plot the model against the human data using the `matplotlib` library by directly passing model output to library functions like `plt.scatter` or `plt.bar`.

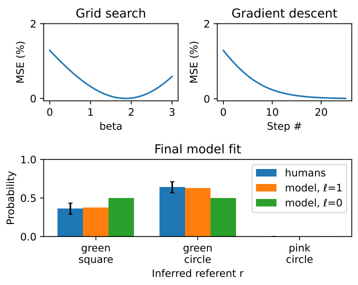


Fig. 3. Using memo, scientists can easily fit models by parallelized grid search (left) or gradient descent (right), and visualize results (bottom).

Figure 3 illustrates fitting $\beta$ to Qing and Franke's data by grid search (within 70ms) and gradient descent (within 250ms), and shows a plot of the data against the best-fit model at $\ell = 1$ and $\ell = 0$. Clearly, the model predicts human behavior well at $\ell = 1$, whereas at $\ell = 0$ the model's predictions differ significantly from humans. While more statistical analysis is certainly needed, this type of result could be used to argue that people are reasoning at $\ell = 1$ rather than $\ell = 0$.

## 2.5 The rest of the language

This concludes our tour of memo. The full abstract syntax of the language is given in Figure 4. It contains only a couple of constructs we have not yet encountered. We highlight them below.

*2.5.1 Programming in the subjunctive mood.* Agents often reason in terms of hypotheticals and counterfactuals, asking "what if?" questions to guide their actions. There is strong evidence that humans are performing this type of simulation when decision-making [17, 62, 145]. To model such reasoning in memo, we introduce the `imagine` expression, which allows *imagining* events in a temporary scope, and then evaluating an expression in that hypothetical world. (`imagine` is like an IIFE in JavaScript, and can be desugared to memo primitives as a call to a freshly-generated model.)

*2.5.2 Thinking about* not *thinking.* Recent work in cognitive science has considered how agents take into account the cost of thinking itself [94, 133]. For example, agents might use an approximation when it is good enough that the marginal cost of additional thinking (in time, energy, memory, etc.) is not worthwhile. Modeling "resource-rationality" has thus far been challenging because it is difficult to reflect on the computational cost of inference in a typical PPL. memo provides a construct for reflecting on computational cost by extracting the number of FLOPs needed from JAX's compile-time analysis. memo models can use this information to have agents make choices based on resource considerations. We show an example of how this feature can be used in Section 4.3.2.

## 3 DESIGN & IMPLEMENTATION

We set out to create a language that helps scientists write correct and efficient RR models. In this section, we will show how memo is designed to achieve these goals. We start by describing memo's front-end, explaining how the static semantics of the language ensure correctness with respect to our intuitive notions of agency (§3.1). Next, we describe how memo models are compiled into array programs to enable efficient inference (§3.2). Finally, we discuss practical considerations for supporting RR practitioners, and how they influenced us (§3.3). We focus here on concrete, intuitive

| | | | | |
|---|---|---|---|---|
| Statement | $s$ | ::= | `a: chooses(x in D, wpp=e)` | *sampling* |
| | | \| | `a: observes [b.x] is c.y` | *conditioning* |
| | | \| | `a: thinks [s*]` | *recursive reasoning* |
| Expression | $e$ | ::= | $\ell$ | *static literal* |
| | | \| | `op(e*)` | *primitive operator* |
| | | \| | $x$ | *choice* |
| | | \| | $a[e]$ | *query agent* |
| | | \| | **E** $[e]$ | *expectation* |
| | | \| | $m\,[a.x*]\,(\ell*)$ | *memo reference* |
| | | \| | `imagine [s*, e]` | *hypotheticals* |
| | | \| | `cost @ m(ℓ*)` | *resource reflection* |
| Static literal | $\ell$ | ::= | $n$ | *numeric literal* |
| | | \| | $\theta$ | *model parameter* |
| | | \| | `op(ℓ*)` | *primitive operator* |
| Model | $\mu$ | ::= | `@memo def m[(x: D)*](θ*): s*; return e` | |

Fig. 4. The syntax of memo. Metavariables $a, b, c$ range over agents, $x, y$ over names of choices, $n$ over real numbers, $D$ over names of domains, $\theta$ over names of parameters, and $m$ over names of models. Stars ($*$) denote comma-separated possibly-empty sequences. **Pr**, **Var**, and **H** (entropy) desugar in terms of **E**.

explanations. Concurrent work [5, anon. ref.] formalizes memo's static and denotational semantics, building on foundations laid by Zhang and Amin [143] for formalizing nested inference.

## 3.1 Front-end: tracking "frames of mind"

As we discussed in Section 1.2.1, memo is designed to make sure models are consistent with our intuitions about agency in the real world. In this section, we will discuss how memo achieves this by giving an overview of the static semantics of the language. Along the way, we will highlight four basic **principles of agency** (marked ①-④), and explain how they are reified by memo's design.

The central data structure in the memo compiler is the "frame of mind," which tracks an agent's beliefs, along with that agent's beliefs about *other* agents' beliefs. To introduce frames, let us continue our running example from Section 2: we will study step by step how frames evolve over the course of compiling RSA. As shown in Figure 5, a frame consists of a set of *agents* (including the self), those agents' *choices*, and pointers to those agents' own frames (thus forming a tree of nested frames). Choices in an agent's frame can be *known* or *uncertain* to that agent.

**Panel A** shows how every memo model begins: with an initial "root" frame, which we will call the *observer's* frame. The observer's frame contains only the two variables $r$ and $u$, which are bound by the axes declared in the top-level model definition ("`def L[r: R, u: U]...`"). These variables are *known* to the observer: there is no uncertainty about their values.

**Panel B** shows what happens when we initialize the listener's mental model of the speaker ("`listener: thinks[speaker: chooses(...)]`"). memo creates a fresh frame for the listener, and another fresh frame for the (listener's model of the) speaker. The observer's frame contains a pointer to the listener's frame, which in turn contains a pointer to the imagined speaker's frame. At this point, the speaker's frame contains two variables $r_{\text{true}}$ and $u_{\text{said}}$, which are both *known* to the speaker because they were chosen by the speaker. However, in the *listener's* frame those same variables are *uncertain*, i.e. random variables. This brings us to the first principle of agency reified by memo: the principle of ① **"no mind reading,"** which is reflected in the fact that agents always *know* their own choices, but are by default *uncertain* of choices other agents have made.

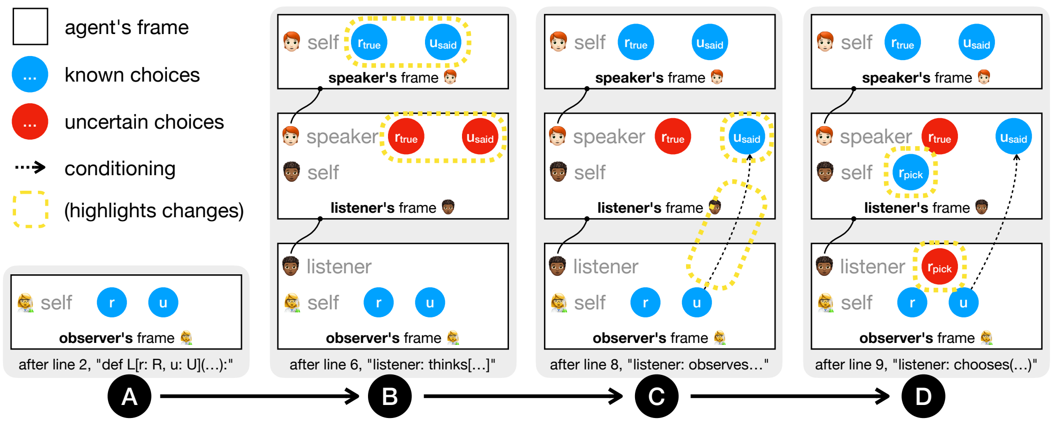


Fig. 5. The memo compiler statically tracks nested "frames of mind" (§3.1). Here, we show in four stages how frames evolve over the course of processing the RSA implementation shown in Figure 2.

**Panel C** shows the result of the listener observing `speaker.`$u_{\text{said}}$ to be $u$. This statement has two effects. First, in the listener's frame, the status of `speaker.`$u_{\text{said}}$ changes from *uncertain* to *known*. Second, in the observer's frame, the listener's `speaker.`$u_{\text{said}}$ gets linked to $u$ (this information is stored in an auxiliary table in the observer's frame). The observer now substitutes $u$ for `speaker.`$u_{\text{said}}$ when modeling the listener's computations. For example, from the observer's perspective, the boolean expression `listener[speaker.`$u_{\text{said}}$` == "green"]` is now equivalent to `u == "green"`.

Notice that this is the only place in the model where the listener's `speaker.`$u_{\text{said}}$ and the observer's $u$ are related. We could also have written `listener: `**`observes`**` [speaker.`$u_{\text{said}}$`] `**`is`**` `$w$, where $w \neq u$. This is by design. It reifies a second principle of agency: that ② **agents can acquire false beliefs**. For example, we can model a "man-in-the-middle" attack as `listener: `**`observes`**` [speaker.`$u_{\text{said}}$`]` **`is`** `attacker.`$u_{\text{fake}}$. The listener would think that the speaker really said $u_{\text{fake}}$, whereas the observer would understand that the listener unwittingly received the attacker's message. Consequently, `listener[speaker.`$u_{\text{said}}$`]` and `attacker.`$u_{\text{fake}}$ would become equivalent in the observer's frame, but not in the listener's frame, which is oblivious to even the presence of an attacker. This reifies the principle of ③ **referential opacity of belief** [56, 110].

Relatedly, notice that the observer's frame does not contain any reference to the speaker's $r_{\text{true}}$ or $u_{\text{said}}$. This is also by design. The observer and listener do not have to have the same mental model of the speaker. For example, the observer might think that in reality, the speaker is babbling randomly, and there is no $r_{\text{true}}$ at all influencing the utterances. The listener may nonetheless be erroneously *interpreting* the speaker's utterances as intentional and *attributing* an $r_{\text{true}}$ being communicated (see [44, 74]). Hence, $r_{\text{true}}$ should not appear in the observer's frame, only in the listener's.

Finally, **Panel D** shows what happens when the listener chooses $r_{\text{pick}}$. Just like our discussion of Panel B, we see that $r_{\text{pick}}$ appears in the listener's frame as a known variable, and in the observer's frame as uncertain. Let us take this opportunity to study one last principle of agency reified by memo. In Section 2.2 we discussed how it is important that the listener *know* the probabilities with which they sample $r_{\text{pick}}$. This constraint effectively says that there is ④ **"no mind control."** If an agent cannot compute why they are making a given choice, then it is as if someone else made the choice for them. For example, if Alice (secretly) picks an ice cream flavor and expects Bob to choose the same one, then her mental model of Bob effectively robs him of his free will. Situations like these give rise to bugs like perpetration confusion (§1.1.1), and more generally to thorny philosophical paradoxes like Newcomb's Problem [102, 135]. memo statically prevents this by tracking whether *expressions* are known or unknown in an agent's frame. Expressions are tainted as unknown if they involve unknown choices (with the exception that `E[`$e$`]` is known even if $e$ is unknown), and memo enforces that agents *know* expressions they use to specify probabilities for their own choices.

## 3.2 Back-end: lowering memo to an array program

As we discussed in Section 1.2.2, RR models often only require discrete enumerative inference, and are thus well-suited to be compiled to array programs. In the rest of this section, we will complete our running example by explaining how memo compiles our RSA model to an array program.

The key idea is to maintain an array for each frame, which represents that agent's current beliefs about the world. The array has a separate dimension for each choice tracked in that frame, and its entries give the joint probability of unknown choices conditioned on known choices. Figure 6 shows how the array associated with the listener's frame evolves over the course of three statements in RSA (at $\ell = 0$ for simplicity). We focus on the listener's frame because it is the most interesting: the observer and speaker have no uncertain choices, so their arrays are simply filled with ones.

The listener's array $\mathfrak{L}$ is initialized as the rank-zero (i.e. dimensionless) array containing the scalar 1: because the listener has not yet modeled any uncertain choices, there is vacuously only one possible world state, to which the listener assigns probability 1.

**Panel B$_1$** shows what happens when the listener models the speaker's choice of $r_{\text{true}}$. First, $\mathfrak{L}$ is expanded with a new size-$|R|$ dimension for $r_{\text{true}}$, making it a 1D array (shown as a bar graph). Next, memo evaluates the choose statement's **wpp** expression (the *likelihood*) for each $r_{\text{true}} \in R$ and normalizes the result. In this case, the likelihood expression is the constant 1 (i.e. uniform), which normalizes to 1/3 for the three referents in $R$. Finally, memo updates $\mathfrak{L}$ by multiplying it pointwise with the normalized likelihoods to get $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$. This should be consistent with our intuitions: thus far, the listener has uniform priors over the three possible referents the speaker may have in mind.

**Panel B$_2$** shows the same process unfold when the listener models the speaker choosing $u_{\text{said}}$. First, $\mathfrak{L}$ is expanded with a new size-$|U|$ dimension for $u_{\text{said}}$, making it a 2D array (shown as a heatmap). Then, memo evaluates the likelihood expression `denotes(`$u_{\text{said}}$`, `$r_{\text{true}}$`)` for each $u_{\text{said}}$ and $r_{\text{true}}$ and normalizes the result along the $u_{\text{said}}$ axis (notice that both of these operations are easily parallelized on modern hardware). Finally, memo updates $\mathfrak{L}$ by multiplying it pointwise with the

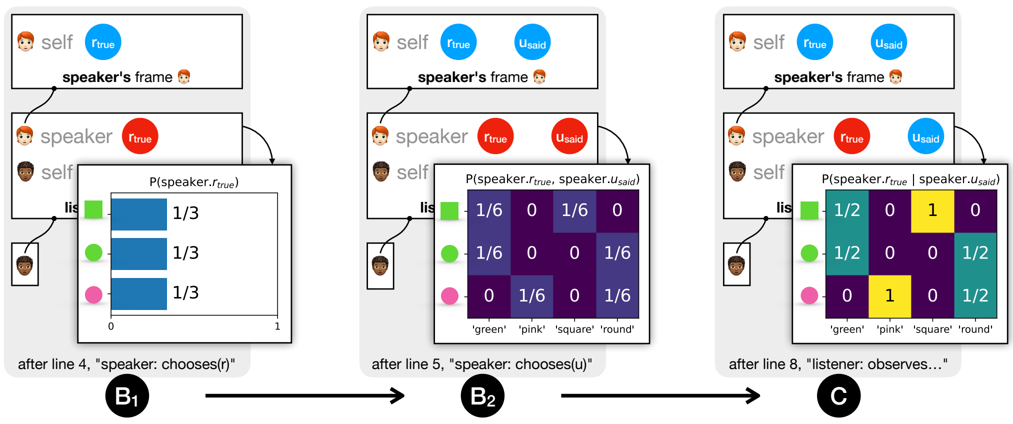


Fig. 6. memo tracks each agent's uncertainty, using arrays to represent conditional distributions (§3.2). Here, we show how the listener's beliefs evolve over the course of the RSA implementation shown in Figure 2.

normalized likelihoods. The update to $\mathfrak{L}$ is thus given by:

$$\mathfrak{L} \leftarrow \underbrace{\overbrace{\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\end{bmatrix}}^{\text{denotes}(u_\text{said}, r_\text{true})} \oslash \overbrace{\begin{bmatrix}2 & \dots\\ 2 & \dots\\ 2 & \dots\end{bmatrix}}^{\text{Normalizer} \rightarrow}}_{\text{Likelihood } p(u_\text{said} \mid r_\text{true})} \odot \underbrace{\overbrace{\begin{bmatrix}1/3 & \dots\\ 1/3 & \dots\\ 1/3 & \dots\end{bmatrix}}^{\text{current } \mathfrak{L}}}_{p(r_\text{true})} = \underbrace{\begin{bmatrix}1/6 & 0 & 1/6 & 0\\ 1/6 & 0 & 0 & 1/6\\ 0 & 1/6 & 0 & 1/6\end{bmatrix}}_{p(r_\text{true}, u_\text{said})}$$

Here, rows correspond to referents (🟩, 🟢, 🔴) and columns to utterances ("green," "pink," "square," "round"); $\oslash$ and $\odot$ denote pointwise operations, and ellipses denote NumPy-style broadcasting. (memo uses broadcasting to minimize redundant computation and storage where possible. For example, the normalizer need only be stored once for each row of $\mathfrak{L}$.)

Lastly, **Panel C** shows what happens when the listener observes the speaker's $u_\text{said}$. That choice becomes known in the listener's frame, and thus the listener must represent the conditional distribution $p(r_\text{true} \mid u_\text{said})$. To do this, memo normalizes $\mathfrak{L}$ along the $r_\text{true}$ axis.

$$\mathfrak{L} \leftarrow \underbrace{\overbrace{\begin{bmatrix}1/6 & 0 & 1/6 & 0\\ 1/6 & 0 & 0 & 1/6\\ 0 & 1/6 & 0 & 1/6\end{bmatrix}}^{\text{current } \mathfrak{L}}}_{p(r_\text{true}, u_\text{said})} \oslash \underbrace{\overbrace{\begin{bmatrix}2/6 & 1/6 & 1/6 & 2/6\\ \vdots & \vdots & \vdots & \vdots\end{bmatrix}}^{\text{Normalizer} \downarrow}}_{p(u_\text{said})} = \underbrace{\begin{bmatrix}1/2 & 0 & 1 & 0\\ 1/2 & 0 & 0 & 1/2\\ 0 & 1 & 0 & 1/2\end{bmatrix}}_{p(r_\text{true} \mid u_\text{said})}$$

Notice that by mechanically following memo's compilation rules, we have essentially recovered "for free" the efficient array-based implementation of RSA described by Pu et al. [106]: alternating row-wise and column-wise normalizations of an $|R| \times |U|$ array! We will see in Section 4.1.3 that memo also recovers the classic tabular value iteration algorithm in the same way.

Using the array $\mathfrak{L}$, we can perform a variety of interesting calculations from the listener's perspective. For example, suppose we were interested in the listener's confidence that $r_\text{true}$ is a green shape, i.e. evaluating the expression `Pr[green(`$r_\text{true}$`)]` in the listener's frame after the listener observes $u_\text{said}$. The expression `Pr[`$e$`]` desugars to `E[`$e$`]` because the expectation of a Bernoulli random variable is its probability. Our goal is thus to compute the expectation $\mathbf{E}[\text{green}(r_\text{true}) \mid u_\text{said}] = \sum_{r_\text{true}} \text{green}(r_\text{true}) \cdot p(r_\text{true} \mid u_\text{said})$. We can implement this computation as a *tensor contraction.* First, we evaluate `green(`$r_\text{true}$`)` in each possible world state and assemble an array $e$ with the result. Then, we contract $e$ with $\mathfrak{L}$ along dimensions representing uncertain variables. In this case, there is only one uncertain dimension, so we can write this tensor contraction as matrix multiplication:

$$\Pr[\text{green}(r_\text{true})] = \underbrace{\begin{bmatrix}\text{green}(🟩) = 1 & \dots\\ \text{green}(🟢) = 1 & \dots\\ \text{green}(🔴) = 0 & \dots\end{bmatrix}^\top}_{e = \text{green}(r_\text{true})} \underbrace{\begin{bmatrix}1/2 & 0 & 1 & 0\\ 1/2 & 0 & 0 & 1/2\\ 0 & 1 & 0 & 1/2\end{bmatrix}}_{\mathfrak{L} = p(r_\text{true} \mid u_\text{said})} = \underbrace{\begin{bmatrix}1 & 0 & 1 & 1/2\\ \vdots & \vdots & \vdots & \vdots\end{bmatrix}}_{\mathbf{E}[\text{green}(r_\text{true})]}$$

This tells us that to the listener, `Pr[green(`$r_\text{true}$`)]` depends on the observed $u_\text{said}$: they think the probability is 1 if they hear "green" (of course!), 0 if they hear "pink," 1 if they hear "square" (the only square referent is green), and 1/2 if they hear "round" (only one of the two circles is green).

*Summary.* Here are key takeaways from the previous two sections: Statically, **memo maintains a tree of frames**, which track agents, their choices, and their knowledge or uncertainty about other agents' choices. **`chooses`** creates a choice that is *known* to the chooser but *uncertain* to the agent modeling the chooser. **`observes`** causes an uncertain choice to become known to the observing agent.

Agents must *know* the probabilities with which they make choices. These rules keep memo models consistent with our intuitions about agency. Dynamically, **memo maintains an array for each frame**, which represents that agent's joint probability over *uncertain* choices conditioned on *known* choices. The array has a dimension for each choice tracked in the frame. `chooses` introduces new dimensions, `observes` normalizes along dimensions, and `E[`$e$`]` is computed by tensor contraction. These operations are easily parallelized on modern hardware, making inference efficient.

## 3.3 Implementation details

A key design consideration for memo was interoperability with scientists' preferred workflows. A common source of friction for scientists is shuttling data back and forth between the PPL used for modeling (e.g. WebPPL), and the scripting language used for data analysis (typically Python, R, or MATLAB). In fact, many of our colleagues reported manually copy-pasting inputs and outputs across files, or writing brittle shell scripts to generate and parse log files.

We thus chose to embed memo in Python, building on the JAX array programming library. This choice had the added benefit of making memo instantly compatible with existing Python tooling (IDEs, notebooks, ...), the vast array-programming-based scientific Python ecosystem (including widely-used packages for data analysis and visualization), and the burgeoning JAX ecosystem (including packages for deep learning, physics simulation, and reinforcement learning).

To embed memo in Python, we provide an `@memo` decorator that parses a definition's source, translates the resulting Python AST into a memo AST, and sends it to the memo compiler. We compile the memo AST to a Python program that calls the JAX API. When a memo model is invoked, the JAX JIT compiler traces the generated Python program and compiles it to native code.

## 4 APPLICATIONS & EVALUATION

We begin this section by highlighting four classic models from the RR paradigm, using them as case studies to evaluate memo (§4.1). For each model, we compare a memo implementation to an existing expert-written implementation in a traditional PPL. We show that **memo is generally faster, while requiring less code.** Next, we briefly discuss four cognitive science projects that are currently using memo, highlighting how **memo is impacting ongoing research** in different ways (§4.2). Finally, we present two examples that showcase **advanced usage of memo**, building models that would be particularly challenging to implement in other PPLs (§4.3).

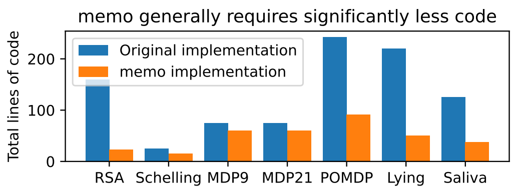
Our results across all of these models are summarized in Table 1. Following our reasoning in Section 1.1.2, we measured the total time taken to compute all relevant conditions of each model, which is the relevant day-to-day workload for practitioners. We excluded the time needed to warm up JIT compilers, because this fixed cost is far outweighed by the variable cost of model fitting and cross-validation. Unless otherwise noted, models were timed on the Apple M2 processor. We also report the number of lines of code (LOC) needed for each model to give a general sense of the effort needed to implement a model, the surface area for bugs, and the leverage provided by memo's domain abstractions. Where relevant, we separate LOC into $\ell = \ell_{\text{specific}} + \ell_{\text{general}}$ where $\ell_{\text{specific}}$ is code specific to the given problem, and $\ell_{\text{general}}$ reflects a general formalism for that type of problem that could be abstracted to a reusable library (e.g. specific maze + general route planner). **We provide code for all examples in §4.1 and §4.3 in the supplementary materials.**

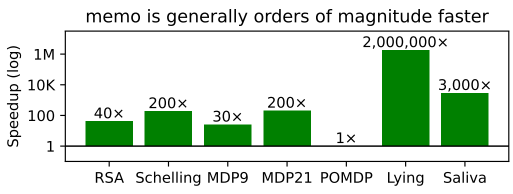
## 4.1 Case studies

We start with four classic models in the RR paradigm: rational communication, two-player games, sequential planning, and belief-space reasoning. We compared existing implementations of these models (sourced from textbooks, documentation, etc.) against re-implementations in memo.

Table 1. Summary of models and results discussed in Section 4.

| **Model** | **Lang**$_{\text{original}}$ | **LOC**$_{\text{original}}$ | **LOC**$_{\text{memo}}$ | **Time**$_{\text{original}}$ (s) | **Time**$_{\text{memo}}$ (s) |
|---|---|---|---|---|---|
| *Classic example models, compared to experts' implementations* | | | | | |
| RSA (§4.1.1) | Pyro | 37+122 | 23 | 25 | 0.6 |
| Schelling (§4.1.2) | WebPPL | 25 | 15 | 1.1 | 0.0058 |
| MDP$^{9\times9}$ (§4.1.3) | WebPPL | 75 | 60 | 12.2 | 0.48 |
| MDP$^{21\times21}$ | WebPPL | 75 | 60 | 79 | 6.9 |
| MDP$^{21\times21}$ (GPU) | WebPPL | 75 | 60 | not supported | 0.377 |
| MDP$^{21\times21}$ (inverse) | — | — | 10 | — | 0.0098 |
| POMDP (§4.1.4) | Julia | 43+199 | 30+61 | 0.025 | 0.026 |
| *Existing in-the-wild models, translated by researchers migrating to memo* | | | | | |
| Lying (§4.2.1) | Gen | 220 | 50 | 100 | $5.5 \times 10^{-5}$ |
| Saliva (§4.2.2) | R | 125 | 38 | 2.13 | $7.3 \times 10^{-4}$ |
| *New models, freshly written in memo* **(see supplementary materials for more examples)** | | | | | |
| Doctor (§4.2.3) | — | — | 120 | — | $9.2 \times 10^{-5}$ |
| Care (§4.2.4) | — | — | 50 | — | 0.433 |
| Fonts (§4.3.1) | — | — | 40+19 | — | 0.124 |
| Takeaway (§4.3.2) | — | — | 43 | — | 0.116 |
| *Summary of the summary* | | | | | |





*4.1.1 Scalar implicature.* Scalar implicature is a linguistic phenomenon where uttering a weaker claim is interpreted as implying that a stronger claim is not true [79]. A classic example is how "some" often implies "not all." For example, if a speaker says "some of my PLDI submissions were accepted this year," then a listener would likely infer that "not all" of the submissions were accepted, even though it is technically not false to say "some" (i.e. $> 0$) were accepted when all were.

The Rational Speech Acts framework predicts this phenomenon well [65]. We set utterances $U = \{\text{none}, \text{some}, \text{all}\}$, referents $R = \{0, 1, \dots, N\}$, and we say that "none" denotes $r = 0$, "some" denotes $r > 0$, and "all" denotes $r = N$. Then, RSA predicts that upon hearing "some," a listener would infer $r$ to be less than $N$, because if the speaker actually intended to communicate $N$, they would have said "all" instead of "some." We compared an implementation of this model taken directly from the example gallery of the Pyro PPL [23] against the memo implementation we developed in Section 2. We chose this example because—unlike the "green-circle" communication game we considered earlier—here it is natural to scale up the size of the problem by scaling up $N$.

As shown in Table 1, the Pyro model is 159 lines of code (37 for the model, 122 for specialized inference routines), while the memo version is 23 lines with no additional code needed for inference. At $N = 10^4$, Pyro takes 25s to perform inference while memo takes only 0.6s.

```
function alice(depth) {
 return Infer(function() {
  var a = sample(prior);
  var b = sample(bob(depth-1));
  condition(a==b);
  return a;
 });
};
```

```
@memo
def alice[a: Bar](depth):
 alice: thinks[
  bob: chooses(b in Bar,
               wpp=bob[b](depth-1)) ]
 alice: chooses(a in Bar,
               wpp=prior(a)*Pr[a==bob.b])
 return Pr[alice.a==a]
```

Fig. 7. The Schelling game (§4.1.2) in WebPPL (adapted from a textbook [47]) and idiomatic memo. The "bob" model (not shown) is defined analogously to "`alice`" in both PPLs. We also omit the definition of "`prior`."

*4.1.2 Schelling coordination games.* Economist Schelling [116] studied situations where two parties are forced to coordinate without communicating. In a now-famous experiment, he asked a sample of 36 students to imagine that they had to meet a stranger in New York City, but that no meeting place or time had been arranged beforehand. Most people said that they would try going to Grand Central Terminal at noon. This "Schelling point" emerges from recursive reasoning: people reason about where others are likely to go, which allows them to bootstrap a salient location into an emergent consensus. Stuhlmüller and Goodman [125] computationally modeled this kind of reasoning with a pair of mutually-recursive probabilistic programs ("Alice" and "Bob"). They considered a simplified setting: a town with two bars, one slightly more popular than the other. With just a few levels of recursion, rational agents converge to overwhelmingly prefer the more popular bar.

We implemented this model in memo, and compared it to an implementation taken directly from a textbook by Evans et al. [47] (Figure 7). For the purposes of timing, we extended the scenario to 100 bars and considered 100 levels of recursive reasoning. As shown in Table 1, the memo version is slightly shorter than the WebPPL version (15 lines vs. 25) and much faster (5.8ms vs 1.1sec).

This example is interesting because it highlights the risk of "perpetration confusion" in traditional PPLs (§1.1.1). The textbook WebPPL implementation happens to give the correct answer in this case, but a slight variation leads to confusion. Suppose we wanted to predict how *confident* Alice is that she will successfully meet Bob. It is tempting to query this by writing a new model `alice′` that infers the probability of `a==b` rather than the probability of `a` itself. But this erroneously returns 100%, "optimistically" manipulating Bob to always choose the same bar as Alice. memo, on the other hand, gives us the correct answer by forcing us to precisely articulate what we wish to compute: `E[alice[Pr[a==bob.b]]]`, an observer's expectation of the probability Alice assigns to meeting Bob.

Why do we need to take the observer's expectation here? This is necessary because the observer has uncertainty over which bar Alice will choose (which may affect her confidence). It is easy to overlook this subtlety, but if we had forgotten the outer `E[...]` memo would have statically raised an error. This is one way memo helps programmers discover hidden assumptions in their models.

Notice also that *Alice's* confidence is different from an observer's *own* confidence that Alice will succeed in meeting Bob. To model the latter in memo, we can have an observer model Alice and Bob choosing $a$ and $b$ respectively, and then compute `Pr[alice.a == bob.b]`. Interestingly, we find that the observer has slightly higher confidence than Alice herself. This is because while Alice at depth $d$ models Bob at depth $d - 1$, the observer at depth $d$ models both Alice *and* Bob at depth $d$.

*4.1.3 MDP planning by value iteration.* A Markov Decision Process (MDP) is a foundational formalism used to model sequential decision-making [19]. Abstractly, an MDP describes a space of *states* ($S$) an agent can inhabit, a space of *actions* ($A$) the agent can take, a stochastic transition function $T(s' \mid s, a)$ describing the dynamics of taking an action $a$ at state $s$, and the single-timestep reward $r_t = R(s_t, a_t) \in \mathbb{R}$ for taking action $a_t$ from state $s_t$. Agents strive to maximize their total utility $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$, where $\gamma \in [0, 1)$ is a "discount factor" that expresses how much more valuable reward is in the present than in the future. An MDP is thus defined by the tuple $\langle S, A, T, R, \gamma \rangle$. We can implement an MDP in memo by treating the agent as reasoning recursively about *itself in the future*. The compiled array program is effectively an implementation of policy iteration.

We compared our implementation to one in WebPPL using a maze navigation MDP, a common setting for testing RR models. We placed an agent in a $9 \times 9$ maze where $S$ was the 81 cells, $A$ was the four cardinal directions, $T$ represented valid moves, and $R$ rewarded reaching a goal square quickly. To plan routes in this maze, we ran 1000 rounds of policy iteration. As shown in Table 1, the WebPPL and memo implementations are comparable in length (75 vs. 60 lines of code), but the memo version is 40× faster (10.3sec vs 0.27sec). When we expanded to a $21 \times 21$ grid, WebPPL took 79 seconds (this required us to override an internal timeout). memo gave the same result in 6.9s.

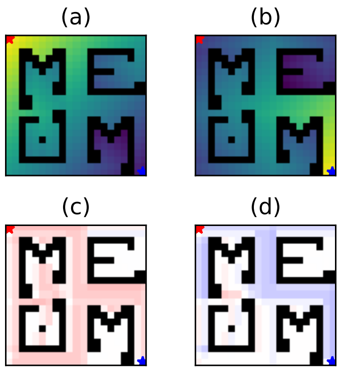


Fig. 8. Planning in a grid-maze MDP. Top: value function if goal is (a) NW and (b) SE. Bottom: inferred goal (**NW**/**SE**) if agent moves (c) west and (d) south. (§4.1.3)

This is the slowest model in this paper, and thus a good opportunity to show the value of GPU acceleration in memo. On an NVIDIA GeForce RTX 4070, the same computation takes only 0.377s (no new code needed). Figures 8(ab) show the computed value function in a $21 \times 21$ memo-themed maze with two possible goals: the northwest (NW) and southeast (SE) corners. As expected, the value function corresponds to the length of the shortest path to the goal.

With this, we can replicate one of the most influential models from computational social cognition: "inverse planning" [13], which models people's ability to understand each other's actions. From a Bayesian perspective, assuming an agent behaves rationally [61, 83, 84], we can infer its (unknown) goal based on its (observed) actions. Figures 8(cd) show such inferences made in memo (requiring only an additional 10 LOC and 9.8ms of inference time). For example, if the agent is located under the "O," then its moving one step west gives a strong impression that it is heading to the NW corner (panel c). However, if the agent is under the lower "M," then its moving west leaves us uncertain about its intentions, because it would move west to go to *either* goal from there (panel d).

*4.1.4 POMDP planning in belief space.* A Partially-Observable Markov Decision Process (POMDP) extends the MDP formalism to describe agents who are unsure of their current state [7, 85]. Upon taking action $a$ to reach state $s'$, agents receive an *observation* $o \in \Omega$ with probability $O(o \mid a, s')$. A POMDP is thus defined by a tuple $\langle S, A, T, R, \Omega, O, \gamma \rangle$. Agents in POMDPs often show "explore-exploit" behavior, first taking epistemic actions that give informative observations, and later using that accrued knowledge to efficiently seek reward. A common approach to planning in POMDPs is *belief-space planning*: we can transform any POMDP into an extended regular MDP where the agent's current *belief* about the true state is itself a type of "belief state," and belief-state transitions are given by the rules of Bayesian belief updating [24].

Here, we considered a classic "hello, world" POMDP, the "crying baby problem" [90]. A baby can be in either of the two states $S = \{\text{sated}, \text{hungry}\}$, and at each timestep the parent can take any of the three actions $A = \{\text{feed}, \text{sing}, \text{ignore}\}$. While it is impossible to know the baby's true state, the parent observes at each timestep whether the baby is $o \in \Omega = \{\text{crying}, \text{quiet}\}$. We use the specifications of $T$ and $O$ provided by Kochenderfer et al. [90]. Abstractly, the optimal solution to this POMDP is for the parent to feed the baby if they are confident the baby is hungry, and otherwise to ignore the baby (but never to sing!).

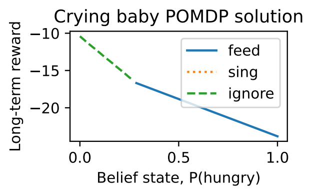


Fig. 9. Replicating Kochenderfer et al. [90, figure F.10] with memo.

We took an existing implementation of the crying baby POMDP [46] and solved it with an off-the-shelf POMDP solver hand-written and optimized by an expert directly in Julia (i.e. not via automated inference in a PPL) [6]. We compared this to a memo implementation. In both cases, we discretized the parent's belief that the baby was hungry by uniformly partitioning the interval [0, 1] into 50 possible belief states. As shown in Table 1, the Julia version required 43 lines to implement the model and 199 lines to implement a POMDP solver. In contrast, memo required 30 lines to implement the model and 61 lines to implement a POMDP solver. (A general version of this solver is built-in as a utility in memo.) The two versions had comparable performance (25ms vs 26ms). The belief-space value function computed by memo is shown in Figure 9, replicating Kochenderfer et al. [90, figure F.10].

## 4.2 memo in the wild

Next, we briefly describe four real-world research projects that are currently using memo. We are grateful to our colleagues for permission to describe their ongoing, early-stage work in this paper.

*4.2.1 Lie production.* Yi et al. [138] are developing a theory of lying. They designed a simple experiment where two parties ("liar" and "judge") each have incomplete information about the world. The "liar" answers a question from the "judge" and is incentivized to lie—but must be careful not to get caught. This is a complicated model because there are many different mental states and beliefs about mental states in play. For example, the liar must reason about what the judge might know, as well as what the judge thinks the liar knows, and what the judge can infer about the liar based on what the liar says. Yi et al.'s original model was implemented in the Gen PPL [40]. We worked with them to re-implement their model in memo. As shown in Table 1, the translation from Gen to memo shortened the model implementation from 220 to 50 lines of code, and sped up inference by a factor of about 2,000,000×.[1] Perhaps most importantly, memo made it easier for these researchers to extend their model. They had long intended to build a model of how people *detect* lies ("call B.S.") by inferring over their model of how people *produce* lies. They had previously budgeted weeks of effort to implementing such a model, but using memo they were able to implement it within an hour and confirm that it predicted the phenomena they wished to capture.

*4.2.2 Risky saliva sharing.* Hung et al. [81] are interested in a computational model of how we make intuitive judgements of each others' social relationships based on their actions. For example, if we see two people sharing a drink, we might have different intuitions about the intimacy of their relationship depending on whether they share a straw, use two straws, or split the drink into two separate cups. Surprisingly, even very young children make such inferences from "risky saliva sharing" [49, 126]. Hung et al. sought to computationally model such judgements. In their model, two agents choose saliva-sharing actions based on intimacy and risk. Then, a third party infers the agents' intimacy based on observed actions. Hung et al. had developed the original version of their model in the R programming language [82]. We worked with them to rewrite the model in memo. As shown in Table 1, the translation from R to memo shortened the implementation from 125 to 38 lines of code, and sped up inference by a factor of about 3000×. This in turn sped up

[1]This dramatic speedup is because the researchers originally implemented all levels of nested inference using Monte Carlo sampling, which Gen is specialized for. Unfortunately, with multiple levels of recursion this led to a combinatorial explosion in the number of samples needed (see §1.1.2). In this regime, memo's exact enumerative inference is much faster. Gen did not at the time support exact enumerative inference (it was only added in late October 2024, as a debugging feature).

statistical analyses (model fitting and cross-validation) from 6 hours to 6 seconds, empowering the researchers to rapidly iterate on their work in a way that was previously impossible.

*4.2.3 Empathatic explanation.* Chandra et al. [30] are interested in the role of emotion in doctor-patient interactions, e.g. how a doctor might explain the cause of a patient's terminal disease in a way that minimizes the regret the patient feels about their life choices. The researchers hypothesized that people expect doctors to take emotion into account, and collected data to test this hypothesis. They then built a computational model of "empathetic explanation" by combining a recent model of explanation [29] with recent models of emotion prediction [80] and intervention [35]. As shown in Table 1, their model is 120 lines of memo and takes 92μs to run. They fit parameters to human data using an off-the-shelf Adam optimizer [42]. They reported that memo helped them catch subtle issues in counterfactual reasoning when modeling inferences about causality and regret.

*4.2.4 Caregiving.* Kleiman-Weiner [88] is interested in studying how parents care for their children. They considered a simple model where a parent could either let a child work through a problem themselves, or "take over" and solve the problem on the child's behalf—at risk of sending their child the signal that they do not believe in them. In the original Python-based version of this model, they used a very restrictive analytically-solvable model of the child out of scalability concerns. memo allowed them to dispense with this restriction, and use a much more flexible model of the child. The model uses memo's built-in POMDP library (mentioned in §4.1.4).

## 4.3 Extensions

We conclude with two demos to showcase new kinds of models that memo has made accessible.

*4.3.1 Font design.* Seven-segment displays are simple electronic components that are used to display letters and numbers on appliances like calculators and microwaves. These displays have seven segments that can be independently turned on or off, for a total of $2^7 = 128$ possible configurations. When using such displays, designers must take care to use an unambiguous "font." For example, if the designer maps "g" to 9, then the result could be confused for an "9." In this case, it could be better to represent capital "G" via 6 (though then there may be a conflict with "6"!).

We can use RSA to design "fonts" for such displays, setting $U$ to be the $2^7$ display configurations and $R$ to be a set of characters to display (e.g. the 36 case-insensitive alphanumerics used in English). The key idea is to apply RSA from the speaker's perspective, where the naïve ($\ell = 0$) "listener" is modeled using a computer vision model of optical character recognition. Because memo is built on JAX, we can draw on its existing deep learning ecosystem to integrate RSA with deep learning. Here, we used the Flax [73] library to train a ResNet [72] that classifies alphanumeric characters in the EMNIST dataset [37]. A naïve strategy for representing a character $r$ with a display setting $u$ is to pick the display setting $u$ that maximizes the network's activation for $r$ when given a rendered image of display setting $u$ as input. However, this strategy leads to an ambiguous "font," as shown in Figure 10. If we instead use the neural network to model the base "listener" in RSA, the speaker distinguishes the confusing pairs.

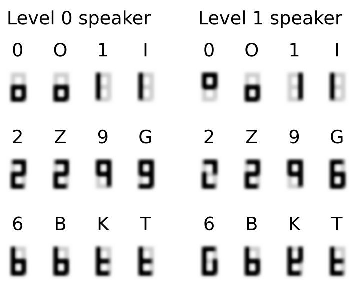


Fig. 10. Generating "fonts" for 7seg displays naïvely (left), and with RSA (right). See §4.3.1.

This model runs efficiently (124ms) because memo internally parallelizes rendering and batches calls to the neural network. This type of deep integration is difficult to achieve with traditional PPLs, where external functions would have to be called in serial (and potentially re-implemented from scratch). While this example is relatively trivial, RR models that reason about other agents' perception and cognition—using computer vision, graphics engines, physics simulators, and language models in the loop—are increasingly of interest to the cognitive science community [20, 28, 32, 39, 98, 144], and there is a growing need for PPLs that allow efficient implementations.

*4.3.2 The game of "Takeaway".* People sometimes make inferences about each other from *how much* they think. For example, if you ask your friend a tricky puzzle and they immediately give the correct answer, you might guess that they knew the solution beforehand. Recent work in cognitive science has sought to model how people take the temporal dynamics of thinking into account [21, 22, 129], and, as we discussed in §2.5.2, memo is designed to facilitate such models.

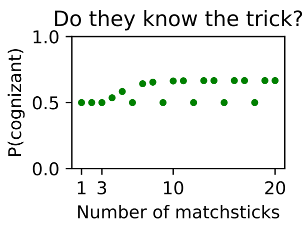


Fig. 11. As $n$ increases, the observer thinks it is likelier that the player knows the "trick" (§4.3.2).

Here, we will show a toy model that makes such inferences. Consider the game of "1-2 Takeaway," a simplified variant of Nim [63]. There are $n$ matchsticks on a table, and players take turns removing either 1 or 2 matchsticks. The player to remove the last matchstick wins. We can easily model this game in memo and recover the optimal strategy for each $n$. It is easy to notice (and to prove by induction) that the optimal strategy is to *leave behind* a multiple of 3 matchsticks. (If $n$ is already a multiple of 3 there is no winning move.) Players who know this "trick" can easily compute optimal moves even when $n$ is large, whereas players who do not know the trick must traverse an exponentially-growing game tree.

Imagine you see someone sit down for a quick round of Takeaway. They count the $n$ matchsticks and then immediately make an optimal move. How likely is it that they know the "trick"? Our intuition is that for small $n$ (e.g. 1 or 2), it is hard to tell because even without knowing the trick, it is easy to compute the correct answer. Similarly, if $n$ is a multiple of 3 (losing position), then it is hard to tell because even with the trick the player would effectively have to choose arbitrarily. However, if $n \gg 1$ and is *not* a multiple of 3, then we might begin to suspect that they *do* know the trick: it is hard to believe that the player performed the full game-theoretic computations correctly in such a short time. (Observing additional correct moves would confirm this suspicion.)

We will model this inference by supposing that the player is either *cognizant* or *oblivious* of the trick. If the player is *cognizant*, they always move optimally. If they are *oblivious*, they pick the correct answer as a softmax over the true utility, where the temperature depends on the amount of computation required (they are noisier if more computation is required). An observer infers whether the player is cognizant or oblivious based on the move made. The critical line is:

```
observer: thinks[
  player: chooses(m in Move, wpp=
    π[n, m](t) if cognizant else exp(π[n, m](t) * α / (cost @ π(t))) ]
```

Here, "**cost @** π(t)" gives the number of FLOPs needed to compute the player's optimal policy $\pi$ for time horizon $t$, and $\alpha$ is a scaling factor to translate FLOPs to units of "mental effort." As shown in Figure 11, this indeed makes the predictions described above: unsure for low $n$ and $n \equiv 0 \pmod 3$, but suspecting the player is cognizant as $n \to \infty$. Such models were previously extremely challenging to implement, but in memo this model is 32 lines of code and runs in 116ms.

*Summary.* memo is expressive enough to encode a wide variety of classic RR models, cutting-edge real-world models, and exotic new models that would be very difficult to implement in existing PPLs. memo code is typically shorter than code in traditional PPLs, and often easier to reason about. Across all of the classic models we considered, memo was faster than expert-written implementations in traditional PPLs (and was comparable to a bespoke hand-written solver in

§4.1.4). The speedup was much more pronounced with cutting-edge real-world models, where memo was orders of magnitude faster than what researchers had previously implemented.

Though we did not discuss their work here, other researchers are using memo to study phenomena like mutual gaze, multimodal inference, and punishment. One university (not affiliated with the authors) is offering a spring semester course on social cognition taught in memo. Finally, **the supplementary materials include many additional examples**: common patterns like empowerment [89] and expected information gain [114], classic economic games like Ultimatum [100], Bayesian Persuasion [86], and "Guess 2/3 of the Mean" [70], and puzzles like Monty Hall [119], Cheryl's Birthday [131], Dining Cryptographers [34], and Newcomb's Paradox [135].

## 5 LIMITATIONS AND FUTURE WORK

Most of memo's limitations derive from its array-oriented backend. Because joint distributions are represented as arrays, memo does not support continuous random variables (unless discretized), and only supports control flow where the sequence of choices (i.e. array axes) is known statically. These limitations are rarely issues for typical real-world RR models. Nonetheless, we are interested in exploring alternate (or hybrid) inference backends that do support these features.

Besides improving memo, we are also interested in *using* it to approach longstanding scientific questions about language and theory of mind. For example, evidence from deaf children's use of sign language suggests that learning language for theory of mind may be critical for being able to reason about false beliefs [41, 99, 107, 108, 117]. Could analyzing the effect of memo's domain abstractions help us understand the power specialized language buys for theory of mind? Similarly, it is hotly debated whether large language models (LLMs) have human-like theory of mind [127]. Could using memo as a "language of thought" [51] *of* thought improve LLMs' theory of mind, like how access to code, theorem provers, and PPLs improves other types of reasoning [105, 136, 142]?

## 6 ADDITIONAL RELATED WORK

We discussed key related work in Section 1. In addition, memo relates to two lines of work.

First, there has been a recent revival of interest in scaling **exact discrete inference**, in languages like Dice [78], SPPL [115], and Genfer [141]. HyBit even "bit-blasts" continuous variables to discrete ones [57]. FSPN [124] and PERPL [36] allow for exact inference in the presence of recursion. These languages seek to scale to *arbitrary* probabilistic programs by cleverly factoring problem structure. A key insight in memo is that for the narrow domain of real-world RR models, one can instead scale by running a relatively straightforward algorithm efficiently on modern parallel hardware (§1.2.2).

Second, memo relates to work on **epistemic logics** [14, 48, 71, 75, 132] and Belief-Desire-Intention logics [59, 60, 137], which have long been applied in domains like economics [9–11] and computer security [18, 26] to reason about agents' knowledge and beliefs. memo takes inspiration from these logics, but builds on the Bayesian tradition of modeling degrees of belief via probabilities. While epistemic logics are primarily used to prove general statements about the nature of knowledge, memo is meant for computing numerical predictions about human behavior in specific scenarios.

Finally, for the sake of transparency, we note that a two-page extended abstract of this paper is concurrently under review for presentation at a non-archival workshop [4, anon. ref.].

## 7 FINAL THOUGHTS

Throughout this paper, we drew liberally from classic literature in cognitive science, linguistics, behavioral economics and AI for examples and case studies. We did this in part to show off memo in a realistic context, but also in part to offer the PL community a glimpse into the many interesting problems raised by theory of mind. We believe this is an exciting, impactful space, and we hope this paper broadly inspires new lines of work on PL for the RR paradigm.

## REFERENCES

[1] Sam Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle Ecanow, Maxwell Nye, Michael Tessler, and Josh Tenenbaum. 2022. Communicating natural programs to humans and machines. *Advances in Neural Information Processing Systems* 35 (2022), 3731–3743.

[2] Nitay Alon, Lion Schulz, Joseph M Barnby, Jeffrey S Rosenschein, and Peter Dayan. 2024. Detecting and Deterring Manipulation in a Cognitive Hierarchy. *arXiv preprint arXiv:2405.01870* (2024).

[3] Nitay Alon, Lion Schulz, Vaughan Bell, Michael Moutoussis, Peter Dayan, and Joseph M Barnby. 2024. (Mal) adaptive Mentalizing in the Cognitive Hierarchy, and Its Link to Paranoia. *Computational Psychiatry* 8, 1 (2024), 159.

[4] Anonymized. 2025. A Domain-Specific Probabilistic Programming Language for Reasoning about Reasoning. *Submission to LAFI workshop* (2025).

[5] Anonymized. 2025. Semantics of the memo probabilistic programming language. *Submission to LAFI workshop* (2025).

[6] Dylan Asmar. 2024. . https://github.com/JuliaPOMDP/BeliefGridValueIteration.jl

[7] Karl Johan Åström. 1965. Optimal control of Markov processes with incomplete state information I. *Journal of mathematical analysis and applications* 10 (1965), 174–205.

[8] Robert J Aumann. 1987. Correlated equilibrium as an expression of Bayesian rationality. *Econometrica: Journal of the Econometric Society* (1987), 1–18.

[9] Robert J Aumann. 1999. Interactive epistemology I: knowledge. *International Journal of Game Theory* 28 (1999), 263–300.

[10] Robert J Aumann. 1999. Interactive epistemology II: probability. *International Journal of Game Theory* 28 (1999), 301–314.

[11] Robert J Aumann. 2016. *Agreeing to disagree.* Springer.

[12] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. 2011. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, Vol. 33.

[13] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. 2009. Action understanding as inverse planning. *Cognition* 113, 3 (2009), 329–349.

[14] Alexandru Baltag and Lawrence S Moss. 2004. Logics for epistemic programs. *Synthese* 139 (2004), 165–224.

[15] JM Barnby, G Bellucci, N Alon, L Schilbach, CD Frith, and V Bell. 2024. Beyond Theory of Mind: A formal interoperable framework for social inference and representation. (2024).

[16] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. 1995. Learning to act using real-time dynamic programming. *Artificial intelligence* 72, 1-2 (1995), 81–138.

[17] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. 2013. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences* 110, 45 (2013), 18327–18332.

[18] Francesco Belardinelli, Ioana Boureanu, Vadim Malvone, and Fortunat Rajaona. 2024. An SMT-based Approach to the Verification of Knowledge-Based Programs. *Formal Aspects of Computing* (2024).

[19] Richard Bellman. 1957. A Markovian decision process. *Journal of mathematics and mechanics* (1957), 679–684.

[20] Marlene Berke, Zhangir Azerbayev, Mario Belledonne, Zenna Tavares, and Julian Jara-Ettinger. 2024. MetaCOG: A Heirarchical Probabilistic Model for Learning Meta-Cognitive Visual Representations. In *The 40th Conference on Uncertainty in Artificial Intelligence.*

[21] Marlene Berke, Ben Sterling, Abi Tenenbaum, and Julian Jara-Ettinger. 2024. No signatures of first-person simulation in Theory of Mind judgments about thinking. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 46.

[22] Marlene Berke, Abigail Tenenbaum, Benjamin Sterling, and Julian Jara-Ettinger. 2023. Thinking about thinking as rational computation. In *Proceedings of the Annual Conference of the Cognitive Science Society.* escholarship. org.

[23] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. 2019. Pyro: Deep universal probabilistic programming. *Journal of machine learning research* 20, 28 (2019), 1–6.

[24] Blai Bonet and Hector Geffner. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems.* 52–61.

[25] Matthew Botvinick and Marc Toussaint. 2012. Planning as inference. *Trends in cognitive sciences* 16, 10 (2012), 485–488.

[26] Michael Burrows, Martin Abadi, and Roger Needham. 1990. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)* 8, 1 (1990), 18–36.

[27] Colin F Camerer, Teck-Hua Ho, and Juin-Kuan Chong. 2004. A cognitive hierarchy model of games. *The Quarterly Journal of Economics* 119, 3 (2004), 861–898.

[28] Matthew Caren, Kartik Chandra, Josh Tenenbaum, Jonathan Ragan-Kelley, and Karima Ma. 2024. Sketching With Your Voice: "Non-Phonorealistic" Rendering of Sounds via Vocal Imitation. In *SIGGRAPH Asia.* https://doi.org/10.1145/3680528.3687679

[29] Kartik Chandra, Tony Chen, Tzu-Mao Li, Jonathan Ragan-Kelley, and Josh Tenenbaum. 2024. Cooperative Explanation as Rational Communication. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 46.

[30] Kartik Chandra, Katie Collines, Jonathan Ragan-Kelley, and Josh Tenenbaum. 2024. Empathy in Explanation. *In preparation* (2024).

[31] Kartik Chandra, Tzu-Mao Li, Rachit Nigam, Joshua Tenenbaum, and Jonathan Ragan-Kelley. 2024. Watchat: Explaining perplexing programs by debugging mental models. *arXiv preprint arXiv:2403.05334* (2024).

[32] Kartik Chandra, Tzu-Mao Li, Joshua Tenenbaum, and Jonathan Ragan-Kelley. 2022. Designing perceptual puzzles by differentiating probabilistic programs. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

[33] Kartik Chandra, Tzu-Mao Li, Joshua Tenenbaum, and Jonathan Ragan-Kelley. 2023. Acting as inverse inverse planning. In *Acm siggraph 2023 conference proceedings*. 1–12.

[34] David Chaum. 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology* 1 (1988), 65–75.

[35] Tony Chen, Sean Dae Houlihan, Kartik Chandra, Josh Tenenbaum, and Rebecca Saxe. 2024. Intervening on Emotions by Planning Over a Theory of Mind. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 46.

[36] David Chiang, Colin McDonald, and Chung-chieh Shan. 2023. Exact recursive probabilistic programming. *Proceedings of the ACM on Programming Languages* 7, OOPSLA1 (2023), 665–695.

[37] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2921–2926.

[38] Katherine M Collins, Ilia Sucholutsky, Umang Bhatt, Kartik Chandra, Lionel Wong, Mina Lee, Cedegao E Zhang, Tan Zhi-Xuan, Mark Ho, Vikash Mansinghka, et al. 2024. Building machines that learn and think with people. *Nature Human Behaviour* 8, 10 (2024), 1851–1863.

[39] Sholei Croom, Hanbei Zhou, and Chaz Firestone. 2023. Seeing and understanding epistemic actions. *Proceedings of the National Academy of Sciences* 120, 49 (2023), e2303162120.

[40] Marco F Cusumano-Towner, Feras A Saad, Alexander K Lew, and Vikash K Mansinghka. 2019. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming language design and implementation*. 221–236.

[41] Jill G De Villiers and Jennie E Pyers. 2002. Complements to cognition: A longitudinal study of the relationship between complex syntax and false-belief-understanding. *Cognitive development* 17, 1 (2002), 1037–1060.

[42] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. 2020. *The DeepMind JAX Ecosystem*. http://github.com/google-deepmind

[43] Judith Degen. 2023. The rational speech act framework. *Annual Review of Linguistics* 9, 1 (2023), 519–540.

[44] Daniel C Dennett. 1989. *The intentional stance*. MIT press.

[45] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. 2013. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 301–308.

[46] Maxim Egorov, Zachary N. Sunberg, Edward Balaban, Tim A. Wheeler, Jayesh K. Gupta, and Mykel J. Kochenderfer. 2017. POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty. *Journal of Machine Learning Research* 18, 26 (2017), 1–5. http://jmlr.org/papers/v18/16-300.html

[47] Owain Evans, Andreas Stuhlmüller, John Salvatier, and Daniel Filan. 2017. Modeling Agents with Probabilistic Programs. http://agentmodels.org. Accessed: 2024-3-8.

[48] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. 2004. *Reasoning about knowledge*. MIT press.

[49] Christine Fawcett. 2022. Kids attend to saliva sharing to infer social relationships. *Science* 375, 6578 (2022), 260–261.

[50] Jaime F Fisac, Monica A Gates, Jessica B Hamrick, Chang Liu, Dylan Hadfield-Menell, Malayandi Palaniappan, Dhruv Malik, S Shankar Sastry, Thomas L Griffiths, and Anca D Dragan. 2020. Pragmatic-pedagogic value alignment. In *Robotics research: the 18th international symposium Isrr*. Springer, 49–57.

[51] JA Fodor. 1975. The language of thought.

[52] Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science* 336, 6084 (2012), 998–998.

[53] Michael Franke and Judith Degen. 2023. The softmax function: Properties, motivation, and interpretation. (2023).

[54] Michael Franke and Gerhard Jäger. 2016. Probabilistic pragmatics, or why Bayes' rule is probably important for pragmatics. *Zeitschrift für sprachwissenschaft* 35, 1 (2016), 3–44.

[55] Michael Franke and Gerhard Jäger. 2016. Reply to commentaries. *Zeitschrift für Sprachwissenschaft* 35, 1 (2016), 117–132.

[56] Gottlob Frege. 1892. On sense and reference.

[57] Poorva Garg, Steven Holtzen, Guy Van den Broeck, and Todd Millstein. 2024. Bit Blasting Probabilistic Programs. *Proceedings of the ACM on Programming Languages* 8, PLDI (2024), 865–888.

[58] Albert Gatt, Roger PG van Gompel, Kees van Deemter, and Emiel Krahmer. 2013. Are we Bayesian referring expression generators. Cognitive Science Society.

[59] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. 1999. The belief-desire-intention model of agency. In *Intelligent Agents V: Agents Theories, Architectures, and Languages: 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings 5*. Springer, 1–10.

[60] Michael P Georgeff and Amy L Lansky. 1987. Reactive reasoning and planning.. In *AAAI*, Vol. 87. 677–682.

[61] György Gergely and Gergely Csibra. 2003. Teleological reasoning in infancy: The naıve theory of rational action. *Trends in cognitive sciences* 7, 7 (2003), 287–292.

[62] Tobias Gerstenberg, Matthew F Peterson, Noah D Goodman, David A Lagnado, and Joshua B Tenenbaum. 2017. Eye-tracking causality. *Psychological science* 28, 12 (2017), 1731–1744.

[63] Solomon W Golomb. 1966. A mathematical investigation of games of "take-away". *Journal of Combinatorial Theory* 1, 4 (1966), 443–458.

[64] Noah D Goodman and Michael C Frank. 2016. Pragmatic language interpretation as probabilistic inference. *Trends in cognitive sciences* 20, 11 (2016), 818–829.

[65] Noah D Goodman and Andreas Stuhlmüller. 2013. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in cognitive science* 5, 1 (2013), 173–184.

[66] Noah D Goodman and Andreas Stuhlmüller. 2014. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org. Accessed: 2024-10-29.

[67] Noah D Goodman, Joshua B. Tenenbaum, and The ProbMods Contributors. 2016. Probabilistic Models of Cognition. http://probmods.org/v2. Accessed: 2024-10-22.

[68] HP Grice. 1975. Logic and conversation. *Syntax and semantics* 3 (1975).

[69] Thomas L Griffiths, Nick Chater, and Joshua B Tenenbaum. 2024. *Bayesian models of cognition: reverse engineering the mind.* MIT Press.

[70] Werner Güth, Rolf Schmittberger, and Bernd Schwarze. 1982. An experimental analysis of ultimatum bargaining. *Journal of economic behavior & organization* 3, 4 (1982), 367–388.

[71] Joseph Y Halpern. 2017. *Reasoning about uncertainty.* MIT press.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[73] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2024. *Flax: A neural network library and ecosystem for JAX.* http://github.com/google/flax

[74] Fritz Heider and Marianne Simmel. 1944. An experimental study of apparent behavior. *The American journal of psychology* 57, 2 (1944), 243–259.

[75] Kaarlo Jaakko Juhani Hintikka. 1962. Knowledge and belief: An introduction to the logic of the two notions. (1962).

[76] Mark K Ho, Fiery Cushman, Michael L Littman, and Joseph L Austerweil. 2021. Communication in action: Planning and interpreting communicative demonstrations. *Journal of Experimental Psychology: General* 150, 11 (2021), 2246.

[77] Mark K Ho, Rebecca Saxe, and Fiery Cushman. 2022. Planning with theory of mind. *Trends in Cognitive Sciences* 26, 11 (2022), 959–971.

[78] Steven Holtzen, Guy Van den Broeck, and Todd Millstein. 2020. Scaling exact inference for discrete probabilistic programs. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (2020), 1–31.

[79] Laurence R Horn and Gregory L Ward. 2004. *The handbook of pragmatics.* Wiley Online Library.

[80] Sean Dae Houlihan, Max Kleiman-Weiner, Luke B Hewitt, Joshua B Tenenbaum, and Rebecca Saxe. 2023. Emotion prediction as computation over a generative theory of mind. *Philosophical Transactions of the Royal Society A* 381, 2251 (2023), 20220047.

[81] Michelle Simona Hung, Ashley J Thomas, Setayesh Radkani, Josh Tenenbaum, and Rebecca Saxe. 2022. Modeling risky food sharing as rational communication about relationships. In *Proceedings of the annual meeting of the cognitive science society*, Vol. 44.

[82] Ross Ihaka and Robert Gentleman. 1996. R: a language for data analysis and graphics. *Journal of computational and graphical statistics* 5, 3 (1996), 299–314.

[83] Julian Jara-Ettinger, Hyowon Gweon, Laura E Schulz, and Joshua B Tenenbaum. 2016. The naïve utility calculus: Computational principles underlying commonsense psychology. *Trends in cognitive sciences* 20, 8 (2016), 589–604.

[84] Julian Jara-Ettinger, Hyowon Gweon, Joshua B Tenenbaum, and Laura E Schulz. 2015. Children's understanding of the costs and rewards underlying rational action. *Cognition* 140 (2015), 14–23.

[85] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101, 1-2 (1998), 99–134.

[86] Emir Kamenica and Matthew Gentzkow. 2011. Bayesian persuasion. *American Economic Review* 101, 6 (2011), 2590–2615.

[87] Justine T Kao, Jean Y Wu, Leon Bergen, and Noah D Goodman. 2014. Nonliteral understanding of number words. *Proceedings of the National Academy of Sciences* 111, 33 (2014), 12002–12007.

[88] Max Kleiman-Weiner. 2024. Computational Principles of Caregiving. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 46.

[89] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. 2005. All else being equal be empowered. In *European Conference on Artificial Life*. Springer, 744–753.

[90] Mykel J Kochenderfer, Tim A Wheeler, and Kyle H Wray. 2022. *Algorithms for decision making*. MIT press.

[91] Joe Kwon, Tan Zhi-Xuan, Joshua Tenenbaum, and Sydney Levine. 2023. When it is not out of line to get out of line: The role of universalization and outcome-based reasoning in rule-breaking judgments. (2023).

[92] Sergey Levine. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909* (2018).

[93] David Lewis. 1969. *Convention: A philosophical study*. John Wiley & Sons.

[94] Falk Lieder and Thomas L Griffiths. 2020. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences* 43 (2020), e1.

[95] R Duncan Luce. 1959. *Individual choice behavior*. Vol. 4. Wiley New York.

[96] Wei Ji Ma, Konrad Paul Kording, and Daniel Goldreich. 2023. *Bayesian models of perception and action: An introduction*. MIT press.

[97] Manasi Malik and Leyla Isik. 2023. Relational visual representations underlie human social interaction recognition. *Nature Communications* 14, 1 (2023), 7317.

[98] Matan Mazor, Rani Moran, and Clare Press. 2024. The role of counterfactual visibility in inference about absence. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 46.

[99] Gary Morgan and Judy Kegl. 2006. Nicaraguan sign language and theory of mind: The issue of critical periods and abilities. *Journal of Child Psychology and Psychiatry* 47, 8 (2006), 811–819.

[100] Rosemarie Nagel. 1995. Unraveling in guessing games: An experimental study. *The American economic review* 85, 5 (1995), 1313–1326.

[101] Andrew Y Ng, Stuart Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *ICML*, Vol. 1. 2.

[102] Robert Nozick. 1969. Newcomb's problem and two principles of choice. In *Essays in honor of carl g. hempel: A tribute on the occasion of his sixty-fifth birthday*. Springer, 114–146.

[103] Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Neeraj Pradhan, Justin Chiu, Alexander Rush, and Noah Goodman. 2019. Tensor variable elimination for plated factor graphs. In *International Conference on Machine Learning*. PMLR, 4871–4880.

[104] Jingwen Pan and Amir Shaikhha. 2023. Compiling Discrete Probabilistic Programs for Vectorized Exact Inference. In *Proceedings of the 32nd ACM SIGPLAN International Conference on Compiler Construction*. 13–24.

[105] Jackson Petty, Sjoerd van Steenkiste, and Tal Linzen. 2024. How Does Code Pretraining Affect Language Model Task Performance? *arXiv preprint arXiv:2409.04556* (2024).

[106] Yewen Pu, Kevin Ellis, Marta Kryven, Josh Tenenbaum, and Armando Solar-Lezama. 2020. Program synthesis with pragmatic communication. *Advances in neural information processing systems* 33 (2020), 13249–13259.

[107] Jennie E Pyers. 2020. Constructing the social mind: Language and false-belief understanding. In *Roots of human sociality*. Routledge, 207–228.

[108] Jennie E Pyers and Ann Senghas. 2009. Language promotes false-belief understanding: Evidence from learners of a new sign language. *Psychological science* 20, 7 (2009), 805–812.

[109] Ciyang Qing and Michael Franke. 2015. Variations on a Bayesian theme: Comparing Bayesian models of referential reasoning. *Bayesian natural language semantics and pragmatics* (2015), 201–220.

[110] WVO Quine. 1960. Word and object. (1960).

[111] Setayesh Radkani and Rebecca Saxe. 2023. What people learn from punishment: joint inference of wrongness and punisher's motivations from observation of punitive choices. In *Proceedings of the annual meeting of the cognitive science society*, Vol. 45.

[112] Setayesh Radkani, Josh Tenenbaum, and Rebecca Saxe. 2022. Modeling punishment as a rational communicative social action. In *Proceedings of the annual meeting of the cognitive science society*, Vol. 44.

[113] Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. 2016. Faster teaching via pomdp planning. *Cognitive science* 40, 6 (2016), 1290–1332.

[114] Anselm Rothe, Brenden M Lake, and Todd M Gureckis. 2018. Do people ask good questions? *Computational Brain & Behavior* 1 (2018), 69–89.

[115] Feras A Saad, Martin C Rinard, and Vikash K Mansinghka. 2021. SPPL: probabilistic programming with fast exact symbolic inference. In *Proceedings of the 42nd acm sigplan international conference on programming language design*

*and implementation*. 804–819.

[116] Thomas C Schelling. 1958. The strategy of conflict. Prospectus for a reorientation of game theory. *Journal of Conflict Resolution* 2, 3 (1958), 203–264.

[117] Brenda Schick, Peter De Villiers, Jill De Villiers, and Robert Hoffmeister. 2007. Language and theory of mind: A study of deaf children. *Child development* 78, 2 (2007), 376–396.

[118] Gregory Scontras, Michael Henry Tessler, and Michael Franke. 2018. Probabilistic language understanding: An introduction to the Rational Speech Act framework. *Retrieved January* 17 (2018), 2021. https://www.problang.org

[119] Steve Selvin. 1975. A problem in probability. *The American Statistician* 29, 1 (1975).

[120] Patrick Shafto, Noah D Goodman, and Thomas L Griffiths. 2014. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive psychology* 71 (2014), 55–89.

[121] Jeffrey Mark Siskind and Barak A Pearlmutter. 2005. Perturbation confusion and referential transparency: Correct functional implementation of forward-mode AD. (2005).

[122] Stephanie Stacy, Siyi Gong, Aishni Parab, Minglu Zhao, Kaiwen Jiang, and Tao Gao. 2024. A Bayesian theory of mind approach to modeling cooperation and communication. *Wiley Interdisciplinary Reviews: Computational Statistics* 16, 1 (2024), e1631.

[123] Dale O Stahl and Paul W Wilson. 1995. On players' models of other players: Theory and experimental evidence. *Games and Economic Behavior* 10, 1 (1995), 218–254.

[124] Andreas Stuhlmüller and Noah D Goodman. 2012. A dynamic programming algorithm for inference in recursive probabilistic programs. *arXiv preprint arXiv:1206.3555* (2012).

[125] Andreas Stuhlmüller and Noah D Goodman. 2014. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. *Cognitive Systems Research* 28 (2014), 80–99.

[126] Ashley J Thomas, Brandon Woo, Daniel Nettle, Elizabeth Spelke, and Rebecca Saxe. 2022. Early concepts of intimacy: Young humans use saliva sharing to infer close relationships. *Science* 375, 6578 (2022), 311–315.

[127] Tomer Ullman. 2023. Large language models fail on trivial alterations to theory-of-mind tasks. *arXiv preprint arXiv:2302.08399* (2023).

[128] Tomer Ullman, Chris Baker, Owen Macindoe, Owain Evans, Noah Goodman, and Joshua Tenenbaum. 2009. Help or hinder: Bayesian models of social goal inference. *Advances in neural information processing systems* 22 (2009).

[129] Tomer D Ullman and Ilona Bass. 2024. The Detection of Automatic Behavior in Other People. (2024).

[130] Priyan Vaithilingam, Yewen Pu, and Elena L Glassman. 2023. The Usability of Pragmatic Communication in Regular Expression Synthesis. *arXiv preprint arXiv:2308.06656* (2023).

[131] Hans van Ditmarsch, Michael Ian Hartley, Barteld Kooi, Jonathan Welton, and Joseph BW Yeo. 2017. Cheryl's Birthday. *arXiv preprint arXiv:1708.02654* (2017).

[132] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. 2007. *Dynamic epistemic logic*. Vol. 337. Springer Science & Business Media.

[133] Edward Vul, Noah Goodman, Thomas L Griffiths, and Joshua B Tenenbaum. 2014. One and done? Optimal decisions from very few samples. *Cognitive science* 38, 4 (2014), 599–637.

[134] Julia White, Jesse Mu, and Noah D Goodman. 2020. Learning to refer informatively by amortizing pragmatic reasoning. *arXiv preprint arXiv:2006.00418* (2020).

[135] David H Wolpert and Gregory Benford. 2013. The lesson of Newcomb's paradox. *Synthese* 190 (2013), 1637–1646.

[136] Lionel Wong, Gabriel Grand, Alexander K Lew, Noah D Goodman, Vikash K Mansinghka, Jacob Andreas, and Joshua B Tenenbaum. 2023. From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672* (2023).

[137] Michael Wooldridge. 2003. *Reasoning about rational agents*. MIT press.

[138] Tan Zhi Yi, Julian Jara-Ettinger, and Marlene Berke. 2024. Reasoning about knowledge in lie production. (2024). https://osf.io/4mu8v/download CogSci.

[139] Erica J Yoon, Michael Henry Tessler, Noah D Goodman, and Michael C Frank. 2020. Polite speech emerges from competing social goals. *Open Mind* 4 (2020), 71–87.

[140] Frances Yung, Jana Jungbluth, and Vera Demberg. 2021. Limits to the rational production of discourse connectives. *Frontiers in Psychology* 12 (2021), 660730.

[141] Fabian Zaiser, Andrzej Murawski, and Chih-Hao Luke Ong. 2024. Exact Bayesian inference on discrete models via probability generating functions: a probabilistic programming approach. *Advances in Neural Information Processing Systems* 36 (2024).

[142] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems* 35 (2022), 15476–15488.

[143] Yizhou Zhang and Nada Amin. 2022. Reasoning about "reasoning about reasoning": semantics and contextual equivalence for probabilistic programs with nested queries and recursion. *Proceedings of the ACM on Programming Languages* 6, POPL (2022), 1–28.

[144] Tan Zhi-Xuan, Lance Ying, Vikash Mansinghka, and Joshua B Tenenbaum. 2024. Pragmatic Instruction Following and Goal Assistance via Cooperative Language-Guided Inverse Planning. *arXiv preprint arXiv:2402.17930* (2024).

[145] Liang Zhou, Kevin A Smith, Joshua B Tenenbaum, and Tobias Gerstenberg. 2023. Mental jenga: A counterfactual simulation model of causal judgments about physical support. *Journal of Experimental Psychology: General* 152, 8 (2023), 2237.