An Introduction to Löb's Theorem in MIRI Research

Patrick LaVictoire

February 23, 2015

Contents

1 Introduction

This expository note is devoted to answering the following question: why do many MIRI research papers cite a 1955 theorem of Martin Löb [12], and indeed, why does MIRI focus so heavily on mathematical logic? The short answer is that this theorem illustrates the basic kind of self-reference involved when an algorithm considers its own output as part of the universe, and it is thus germane to many kinds of research involving self-modifying agents, especially when formal verification is involved or when we want to cleanly prove things in model problems. For a longer answer, well, welcome!

I'll assume you have some background doing mathematical proofs and writing computer programs, but I won't assume any background in mathematical logic beyond knowing the usual logical operators, nor that you've even heard of Löb's Theorem before.

To motivate the mathematical sections that follow, let's consider a toy problem. Say that we've designed `Deep Thought 1.0`, an AI that reasons about its possible actions and only takes actions that it can show to have good consequences on balance. One such action is designing a successor, `Deep Thought 2.0`, which has improved deductive abilities. But if `Deep Thought 1.0` (hereafter called DT1) is to actually build `Deep Thought 2.0` (DT2), DT1 must first conclude that building DT2 will have good consequences on balance.

There's an immediate difficulty—the consequences of building DT2 include the actions that DT2 takes; but since DT2 has increased deductive powers, DT1 can't actually figure out what actions DT2 is going to take. Naively, it seems as if it should be enough for DT1 to know that DT2 has the same goals as DT1, that DT2's deductions are reliable, and that DT2 only takes actions that it deduces to have good consequences on balance.

Unfortunately, the straightforward way of setting up such a model fails catastrophically on the innocent-sounding step "DT1 knows that DT2's deductions are reliable". If we try and model DT1 and DT2 as proving statements in two formal systems (one stronger than the other), then the only way that DT1 can make such a statement about DT2's reliability is if DT1 (and thus both) are in fact unreliable! This counterintuitive roadblock is best explained by reference to Löb's theorem, and so we turn to the background of that theorem.

2 Crash Course in Löb's Theorem

2.1 Gödelian self-reference and quining programs

Löb's Theorem makes use of the machinery of Kurt Gödel's incompleteness theorems [10], so we will discuss those first. Informally, Gödel found a way to import self-reference into a mathematical system that was simply trying to talk about properties of natural numbers, and then pointed out the odd consequences of a mathematical statement that asserts its own

unprovability.

One (anachronistic) way of stating Gödel's key insight is that you can use computer programs to search for proofs, and you can prove statements about computer programs. If we think about any conjecture in mathematics that can be stated in terms of arithmetic, you can write a rather simple program that loops over all possible strings, checks whether any of them is a valid proof of the conjecture, and halts if and only if it finds one.

Thus, instead of trying to prove that conjecture directly, we could instead try to show (or prove) that the program halts. Now, this generally doesn't make things easier at all, since we're just restating the same problem in a more complicated way, and actually looping over all strings is basically the worst way to try and find proofs of theorems. However, this reformulation makes it more intuitive that we can embed self-reference in mathematics, because we can embed self-reference in computer code!

The kind of self-reference I'm talking about is called a quine: a program that is able to reproduce its own source code without taking any inputs. One way to do this is for the program to include a string with a variable, and for it to replace that variable with the original string itself, such that the resulting expanded string is the entire source code of the program itself.

Exercise. Write a quining program in your favorite language. No fair copying one from the Wikipedia article, and no fair making calls to external programs or the filesystem.

In addition to quines that merely print their source code, programs can be made which perform arbitrary tasks using their own source code. Indeed, one prominent example of a quining program that does other tasks is Ken Thompson's famous C compiler Trojan horse [15], which uses the quining trick to avoid ever being visible outside of machine code.

Thus we can have a program G which refers to itself in this way, and searches for proofs in arithmetic related to its own source code. In particular, we consider G which searches for a proof of the statement "G runs forever", and halts if and only if it succeeds at finding one.

Now, we claim that G never finds a proof, and also that we can never *prove* that G runs forever! For if we could prove that G ran forever, then G would find that proof, and it would halt—and we could prove that it halted, and thereby produce a contradiction! On the other hand, if G actually halted after some finite number of steps, then the number at which it halts encodes a proof that G runs forever, which again leads to a contradiction![1]

[1] I apologize for all of the exclamation points in the previous paragraph; in my defense, if any result deserves exclamation points, it's Gödel's First Incompleteness Theorem!

Also, there's a sleight of hand here in the demonstration that G never halts. We'll get further into that in Section 4.

It seems silly to ask whether we could prove that G halts, given that G actually runs forever. But it actually wouldn't be a contradiction if we asserted that G actually halted, so long as we didn't say anything about how long it took! It's only a claim like "G halts in fewer than a googolplex steps" that would be an actual contradiction. It turns out that we could add *either* "G never halts" or "G halts" (but not both!) as a new axiom of arithmetic, and not introduce a contradiction by doing so. We express this by saying that G is *undecidable*, and therefore we've proved

Theorem 2.1 (Gödel's First Incompleteness Theorem) *If the theory of arithmetic is consistent, then there exist undecidable statements in the theory of arithmetic.*

Remark. Gödel's Theorem is a bit different from the Halting Problem; the latter shows there's no *one* program X that can tell whether *every* program Y halts or not, but of course there may be *particular* programs X that tell you this fact for *particular* programs Y. But this is saying that there's *no* program X that can tell you definitively whether the program G halts or runs forever.

So Gödel found that we can write undecidable statements about properties of natural numbers, and furthermore showed that adding new axioms won't fix it, since you can repeat the process with the new rule system for whether a statement is a theorem. (There's only one loophole, and it's not a very exciting one: if we use inconsistent axioms, then everything is a theorem, thanks to the Principle of Explosion, and therefore everything is decidable).

Now we're not here to bury mathematical logic, but to use it. So from Gödel's Theorem we move on to...

2.2 Löb's Theorem

Löb's Theorem [12] takes the same framework as Gödel's First Incompleteness Theorem, and constructs programs of the following sort:

- Let X be any logical statement (the sort of thing that could be proved or disproved).
- Now let $\mathsf{ProofSeeker}(X)$ be the program that searches all possible proofs, and halts if and only if one of them is a valid proof of the statement X.
- Finally, let $L(X)$ be the statement "if $\mathsf{ProofSeeker}(X)$ halts, then X".

Let's ponder for a moment whether $L(X)$ should be true or not, using three different kinds of statements X.

- If X is a provable statement, for example "$2+2=4$", then $\mathsf{ProofSeeker}(X)$ halts, and $L(X)$ is "if [true thing], then [true thing]", which is a valid statement.

- If X is disprovable, for example the statement "$2 + 2 = 5$", then $\mathsf{ProofSeeker}(X)$ does not halt, and $L(X)$ is "if [false thing], then [false thing]", which is also a valid statement.

- If X is neither provable nor disprovable, for example Gödel's statement G, then again $\mathsf{ProofSeeker}(X)$ does not halt, and $L(X)$ is "if [false thing], then [maybe true thing]", which is also a true statement (remember your propositional calculus).

So it seems like $L(X)$ is always true. And it would certainly be handy if $L(X)$ were provable for every X: for instance, you could use the second case above to show that mathematics proves no contradictions! Because if we could prove "if $\mathsf{ProofSeeker}(\text{"}2+2=5\text{"})$ halts, then $2+2=5$", then we would have the contrapositive "if $2+2 \neq 5$, then $\mathsf{ProofSeeker}(\text{"}2+2=5\text{"})$ never halts", and since we can prove $2+2 \neq 5$, then we could *prove* that there is no contradictory proof of $2 + 2 = 5$.

Alas, that proof of mathematical consistency is too good to be true. As in the case of Gödel's Theorem, something being true is no guarantee of it being provable. And in fact, we find that $L(X)$ is only provable in the first of the three cases above:

Theorem 2.2 (Löb's Theorem) *For all statements X, if $L(X)$ is provable, then X is provable.*

(There is a neatly presented formal proof of this theorem at The Cartoon Guide to Löb's Theorem [16].)

So in particular, if we could prove that mathematics would never prove a contradiction, then in fact mathematics *would* prove that contradiction! (Note that by the Principle of Explosion, this is indeed a possible state of affairs: if your mathematical axioms lead to a contradiction, then they can prove every statement in the language, including the statement that your mathematical axioms *don't* lead to a contradiction!) Thus the only systems that prove their own consistency are the inconsistent ones; incidentally, this is precisely Gödel's Second Incompleteness Theorem, although he originally proved it without Löb's Theorem.

Remark. If you're interested in going deeper on these topics, Computability and Logic by Boolos, Burgess, and Jeffrey [5] is a good reference.

Remark. There is also a version of Löb's Theorem for bounded proof searches, in the sense of "look through all formulas of length $\leq N$ and see if any of them are a proof of ϕ", and it controls the length of the proof of ϕ in terms of the length of the proof of $\Box\phi \to \phi$. In the limit of arbitrarily large computational resources, the phenomena we care about happen in the same way that they do in the case of infinite computation (i.e. access to halting oracles), and so we will generally discuss the latter case because the proofs are simpler and clearer.

3 Direct Uses of Löb's Theorem in MIRI Research

We can now exhibit three simple cases where Löb's Theorem comes up in MIRI research topics: one where it forms an unexpected obstacle to justifying self-modifications, one where it neatly enables mutual cooperation in a nonstandard Prisoner's Dilemma setup, and one where it frustrates a naive decision algorithm.

3.1 "The Löbstacle"

Let's return to the problem we discussed in the Introduction: `Deep Thought 1.0` wants to verify that switching on its successor, `Deep Thought 2.0`, will have good consequences. Because DT2 has better deductive capacities, DT1 cannot deduce exactly what actions DT2 will take, but it does know that DT2 has the same utility function, and that it too will only take actions it deduces to be good.

Intuitively, this should be enough for DT1 to "trust" DT2, to say that whatever DT2 does, DT2 must have deduced that to be good, and therefore it must actually be good. But that last clause is analogous to the Löbian statement $L(X)$: "if the action is deduced to be good, then it must actually be good"! And therefore DT1 cannot generally prove that clause (unless its reasoning is inconsistent), since then it could prove that every action is good.

This is not just an analogy; when we consider a simple mathematical model of a self-modifying agent that uses proofs in some consistent formal system to justify its actions, that agent has precisely this problem. In models that presume infinite computing power and represent different deductive powers with different axiom systems, a simple agent with a utility function will only create successors whose formal systems are strictly weaker than its own, since only those are fully trusted by the current system.

There are a number of partial and potential remedies to this "Löbstacle", some of them more appealing than others. For more details, see the MIRI preprints Tiling Agents for Self-Modifying AI, and the Löbian Obstacle [18] and Problems of self-reference in self-improving space-time embedded intelligence [9].

3.2 Löbian cooperation

The second topic concerns a variation on the usual Prisoner's Dilemma. Rather than playing directly, you write an algorithm to play against other algorithms on your behalf, as in Robert Axelrod's famous algorithmic Prisoner's Dilemma tournament [1]. However, instead of that setup, in which the algorithms play *iterated* games against one another, in this case your algorithm and theirs get to *read the opponent's source code*, calculate for as long as they like, and then play only once.

Using quining (recall Section 2.1), one can write a program that cooperates if and only

if the opponent's source code is identical to its own. However, there are many ways to write such programs, and different ones will not cooperate with each other; this is therefore a fragile form of mutual cooperation.

There's a different algorithm which (at the cost of using lots of computation) avoids that fragility. We call this agent `FairBot`, and it operates as follows:

- Both `FairBot` and its opponent `X` are functions of one argument, which take the opponent's source code and output either C or D.
- When `FairBot()` is called on the input `X`, it searches through all proofs of length $\leq N$ to see whether any are valid proofs of the statement "`X(FairBot)`$= C$". If yes, then it outputs C; if no, then it outputs D.

(Here N is a parameter that doesn't depend on `X`; we'll think of it as some extremely large number. The only reason we have that parameter at all is so that our algorithm does in fact always return an output in finite time.)

Some things are clear from the definition of `FairBot`. One is that it is capable of cooperation: if `X` is a simple algorithm that always returns C, then `FairBot` will return C as well. Another is that (unless arithmetic is inconsistent) `FairBot` will never be exploited (cooperate while its opponent defects).

What is less immediately clear is the outcome when `FairBot` plays against itself. Intuitively, it seems like both mutual cooperation and mutual defection are stable fixed points of the situation. However, a Löbian statement breaks the deadlock in favor of cooperation!

To see this, ignore for the moment the parameter N. Then if we consider the statement $L($"`FairBot(FairBot)`$= C$"$)$, we find that it follows directly from the code of `FairBot`. For if there is a proof that `FairBot(FairBot)`$= C$, then `FairBot` will discover that proof and output C. But then, by Löb's Theorem, there must be an actual proof of the statement "`FairBot(FairBot)`$= C$"![2]

As mentioned in Section 2.2, there is a quantitatively bounded version of Löb's Theorem such that this works even with the parameter N, for sufficiently large values of N.

Furthermore, this form of cooperation is much more robust: two such programs don't need to figure out that they are functionally identical, they only need to search for proofs about one another's output. Moreover, even algorithms that are not functionally identical to `FairBot` can achieve mutual cooperation with it.

[2]Note the asymmetry: a proof of defection does not directly lead to defection, since `FairBot` acts based only on whether it finds a proof of cooperation, and it nowhere presumes that finding a proof of defection precludes finding another proof of cooperation. Again, the formal system can't correctly know that it is consistent!

The FairBot algorithm is due to Vladimir Slepnev; for more on Löbian cooperation, see the MIRI paper Program Equilibrium in the Prisoner's Dilemma via Löb's Theorem [11]. We will return to this topic in Section 7.1 when we have developed a few more tools.

3.3 Spurious counterfactuals

The third topic concerns an unexpected issue that comes up when the agent is a part of the same universe that the agent is proving theorems about.

The setting is a pure computational universe $U()$ which takes no input, and which just runs and eventually outputs a number. Within that universe there is an agent $A()$ which knows the code for $U()$, which does some computations as part of the natural order of the universe and then outputs something.

This may seem like a rather extreme setup, and it is: we want to see what we can say about agents that know the source code of the universe and wield arbitrarily large computational resources, because it is often easier to show what happens with such agents than what happens with more realistic, messy, and bounded agents. Moreover, any obstacles we discover in the ideal setting are likely to correspond in the real setting to obstacles that can't be overcome merely by adding more knowledge and computing power to the system.

Let's first consider a really simple sort of universe (Algorithm 1), where $A()$ must try to figure out whether to open door number 1, door number 2, or neither.

Algorithm 1: $U()$

```
if A() = 1 then
    return 100
else if A() = 2 then
    return 1
else
    return −∞
end
```

This requires us to specify the function $A()$. Now of course we could consider a function $A()$ hard-coded to return 1, but that's not especially interesting. Instead, we'll write one (Algorithm 2) which tries to deduce which door (if either) to open, by looking through proofs of arithmetical statements.

It seems obvious that there should be proofs of $A() = 1 \to U() = 100$ and $A() = 2 \to U() = 1$ whose Gödel numbers are smaller than a googolplex, and therefore A should correctly prove those statements and choose Door 1.

Algorithm 2: $A()$

```
Door1 ← −∞;
Door2 ← −∞;
n = 1;
while Door1 + Door2 = −∞ and n < 10^(10^100) do
    if n encodes a valid proof of "A() = 1 → U() = c" for some c then
        Door1 ← c;
    if n encodes a valid proof of "A() = 2 → U() = c" for some c then
        Door2 ← c;
    n+ = 1;
end
if Door1 ≥ Door2 then
    return 1
else
    return 2
end
```

But there's something very strange that can happen instead: A could prove $A() = 2 \rightarrow U() = 1$, and then prove a "spurious counterfactual" such as $A() = 1 \rightarrow U() = -1$, which then makes $A()$ stop looking early and select door number 2! This wouldn't be a contradiction, because if $A() = 2$, then $A() = 1 \rightarrow U() = -1$ is formally true[3]. But why should it actually happen?

It's clearest to see in the case where I can make one innocent-seeming change to $A()$: I ask that before it starts searching through all $n < 10^{10^{100}}$ to see if they encode proofs, I want it to check two particular proofs. The first is the simple proof that $A() = 2 \rightarrow U() = 1$, and the second is a statement ϕ. Now if ϕ really is a proof of the spurious counterfactual $A() = 1 \rightarrow U() = -1$, it's clear from the rest of the source code of $A()$ that it will break the while loop and choose Door 2. In fact, we can formalize that reasoning: there's a proof that if ϕ is a proof of $A() = 1 \rightarrow U() = -1$, then in fact it's true that $A() = 2$ and thus $A() = 1 \rightarrow U() = -1$.

But this is almost a Löbian statement, just with the addition of a specific ϕ rather than an existential quantifier! By the machinery of quining, we can find a ϕ that serves our purposes, so it works just as before: since the Löbian statement is provable, so is the conclusion $A() = 1 \rightarrow U() = -1$, by means of that particular ϕ.

[3]Note, however, that it's impossible to prove a spurious counterfactual about the choice you actually take, and thus it could not be consistent to find a spurious proof that $A() = 1 \rightarrow U() = 2$ or that $A() = 2 \rightarrow U() = 11$

Even without that malicious change, these kinds of agents and universes are susceptible to spurious counterfactuals. In Section 7.2, we'll consider a similar setup, but with a more careful ordering of which proofs it pays attention to, and that agent can be shown to make the correct decisions (given enough deductive power). The idea was originally due to Benja Fallenstein; Tsvi Benson-Tilson's paper UDT with Known Search Order [3] discusses spurious counterfactuals and a different response to them.

4 Crash Course in Model Theory

For the later topics, we'll need some additional background in mathematical logic. Let's start out by just talking formally about the natural numbers with addition and multiplication.

4.1 Axioms and theories

We begin with a language of symbols, and we don't start out assuming anything about what they mean. We'll take some symbols of the propositional calculus, set theory, arithmetic, and three special symbols:

$$\{(,), \wedge, \vee, \neg, \rightarrow, \leftrightarrow, \in, \forall, \exists, =, +, \cdot, \mathsf{O}, \mathsf{S}, \mathbf{N}\}$$

Here we'll think of $\mathbf{N}$ as standing in for the set of natural numbers, O as standing in for zero, and S as standing in for the successor (so that SO represents 1, SSO represents 2, etc.). That way we don't need rules for dealing with infinitely many number symbols, or rules for dealing with digits. We'll also add an infinite family of symbols like x and y for variables (so long as they're quantified over with $\forall$ or $\exists$). Finally, we'll use letters like ϕ to stand in for a formula within the language, but note that ϕ is not itself part of the language, so we can't say things like $\forall\phi$.

We'll import the standard rules of the propositional calculus and set theory as applied to the appropriate symbols (for instance, whenever $\phi \wedge \psi$ is a theorem, then both ϕ and ψ are theorems), but we won't yet assume any properties of $+, \cdot, \mathsf{O}, \mathsf{S}$, or $\mathbf{N}$. That's because we'll want to pick our own axioms about arithmetic on the natural numbers! This creates a theory, the set of all theorems that follow from the axioms; different sets of axioms can give rise to different theories.

So let's take some true statements about the natural numbers, make them axioms of our logical system, and see what kind of theory we get. (Parenthetical comments are included for our intuition, but are not actually a part of the theory.)

1. $\mathsf{O} \in \mathbf{N}$ (zero is a natural number)
2. $\forall x \in \mathbf{N}\, \mathsf{S}x \in \mathbf{N}$ (if x is a natural number, so is its successor)

3. $\forall x \in \mathbf{N} \forall y \in \mathbf{N} \, (x + y \in \mathbf{N}) \wedge (x \cdot y \in \mathbf{N})$ (the product and sum of two natural numbers are also natural numbers)

4. $\forall x \in \mathbf{N} \forall y \in \mathbf{N} \, (\mathsf{S}x = \mathsf{S}y) \rightarrow (x = y)$ (if the successors are equal, so are the originals)

5. $\forall y \in \mathbf{N} (y = \mathsf{O}) \vee (\exists x \in \mathbf{N} \, \mathsf{S}x = y)$ (every nonzero number has a predecessor)

6. $\forall x \in \mathbf{N} \, (x + \mathsf{O} = x) \wedge (x \cdot \mathsf{O} = \mathsf{O})$ (how zero interacts with addition and multiplication)

7. $\forall x \in \mathbf{N} \forall y \in \mathbf{N} \, (x + \mathsf{S}y = \mathsf{S}(x + y)) \wedge (x \cdot \mathsf{S}y = (x \cdot y) + x)$ (how successor interacts with addition and multiplication)

A theory can contain contradictions, of course: some theorem ϕ such that $\neg\phi$ is also a theorem. So how can we tell whether the theory built on these axioms contains any contradictions? We could try proving things and see if we ever find a contradiction, but as there are infinitely many theorems, we couldn't be sure that we just hadn't found a contradiction *yet*.

But there's another thing we could do: exhibit an example of a set for which all of the axioms hold! That serves as conclusive evidence that the theory does not contain contradictions, because ϕ is either actually true or actually false for that specific set, and so at most one of ϕ and $\neg\phi$ can actually be a theorem.

We need to do a little more than exhibit an object: we need to have a correspondence between the symbols in the language and the parts of the object. So for instance, we identify the symbol O with the number 0, SO with 1, and so on[4]; we identify $\mathbf{N}$ with the full set of natural numbers $\mathbb{N}$; and we identify addition and multiplication with the usual operations on $\mathbb{N}$. We call such an object a model, and such a correspondence an interpretation of the theory.

However, the same theory can have many models, some of them not at all what you were thinking of when you made the axioms...

4.2 Alternative and nonstandard models

For instance, an alternate model of the theory above is the set with a single element "Alice"; we identify O with Alice, and then SO with Alice as well (so $\mathsf{SO} = \mathsf{O}$), and so on; we declare that Alice+Alice=Alice·Alice=Alice; and then we identify $\mathbf{N}$ with the set {Alice}. Note that all of the axioms above are true of this alternate model, even if their intuitive meanings aren't!

Fine, let's patch this. We can't add an axiom directly saying it's not this particular model,

[4]Of course, this example does take a leap of faith in accepting the existence of infinitely many natural numbers; some mathematicians reject that, call themselves strict finitists, and assert that we have no idea whether the axioms of arithmetic are consistent. But they do accept that any theory with a completely specified finite model must be consistent.

since Alice isn't part of the language of the theory itself. But we can add an axiom saying $\mathsf{SO} \neq \mathsf{O}$, which is true of the model of the natural numbers we want, but not true of the Alice model.

Well, we excluded that alternate model, but there are still others we haven't excluded. In particular, we can consider arithmetic mod n as a model for any n! In order to exclude these, we could add axioms saying $\mathsf{SSO} \neq \mathsf{O}$, $\mathsf{SSSO} \neq \mathsf{O}$, etc, or we could be more economical and just use the axiom $\forall x \in \mathbf{N}\, \mathsf{S}x \neq \mathsf{O}$. (The resulting theory is known as Robinson arithmetic, and is interesting in its own right.)

Have we eliminated all alternate models from this theory? Hardly! We're just getting started. Consider the model where $\mathbf{N}$ refers to the natural numbers combined with one additional entity, "Bob"; we identify S(Bob)=Bob, Bob$\cdot 0 = 0$, Bob$\cdot x$ =Bob for all $x \neq 0$, and Bob$+x$ =Bob for all x. Then we again see that this satisfies all of our axioms so far. We could patch this with the axiom $\forall x \in \mathbf{N}\, \mathsf{S}x \neq x$, but again we can build more and more complicated alternate models. (One particularly interesting model for Robinson arithmetic consists of all polynomials with integer coefficients whose leading term is nonnegative.)

It turns out that what we're missing from the natural numbers is the principle of induction. (Indeed, with Robinson arithmetic you can't even prove that addition is commutative!) In order to express induction in the language (which doesn't have variables for properties, only for numbers), we must resort to an infinite family of new axioms: for every logical formula $\phi(x)$ (with no free variables except for x), we want to add the axiom $(\phi(0) \wedge (\forall x \in \mathbf{N}\, \phi(x) \to \phi(\mathsf{S}x))) \to \forall y \in \mathbf{N}\, \phi(y)$.[5]

With all of that completed, we've got the axioms for the theory of Peano Arithmetic. *Now* are we done with alternative models? Of course not! Remember how Gödel's self-referential formula G was undecidable? There are models of Peano arithmetic where G holds, and other models where G fails to hold.

The models where G holds include our standard intuitive model of the natural numbers, since as we discussed before, G definitely does not halt at any finite number. But then, what kind of model of Peano Arithmetic could G fail to halt on?

It may help to think of the model of $\mathbf{N}$ with Bob. If we do Gödel's construction in Robinson Arithmetic, there will be no contradiction if we assert that the special extra number Bob in fact satisfies the property that the formula G is checking (the property which represents

[5] Actually, we need to get a bit more complicated to do *all* inductive proofs on the natural numbers; what we truly need is that if $\phi(x, z_1, \ldots, z_k)$ is a logical formula, then

$$\forall z_1 \in \mathbf{N} \ldots \forall z_k \in \mathbf{N}\, (\phi(0, z_1, \ldots, z_k) \wedge (\forall x \in \mathbf{N} \phi(x, z_1, \ldots, z_k) \to \phi(\mathsf{S}x, z_1, \ldots, z_k))) \to \forall y \in \mathbf{N} \phi(y, z_1, \ldots, z_k).$$

"this is a valid proof that no number is a valid proof of G"). This is okay, because Bob doesn't actually have a representation in terms of lots of S's and an O, so this assertion can't be used to actually construct a finite sequence of logical formulas that comprise a proof that no number is a valid proof of G. This loophole suffices to maintain consistency for the new system!

We're using Peano Arithmetic, though, so our nonstandard models will be weirder than the natural numbers plus Bob. The key to understanding these is that G never halts at any finite number, but we can't actually define in our formal language what "finite" means. The nonstandard models of Peano Arithmetic are those which have all the usual numbers but also lots of extra numbers that are "too large" to ever be written as lots of S's followed by an O, but which nonetheless are swept along in any inductive statement. (I still don't know quite how to visualize this model, but you can rigorously construct it using set theory.) And again, a "number" that can't actually be written in S's and an O can be asserted to represent a valid proof, but that proof can't be extracted from the number and used against itself, so it all works out consistently.

Remark. Since nonstandard models of arithmetic contain numbers larger than any of the usual natural numbers, they can be used in other areas of mathematics. If you take the reciprocal of one of those numbers, you get an infinitesimal; thus you can use these to define the concepts of calculus without using limits! This is known as nonstandard analysis.

And it doesn't end there! As I mentioned before, we can add either G or $\neg G$ as an extra axiom of Peano Arithmetic, and since adding a new axiom changes the rules for what counts as a valid proof, we can redo Gödel's construction in our new formal system, and have statements that are undecidable given the new rules. Or we can directly talk about the consistency of Peano Arithmetic (by making a statement that asserts that there is no proof in PA that $0 = 1$), and the consistency of the system with all the rules of Peano Arithmetic plus the axiom that PA is consistent, and so on. (Ponder for a moment the formal system which has all the axioms of PA, plus the axiom that PA is consistent, plus the axiom that "the system which has all the axioms of PA, plus the axiom that PA is consistent" is inconsistent. As it turns out, this is a perfectly consistent system[6]!) We'll work with a particular hierarchy of such formal systems in Section 6, but before then, we'll discuss another MIRI paper that relates to theories and models.

Remark. Again, Computability and Logic by Boolos, Burgess, and Jeffrey [5] covers all of this material at a much deeper level.

[6]Assuming that the standard ZFC axioms of set theory are consistent, as we do throughout this paper, since we use these to construct models of all the weird axiomatizations of Peano Arithmetic.

5 Uses of Model Theory in MIRI Research

5.1 Reflection in probabilistic logic

The topic of the first MIRI mathematical workshop paper, Definability of Truth in Probabilistic Logic [7], isn't directly entwined with Löb's Theorem. But we've just covered the necessary background to understand it, and we'll need another ingredient before getting to the other MIRI papers in these notes, so we might as well detour here!

This paper concerns probabilistic logic: assigning probabilities in $[0, 1]$ to logical statements, rather than just assigning them the labels "provable", "disprovable[7]", and "undecidable". We'll want to make sure we assign the value 1 to all provable statements and 0 to all disprovable statements, and that we assign values to the undecidable statements in a consistent way (so if ϕ and ψ are undecidable but $\phi \to \psi$ is provable, then the probability we assign to ϕ should be less than or equal to the probability we assign to ψ).

Why is this an important topic for MIRI to study? Probabilistic logic is an interesting, clean, and rich model for Bayesian reasoning and for bounded inference. Consider an AI that cannot deduce with certainty whether $P = NP$; knowing it one way or the other should certainly change its actions (for instance, its cryptographic precautions against other agents), and so the most sensible way to proceed seems to be assigning it a tentative probability based on the available evidence, using that probability to do consequentialist calculations, and updating it in the light of newer evidence. So, in order to understand bounded reasoning, we might start with the nice and symmetrical (if unrealistic) case of a coherent probability assignment over all logical statements.

So let's consider a Gödelian obstacle that bedevils logic, and see if it looks different within probabilistic logic. That obstacle is Tarski's undefinability of truth [14].

OK, so we've previously established that some statements are undecidable, and in fact, undecidable statements hold in some models of a theory but not others. We might want to endorse some particular model as the "true" one (for instance, our standard model of the natural numbers, without all of those weird nonstandard numbers), and say that logical statements are true if they hold in that model and false if they don't. This truth predicate exists outside the language, and so the logical statements can't talk about the truth predicate, only about weaker notions like provability. All that is perfectly fine, thus far.

The trouble comes when we try to construct a language that contains its own truth predicate T, either by finding a formula for it using the previously existing symbols, or by directly adding a new symbol with some rules of usage, such that for all ϕ, it's true that $T(\phi) \leftrightarrow \phi$. Either approach is doomed, by an argument that should seem familiar by now: there exists

[7] By this, we mean statements whose negation is provable, not statements which cannot be proven.

a sentence X that refers to itself by quining, so that it's provable that $X \leftrightarrow T(\neg X)$. And now, clearly, neither $T(X)$ nor $T(\neg X)$ can hold.

All right, so what changes when you incorporate probabilities?

Let's introduce a symbol P which represents the probability of a logical statement. We'd like P to satisfy some mathematical coherence properties, like $P(\phi \wedge \psi) \leq P(\phi)$, and we'd also like P to be able to discuss itself. Now what happens when we consider the statement Y constructed such that $Y \leftrightarrow (P(Y) < 0.5)$? Does this create a contradiction?

It depends on what kind of statements P is allowed to make about itself! Namely, if P isn't allowed to make exact statements about its own values, but only arbitrarily precise approximations, then everything can work out consistently. Namely, in the place of the failed reflection principle $T\left(T(\phi) \leftrightarrow \phi\right)$, we take the reflection principle

$$\forall \phi \forall a, b \in \mathbb{Q}\ (a < P(\phi) < b) \rightarrow (P(a < P(\phi) < b) = 1)\,. \tag{5.1}$$

In the paper, they prove that with this reflection principle and the natural coherence conditions, there indeed is a probability valuation that works for the values of P.

How does this avoid a collision with the statement Y above? It turns out that the true value of $P(Y)$ is 0.5, but P isn't able to know whether $P(Y)$ is exactly 0.5, or slightly above it, or slightly below it, and so $P(P(Y) < 0.5)$ is 0.5 as well, and so on. For any rational $\epsilon > 0$, you can prove with certainty that $P(0.5 - \epsilon < P(Y) < 0.5 + \epsilon) = 1$, but you can't get from this anything sharp enough to produce a contradiction; in particular, you can't prove the generalization that $\forall \epsilon > 0\, P(0.5 - \epsilon < P(Y) < 0.5 + \epsilon) = 1$.

Remark. Nonstandard arithmetic isn't needed for this result, but it can shed some light on the fact that $P(0.5 - \epsilon < P(Y) < 0.5 + \epsilon) = 1$ for any rational ϵ. Since some models of **N** include nonstandard natural numbers, P can't rule out the possibility that reciprocals of nonstandard natural numbers (infinitesimals) exist, and so $P(Y)$ may as well be imagined as 0.5 plus or minus an infinitesimal...

Remark. Since the result in the Definability of Truth paper is nonconstructive, it's worth noting that there's a followup paper by Christiano on computationally tractable approximations of probabilistic logic: Non-Omniscience, Probabilistic Inference, and Metamathematics [6].

6 Crash Course in Gödel-Löb Modal Logic

The final topics in this survey require some topics outside the usual first course in mathematical logic, namely the Gödel-Löb modal logic of provability. This modal logic captures exactly the parts of Peano Arithmetic that relate to self-reference, and by leaving out the rest, it has a pleasingly simple structure and plenty of powerful tools for analysis. Thus it's a great setting for model problems in decision theory, as we shall see.

6.1 The modal logic of provability

You may have heard of modal logic before, in the context of philosophy: Aristotle defined ways of arguing about which facts are necessary or possible rather than merely true or false. By the twentieth century, people had formalized various kinds of modal logic, where you take the usual language of propositional logic and add the symbols $\Box$ and $\diamond$, as well as new axioms and rules incorporating them. $\Box\phi$ is usually interpreted as the statement "it is necessary that ϕ", and $\diamond\phi$ as "it is possible that ϕ". (We actually only need $\Box$, since in all modal logics of interest to us, $\diamond\phi \leftrightarrow \neg\Box\neg\phi$.)

Philosophers have tried out various sets of axioms, mostly in order to write intimidatingly technical-looking arguments for their preferred metaphysical conclusions[8], but we're more interested in a particular modal logic that constitutes a perfectly rigorous—and quite useful—reflection of Löbian phenomena in Peano Arithmetic and other such systems. This is the Gödel-Löb modal logic (alternately, 'the modal logic of provability' or simply 'provability logic'), which we will denote as GL.

We can construct GL in two different ways. First, we can let $\Box\phi$ represent the formula of Peano Arithmetic that asserts that ϕ is provable in Peano Arithmetic, and then restrict ourselves to the formulas and proofs that use only $\Box$, the logical operators, and Boolean variables (including the constants $\top$ for true and $\bot$ for false, but no numbers, arithmetical operations, or quantifiers). Or, equivalently, we can start with those elements of the language, and then add the following axioms and rules:

- All tautologies of the propositional calculus are axioms of GL, including tautologies where an expression including $\Box$ has been substituted for a variable (for instance, $\Box p \vee \neg\Box p$ is an axiom).
- Modus Ponens Rule: if the expressions A and $A \to B$ are theorems, then so is B.
- Distribution Axioms: for any expressions A and B in the language, we have the axiom $\Box(A \to B) \to (\Box A \to \Box B)$.
- Generalization Rule: if the expression A is a theorem, then so is $\Box A$.
- Gödel-Löb Axioms: for any expression A, we have the axiom $\Box(\Box A \to A) \to \Box A$.

It's a nontrivial theorem that these approaches give us the same modal logic! So GL really does capture the self-referential parts of Peano Arithmetic, while leaving aside the arithmetical parts; for instance, the statement that Peano Arithmetic is consistent is simply $\neg\Box\bot$.

Exercise. Show, using the rules of GL, that $\Box\bot \leftrightarrow \Box \diamond p$; recall that $\diamond p = \neg\Box\neg p$. Informally, that is, no statement can be proven consistent with Peano Arithmetic unless Peano

[8] Gödel himself, toward the end of his life, circulated a modal logic version of Anselm's ontological proof of the existence of God. The proof is of course formally valid, but the axioms are a bit overpowered and underjustified.

Arithmetic is inconsistent. [Hint: First prove that $\Box\bot \rightarrow \Box \diamond p$, then that $\Box \diamond p \rightarrow \Box\bot$. For the latter, it will help to rewrite $\neg\Box\bot$ as $\Box\bot \rightarrow \bot$, then consider the Löbian axiom $\Box(\Box\bot \rightarrow \bot) \rightarrow \Box\bot$.]

While every specific deduction we would want to make in GL can be made via formal manipulations in the system, this is a computationally intractable way to handle deduction. (Deducing whether a given modal statement is a theorem of GL is NP-complete in general, just as it is in Peano Arithmetic.) However, there are some special cases where there *are* efficient algorithms for deducing provability in GL, and these happen to include cases of direct interest to decision theory...

6.2 Fixed points of modal statements

In Section 3.2, we figured out what happens when two different programs try to prove theorems about one another. In that case, the programs were simple enough that we could directly see where to apply Löb's Theorem, but in general we'd like to be able to handle more complicated phenomena. Fortunately, modal logic comes equipped with a powerful tool for doing precisely that: the theory of fixed points for modal sentences.

In Peano Arithmetic, the Gödel statement G refers to itself via quining, in order to claim that it cannot be proved in PA; in GL, G corresponds to the formula $p \leftrightarrow \neg\Box p$. Similarly, we can have all sorts of formulas that refer to themselves and each other by quining, and these are represented by formulas $p \leftrightarrow \phi(p, q_1, \ldots, q_k)$ that are *modalized* in p: every occurrence of p in ϕ happens within the scope of some $\Box$. (After all, quining can only refer to the provability of the statement's own Gödel number, not directly to the statement itself!)

As it happens, whenever you have this setup in GL, with p equivalent to a formula that is modalized in p, then p is equivalent to some other formula which doesn't use p at all! For example, the Gödel statement is equivalent to the inconsistency of arithmetic: $\Box(p \leftrightarrow \Box\neg p) \leftrightarrow \Box(p \leftrightarrow \Box\bot)$.

In the case of one-variable formulas $p \leftrightarrow \phi(p)$ with $\phi(p)$ modalized in p, there's a neat tool that helps us calculate these fixed points (where the result will involve no variables at all, just logical connectives, $\Box$, $\top$, and $\bot$).

First, for many modal logics including GL, there's a corresponding class of sets and relations (called Kripke frames) such that a formula is provable in the modal logic if and only if a corresponding property holds for every Kripke frame in that class. And secondly, in the special case of sentences without variables in GL, we can reduce to checking a linear hierarchy corresponding to a world in which $\Box\bot$ holds, a world in which $\Box\Box\bot \wedge \neg\Box\bot$ holds, and so on up the ladder of $\Box^{n+1}\bot \wedge \neg\Box^n\bot$ for each n[9]. If you'll grant me that result (known as an

[9]Note that these all correspond to nonstandard models of Peano Arithmetic, but that in some sense they approach the standard model as $n \rightarrow \infty$.

adequacy result for GL), I can show you how to construct a "truth table" and a variable-free fixed point for such a formula!

Let's take as our example the formula $p \leftrightarrow (\Box\Box p \rightarrow \Box(p \rightarrow \Box\neg p))$, which has p equivalent to a modalized expression of p. As with truth tables in propositional calculus, we'll break the formula down into constituent parts. But instead of assuming p to be $\top$ or $\bot$, we'll use some rules involving $\Box$ to propagate it to the expression $(\Box\Box p \rightarrow (p \rightarrow \Box\neg p))$ and thereby to p.

World	$\Box p$	$\Box\Box p$	$\Box\neg p$	$\Box(p \rightarrow \Box\neg p)$	$\Box\Box p \rightarrow \Box(p \rightarrow \Box\neg p)$	p	$p \rightarrow \Box\neg p$
$\Box\bot$	?	?	?	?	?	?	?
$\Box\Box\bot \wedge \neg\Box\bot$	?	?	?	?	?	?	?
$\Box^3\bot \wedge \neg\Box^2\bot$	?	?	?	?	?	?	?

We then notice that if $\Box\bot$, then $\Box X$ for any modal expression X, so we can fill in several of the entries in the first row.

World	$\Box p$	$\Box\Box p$	$\Box\neg p$	$\Box(p \rightarrow \Box\neg p)$	$\Box\Box p \rightarrow \Box(p \rightarrow \Box\neg p)$	p	$p \rightarrow \Box\neg p$
$\Box\bot$	1	1	1	1	?	?	?
$\Box\Box\bot \wedge \neg\Box\bot$	?	?	?	?	?	?	?
$\Box^3\bot \wedge \neg\Box^2\bot$	?	?	?	?	?	?	?

(Note: we'll use 1 and 0 instead of $\top$ and $\bot$ so that humans can read these tables.) Now we can use the rules of predicate logic to figure out the big expression, and since we're assuming it's equivalent to p itself[10], we can fill in the last two entries on the first row:

World	$\Box p$	$\Box\Box p$	$\Box\neg p$	$\Box(p \rightarrow \Box\neg p)$	$\Box\Box p \rightarrow \Box(p \rightarrow \Box\neg p)$	p	$p \rightarrow \Box\neg p$
$\Box\bot$	1	1	1	1	1	1	1
$\Box\Box\bot \wedge \neg\Box\bot$	?	?	?	?	?	?	?
$\Box^3\bot \wedge \neg\Box^2\bot$	?	?	?	?	?	?	?

Here's where things start to get interesting. In the next world (defined by $\Box\Box\bot \wedge \neg\Box\bot$), it's not the case that all $\Box A$ are provable. Instead, we'll use the following lemma:

Lemma 6.1 *For all modal expressions A, GL proves $(\Box^{n+1}\bot \wedge \neg\Box^n\bot) \rightarrow \Box A$ if and only if GL proves $(\Box^{k+1}\bot \wedge \neg\Box^k\bot) \rightarrow A$ for all $k < n$.*

Exercise. Prove this lemma by induction on n.

Then, since p holds in the $\Box\bot$ world (which we'll denote as world 0, since there $n = 0$), $\Box p$ holds in world 1 (defined by $\Box\Box\bot \wedge \neg\Box\bot$), while $\neg p$ does not hold in world 0, so $\Box\neg p$ does not hold in world 1. Similarly, since $\Box p$ and $p \rightarrow \Box\neg p$ hold in world 0, $\Box\Box p$ and $\Box(p \rightarrow \Box\neg p)$ hold in world 1, and we can complete the first row (note that now, $p \rightarrow \Box\neg p$ will be false):

[10]This part requires our assumption that the expression is modalized in p, since otherwise we couldn't fill in the value for it without first knowing the value for p.

World	$\Box p$	$\Box\Box p$	$\Box\neg p$	$\Box(p \to \Box\neg p)$	$\Box\Box p \to \Box(p \to \Box\neg p)$	p	$p \to \Box\neg p$
$\Box\bot$	1	1	1	1	1	1	1
$\Box\Box\bot \wedge \neg\Box\bot$	1	1	0	1	1	1	0
$\Box^3\bot \wedge \neg\Box^2\bot$	?	?	?	?	?	?	?

Continuing with the second row, we find that p flips to false:

World	$\Box p$	$\Box\Box p$	$\Box\neg p$	$\Box(p \to \Box\neg p)$	$\Box\Box p \to \Box(p \to \Box\neg p)$	p	$p \to \Box\neg p$
$\Box\bot$	1	1	1	1	1	1	1
$\Box\Box\bot \wedge \neg\Box\bot$	1	1	0	1	1	1	0
$\Box^3\bot \wedge \neg\Box^2\bot$	1	1	0	0	0	0	1
$\Box^4\bot \wedge \neg\Box^3\bot$	?	?	?	?	?	?	?

And with the third row, when considering $\Box\neg p$, it is important to remember that the lemma above requires A to be true in *all* previous rows, so even though $\neg p$ is now true, $\Box\neg p$ remains false. We will fast-forward through the next few rows:

World	$\Box p$	$\Box\Box p$	$\Box\neg p$	$\Box(p \to \Box\neg p)$	$\Box\Box p \to \Box(p \to \Box\neg p)$	p	$p \to \Box\neg p$
$\Box\bot$	1	1	1	1	1	1	1
$\Box\Box\bot \wedge \neg\Box\bot$	1	1	0	1	1	1	0
$\Box^3\bot \wedge \neg\Box^2\bot$	1	1	0	0	0	0	1
$\Box^4\bot \wedge \neg\Box^3\bot$	0	1	0	0	0	0	1
$\Box^5\bot \wedge \neg\Box^4\bot$	0	0	0	0	1	1	0
$\Box^6\bot \wedge \neg\Box^5\bot$	0	0	0	0	1	1	0

And here it stabilizes: every further row is identical[11]. And now that we've found the truth values of p, if we find a constant formula with the same truth values, then by the adequacy result I mentioned in passing, there must be a proof of equivalence in GL!

We can get such a formula by taking Boolean combinations of the formulas that define the worlds (since the truth table stabilizes eventually, one only needs a finite combination of these); in the present case, that formula is $\neg(\Box^3\bot \wedge \neg\Box^2\bot) \wedge \neg(\Box^4\bot \wedge \neg\Box^3\bot)$, or equivalently $\Box^4\bot \to \Box^2\bot$. So in the end, we have the fixed point

$$\Box(p \leftrightarrow (\Box\Box p \to \Box(p \to \Box\neg p))) \leftrightarrow \Box(p \leftrightarrow (\Box^4\bot \to \Box^2\bot)),$$

and our algorithm for finding it is polynomial in the complexity of the statement (in fact, quadratic: we add columns for sub-expressions, and there's a linear bound for how many rows we need to calculate).

Remark. We demonstrated the fixed-point algorithm only for one-variable expressions, but you can also take a statement of the form $p \leftrightarrow \phi(p, q_1, \ldots, q_k)$, where ϕ is modalized in p

[11] Since every expression $\Box A$ either remains true forever or switches over to false and remains that way forever, we can see that every expression must eventually stabilize; and if we've decomposed it at each step involving a $\Box$, once two rows are identical, all subsequent rows must be as well.

(not necessarily in any of the q_i), and find an equivalent expression $p \leftrightarrow \psi(q_1, \ldots, q_k)$. The machinery for this is more difficult, as one needs to deal with more complicated Kripke frames (in the general case of GL, a Kripke frame is a set equipped with a transitive, irreflexive, well-founded relation), but it can be mechanized as well.

I swept quite a lot under the rug in this section; if you want to get to know the topic on a much sounder basis, read The Logic of Provability [4] by Boolos. (This is an advanced textbook, so wait until after your first course in mathematical logic!)

7 Uses of Gödel-Löb Modal Logic in MIRI Research

7.1 Modal Combat in the Prisoner's Dilemma

We're now ready to handle the preprint Robust Cooperation in the Prisoner's Dilemma: Program Equilibrium via Provability Logic [2], which encompasses the results from Section 3.2, and studies "modal agents" represented by expressions in modal logic.

Recall the idea of Löbian cooperation from Section 3.2. One embarrassing thing about `FairBot` is that it doesn't check whether its potential cooperation would actually make any difference. It happily cooperates with `CooperateBot`, the constant strategy that simply returns C for every opponent. That's a bit like cooperating against a rock because the rock "cooperated" with you.

Let's consider the following alternative: first we define `DefectBot` as the constant strategy that simply returns D for every opponent; then we define `PrudentBot` as the algorithm that, given the source code of an opponent `X`, searches in Peano Arithmetic for a proof of the statement "`X` cooperates with `PrudentBot`, and if Peano Arithmetic is consistent[12], then furthermore `X` defects against `DefectBot`", and cooperates if and only if it finds such a proof[13].

Now it's more difficult to figure out what happens when `FairBot` plays the open-source-code Prisoner's Dilemma against `PrudentBot`, if we view these as proof searches in Peano Arithmetic. But if we assume infinite computational power (i.e. the ability to consult a halting oracle about proof searches in Peano Arithmetic), then they can be written out as simple statements in modal logic. Let p = "`FairBot` cooperates with `PrudentBot`", q = "`PrudentBot` cooperates with `FairBot`", r = "`FairBot` cooperates with `DefectBot`", and

[12] Why does `PrudentBot` include the consistency requirement? Because certain actions that depend on agents require the consistency of Peano Arithmetic; for instance, `FairBot` does defect against `DefectBot`, but knowing this requires knowing that Peano Arithmetic is consistent, since otherwise `FairBot` might find a proof that `DefectBot` cooperates, using the Principle of Explosion.

[13] Also, why not directly check whether your cooperation causes the opponent's cooperation, by defining `MagicBot` which cooperates if and only if it finds a proof of the statement "`X` cooperates with `MagicBot` if and only if `MagicBot` cooperates with `X`"? Because this makes it susceptible to spurious counterfactuals as in Section 3.3, and in fact `MagicBot` is equivalent to `FairBot` when they are rendered as modal statements.

$s =$ "`DefectBot` cooperates with `FairBot`". Then we have

$$\begin{aligned} p &\leftrightarrow \Box q \\ q &\leftrightarrow \Box(p \wedge (\neg\Box\bot \rightarrow \neg r)) \\ r &\leftrightarrow \Box s \\ s &\leftrightarrow \bot. \end{aligned}$$

If we look for the fixed point of this modal system, we can use the truth table formalism of the fixed-point theorem (we do all of these simultaneously, since every variable is equivalent to a fully modalized expression) to show that $p \leftrightarrow \top$, $q \leftrightarrow \top$, $r \leftrightarrow \Box\bot$, and of course $s \leftrightarrow \bot$.

Exercise. Show this formally.

So what actually happens when you run the bounded versions of these agents against one another (with sufficiently high bounds on how deep they search for proofs before giving up)? Again, the bounded analogue of Löb's Theorem helps any valid proof in GL to go through if the proof bounds are sufficiently large. And since we really don't expect there to be a contradiction in Peano Arithmetic or any of the systems built atop it by addition of the axioms $\neg\Box^n\bot$[14], in the bounded case we should expect that every search for $\Box^n\bot$ comes up empty. Thus p and q are true, and r and s are false. (This is equivalent, in the infinite computation case, to specifying that we care about what actually holds in the standard model of the natural numbers.)

Therefore we can use these tools to analyze interactions between "modal agents" represented by families of modal statements like `FairBot` and `PrudentBot`, and moreover, we can figure out what actually happens using an efficient algorithm rather than the huge brute-force proof search!

It's a little tricky to define these modal agents in one fell swoop, so let's start with the simplest case: agents that don't check their opponent's action against any third parties, but only against themselves. (We will call these "modal agents of rank 0".) A rank 0 modal agent is represented by a formula $p \leftrightarrow \phi(p, q)$, where ϕ is modalized in both p and q (we don't allow modal agents to run the other agent, only to prove things about them; this avoids the infinite regress of two agents simulating each other forever, waiting for the other to 'make the first move'). By the modal fixed-point theorem, this is equivalent in GL to $p \leftrightarrow \tilde{\phi}(q)$ for some $\tilde{\phi}$ modalized in q, so we will define rank 0 modal agents using those formulas.

Then if we have two modal agents of rank 0, represented by $p \leftrightarrow \phi(q)$ and $q \leftrightarrow \psi(p)$, the simultaneous fixed point of these formulas gives us constant sentences (combinations of

[14] If this step worries you, you may be reassured to know that the standard axioms of set theory are strong enough to assert that this tower upon Peano Arithmetic is consistent all the way up. Of course, that simply means that maybe set theory is contradictory, but if so then the contradiction has been hiding pretty effectively from the mathematical community for the last few decades...

$\Box^n \bot$) from which we can deduce their actions.

Now let's allow calls to third parties, like `PrudentBot`'s call to the opponent's action against `DefectBot`. We'll insist that it bottom out in a finite number of statements: for this, it suffices that every modal agent can only predict its opponent's action against itself and against modal agents of strictly lower rank. That is,

Definition. $X = \langle \phi, Z_1, \ldots, Z_k \rangle$ is a modal agent of rank $n \geq 0$ if ϕ is a fully modalized formula of $k+1$ variables $p, r_1, \ldots, r_k$, and each Z_i is a modal agent of rank $< n$.

Given two modal agents $X = \langle \phi_X, Z_1, \ldots, Z_k \rangle$ and $Y = \langle \psi_Y, W_1, \ldots, W_l \rangle$, we then construct the family of formulas given by recursively applying the modal agents to each other and applying the third parties to the originals:

$$\begin{aligned} p_X &\leftrightarrow \phi_X(q_Y, r_{Z_1}, \ldots, r_{Z_k}) \\ q_y &\leftrightarrow \psi_Y(p_X, s_{W_1}, \ldots, s_{W_l}) \\ r_{Z_i} &\leftrightarrow \phi_{Z_i}(t_{(Y,Z_i)}, u_1^{Z_i}, \ldots, u_{m_i}^{Z_i}) \\ t_{(Y,Z_i)} &\leftrightarrow \psi_Y(r_{Z_i}, v_{(W_1,Z_i)}, \ldots, v_{(W_l,Z_i)}) \end{aligned}$$

and so on until it bottoms out. (Please excuse the proliferation of indices and the imperfect rigor, and take another look at the example above of `PrudentBot` versus `FairBot`, where `FairBot`$=\langle \Box p \rangle$, `PrudentBot`$=\langle \Box(p \wedge (\neg\Box\bot \rightarrow \neg r)), \texttt{DefectBot} \rangle$, and `DefectBot`$=\langle \bot \rangle$.)

Thus any two modal agents can be played against each other, and the result of that "modal combat[15]" computed by an effective algorithm. So there's an actual Haskell program, written by Mihaly Barasz and Marcello Herreshoff, that checks the result of the modal combat, and that program helped the authors to find many other patterns about modal agents (including the discovery of `PrudentBot`, which is never exploited, achieves mutual cooperation with itself and with `FairBot`, and which correctly defects against `CooperateBot`).

Among the other results in that paper:

- Third parties are necessary to get cooperation from `FairBot` while defecting against `CooperateBot`; no modal agent of rank 0 can achieve both.
- No modal agent globally dominates another modal agent, since it is possible to write another modal agent which punishes or rewards agents for their other decisions. (For example, consider `TrollBot`, which cooperates with its opponent if and only if it proves that its opponent cooperates with `DefectBot`.)
- In another obstacle to strong definitions of optimality, consider `WaitFairBot`$_N$ defined by $\langle \neg\Box^{N+1}\bot \wedge \Box(\neg\Box^N\bot \rightarrow p) \rangle$. Then any modal agent that defects against `DefectBot` will fail to elicit mutual cooperation from `WaitFairBot`$_N$ for sufficiently large N.

[15] I came up with that pun. I'm proud of it.

- Finally, there are some modal agents that can be exploited by non-modal agents but not by any modal agent. Consider `MimicFairBot`, which cooperates with `X` if and only if Peano Arithmetic proves that `X` cooperates with `FairBot`. (That is, `MimicFairBot`$=\langle\Box r, \texttt{FairBot}\rangle$.) It is possible for an algorithm to exploit `MimicFairBot` (consider the non-modal agent that cooperates if and only if the opponent's source code is identical to the source code of `FairBot`), but every modal agent treats `MimicFairBot` and `FairBot` identically.

The field of modal combat is new and still wide-open, and I expect to see many more results soon! Furthermore, it inspired an unexpected development in decision theory for one-player games as well...

7.2 Modal Decision Theory

Recall the setup in Section 3.3: we have a universe $U()$ which contains an agent $A()$. The universe runs some computation and returns some outcome; as a part of that computation, the agent runs some computation and returns some action. Everything is deterministic, but we can think of a decision theory as a condition on the allowable form of $A()$; if you have a certain decision theory, then you find yourself only in certain universes and you perform some specific algorithm in those universes.

In order to define what we mean by a decision theory, then, we need to specify where in the universe the agent goes. So we start with some utility values over outcomes, and a universe template $U(\cdot)$ with a spot for an agent; then our decision theory specifies an agent $A()$ that goes there, such that we now have the universe $U_A() = U(A)$ which computes a certain outcome; and we judge the decision theory based on how highly it values that outcome.

Causal decision theory (CDT) and evidential decision theory (EDT) are two famous philosophical examples of decision theory. Both of these require the universe template to supply the agent with some additional information: in the case of CDT, a causal graph of the universe, and in the case of EDT, the actions and outcomes of other agents in the same universe template.

Because CDT and EDT can be shown to make bad decisions on some problems even when given the right info, several people have proposed alternatives, including timeless decision theory [17] (TDT) and updateless decision theory (UDT). (For more on the philosophical side of things, see Toward Idealized Decision Theory [13].) While working out models of UDT, in particular, Vladimir Slepnev and Benja Fallenstein found a nice formalization of a simple decision theory using modal logic, one that they could even prove optimal on a certain class of modally defined problems!

Consider a universe template $U(\cdot)$ with a finite set of possible actions $\mathcal{A}$ and a finite set

of possible outcomes $\mathcal{O}$. We have some preference ordering over $\mathcal{O}$, and we pick some default action a_0 to return if we can't prove anything at all. We then define the decision theory which takes $U(\cdot)$ and returns the agent $A()$ given by Algorithm 3.

Algorithm 3: $A()$ (Slepnev-Fallenstein model of UDT)

```
for x ∈ 𝒪 (in descending order of preference) do
    for a ∈ 𝒜 do
        if Peano Arithmetic proves "A() = a → U() = x" then
            return a
        end
    end
end
return a_0
```

Note that this decision theory is immune to the spurious conterfactuals of Section 3.3: if it ever proves a counterfactual, it immediately makes the antecedent true by returning that action, so the consequent must be true as well. Moreover, if it proves any counterfactual, then it has already tried and failed to achieve every outcome higher in its preference order.

Before we show an optimality result for Algorithm 3 in a special domain, we will have to show what can go wrong with it in other domains. Firstly, the universe template could be 'unfair', in the sense that it cares about aspects of the agent that aren't significantly related to its output. For instance, we could have a universe template that gives the best outcome if and only if $A()$ is a tab-indented Python program, or if and only if $A()$ can be proved to always select the first action of A in alphabetical order (regardless of the details of $U(\cdot)$. We will not worry about how to succeed on such pathological problems in general.

A very strong fairness condition for a universe template is that $U(\cdot)$ should be a function of only the output of A, rather than depending on any other features[16]. We can define this in Peano Arithmetic by choosing an encoding for agents in a particular universe template, then writing a sentence that asserts $(A() = B()) \to (U_A() = U_B())$; the universe template is 'provably extensional' if that statement is a theorem of Peano Arithmetic.

Everything thus far can be represented in modal logic. We will represent $\mathcal{O}$ and $\mathcal{A}$ by *provably mutually exclusive and exhaustive* (p.m.e.e.) families of modal statements. For example, $\{p \wedge q, p \wedge \neg q, \neg p\}$ is a p.m.e.e. family. Then if the universe template and the agent are modal expressions, the universe template is provably extensional if $(a \leftrightarrow b) \to (U(a) \leftrightarrow U(b))$ is a

[16]Note that this condition is too strong to include modal combat, since there it also matters whether a given formal system can prove what the algorithm's output is, not only what the output is. Thus our optimality theorem will not cover modal combat, which is consistent, given that we mentioned in the last section that we lack a good optimality result for modal combat.

theorem of GL. (You can check that the universe from Algorithm 1 corresponds to a provably extensional modal universe with 3 actions and 3 outcomes.)

However, the modal version of Algorithm 3 can fail even on provably extensional modal universe templates; indeed, for any modally defined decision theory there is an "evil problem" which it fails on. (This can be shown by representing the modal decision theory itself in the universe template, and punishing any agent that acts identically to the action of the modal decision theory.) So our definition of optimality must allow some way around this.

As it happens, although for any decision theory one can write a specific universe template that frustrates it, for any fixed universe template there is a modification of Algorithm 3 which succeeds on it, and this modification is simply a replacement of "proves in Peano Arithmetic" with "proves in Peano Arithmetic given $\neg\Box^n\bot$" for sufficiently large n. We can then show that this modification achieves the best outcome that is achievable by any modal decision theory.

This result shows that we're on to something: in this restricted domain, we can do as well as it's possible to do, as long as we're bringing enough deductive capacity to bear! Note that here, the universe acts on the agent only by running it and using the output, while the agent interacts with the source code of the universe only by proving things about it. So in particular, it's less rich than modal combat, in which your agent is proving things about the universe, but another part of that universe (the opponent) is proving things about the agent as well.

So before we count ourselves done with using modal logic in decision theory, we have many open questions about competition and bargaining and other phenomena. It's a beautiful Löbian universe, waiting for us to explore!

In addition to the contributions of Vladimir Slepnev and Benja Fallenstein, the "evil decision problems" were introduced by Nick Bone. If you'd like to get into this topic, I recommend Benja's recent posts on evil decision problems in provability logic, optimality of the model of UDT [8], and its sequel.

8 Acknowledgments

I originally prepared about half of these notes for a talk at a MIRIx workshop in Los Altos, California in November 2014; the feedback from the participants was especially helpful. When writing up the notes, I got some great assistance from Benja Fallenstein, Marcello Herreshoff, Elliot Jin, and Nathaniel Thomas.

References

[1] R. Axelrod and W. D. Hamilton. The Evolution of Cooperation. *Science*, 211:1390–1396, March 1981.

[2] Mihaly Barasz, Paul Christiano, Benja Fallenstein, Marcello Herreshoff, Patrick LaVictoire, and Eliezer Yudkowsky. Robust cooperation in the Prisoner's Dilemma: Program equilibrium via provability logic. http://arxiv.org/pdf/1401.5577v1.pdf.

[3] Tsvi Benson-Tilsen. UDT with known search order. http://intelligence.org/files/UDTSearchOrder.pdf, 2014.

[4] G. Boolos. *The Logic of Provability*. Cambridge University Press, 1995.

[5] G.S. Boolos, J.P. Burgess, and R.C. Jeffrey. *Computability and Logic*. Cambridge University Press, 5 edition, 2007.

[6] Paul Christiano. Non-omniscience, probabilistic inference, and metamathematics. http://intelligence.org/files/Non-Omniscience.pdf.

[7] Paul F. Christiano, Eliezer Yudkowsky, Marcello Herreshoff, and Mihaly Barasz. Definability of truth in probabilistic logic. http://intelligence.org/files/DefinabilityTruthDraft.pdf.

[8] Benja Fallenstein. An optimality result for modal UDT. http://forum.intelligence.org/item?id=50, 2014.

[9] Benja Fallenstein and Nate Soares. Problems of self-reference in self-improving space-time embedded intelligence. Number 8598 in Lecture Notes in Artificial Intelligence, pages 21–32. Springer, 2014.

[10] Kurt Gödel. Über formal unentscheidbare sätze der Principia Mathematica und verwandter systeme I. *Monatshefte für Mathematik*, 38(1):173–198, 1931.

[11] Patrick LaVictoire, Benja Fallenstein, Eliezer Yudkowsky, Mihaly Barasz, Paul Christiano, and Marcello Herreshoff. Program equilibrium in the prisoner's dilemma via Löb's Theorem. In *Multiagent Interaction without Prior Coordination: Papers from the AAAI-14 Workshop*. AAAI Publications, 2014.

[12] M. H. Löb. Solution of a Problem of Leon Henkin. *The Journal of Symbolic Logic*, 20(2):pp. 115–118, 1955.

[13] Nate Soares and Benja Fallenstein. Toward idealized decision theory. http://intelligence.org/files/TowardIdealizedDecisionTheory.pdf, 2014.

[14] Alfred Tarski. The concept of truth in formalized languages [Der wahrheitsbegriff in den formalisierten sprachen]. In J. H. Woodger, editor, *Logic, Semantics, Metamathematics*, pages 152–278. Hackett, 2 edition, 1983.

[15] Ken Thompson. Reflections on Trusting Trust. *Communications of the ACM*, 27(8):761–763, 1984.

[16] Eliezer Yudkowsky. A cartoon guide to Löb's theorem. http://www.yudkowsky.net/assets/pdf/LobsTheorem.pdf.

[17] Eliezer Yudkowsky. Timeless decision theory. http://intelligence.org/files/TDT.pdf, 2010.

[18] Eliezer Yudkowsky and Marcello Herreshoff. Tiling agents for self-modifying AI, and the Löbian obstacle. http://intelligence.org/files/TilingAgents.pdf, 2013.