Parametric Bounded Löb's Theorem and Robust Cooperation of Bounded Agents

Andrew Critch

Machine Intelligence Research Institute

critch@intelligence.org

Abstract

Löb's theorem and Gödel's theorems make predictions about the behavior of self-reflective systems with unbounded computational resources with which to write and evaluate proofs. However, in the real world, self-reflective systems will have limited memory and processing speed, so in this paper we introduce an effective version of Löb's theorem theorem which is applicable given such bounded resources. These results have powerful implications for the game theory of bounded agents who are able to write proofs about themselves and one another, including the capacity to out-perform classical Nash equilibria and correlated equilibria, attaining mutually cooperative program equilibrium in the Prisoner's Dilemma. Previous cooperative program equilibria studied by Tennenholtz [1] and Fortnow [2] have depended on tests for program equality, a fragile condition, whereas "Löbian" cooperation is much more robust and agnostic of the opponent's implementation.

1 Background and Overview

The arc of this paper begins and ends with a discussion of the Prisoner's Dilemma, but it passes through a new result in provability logic. Thus, it will hopefully be of interest to game theorists and logicians alike.

1.1 Open-source Prisoner's Dilemma

Consider the Prisoner's Dilemma, a game with two possible actions C (Cooperate) and D (Defect), with the following payoff matrix:

		Player 2	
		C	D
Player 1	C	$(2,2)$	$(0,3)$
	D	$(3,0)$	$(1,1)$

In other words, by choosing D over C, each player can destroy 2 units of its opponent's utility to gain 1 unit of its own. As long as the payoffs are truly represented in the matrix—for example, there are no reputational costs of choosing D that are not already imputed in the payoffs—then (D, D) is the only Nash equilibrium, and the only correlated equilibrium. In fact, irrespective of the opponent's move, it is better to defect. It is therefore broadly believed that (D, D) is an inevitable outcome between "rational" agents in a truly represented (non-iterated) Prisoner's Dilemma.

But consider a version of the game — as first studied by [1] — wherein each player is an algorithm which can read its opponent's source code, as well as its

own, before the game. Is D still the obvious correct strategy? As a warm-up, one can imagine designing various algorithmic "agents" to compete in such games. For example, an agent who always cooperates:

```
def CooperateBot(Opponent) :
        return C
```

Tenenholtz [1] considers a simple agent which cooperates if and only if the opponent is identitically equal to itself:

```
def IsMeBot(Opponent) :
        if Opponent=IsMeBot
                return C
        else
                return D
```

When playing against IsMeBot, the opponent is incentivized to "be IsMeBot", and in particular, cooperate. To capture this intuition, Tenenholtz defines a **program equilibrium** to be a pair of agents (programs) competing in a game, with access to one another's source code, such that replacing either agent by a different agent would decrease its expected payoff. Thus, a program equilibrium is a Nash equilibrium of the 'meta-game' of choosing which program to play.

Agents in a program equilibrium can return outputs that do not constitute a Nash equilibrium (of the object-level game), even in a one-shot game, as can be seen here: (IsMeBot,IsMeBot) is a program equilibrium, returning outputs (C,C) and payoffs (2,2). This program equilibrium of IsMeBot is highly fragile, however: if we let IsMeBot' be the same program but with a tiny irrelevant change to its code — a comment perhaps — then IsMeBot will defect against it. Agents studied by [2] are similarly fragile.

To the end of someday designing real-world cooperative agents, it is therefore interesting to design a more "robust" cooperative agent, whose behavior does not depend too heavily on the details of the implementation of its opponent, but which nonetheless incentivizes its opponent to cooperate. For this, consider:

```
def FairBot_k(Opponent) :
        search for a proof of length k that
                Opponent(FairBot_k) = C
        if found,
                return C
        else
                return D
```

Here a 'proof of length k' means a mathematical proof — say, in some implementation of Peano Arithmetic — using fewer than k characters (symbols) to write out as a text file. To begin thinking about these agents, observe that

- CooperateBot(FairBot$_k$) = C, because CooperateBot always returns C;

- FairBot$_k$(CooperateBot) = C when k is large enough to complete the shortest proof that CooperateBot(FairBot$_k$) = C (which, given the simplicity of CooperateBot, will be very short), and D when k is too small to complete the proof.

1.2 The Example of FairBot vs FairBot

The first interesting question that arises is then:

$$\text{What is FairBot}_k(\text{FairBot}_k)?$$

It is not so hard to see that if k is too small to complete any proofs, each Fair-Bot returns D. But suppose k is extremely large, for example, 10^{100}. Does $\text{FairBot}_k(\text{FairBot}_k)$ find a proof that $\text{FairBot}_k(\text{FairBot}_k) = C$ and therefore return C, validating the proof? Or does it continue searching for a proof that $\text{FairBot}_k(\text{FairBot}_k) = C$ until the proof bound is reached, and having found no such proof, return D, consistent with the failed proof search?

It is worth pausing a moment to reflect on this question, since, when given no hints, 100% of the dozens of mathematicians and computer scientists I've seen asked it have answered incorrectly at first (myself included).

Consider that each instance of FairBot_k is waiting for a proof that the other FairBot_k will return C before it will return C itself, and since neither algorithm has a clause in its code to take a "leap of faith" in such a situation, it seems that neither algorithm will "make the first move", so their proof searches must simply keep searching until they reach their limit k and return D.

However, this reasoning turns out to be incorrect, because of a version of Löb's Theorem that is the main result of this paper, proven in Section 5. It implies that $\text{FairBot}_k(\text{FairBot}_k) = C$ for large k. Aside from being surprising, this result opens up a whole class of behaviors that can outperform the classical (D, D) equilibrium in a truly formulated, non-iterated Prisoner's Dilemma. Moreover, this performance can be made more robust, into a statement about any two agents willing to cooperate based on a proof of their opponents' cooperation.

Such interesting "Löbian" behavior first seemed plausible from the work of Bárász and LaVictoire et al. [3][4], who illustrated something like program equilibria among certain non-computable logical entities they called "modal agents", including an analog of FairBot in that context.

1.3 Robust Cooperative Program Equilibria

The main application of this paper is to establish robust cooperative program equilibria for computationally bounded agents. In particular, it is possible to write algorithms which are unexploitable in a Prisoner's Dilemma—that is, they never receive the undesirable outcome (C, D) as Player 1—and which achieve the outcome (C, C) against a variety of opponents, such that there is no incentive for their opponents to deviate from cooperation, even though there is no iteration or reputation to be earned in the game. This is what we mean by "robust cooperation".

To summarize the result, we write $\square_k p$ for the statement "p can be proven using k or fewer written symbols". Given a nonnegative increasing function G, we say that an agent A_k taking a parameter $k \in \mathbb{N}$ is **G-fair** if

$$\vdash \square_{k+G(\text{LengthOf}(Opp))}[Opp(A_k) = Cooperate] \rightarrow A_k(Opp) = Cooperate$$

In other words, if A_k finding a proof that its opponent cooperates is sufficient for A_k to cooperate, we say it is G-fair, provided the proof lengths in the search did not exceed $k + G(\text{LengthOf}(Opp))$. Then we have, in terms to be made precise later,

Theorem (**Robust cooperation of bounded agents**). *If certain bounds are satisfied by the Gödel encoding of our proof system, and the function G exceeds a certain asymptotic lower bound, then for any G-fair agents A_k and B_k, we have for all sufficiently large m, n,*

$$A_m(B_n) = B_n(A_m) = Cooperate$$

This result depends crucially on a new version of Lö's Theorem.

1.4 Löb's Theorem

Löb's Theorem states that, if $\square p$ denotes the provability of statement p in Peano Arithmetic (or any extension of it), then

$$\square(\square p \rightarrow p) \rightarrow \square p$$

If the reader has never encountered this result, consider the case where p is the Riemann Hypothesis, RH. Suppose that the Riemann Hypothesis is, unbeknownst to us, false. Without yet knowing whether RH is true, it is tempting for us to claim at least that *if RH is provable, then RH is true*, i.e. $\Box RH \to RH$. However, if that claim were itself provable, i.e. if $\Box(\Box RH \to RH)$, then Löb's Theorem tells us that $\Box RH$—the Riemann Hypothesis is provable—which is very bad news for the soundness of our proof system if the Riemann Hypothesis is actually false!

Thus, Löb's Theorem defies the intuition that we might soundly prove the "self-trust" statement that *if we prove p, then p is true.* This counterintuitiveness is in fact the same phenomenon as the surprising outcome that $\text{FairBot}_k(\text{FairBot}_k) = C$ from earlier, except that the FairBots—being algorithms which halt—only concern proofs up to a certain bounded length, k. Hence the motivation of this paper: to establish a version of Löb's theorem for proofs bounded in length by a parameter, k. In rough terms, we prove:

Theorem (Parametric Bounded Löb). *Suppose $p(-)$ is a logical formula with a single unquantified variable, and that $f : \mathbb{N} \to \mathbb{N}$ is computable and exceeds a certain asymptotic lower bound. Then $\exists \hat{k}$:*

$$\vdash \forall k, \ \Box_{f(k)} p(k) \to p(k)$$
$$\Rightarrow \ \vdash \forall k > \hat{k}, \ p(k)$$

1.5 Comparison to previous work

As mentioned, [1] first defined program equilibria and studied various 'non-robust' examples similar to IsMeBot above, which depend on program equality. In particular, if one agent is written in C++ while the other is written in Python, they will defect against each other. Examples studied by [2] are similarly fragile.

Later, Peters and Szentes [5] consider agents encoded as a first-order formulas over the integers which can reference the Gödel-numbering of the formula for the other player as well as its own, but these agents are non-computable in a way similar to those of Bárász and LaVictoire et al. [3][4].

By comparison, the program equilibria exhibited here are both computable and *robust*, in that they do not depend on tests for program equality, and generally exist between many pairs of agents provided they both follow a certain principle of fairness, in which a new bounded Löb's Theorem plays a crucial role.

1.6 Long-Term Relevance

As automated reasoning and decision-making systems improve, it is plausible that some such systems might exhibit a capacity to reason in generality about their own design principles, and those of other systems. As an illustrative example, such a system can be designed expressly today: a theorem-prover can be handed a copy of its own source code and queried to write proofs about it. Less contrivedly, there might be economic value in creating systems that can reason about themselves and others, such as for collaboration or negotiation. For example, a human can reason that he is mentally outclassed in the middle of a competitive game of Go against a new player, and therefore resign to hedge his losses. Such reasoning invokes a theory of the reasoning capacity of one's opponent, and of oneself: algorithms reasoning about algorithms.

It therefore seems prudent to explore what game-theoretic dynamics emerge from algorithms reasoning about each other, beginning with the simplest cases we can currently state and examine, similar in spirit to the way RAND Corporation's Thomas Schelling began his understanding of nuclear deterrence [6][7], by analyzing simple examples of non-zero-sum games [8].

In this paper, we find that classical game theory—and more generally, causal decision theory [9]—is not an adequate framework for describing the competitive interactions of algorithms that reason about the source codes of their opponent

algorithms and themselves. When given read access to one another's source code—an extreme scenario for two humans, but trivial for computer systems—competing algorithms can exhibit counterintuitive "Löbian" behaviors which, among other things, can robustly achieve cooperative outcomes that outperform classical Nash equilibria and correlated equilibria. Moreover, the time at which each algorithm outputs its cooperative decision occurs later in time than the causal pathway by which it benefits from the decision (namely, the pathway wherein its opponent predicts its behavior using its source code; see Section 6.1).

Thus, without further investigation, our more classical intuitions about what group-level behaviors will emerge from such algorithms may miss the mark entirely.

2 Fundamentals

Here we begin building up the main technical result of the paper. The algorithms examined here will make use of provability logic as a way of "reasoning about reasoning", and the main resource bounds on the algorithms, for simplicity, will be the lengths of the proofs they may discover.

2.1 Proof Length and Notation

If the first line of a three-line proof is so long that it would not fit on any physical computer system, saying the proof is "only three lines long" is not very descriptive. Therefore, we will measure proof length in *characters* instead of lines, the way one might measure the size of a text file on a computer. An extensive analysis of proof lengths measured in characters is covered by Pudlák [10].

We will fix a proof system S (e.g. an extension of Peano Arithmetic) throughout, and write

$$S \vdash_n \phi, \quad \text{or simply} \quad \vdash_n \phi$$

to mean that there exists an S-proof of ϕ using n or fewer characters. After a choice of Gödel encoding for S, it is customary to write $\Box\phi$ for $\exists n : Bew(n, \ulcorner\phi\urcorner)$, i.e., there exists a number n encoding a proof of ϕ. This allows S to indirectly talk about the existence of proofs in S, indirectly. We will extend this definition to talk about proof lengths:

$$\Box_n\phi \quad \text{means} \quad \exists m : Bew(m, \ulcorner\phi\urcorner) \text{ and } \text{ProofLength}(m) < n$$

where $\text{ProofLength}(m)$ denotes the length, in characters, of the proof encoded by m. In other wrods, $\Box_n\phi$ is the S-encoded statement that ϕ can be proven in S with n or fewer characters.

2.2 Proof System

We let S be any first-order proof system that

1) can represent computable functions in the sense of Section 2.4,

2) can write any number $k \in \mathbb{N}$ using $\mathcal{O}\lg k$ symbols, and

3) allows the definition and expansion of abbreviations during proofs.

For example, we could take Peano Arithmetic, where each proof line is either

- an axiom, or

- an application of Modus Ponens from lines above it,

and additionally allow ourselves to write numbers in a binary format, and allow proof lines lines which are

- the definition of an abbreviation that may be used in subsequent lines, or

5

- an expansion of an abbreviation used in a previous line.

We have chosen to allow abbreviations in our proof system for two reasons. The first is that real-world automated proof systems will tend to use abbreviations because of memory constraints. The second is that abbreviations make the lengths of shortest-proofs in this system are slightly easier to analyze: for example, if a number N with a very large number of digits occurs in the shortest proof of a proposition, it will not occur multiple times; instead, it will occur only once, in the definition of an abbreviation for it. Then, we don't have to carefully count the number of times the numeral occurs in the proof to determine its contribution to the proof length; its contribution will simply be linear in its length, or $\lg N$.

We write

Lang(S) for the language of S,

Lang$_r(S)$ for the formulas in Lang(S) with r free variables, and

Const(S) for the set of constants in S (e.g. 0, $\mathcal{S}0$, etc.).

2.3 Gödel Encoding

We fix throughout a Gödel numbering

$$\#(-) : \mathrm{Lang}(S) \to \mathbb{N}$$

and a "numeral" mapping

$$^{\circ}(-) : \mathbb{N} \to \mathrm{Const}(S) \subseteq \mathrm{Lang}(S)$$

for expressing naturals as constants in S. Note that in traditional $\mathcal{PA}$, for example, $^{\circ}5 = \mathcal{SSSSS}0$. However, to be more realistic we have assumed that S uses a binary encoding to be more efficient, so e.g.,

$$^{\circ}5 = 101.$$

The maps $\#(-)$ and $^{\circ}(-)$ combine to form a Gödel encoding

$$\ulcorner(-)\urcorner : \mathrm{Lang}(S) \to \mathrm{Const}(S)$$

$$\ulcorner\phi\urcorner := {^{\circ}}\#\phi$$

which allows S to write proofs about itself.

2.4 Convention for Representing Computable Functions

The astute reader will notice that throughout, although $\mathcal{PA}$ and related first-order theories typically have no symbols for functions, we will often write objectionable expressions like

$$\vdash \ldots \text{something about } f(x) \ldots$$

where $f : \mathbb{N} \to \mathbb{N}$ is some computable function.

However, there is a convention for interpreting such statements. It is known (see, e.g. Theorem 6.8 of Cori and Lascar, Part II [11]) that for any computable function $f : \mathbb{N} \to \mathbb{N}$, there exists a "graph" predicate $\Gamma_f(-, -) \in \mathrm{Lang}_2(\mathcal{PA})$ such that

$$\forall x \in \mathbb{N}, \ \mathcal{PA} \vdash \forall y, \ \Gamma_f(^{\circ}x, y) \leftrightarrow y = {^{\circ}}f(x)$$

We have assumed that S is capable of representing computable functions in this way (e.g., by being an extension of $\mathcal{PA}$).

The two-place predicates Γ_f are cumbersome in writing because each usage introduces a quantifier. For example, if we we have functions f, g and h and we

want to say that S proves that for any x value, $f(x) < g(x) + h(x)$, technically we should write

$$\vdash \forall x \forall y_1 \forall y_1 \forall y_3, \ \Gamma_f(x, y_1) \text{ and } \Gamma_g(x, y_2) \text{ and } \Gamma_h(x, y_3) \to y_1 < y_2 + y_3$$

However, for easier reading, in such cases we will abuse notation and write

$$\vdash \forall x, \ f(x) < g(x) + h(x),$$

leaving the expansion in terms of Γ's and $\forall y$'s as an exercise to any willing reader.

2.5 Asymptotic Notation

We use the convention that $f \prec g$ means that for any $M \in \mathbb{N}$, there exists and $N \in \mathbb{N}$ such that $\forall n > N, Mf(n) < g(n)$. We write $\mathcal{O}g$ for the set of functions $f \preceq g$, and for a specific function $\mathcal{E}$ we will sometimes write write $\mathcal{E}\mathcal{O}g$ for the set of functions of the form $\mathcal{E} \circ f$ where $f \in \mathcal{O}g$.

3 A Parametric Diagonal Lemma

Löb's Theorem can be proven via the classical Diagonal Lemma [12], which states that for any formula $F(-) \in \mathrm{Lang}_1(S)$ (having one free variable), there exists a sentence $\psi \in \mathrm{Lang}_0(S)$ (with no free variables) such that

$$\vdash \psi \leftrightarrow F(\ulcorner \psi \urcorner).$$

However, to reason about computer systems with certain as-yet unset parameters, we will need a generalization of the Diagonal Lemma for formulas with free variables to represent those parameters in a way that avoids writing a separate proof for every instance of the parameters:

Proposition 1 (Parametric Diagonal Lemma). *Suppose S is a first-order theory capable of representing all computable functions, as in Section 2.4. Then for any predicate $G \in \mathrm{Lang}_{r+1}(S)$, there exists a predicate $\psi \in \mathrm{Lang}_r(S)$ such that*

$$\vdash \forall \bar{k} = (k_1, \ldots, k_r), \ \psi(\bar{k}) \leftrightarrow G(\ulcorner \psi \urcorner, \bar{k})$$

Proof. We define a "partial self-evaluation function" $e : \mathbb{N} \to \mathbb{N}$ as follows:

$$e(n) = \begin{cases} \# \left[\theta(\ulcorner \theta \urcorner, -, \ldots, -) \right] & \text{if } n = \#\theta \text{ for some } \theta \in \mathrm{Lang}_{r+1}(S) \\ 0 & \text{otherwise} \end{cases}$$

Now, e is computable, and therefore representable in $\mathrm{Lang}(S)$, so we can define $\beta \in \mathrm{Lang}_{r+1}(S)$ by

$$\beta(n, \bar{k}) := G(e(n), \bar{k})$$

(using the notational convention of Section 2.4 to avoid writing extra quantifiers and Γ_e's). Then, $\forall \theta \in \mathrm{Lang}_{r+1}(S)$,

$$\vdash \forall \bar{k} \ \beta(\ulcorner \theta \urcorner, \bar{k}) \leftrightarrow G(\ulcorner \theta(\ulcorner \theta \urcorner, -, \ldots, -) \urcorner, \bar{k})$$

Now let $\theta = \beta$, so we have

$$\vdash \forall \bar{k} \ \beta(\ulcorner (\urcorner \beta), \bar{k}) \leftrightarrow G(\ulcorner \beta(\ulcorner \beta \urcorner, -, \ldots, -) \urcorner, \bar{k})$$

Finally, taking $\psi(\bar{k}) = \beta(\ulcorner \beta \urcorner, \bar{k})$ yields the desired result

$$\vdash \forall \bar{k} \ \psi(\bar{k}) \leftrightarrow G(\ulcorner \psi \urcorner, \bar{k})$$

$\square$

4 A Bounded Provability Predicate, $\Box_k$

4.1 Defining $\Box_k$

Given a choice of Gödel encoding for Peano Arithmetic, it is classical that a predicate $Bew(-, -) \in \mathrm{Lang}_2(S)$ exists such that $Bew(m, n)$ means, in natural language, that the number m encodes a proof in $\mathcal{PA}$, and that the number n encodes the statement it proves. So, the standard provability operator $\Box : \mathrm{Lang}(\mathcal{PA}) \to \mathrm{Lang}(\mathcal{PA})$ can be defined as

$$\Box \phi := \exists m : Bew(m, \ulcorner \phi \urcorner).$$

We take for granted that Bew exists for S and can be extended to a three-place predicate $Bew(-, -, -) \in \mathrm{Lang}_2(S)$ such that $Bew(m, n, k)$ means that

- m encodes a proof in S,

- n encodes the statement it proves, and

- the proof encoded by m uses at most k characters when written in the language of S (*not* when written using the encoding.)

Then we can define a "bounded" box operator:

$$\Box_k \phi = \exists m : Bew(m, \ulcorner \phi \urcorner, k).$$

We also take for granted a computable "single variable evaluation" function, $Eval_1 : \mathbb{N} \to \mathbb{N}$, such that for any $\phi(-) \in \mathrm{Lang}_1(S)$,

$$Eval_1(\ulcorner \phi \urcorner, k) = \ulcorner \phi(^\circ k) \urcorner$$

Since $Eval_1$ is computable, it can be represented in $\mathrm{Lang}(S)$ as in Section 2.4. This allows us to extend the $\Box_k$ operator to act on sentences $\phi(-)$ with an unbound variable:

$$(\Box_k \phi)(\ell) := \exists m : Bew(m, Eval_1(\ulcorner \phi \urcorner, \ell), k)$$

In words, "There is a proof using k or fewer characters of the formula $\phi(\ell)$".

4.2 Basic Properties of $\Box_k$

Each of the following properties will be needed multiple times during the proof of Parametric Bounded Löb. Since the proof is already highly symbolic, we give these properties English names to recall them.

Property 1 (Implication Distribution). *There is a constant $c \in \mathrm{Const}(S)$ such that for any $p, q \in \mathrm{Lang}(S)$,*

$$\vdash \forall a \forall b, \ \Box_a(p \to q) \to (\Box_b p \to \Box_{a+b+c}).$$

Proof sketch. The fact that one can combine a proof of an implication with the proof of its antecedent to obtain a proof of its consequent can be proven in general, with quantified variables in place of the Gödel numbers of the particular statements involved. Let us suppose this general proof has length c_0. Then, we need only instantiate the statements in it to p and q. However, if p and q are long expressions, they can have been abbreviated in the earlier proofs without lengthening them, so they can be written in abbreviated form again during this step. Hence, the total cost of combining the two proofs is around $c = 2c_0$, which is constant with respect to p and q. $\qquad\Box$

Property 2 (Quantifier Distribution). *There is a constant $C \in \mathrm{Const}(S)$ such that for any $\phi(-) \in \mathrm{Lang}_1(S)$,*

$$\vdash \Box_N (\forall k \phi(k))$$

$$\Rightarrow \ \vdash \forall k \, \Box_{C+2N+\lg k} \phi(k), \ \textit{which in turn}$$

$$\Rightarrow \ \vdash \forall k \, \Box_{\mathcal{O}\lg k} \phi(k)$$

Proof. An encoded proof of $\phi(°K)$ for a specific K can be obtained by specializing the conclusion of an N-character encoded proof of $\forall k\phi(k)$ and appending the specialization with $°K$ in place of k at the end. To avoid repeating $°K$ numerous times in the final line (in case it is large), we will use an abbreviation for ϕ. Thus the appended lines can say:

(1) let Φ stand for $\ulcorner\phi\urcorner$

(2) $\Phi(°K)$

Let us analyze how many characters are needed to write such lines. First, we need a string Φ to use as an abbreviation for ϕ. Since no string of length $\frac{N}{2}$ has yet been used as an abbreviation in the earlier proof (otherwise we can shorten the proof by not defining and using the abbreviation), we can surely have $\text{Length}(\Phi) < \frac{N}{2}$. We also need some constant c number of characters to write out the system's equivalent of "let", "stand for", "(", and ")". Finally, we need $\lg K$ characters to write $°K$. Altogether, the proof was extended by $C + N + \lg(k)$ characters, for a total length of $2N + c + \lg k$. □

5 Parametric Bounded Löb

Definition 2 (Proof expansion function). We choose a computable function $\mathcal{E} : \mathbb{N} \to \mathbb{N}$ to bound the expansion of proof lengths when we Gödel-encode them. Its definition is that it must be large enough to satisfy the following two properties:

Property 3 (Bounded Necessitation). $\forall \phi \in \text{Lang}(S)$,

$$\underset{k}{\vdash} \phi \tag{5.1}$$

$$\Rightarrow \underset{\mathcal{E}k}{\vdash} \Box_k \phi \tag{5.2}$$

Property 4 (Bounded Inner Necessitation). *For any* $\phi \in \text{Lang}(S)$,

$$\vdash \Box_k \phi \to \Box_{\mathcal{E}k} \Box_k \phi.$$

Estimating $\mathcal{E}$. How large must $\mathcal{E}$ be in practice? Gödel numberings for sequences of integers can be achieved in $\mathcal{O}n$ space [13], as can Gödel numberings of term algebras [14]. To check that one line is an application of Modus Ponens from previous lines, if the proof encoding indexes the implication to which MP is applied, is a test for string equality that is linear in the length the of lines. Finally, to check that an abbreviation has been applied or expanded, if the proof encoding indexes where the abbreviation occurs, is also a linear time test for string equality. Thus, it seems reasonable to expect $\mathcal{E} \in \mathcal{O}k$ for real-world theorem-provers. But however large it may be, in any case we have:

Theorem 3 (Parametric Bounded Löb). *Suppose* $p(-) \in \text{Lang}_1(S)$ *is a formula with a single unquantified variable, and that* $f : \mathbb{N} \to \mathbb{N}$ *is computable and satisfies* $f(k) \succ \mathcal{E}\mathcal{O}\lg k$. *Then* $\exists \hat{k}$:

$$\vdash \forall k, \; \Box_{f(k)} p(k) \to p(k)$$

$$\Rightarrow \; \vdash \forall k > \hat{k}, \; p(k)$$

Note: In fact a weaker statement

$$\vdash \forall k > k_1, \; \Box_{f(k)} p(k) \to p(k)$$

is sufficient to derive the consequent, since we could just redefine $f(k)$ to be 0 for $k \leq k_1$ and then $\Box_{f(k)} p(k) \to p(k)$ is vacuously true and provable for $k \leq k_1$ as well.

Proof. (In this proof, each centered equation will follow directly from the one above it unless otherwise noted.)

We begin by choosing some function $g(k)$ such that $\lg k \prec g(k)$ and $\mathcal{E}g(k) \prec f(k)$. For example, we could take $g(k) = \lfloor \sqrt{(\lg k)(\mathcal{E}^{-1}f(k))} \rfloor$. Define a predicate $G(-,-) \in \mathrm{Lang}_2(S)$ by

$$G(n,k) := (\exists m : Bew(m, Eval_1(n,k), g(k))) \to p(k)$$

so that for any $\phi(-) \in \mathrm{Lang}_1(S)$,

$$G(\ulcorner \phi \urcorner, k) = \square_{g(k)} \phi(k) \to p(k).$$

Now, by the Parametric Diagonal Lemma, $\exists \psi(-) \in \mathrm{Lang}_1(S)$ such that in some number of characters n,

$$\underset{n}{\vdash} \forall k\ \psi(k) \leftrightarrow G(\ulcorner \psi \urcorner, k) \tag{5.3}$$

By Bounded Necessitation,

$$\vdash \square_n \left(\forall k\ \psi(k) \leftrightarrow G(\ulcorner \psi \urcorner, k) \right)$$

By Quantifier Distribution, since n is constant with respect to k,

$$\vdash \forall k\ \square_{\mathcal{O}\lg k} \left(\psi(k) \leftrightarrow G(\ulcorner \psi \urcorner, k) \right),$$

in which we can specialize to the forward implication,

$$\vdash \forall k\ \square_{\mathcal{O}\lg k} \left(\psi(k) \to G(\ulcorner \psi \urcorner, k) \right)$$

By Implication Distribution of $\square_{\mathcal{O}\lg k}$,

$$\vdash \forall k \forall a\ \square_a \psi(k) \to \square_{a+\mathcal{O}\lg k} G(\ulcorner \psi \urcorner, k)$$

By Implication Distribution again, this time of $\square_{a+\mathcal{O}\lg k}$ over the implication $G(\ulcorner \psi \urcorner, k) = \square_{g(k)} \phi(k) \to p(k)$, we obtain

$$\vdash \forall k \forall a \forall b\ \square_a \psi(k) \to \left(\square_b \square_{g(k)} \psi(k) \to \square_{a+b+\mathcal{O}\lg k} p(k) \right)$$

Now we specialize this equation to $a = g(k)$ and $b = h(k)$, where $h : \mathbb{N} \to \mathbb{N}$ is a computable function satisfying $\mathcal{E}g(k) \prec h(k) \prec f(k)$, for example $h(k) = \lfloor \sqrt{f(k)\mathcal{E}g(k)} \rfloor$:

$$\vdash \forall k\ \square_{g(k)} \psi(k) \to \left(\square_{h(k)} \square_{g(k)} \psi(k) \to \square_{g(k)+h(k)+\mathcal{O}\lg k} p(k) \right)$$

Then since $g(k) + h(k) + \mathcal{O}\lg k < f(k)$ after some bound $k > k_1$, we have

$$\vdash \forall k > k_1,\ \square_{g(k)} \psi(k) \to \left(\square_{h(k)} \square_{g(k)} \psi(k) \to \square_{f(k)} p(k) \right)$$

Now, by hypothesis, $\vdash \forall k\ \square_{f(k)} p(k) \to p(k)$, thus

$$\vdash \forall k > k_1,\ \square_{g(k)} \psi(k) \to \left(\square_{h(k)} \square_{g(k)} \psi(k) \to p(k) \right) \tag{5.4}$$

Also, without any of the above, from Bounded Inner Necessitation we can write

$$\vdash \forall k \forall a\ \square_a \psi(k) \to \square_{\mathcal{E}a} \square_a \psi(k)$$

From this, with $a = g(k)$, we have

$$\vdash \forall k\ \square_{g(k)} \psi(k) \to \square_{\mathcal{E}g(k)} \square_{g(k)} \psi(k)$$

Now, since $\mathcal{E}g(k) < h(k)$ after some bound $k > k_2$, we have

$$\vdash \forall k > k_2 \; \Box_{g(k)}\psi(k) \to \Box_{h(k)}\Box_{g(k)}\psi(k) \tag{5.5}$$

Next, from Equations 5.4 and 5.5, assuming we chose $k_2 \geq k_1$ for convenience, we have

$$\vdash \forall k > k_2, \; \Box_{g(k)}\psi(k) \to p(k) \tag{5.6}$$

But from Equation 5.3, the implication here is equivalent to $\psi(k)$, so we have

$$\underset{N}{\vdash} \forall k > k_2, \; \psi(k),$$

where N is the number of characters needed for the proof above. From this, by Bounded Necessitation, we have

$$\vdash \Box_N[\forall k > k_2, \; \psi(k)].$$

By Quantifier Distribution of $\Box_N$,

$$\vdash \forall k > k_2, \; \Box_{\mathcal{O}\lg k}\psi(k)$$

and since $\mathcal{O}\lg k < g(k)$ after some bound $k > \hat{k}$, taking $\hat{k} \geq k_2$ for convenience, we have

$$\vdash \forall k > \hat{k}, \; \Box_{g(k)}\psi(k). \tag{5.7}$$

Finally, from Equations 5.6 and 5.7 we have

$$\vdash \forall k > \hat{k}, \; p(k),$$

as required. $\qquad\square$

6 Robust Cooperation of Bounded Agents in the Prisoner's Dilemma

Bárász, LaVictoire and others [3][4] have exhibited various proof-based agents who robustly cooperate in the Prisoner's Dilemma by basing their decisions on proofs about each others' cooperation. However, their agents are purely logical entities which can discover proofs of unbounded length, and so are impossible to run on a physical computer. This leaves open the question of whether such behavior is achievable by agents with bounded computational resources.

So, consider the following bounded agent, where G is some increasing, non-negative function to be determined later, and $G = 0$ recovers the definition of FairBot from Section 1:

```
def FairBot_k(Opponent) :
        let B = k + G(LengthOf(Opponent))
        search for proof of length at most B that
                Opponent(FairBot_k) = Cooperate
        if found,
                return Cooperate
        else
                return Defect
```

Question: What is $\text{FairBot}_k(\text{FairBot}_k)$? It seems intuitive that each FairBot is waiting for the other to provably cooperate, in a bottomless regression that will exhaust the proof bound B. Thus, they will find no proof of cooperation, and hence defect.

However, this turns out not to be the case, as a consequence of Parametric Bounded Löb. We let

$$p(k) := [\text{FairBot}_k(\text{FairBot}_k) = Cooperate].$$

Since $G \geq 0$, $k \leq B$ in the definition of FairBot, so we have

$$\vdash \Box_k p(k) \to \Box_B p(k).$$

Now since $\Box_B p(k)$ is FairBot's criterion for cooperation, we also have

$$\vdash \Box_B p(k) \to p(k), \quad \text{so}$$

$$\vdash \forall k, \; \Box_k p(k) \to p(k),$$

whence for sufficiently large $\hat{k}$, by Parametric Bounded Löb,

$$\vdash \forall k > \hat{k}, \; p(k).$$

In other words, $FairBot_k$ cooperates with $FairBot_k$ for large k.

This result is interesting for three reasons:

1. It is *surprising.* 100% of the dozens of mathematicians and computer scientists that I've asked to guess the output of $\text{FairBot}_k(\text{FairBot}_k)$ have guessed incorrectly (expecting the proof searches to enter an infinite regress and thus reach their bounds), or have given an invalid argument for cooperation (such as "it would be better to cooperate, so they will").

2. It is *advantageous.* FairBot outperforms the classical Nash/correlated equilibrium solution (Defect, Defect) to the Prisoner's Dilemma, in a one-shot game with no iteration or future reputation. Moreover, it does so *while being unexploitable*: if an opponent will defect against FairBot, FairBot will find no proof of the opponent's cooperation, so it will also defect.

3. It is *robust.* Previous examples of cooperative program equilibria studied by [1] and [2] all involved cooperation based on *equality of programs*, a very fragile condition. For example, the agent IsMeBot from the introduction will mutually defect against an identical opponent written in a different programming language, or even in a slightly different style. Such fragility is not desirable if we wish to build real-world cooperative systems.

Taking this robustness further, we next demonstrate mutual cooperative program equilibria among a wide variety of (unequal) agents, provided only that they employ a certain "principle of fairness". Given a non-negative increasing function G, we say that an agent A_k taking a parameter $k \in \mathbb{N}$ is **G-fair** if

$$\vdash \Box_{k + G(\text{LengthOf}(Opp))}[Opp(A_k) = C] \to A_k(Opp) = C$$

In other words, if A_k finding a proof that its opponent cooperates is sufficient for A_k to cooperate, we say it is G-fair, provided the proofs in the search did not exceed length $k + G(\text{LengthOf}(Opp))$. The agents FairBot_k defined above are G-fair, and the reader is encouraged to keep these examples in mind for the following result:

Theorem 4 (Robust cooperation of bounded agents). *Suppose that*

- *the proof expansion function $\mathcal{E}$ (defined in Section 5) of our proof system satisfies $\mathcal{E}\mathcal{O}\lg k \prec k$,*

- *f is any function satisfying $\mathcal{E}\mathcal{O}\lg k \prec f(k) \prec k$, and*

- *G is any increasing function satisfying $G(\ell) > 6f(2^{\ell})$.*

Then, for any G-fair agents A_k and B_k, we can choose a threshold r such that for all $m, n > r$,

$$A_m(B_n) = B_n(A_m) = Cooperate$$

Feasibility of bounds. Before proceeding, recall from Section 5 that we can achieve $\mathcal{E} \in \mathcal{O}k$ for automatic proof systems that are designed for easy verifiability, in which case $\mathcal{E}\mathcal{O}\lg k = \mathcal{O}\lg k$, well below the $\prec k$ requirement.

Proof. For brevity, we let

$$a(k) := G(\text{LengthOf}(A_k)), \tag{6.1}$$
$$b(k) := G(\text{LengthOf}(B_k)), \tag{6.2}$$
$$\alpha(m, n) := [A_m(B_n) = Cooperate], \text{ and} \tag{6.3}$$
$$\beta(n, m) := [B_n(A_m) = Cooperate] \tag{6.4}$$

so we can write the G-fairness conditions more compactly as

$$\vdash\Box_{m+b(n)}\beta(n, m) \to \alpha(m, n) \text{ and} \tag{6.5}$$
$$\vdash\Box_{n+a(m)}\alpha(m, n) \to \beta(n, m).$$

Now, $\text{LengthOf}(A_k) > \lg k$ and $\text{LengthOf}(B_k) > \lg k$ since they must reference the parameter k in their code. Applying G to both sides yields

$$a(k), b(k) > G(\lg k) > 6f(k). \tag{6.6}$$

Define an "eventual cooperation" predicate:

$$p(k) := \forall m > k, \ \forall n > k, \ \alpha(m, n) \text{ and } \beta(n, m).$$

Using Quantifier Distribution once on the definition of $p(k)$,

$$\vdash \forall k[\Box_{f(k)}p(k) \to \forall m > k, \ \Box_{C+2f(k)+\lg m}[\forall n > k, \ \alpha(m, n) \text{ and } \beta(n, m)]]$$

Applying Quantifier Distribution again,

$$\vdash \forall k[\Box_{f(k)}p(k) \to \forall m > k, \forall n > k, \ \Box_{3C+4f(k)+2\lg m+\lg n}[\alpha(m, n) \text{ and } \beta(n, m)]]$$
(6.7)

Now, for m, n large and $> k$, we have

$$3C + \lg n < n \qquad\qquad \text{and by (6.6),}$$
$$4f(k) + 2\lg m < 6f(m) < a(m).$$

Adding these inequalities yields

$$3C + 4f(k) + 2\lg m + \lg n < n + a(m),$$

so for some k_1, from (6.7) we derive

$$\vdash \forall k > k_1, \ [\Box_{f(k)}p(k) \to \forall m > k, \forall n > k, \ \Box_{n+a(m)}\alpha(m, n)].$$

Similarly, we also have

$$3C + 2\lg m < m \qquad\qquad \text{and}$$
$$4f(k) + \lg n < 5f(n) < b(n), \qquad\qquad \text{so for some } k_2 \geq k_1,$$

$$\vdash \forall k > k_2 \ [\Box_{f(k)}p(k) \to \forall m > k, \forall n > k, \ \Box_{n+a(m)}\alpha(m, n) \text{ and } \Box_{m+b(n)}\beta(n, m)]$$

Thus by (6.5),

$$\vdash \forall k > k_2[\Box_{f(k)}p(k) \to \forall m > k, \forall n > k, \ c(n, m) \text{ and } c(m, n)], \text{ i.e.}$$

13

$$\vdash \forall k > k_2,\ \Box_{f(k)} p(k) \rightarrow p(k)$$

Therefore, by Parametric Bounded Löb (and the note following it), for some $\hat{k}$ we have

$$\vdash \forall k > \hat{k},\ p(k).$$

In other words, for all $m, n > \hat{k} + 1$,

$$A_m(B_n) = B_n(A_m) = Cooperate.$$

$\square$

6.1 Ramifications for Causal Decision Theory

Causal Decision Theory [9] is a framework for evaluating the desirability of an action by assessing the causal consequences of the action itself. The interaction of $FairBot_m$ and $FairBot_n$ present a challenge to Causal Decision Theory, in a way similar to Newcomb's Problem [15], a classic scenario wherein one agent is able to predict the actions of another.

Concretely, imagine $FairBot_m$ and $FairBot_n$ are played against each other while being run on separate computers in separate rooms, and that they will print their final responses, C or D, at the same time. When $FairBot_m$ decides to cooperate with $FairBot_n$, it does so *after* computing a proof that $FairBot_n(FairBot_m) = C$, but *before* its opponent $FairBot_n$ actually prints its response. There is therefore no causal effect transmitted from the value that $FairBot_n$ prints to its screen to the value that $FairBot_m$ prints to its screen. So from a purely causal perspective, there is an "incentive" for $FairBot_n$ to print D instead of C, since that would have "no effect" on its opponent, and would counterfactually yield the better outcome (D, C) in place of (C, C). Thus one might argue that $FairBot_n$ is acting sub-optimally in this scenario: its response could be changed to obtain a better outcome, (D, C).

However, such reasoning is misplaced from a strategic standpoint. $FairBot_n$ cannot output D while its opponent $FairBot_m$ outputs C, for that outcome would be logically incoherent. Although the instance of $FairBot_n$ running as Player 2 has no causal effect on the $FairBot_m$ running as Player 1, it cannot treat its decision as independent: the outcome (C, D) is simply not attainable by any agent under any circumstances when Player 1 is $FairBot_m$.

This prompts a re-thinking of what it means to make an optimal decision as an algorithm whose source code is transparent. Such questions, and some of their long-term relevance, have already been considered at length in [16].

7 Summary

We have discovered a version of Löb's Theorem which can be applied to algorithms with bounded computational resources. This result, in turn, can be used by algorithmic agents that have access to one another's source codes to achieve cooperative outcomes (among other things) that out-perform classical Nash equilibria and correlated equilibria, via conditions that are much more robust than previously known examples depending on program equality. Moreover, the causal pathway by which each agent benefits from its own decision to cooperate happens *before* the agent actually computes its decision, which prompts a re-thinking of the causal analysis of optimal decision-making known as Causal Decision Theory in a setting where decision-making agents are algorithms with transparent source-codes.

In light of these findings, classical game theoretic results and the intuitions we derive from them may be quite far from describing what we should actually expect from systems of agents capable of reasoning about each other's design. In order to ensure robust and beneficial long-term deployment of advanced AI technologies in the future, as described in [17] and supported by over 100 researchers in the Future of Life Institute's Open Letter [18], it seems prudent to investigate these dynamics

ahead of time, so as to be prepared for the sorts of game-theoretic scenarios that might arise between algorithmic agents in the future.

As a direction for potential future investigation, it seems inevitable that other agents described in the purely logical (non-computable) setting of Bárász and LaVictoire et al. [3][4] will likely have bounded, algorithmic analogs, and that many more general consequences of Löb's Theorem—perhaps all the theorems of Gödel–Löb provability logic—will have resource-bounded analogs as well.

Acknowledgements

References

[1]   Moshe Tennenholtz. "Program equilibrium". In: *Games and Economic Behavior* 49.2 (2004).

[2]   Lance Fortnow. "Program equilibria and discounted computation time". In: *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge.* ACM. 2009.

[3]   Mihály Bárász et al. *Robust Cooperation in the Prisoner's Dilemma. Program Equilibrium via Provability Logic.* eprint. 2014. arXiv: `1401.5577 [cs.GT]`.

[4]   Patrick LaVictoire et al. "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. URL: `http://www.aaai.org/ocs/index.php/WS/AAAIW14/paper/view/8833`.

[5]   Michael Peters and Balázs Szentes. "Definable and contractible contracts". In: *Econometrica* 80.1 (2012).

[6]   Thomas C. Schelling. "The Strategy of Conflict Prospectus for a Reorientation of Game Theory". In: *Journal of Conflict Resolution* 2.3 (1958).

[7]   Thomas C. Schelling. *Arms and Influence.* Henry L. Stimson Lectures. Written under the auspices of the Center for International Affairs, Harvard University. New Haven and London: Yale University Press, 1966.

[8]   Thomas C. Schelling. *Re-Interpretation of the Solution Concept for "Non-Cooperative" Games.* P-1385. Santa Monica, CA: RAND Corporation, 1958. URL: `http://www.rand.org/pubs/papers/P1385.html`.

[9]   Allan Gibbard and William L. Harper. "Counterfactuals and Two Kinds of Expected Utility. Theoretical Foundations". In: *Foundations and Applications of Decision Theory.* Ed. by Clifford Alan Hooker, James J. Leach, and Edward F. McClennen. Vol. 1. The Western Ontario Series in Philosophy of Science 13. Boston: D. Reidel, 1978.

[10]   Pavel Pudlák. "The Lengths of Proofs". In: *Handbook of Proof Theory.* Ed. by Samuel R. Buss. Vol. 137. Studies in Logic and the Foundations of Mathematics. Amsterdam: Elsevier, 1998. ISBN: 978-0-444-89840-1. DOI: `http://dx.doi.org/10.1016/S0049-237X(98)80023-2`.

[11]   René Cori and Daniel Lascar. *Mathematical Logic: A Course with Exercises. Recursion Theory, Gödel's Theorems, Set Theory, Model Theory.* Trans. by Donald Pelletier. Vol. Part II. New York: Oxford University Press, 2001.

[12]   Rudolf Carnap. *Logische Syntax der Sprache.* Vol. 8. Schriften zur Wissenschaftlichen Weltauffassung. Springer Berlin Heidelberg, 1934. DOI: `10.1007/978-3-662-25375-5`.

[13]   Shi-Chun Tsai, Jen-Chun Chang, and Rong-Jaye Chen. "A Space-efficient Gödel Numbering with Chinese Remainder Theorem". In: *Proceedings of the 19th Workshop on Combinatorial Mathematics and Computation Theory, Kaohsiung, Taiwan, Mar. 29-30, 2002.* 2002. URL: `http://par.cse.nsysu.edu.tw/~algo/paper/paper02/A0222.pdf`.

[14]   Paul Tarau. "Bijective Size-proportionate Gödel Numberings for Term Algebras".
       Unpublished manuscript. 2013. URL: http://logic.cse.unt.edu/tarau/research/
       2013/cgoedel.pdf.

[15]   Robert Nozick. "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*. Ed.
       by Nicholas Rescher. Synthese Library 24. Dordrecht, The Netherlands: D. Reidel,
       1969.

[16]   Nate Soares and Benja Fallenstein. *Toward Idealized Decision Theory*. eprint. 2015.
       arXiv: 1507.01986 [cs.AI].

[17]   Stuart Russell, Daniel Dewey, and Max Tegmark. *Research Priorities for Robust
       and Beneficial Artificial Intelligence*. The Future of Life Institute, 2015. URL: http:
       //futureoflife.org/static/data/documents/research_priorities.pdf.

[18]   Max Tegmark. *Research Priorities for Robust and Beneficial Artificial Intelligence:
       an Open Letter*. Future of Life Institute. 2015. URL: http://futureoflife.org/
       misc/open_letter (visited on 02/09/2016).