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

Andrew Critch
Machine Intelligence Research Institute
critch@intelligence.org

Abstract

Löb's Theorem makes 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 this paper introduces a bounded version of Löb's Theorem which is applicable given such resource limitations. These results have 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.

1 Introduction

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

$$\Box(\Box p \to p) \to \Box p.$$

This result defies the intuition that we might soundly prove the "self-trust" statement that *if we prove p, then p is true.* Indeed, when p is the statement "$1 = 0$", Löb's Theorem reduces to Gödel's First Incompleteness Theorem: that arithmetic cannot prove its own consistency.

If we would ever hope to apply Löb's Theorem to the analysis of real-world algorithms which analyze other algorithms, such as compilers or formal verification software, a version taking into account computational resource bounds is needed. In particular, given any such system, there will be some bound (measured in characters) on the length of the longest proof that it can write or validate.

Some need has already arisen for a resource-bounded version of Löb's theorem: in the game theory of agents who can read on another's source code. Work of Bárász and LaVictoire et al. [1][2] have found that Löb's Theorem can be be used to design logical entities that resemble "agents" which achieve robust cooperative equilibria in games such as the Prisoner's Dilemma. However, these "modal agents" were defined as uncomputable logical functions, and to exhibit computable versions, a resource-bounded version of Löb's Theorem is required.

Previous work of Pudlák [3] on provability of finitistic consistency statements were suggestive that such a result might be possible

elaborate, and in fact this paper proves a version that suits both of the above use cases:

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*

Research supported by the Machine Intelligence Research Institute (intelligence.org).

asymptotic lower bound. Then $\exists \hat{k}$ *:*

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

Outline. Sections 2 through 4 are devoted to proving this bounded generalization of Löb's Theorem, and Sections **??** and 6.1 explore its consequences for the robust cooperation of bounded agents and for the "self-trust" of bounded logical proof systems.

2 Fundamentals

Since this paper draws from work in several disciplines, this section is provided to clarify the use of notation and conventions throughout.

Proof length conventions 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, proof length will be measured 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 [3].

Throughout this paper, S refers to a fixed proof system (e.g. an extension of Peano Arithmetic). Writing

$$S \underset{n}{\vdash} \phi, \quad \text{or simply} \quad \underset{n}{\vdash} \phi$$

means that there exists an S-proof of ϕ using n or fewer characters.

Proof system. Let S be any first-order proof system that

1) can represent computable functions in the sense of Section 2,

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

3) allows the definition and use 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.

For concreteness, an illustration of a proof using abbreviations is given in the Appendix.
Abbreviations are allowed in the 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 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, one does not need 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 $\lg N$.

Write

$\mathrm{Lang}(S)$ for the language of S,

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

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

Choosing a Gödel encoding. Along with the proof system S, a single Gödel

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

is chosen and fixed throughout, as well as "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 it is 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.

Representing computable functions. It is known (see, e.g. Theorem 6.8 of Cori and Lascar, Part II [4]) 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

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

It is 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, given functions f, g and h, to say that S proves that for any x value, $f(x) < g(x) + h(x)$, technically one 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 one can 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 to the reader.

Asymptotic Notation. The notation $f \prec g$ will mean that for any $M \in \mathbb{N}$, there exists and $N \in \mathbb{N}$ such that $\forall n > N, Mf(n) < g(n)$. The expression $\mathcal{O}(g)$ stands for the set of functions $f \preceq g$, and for any specific function $\mathcal{E}$, $\mathcal{E}(\mathcal{O}(g))$ will stand for the set of functions of the form $\mathcal{E} \circ f$ where $f \in \mathcal{O}(g)$.

3 A Parametric Diagonal Lemma

To reason about computer systems with certain as-yet unset parameters, such as memory constraints, a generalization of the Diagonal Lemma is needed for formulas with free variables to represent those parameters in a way that avoids writing a separate proof for every instance of the parameters.

Lemma 1 (Parametric Diagonal Lemma)**.** *Suppose S is a first-order theory capable of representing all computable functions, as in Section 2. Then for any predicate $G \in \mathrm{Lang}_{r+1}(S)$, there exists a predicate $\psi \in \mathrm{Lang}_r(S)$ such that:*

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

Since the first draft of this paper, it has been pointed out[1] that this result can be found on p. 53 of [5]. However, since the proof exposited here is quite short, and simpler to follow in the context of this paper, is it kept here for the purpose of self-containment:

Proof. 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 $\mathrm{Lang}(S)$, 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 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 that

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

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

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

□

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

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

It is taken 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.)

[1] Thanks to Professor Dana Scott.

Then one can define a "bounded" box operator:

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

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

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

Since $Eval_1$ is computable, it can be represented in $\text{Lang}(S)$ as in Section 2. 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, these properties are given English names to recall them.

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

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

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, one needs only to 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. □

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

$$\begin{aligned} & \vdash \Box_N\ (\forall k \phi(k)) \\ \Rightarrow\ & \vdash \forall k\ \Box_{C+2N+\lg k} \phi(k), \textit{ which in turn} \\ \Rightarrow\ & \vdash \forall k\ \Box_{\mathcal{O}(\lg k)} \phi(k) \end{aligned}$$

Proof. An encoded proof of $\phi(^\circ K)$ 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 $^\circ K$ in place of k at the end. To avoid repeating $^\circ K$ numerous times in the final line (in case it is large), an abbreviation will be used for ϕ. Thus the appended lines can say:

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

(2) $\Phi(^\circ K)$

Let us analyze how many characters are needed to write such lines. First, a string Φ is needed 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 one 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 $^\circ K$. 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

$$\mathcal{E} : \mathbb{N} \to \mathbb{N}$$

bounding 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)**.** $\forall \phi \in \text{Lang}(S)$,

$$\vdash_k \phi \tag{5.1}$$

$$\Rightarrow \quad \vdash_{\mathcal{E}(k)} \Box_k \phi \tag{5.2}$$

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

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

Achieving $\mathcal{E}(k) \in \mathcal{O}(k)$**.** How large must $\mathcal{E}$ be in practice? Gödel numberings for sequences of integers can be achieved in $\mathcal{O}(k)$ space [6] (where k is the length of a standard binary encoding of the sequence), as can Gödel numberings of term algebras [7]. 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, one can straightforwardly achieve $\mathcal{E}(k) \in \mathcal{O}(k)$ 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 there is a threshold* $\hat{k} \in \mathbb{N}$*, depending on* $p(-)$*, such that*

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

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

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 $G(-,-) \in \text{Lang}_2(S)$ by

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

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

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

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

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

By Bounded Necessitation,

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

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

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

in which we can specialize to the forward implication,

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

By Implication Distribution of $\Box_{\mathcal{O}(\lg k)}$,

$$\vdash \forall k \forall a\ \Box_a \psi(k) \rightarrow \Box_{a+\mathcal{O}(\lg k)} G(\ulcorner\psi\urcorner, k)$$

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

$$\vdash \forall k \forall a \forall b\ \Box_a \psi(k) \rightarrow \left(\Box_b \Box_{g(k)} \psi(k) \rightarrow \Box_{a+b+\mathcal{O}(\lg k)} p(k)\right)$$

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

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

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

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

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

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

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

$$\vdash \forall k \forall a\ \Box_a \psi(k) \rightarrow \Box_{\mathcal{E}(a)} \Box_a \psi(k)$$

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

$$\vdash \forall k\ \Box_{g(k)} \psi(k) \rightarrow \Box_{\mathcal{E}(g(k))} \Box_{g(k)} \psi(k)$$

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

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

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

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

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 $\Box_N$,

$$\vdash \forall k > k_2,\ \Box_{C+2N+\lg k}\psi(k)$$

and since $C + 2N + \lg k < g(k)$ 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. □

6 Applications

6.1 An Obstacle to Logical Self-Trust

Löb's Theorem can be viewed as an obstacle to a formal system of logic "trusting itself" to soundly prove any statement p. Intuitively, we might like it if a system could "trust itself about p" by proving that "If I prove p, then p is true." But this is just the hypothesis $\vdash \Box p \to p$ of Löb's Theorem, whence the conclusion $\Box p$ follows, which is undesirable (makes our system unsound) if p turns out to actually be false.

Previously, one might have thought this obstacle was merely a quirk of infinities arising from the the unbounded proof-existence predicate $\Box$. However, we see now that some bounded obstacle remains: namely, that a bounded logical system cannot trust itself "about moderately long proofs in general". To see this interpretation, let $p(k)$ be any statement with a free parameter k, and $f(k) \succ \lg(k)$ be any function, representing "moderate largeness". Then the hypothesis $\vdash \forall k, \Box_{f(k)}p(k) \to p(k)$ of Parametric Bounded Löb's Theorem says that our logical system generally trusts its proofs about $p(k)$, even if they are moderately long. However, this will imply that $\vdash \forall k > \hat{k}, p(k)$, which is bad news if $p(k)$ is sometimes false.

6.2 Application: Robust Cooperation of Bounded Agents in the Prisoner's Dilemma

Bárász, LaVictoire and others [1][2] 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 $G : \mathbb{N} \to \mathbb{N}$ is a function to be specified later:

```
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 [8] and [9] all involved cooperation based on *equality of programs*, a very fragile condition. 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 for any opponent Opp, we have

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

where $\text{LengthOf}(Opp)$ denotes the number of symbols in the opponent's source code.

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 need not exceed length $k+G(\text{LengthOf}(Opp))$. The agents FairBot_k defined above are G-fair (where G is the function appearing in line 2 of their source code), and the reader is encouraged to keep them in mind as a motivating example for the following result:

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

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

- $G(\ell)$ *is any non-decreasing function satisfying* $G(\ell) \succ \mathcal{E}(\mathcal{O}(\ell))$.

Then, for any G-fair agents A_k *and* B_k, $\exists r \gg 0$ *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 $\mathcal{E}(k) \in \mathcal{O}(k)$ 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 of Theorem 4. 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

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

For later convenience, we also choose a non-decreasing computable function

$$f(k) \succ \mathcal{E}(\mathcal{O}(lg(k)))$$

such that

$$6f(2^\ell) \leq G(\ell).$$

For example, we could take $f(k) = \lfloor G(\lfloor \lg k \rfloor)/6 \rfloor$.

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

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

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,

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

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

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

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

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

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

Thus by (6.5),

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

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

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 $m, n > \hat{k} + 1$,

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

□

7 Summary

We have exhibited a version of Löb's Theorem which can be applied to algorithms which read and write proofs using bounded computational resources, such as formal verification software. 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 ([8]).

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. [1][2] 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.

A Inspiration: a modal proof of Löb's Theorem

The proof of our bounded version of Löb's Theorem was inspired by a particular proof of Löb's original theorem using a few simple properties of $\Box$, along with the classical Diagonal Lemma [10].[2]

- **Necessitation:** From $\vdash A$ conclude $\vdash \Box A$: Informally, this says that if A is a theorem, then it is provable.
- **Internal necessitation:** $\vdash \Box A \to \Box\Box A$: If A is provable, then it is provable that it is provable.
- **Box distributivity:** $\vdash \Box(A \to B) \to (\Box A \to \Box B)$: This rule allows applications of modus ponens inside the provability operator. If it is provable that A implies B, and A is provable, then B is provable.
- **Diagonal Lemma:** for any formula $F(-) \in \text{Lang}_1(S)$ (having one free variable), there exists a sentence $\psi \in \text{Lang}_0(S)$ (with no free variables) such that

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

[2]Thanks to Professor Leo Harrington for the advice to include this proof as a way of motivating the remainder of the paper.

Proof of Löb's Theorem:

1. Suppose $\vdash \Box p \to p$.
   Define $F(x) \in \text{Lang}_1(\mathcal{PA})$ by $F(\ulcorner\psi\urcorner) = (\Box\psi \to p)$ for any $\psi \in \text{Lang}_0(\mathcal{PA})$ and $F(x) = 0$ otherwise. From the Diagonal Lemma, F has a fixed point ψ:
2. $\vdash \psi \leftrightarrow (\Box\psi \to p)$. In particular, we have the forward implication:
3. $\vdash \psi \to (\Box\psi \to p)$. By the necessitation rule:
4. $\vdash \Box(\psi \to (\Box\psi \to p))$. By the box distributivity rule:
5. $\vdash \Box\psi \to \Box(\Box\psi \to p)$. Box distributivity with $A = \Box\psi$ and $B = p$ gives:
6. $\vdash \Box(\Box\psi \to p) \to (\Box\Box\psi \to \Box p)$. From this and 5 we have:
7. $\vdash \Box\psi \to (\Box\Box\psi \to \Box p)$. By the internal necessitation rule:
8. $\vdash \Box\psi \to \Box\Box\psi$. From this and 7:
9. $\vdash \Box\psi \to \Box p$. From this and 1:
10. $\vdash \Box\psi \to p$. From 2:
11. $\vdash (\Box\psi \to p) \to \psi$. From this and 10:
12. $\vdash \psi$. Thus, by the necessitation rule:
13. $\vdash \Box\psi$. Finally, from this and 10 we have:
14. $\vdash p$.

□

B String abbreviations in proofs

In section 2, we required that our proof system allow the definition and use of abbreviations, because real-world proof systems (such as MetaMath [11]) do this, and because it makes our analysis easier. Abbreviations are just string substitutions which must be replaced by their previously defined values when reading a proof. For the sake of concreteness, we illustrate this with an example.

Recall the axioms of Peano Arithmetic in first-order logic (FOL):

$$\begin{aligned}
P1: &\quad \forall x(Sx \neq 0)\\
P2: &\quad \forall x\forall y(Sx = Sy \to x = y)\\
P3: &\quad \forall x(x + 0 = x)\\
P4: &\quad \forall x\forall y(x + sy = s(x + y))\\
P5: &\quad \forall x(x \cdot 0 = 0)\\
P6: &\quad \forall x\forall y(x \cdot Sy = (x \cdot y) + x)
\end{aligned}$$

along with, for every formula $A(x, \bar{y}) = A(x, y_1, \ldots, y_k)$, the induction axiom

$$Ind(A): \quad \forall y_1 \ldots y_k[(A(0, \bar{y}) \text{ and } (\forall x(A(x, \bar{y}) \to A(Sx, \bar{y}))) \to \forall x A(x, \bar{y}))]$$

Below is a proof of the associativity of addition, which does not use abbreviations. Some sections involving first-order logic are summarized in a single step,

but this is not necessary. This proof uses the induction axiom for the formula $A(z) = ((x + y) + z = x + (y + z))$:

By P3:
$$(x + y) + 0 = x + y \tag{B.1}$$
By P3:
$$(x + y) + 0 = x + (y + 0) \tag{B.2}$$
By FOL (substitution):
$$(x + y) + z = x + (y + z) \rightarrow S((x + y) + z) = S(x + (y + z)) \tag{B.3}$$
By P4:
$$S((x + y) + z) = (x + y) + Sz \tag{B.4}$$
By FOL (substitution of B.4 in B.3):
$$(x + y) + z = x + (y + z) \rightarrow (x + y) + Sz = S(x + (y + z)) \tag{B.5}$$
By P4:
$$S(x + (y + z)) = x + S(y + z) \tag{B.6}$$
By P4:
$$S(y + z) = y + Sz \tag{B.7}$$
By FOL (substitution of B.7 in B.6):
$$S(x + (y + z)) = x + (y + Sz) \tag{B.8}$$
By FOL (substitution of B.8 in B.5):
$$(x + y) + z = x + (y + z) \rightarrow (x + y) + Sz = x + (y + Sz) \tag{B.9}$$
By $Ind(A(z))$ from B.9:
$$\forall x \forall y \forall z((x + y) + z = x + (y + z)) \tag{B.10}$$

Part of this proof can be made slightly shorter (in characters) using an abbreviation; see below. An abbreviation, as it is meant here, is nothing more than a literal string to be replaced by another string upon reading and checking the proof; it has no particular type or grammatical role in the proof language aside from that. If we chose to give more structure to the types of abbreviations that are allowed, we could make the checking of our proofs more efficient, and indeed, this is done in real-world proof systems like MetaMath [11]. This restriction is not needed for the proofs in this paper, however, so we allow abbreviations to be any literal string substitution for simplicity.

Here is essentially the same proof as a above, using an abbreviation, "$\%HYP$", defined on the third line of the proof:

By P3:

$$(x + y) + 0 = x + y \tag{B.11}$$

By P3:

$$(x + y) + 0 = x + (y + 0) \tag{B.12}$$

Define abbreviation:

$$\%HYP == \text{“}(x + y) + z = x + (y + z)\text{”} \tag{B.13}$$

By FOL (substitution):

$$\%HYP \rightarrow S((x + y) + z) = S(x + (y + z)) \tag{B.14}$$

By P4:

$$S((x + y) + z) = (x + y) + Sz \tag{B.15}$$

By FOL (substitution of B.15 in B.14):

$$\%HYP \rightarrow (x + y) + Sz = S(x + (y + z)) \tag{B.16}$$

By P4:

$$S(x + (y + z)) = x + S(y + z) \tag{B.17}$$

By P4:

$$S(y + z) = y + Sz \tag{B.18}$$

By FOL (substitution of B.18 in B.17):

$$S(x + (y + z)) = x + (y + Sz) \tag{B.19}$$

By FOL (substitution of B.19 in **??**):

$$\%HYP \rightarrow (x + y) + Sz = x + (y + Sz) \tag{B.20}$$

By Ind(A(z)) from B.20:

$$\forall x \forall y \forall z((x + y) + z = x + (y + z)) \tag{B.21}$$

References

[1] Mihály Bárász et al. “Robust Cooperation in the Prisoner’s Dilemma. Program Equilibrium via Provability Logic”. In: (2014). arXiv: `1401.5577 [cs.GT]`.

[2] 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.

[3] 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, pp. 547–637.

[4] 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.

[5] George Boolos. *The Logic of Provability*. New York: Cambridge University Press, 1993.

[6] Shi-Chun Tsai, Jen-Chun Chang, and Rong-Jaye Chen. “A Space-efficient Gödel Numbering with Chinese Remainder Theorem”. In: *19th Workshop on Combinatorial Mathematics and Computation Theory*. 2002, pp. 192–195.

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

[8] Moshe Tennenholtz. “Program Equilibrium”. In: *Games and Economic Behavior* 49.2 (2004), pp. 363–373.

[9] Lance Fortnow. “Program Equilibria and Discounted Computation Time”. In: *TARK ’09: 12th Conference on Theoretical Aspects of Rationality and Knowledge*. New York: ACM Press, 2009, pp. 128–133.

[10] Rudolf Carnap. *Logische Syntax der Sprache*. Vol. 8. Schriften zur Wissenschaftlichen Weltauffassung. Springer Berlin Heidelberg, 1934.

[11] Norman Megill. *Metamath, A Computer Language for Pure Mathematics*. Morrisville, NC, USA: Lulu Press, 2007.