The Journal of Symbolic Logic
Volume 84, Number 4, December 2019

A PARAMETRIC, RESOURCE-BOUNDED GENERALIZATION OF LÖB'S THEOREM, AND A ROBUST COOPERATION CRITERION FOR OPEN-SOURCE GAME THEORY

ANDREW CRITCH

Abstract. This article presents two theorems: (1) a generalization of Löb's Theorem that applies to formal proof systems operating with bounded computational resources, such as formal verification software or theorem provers, and (2) a theorem on the robust cooperation of agents that employ proofs about one another's source code as unexploitable criteria for cooperation. The latter illustrates a capacity for outperforming classical Nash equilibria and correlated equilibria, attaining mutually cooperative program equilibrium in the Prisoner's Dilemma while remaining unexploitable, i.e., sometimes achieving the outcome (Cooperate, Cooperate), and never receiving the outcome (Cooperate, Defect) as player 1.

§1. Introduction. In the game theoretic analysis of computerized agents who may read one another's source code, and more generally in the analysis of program verification systems that use proofs to verify other program verification systems, a need has arisen for a version of Löb's Theorem that applies to proof systems with bounded computational resources. The first main theorem of this article is a generalization of Löb's Theorem that applies in such cases, developed and proven in Section 2 through Section 4.

The second main theorem, presented in Section 5, makes use of the first theorem and some further proof-theoretic analysis to derive some implications for the game theory of agents who read one another's source code. Specifically, game theoretic work of Bárász, Christiano, Fallenstein et al. [1] and LaVictoire, Fallenstein, Yudkowsky et al. [5] found that Löb's Theorem can be used to design entities in modal logic that resemble "agents" who achieve robust cooperative equilibria in games such as the Prisoner's Dilemma. However, these so-called "modal agents" were defined as strings representing uncomputable functions; strings of the form "if $(\cdots)$ is provable, return 1, else return 0". It remained open whether their results would arise naturally for computable agents, a question answered affirmatively in Section 5.

1.1. Related work. The work of Pudlák [7] on the lengths of proofs of finitistic consistency statements are suggestive of the first theorem of this article. Specifically,

Received July 10, 2016.

2010 *Mathematics Subject Classification.* 03B70, 03B45, 03F03, 68T27, 91A35, 62C99, 91A05, 91A06, 91A10.

Key words and phrases. Löbian cooperation, bounded rationality, program equilibrium, proof length, Prisoner's Dilemma.

0022-4812/19/8404-0004
DOI:10.1017/jsl.2017.42

Pudlák shows that, for a consistent system, there exists some $\varepsilon > 0$ such that for all N, any proof of a statement of the form "there does not exist a proof of $\bot$ using N or fewer characters" must use at least N^{ε} characters. This means there is some obstruction to self-trust for a resource-bounded proof system, which suggests that a resource-bounded version of Löb's Theorem—a generic statement of self-trust about a family of sentences parametrized by a resource bound—might also hold.

§**2. Fundamentals.** Since this article draws from work in several disciplines, this section is provided to clarify the use of notation and conventions throughout.

2.1. Proof length conventions and notation. 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 [8].

Throughout this article, 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.

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

1) can represent computable functions (e.g., by being an extension of $\mathcal{PA}$; see Section 2.4),
2) can write any number $k \in \mathbb{N}$ using $\mathcal{O}(\lg(k))$ symbols (e.g., using binary), and
3) allows the definition and use of abbreviations during proofs (see the Appendix for details).

Compact numeral representations and abbreviations are allowed in the proof system for two reasons. The first is that real-world automated proof systems will tend to use these because of memory constraints. The second is that abbreviations make the lengths of shortest proofs 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 the contribution of its size to the proof length; the contribution will simply be linear in its length, or $\lg(N)$.

Write

L_S for the language of S,
$L_S(r)$ for the set of formulas in L_S with r free variables, and
$\mathrm{Const}(S)$ for the set of closed-form constant expressions in S (e.g., 0, $\mathcal{S}0$, $\mathcal{S}0 + \mathcal{S}0$, etc.).

When $\phi \in L_S(r)$, given any closed-form expressions $c_1, \ldots, c_r$ (such as constants, or free variables), write

$$\phi[\bar{c}] = \phi[c_1, \ldots, c_r]$$

for the result of substituting the c_i for the free variables in ϕ.

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

$$\#(-) : L_S \to \mathbb{N}$$

is chosen and fixed throughout, as well as a "numeral" mapping

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

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

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

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

which allows S to write proofs about itself.

2.4. Representing computable functions. It is assumed that for any computable function $f : \mathbb{N} \to \mathbb{N}$, there exists a "graph" formula $\Gamma_f \in L_S(2)$ such that

$$\text{for all } x \in \mathbb{N},\ S \vdash (\forall y)\big(\Gamma_f[^\circ x, y] \leftrightarrow y = {}^\circ f(x)\big).$$

For example, this condition holds if S is an extension of $\mathcal{PA}$; see, e.g., Theorem 6.8 of Cori and Lascar ([3], Part II).

Sometimes we abuse notation and write symbols for computable functions in-line with logical expressions to save space. 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

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

but instead, we abuse notation and write

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

2.5. Asymptotic notation. The notation $f \prec g$ will mean that for any $M \in \mathbb{N}$, there exists an $N \in \mathbb{N}$ such that for all $n > N$, $Mf(n) < g(n)$. The expression $\mathcal{O}(g)$ stands for the set of functions $f \preceq g$. If $f : \mathbb{N} \to \mathbb{N}$ is any specific function, $f(\mathcal{O}(g))$ will stand for the set of functions of the form $e \circ f$ where $f \in \mathcal{O}(g)$.

§3. A bounded provability predicate, $\Box_k$. Here a predicate $\Box_k$ is defined for asserting provability using a proof with length bounded by k.

3.1. Defining $\Box_k$. Given a choice of Gödel encoding for Peano Arithmetic, it is classical that a formula $Bew[m, n] \in L_S(2)$ exists that 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 $\Box : L_{\mathcal{PA}} \to L_{\mathcal{PA}}$ can be defined as

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

It is taken for granted that $Bew[m, n]$ exists for S and can be extended to a "bounded Bew" formula $BBew[m, n, k] \in L_S(3)$ that means

- m encodes a proof in S,
- n encodes the statement it proves, and
- the proof encoded by m, before being encoded as m, uses at most k characters when written in the language of S. (Note that in general, m itself will be much larger than k as a member of $\mathbb{N}$.)

Then one can define a "bounded" box operator, which, given any sentence ϕ and closed expression k (such as a free variable, or a constant), returns

$$\Box_k(\phi) := (\exists m)(BBew[m, \ulcorner\phi\urcorner, k]).$$

Also taken for granted is a computable "single variable evaluation" function, $Eval_1 : \mathbb{N} \to \mathbb{N}$, such that for any $\phi \in L_S(1)$,

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

Since $Eval_1$ is computable, it can be represented in L_S as in Section 2.4. This allows us to extend the $\Box_k$ operator to act on sentences with one unbound variable. Specifically, if ϕ is a sentence with an unbound variable ℓ, then

$$\Box_k(\phi) := (\exists m)\big(BBew[m, Eval_1(\ulcorner\phi\urcorner, \ell), k]\big).$$

Then $\Box_k(\phi)$ itself has ℓ as an unbound variable, and in natural language stands for "There is a proof using k or fewer characters of the formula ϕ".

3.2. Basic properties of $\Box_k$. Each of the following properties will be needed multiple times during the proof of the main results. 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 $p, q \in L_S$,

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

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, it may be that they were 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 $\phi \in L_S(1)$,

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

In the final subscript, $\mathcal{O}(\lg(k))$ stands for a closed expression representing a function of k that is asymptotically less than some positive constant times $\mathcal{O}(\lg(k))$.

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 read as follows:

(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), one can achieve $\text{Length}(\Phi) < \frac{N}{2}$. As well, some constant c number of characters are needed to write out the system's equivalent of "let", "stand for", "$\ulcorner$", and "$\urcorner$". Finally, $\lg(K)$ characters are needed to write $^\circ K$. Altogether, the proof was extended by $C + N + \lg(k)$ characters, for a total length of $2N + c + \lg(k)$. ⊣

§4. A bounded generalization of Löb's Theorem. This section exhibits the first main theorem: a generalization of Löb's Theorem applicable to the analysis of resource-bounded proof systems.

DEFINITION 4.1 (Proof expansion function). Let

$$e : \mathbb{N} \to \mathbb{N}$$

be any computable function bounding the expansion of S-proof lengths when they are Gödel encoded. That is, its definition is only that it must be large enough to satisfy the following two properties:

PROPERTY 3 (Bounded Necessitation). For all $\phi \in L_S$,

$$\vdash_k \phi \tag{4.1}$$

$$\Rightarrow \quad \vdash_{e(k)} \Box_k(\phi). \tag{4.2}$$

PROPERTY 4 (Bounded Inner Necessitation). For any $\phi \in L_S$,

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

Then the following theorem holds:

THEOREM 4.2 (Resource-bounded generalization of Löb's Theorem). *Let $p[k] \in L_S(1)$ be a formula with a single unquantified variable k, and suppose that $f : \mathbb{N} \to \mathbb{N}$ is computable and satisfies $f(k) \succ e(\mathcal{O}(\lg(k)))$. Then there is a threshold $\hat{k} \in \mathbb{N}$, depending on $p[k]$, such that*

$$\vdash (\forall k)\big(\Box_{f(k)}(p[k]) \to p[k]\big)$$

$$\Rightarrow \quad \vdash \Big(\forall k > \hat{k}\Big)(p[k]).$$

Note: In fact a weaker statement,

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

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.

The proof of Theorem 4.2 makes use of a version of Gödel's diagonal lemma that allows free variables to appear in the formula being diagonalized:

PROPOSITION 4.3. *Suppose S is a first-order theory capable of representing all computable functions, as in Section* 2.4. *Then for any formula $G \in L_S(r+1)$, there exists a formula $\psi \in L_S(r)$ such that*

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

where $\bar{k} = (k_1, \ldots, k_r)$ are free variables.

PROOF. This result can be found on p. 53 of Boolos [2]. ⊣

PROOF OF THEOREM 4.2. (*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 $e(g(k)) \prec f(k)$. For example, we could take $g(k) = \lfloor \sqrt{(\lg(k))(e^{-1}(f(k)))} \rfloor$. Define a formula $G \in L_S(2)$ by

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

so that for any $\phi \in L_S(1)$,

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

Now, by Proposition 4.3, there is some $\psi \in L_S(1)$ such that in some number of characters n,

$$\underset{n}{\vdash} (\forall k)(\psi[k] \leftrightarrow G[\ulcorner \psi \urcorner, k]). \tag{4.3}$$

By Bounded Necessitation,

$$\vdash \Box_n((\forall k)(\psi[k] \leftrightarrow G[\ulcorner \psi \urcorner, k])).$$

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

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

in which we can specialize to the forward implication,

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

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

$$\vdash (\forall k)(\forall a)\left(\Box_a(\psi[k]) \to \Box_{a+\mathcal{O}(\lg(k))}(G[\ulcorner \psi \urcorner, k])\right).$$

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

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

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 $e(g(k)) \prec h(k) \prec f(k)$, for example, $h(k) = \lfloor \sqrt{f(k)e(g(k))} \rfloor$:

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

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

$$\vdash (\forall k > k_1)\big(\Box_{g(k)}(\psi[k]) \to \big(\Box_{h(k)}\big(\Box_{g(k)}(\psi[k])\big) \to \Box_{f(k)}(p[k])\big)\big).$$

Now, by hypothesis, $\vdash (\forall k)\big(\Box_{f(k)}(p[k]) \to p[k]\big)$, thus

$$\vdash (\forall k > k_1)\big(\Box_{g(k)}(\psi[k]) \to \big(\Box_{h(k)}\big(\Box_{g(k)}(\psi[k])\big) \to p[k]\big)\big). \tag{4.4}$$

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

$$\vdash (\forall k)(\forall a)\big(\Box_a(\psi[k]) \to \Box_{e(a)}(\Box_a(\psi[k]))\big).$$

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

$$\vdash (\forall k)\big(\Box_{g(k)}(\psi[k]) \to \Box_{e(g(k))}\big(\Box_{g(k)}(\psi[k])\big)\big).$$

Now, since $e(g(k)) < h(k)$ after some bound $k > k_2$, we have

$$\vdash (\forall k > k_2)\big(\Box_{g(k)}(\psi[k]) \to \Box_{h(k)}\big(\Box_{g(k)}(\psi[k])\big)\big). \tag{4.5}$$

Next, from Equations 4.4 and 4.5, assuming we chose $k_2 \geq k_1$ for convenience, we have

$$\vdash (\forall k > k_2)\big(\Box_{g(k)}(\psi[k]) \to p[k]\big). \tag{4.6}$$

But from Equation 4.3, 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 $\Box_N$,

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

and since $C + 2N + \lg(k) < g(k)$ after some bound $k > \hat{k}$, taking $\hat{k} \geq k_2$ for convenience, we have

$$\vdash \left(\forall k > \hat{k}\right)\big(\Box_{g(k)}(\psi[k])\big). \tag{4.7}$$

Finally, from Equations 4.6 and 4.7 we have, as needed,

$$\vdash \left(\forall k > \hat{k}\right)(p[k]).$$

⊣

4.1. Interpretation. Löb's Theorem may be viewed as an obstacle to a formal system of logic "trusting itself" to soundly prove any statement p. Previously, one might have thought this obstacle was merely a quirk of infinities arising from 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 $f(k) \succ e(\mathcal{O}(\lg(k)))$ be any function, representing

"moderate largeness." Then the hypothesis $\vdash (\forall k)\big(\Box_{f(k)}(p[k]) \to p[k]\big)$ of Theorem 4.2 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.

4.2. Making $f(k)$ small. The statement of Theorem 4.2 becomes stronger as one makes the function $f(k)$ smaller, but it must remain $\succ e(\mathcal{O}(\lg(k)))$ for the theorem to apply. The obstruction to making $f(k)$ small is hence the size of the proof expansion function e, which in real-world software for writing proofs about proofs will be under some design pressure to be made small, to manage computational resources.

How small can e be made in practice? Gödel numberings for sequences of integers can be achieved in $\mathcal{O}(n)$ space (Tsai, Chang, and Chen, [11]) (where n is the length of a standard binary encoding of the sequence), as can Gödel numberings of term algebras (Tarau, [9]). 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 $e(k) \in \mathcal{O}(k)$ for real-world theorem-provers. In that case, the condition $f(k) \succ e(\mathcal{O}(\lg(k)))$ amounts only to saying that $f(k) \succ \lg(k)$.

§5. Robust cooperation of bounded proof-based agents in the Prisoner's Dilemma. Bárász, Christiano, Fallenstein et al. [1], LaVictoire, Fallenstein, Yudkowsky et al. [5], and others have exhibited various agent-like logical formulae who can be viewed as playing the Prisoner's Dilemma by basing their "decisions" on proofs about each others' definitions (as strings). In particular, they proffer proof of the opponent's cooperation as an unexploitable condition for cooperation. However, their "agents" are purely mathematical entities who decide whether to cooperate based on undecidable logical conditions. This leaves open the question of whether their results are achievable by real software with bounded computational resources.

So, consider the following program, 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 a proof of length at most B that
                Opponent(FairBot_k) = Cooperate
        if found,
                return Cooperate
        else
                return Defect
```

In this program, the subroutine "search for a proof of length at most B that $(\cdots)$" is defined as a process which searches, in lexicographic order, through all strings of length $\leq B$, checking each string for whether it is a proof of $(\cdots)$. If any string turns

up that is a proof of $(\cdots)$, the search halts and sets `found` $=$ `true`. If no such proof exists, the search continues until the (finite) set of strings of length $\leq B$ have been exhausted, then halts and sets `found` $=$ `false`. In what follows, all that matters is the functional behavior of the proof search procedure: that it sets `found` $=$ `true` if a proof of length $\leq B$ exists, and `found` $=$ `false` otherwise.

The program $\mathrm{FairBot}_k$ may be viewed as a kind of proof-based "agent" that plays the Prisoner's Dilemma in the following sense. Given any string, `Opponent`, representing the source code of another program, we can compute the pair[1]

$$\mathrm{R}(\mathrm{FairBot}_k, \mathrm{Opp}) := \big(\mathrm{FairBot}_k(\mathrm{Opp}), \mathrm{Opp}(\mathrm{FairBot}_k)\big)\,.$$

Viewed as such, $\mathrm{FairBot}_k$ has the desirable property that the outcome

$$\mathrm{R}(\mathrm{FairBot}_k, \mathrm{Opp}) = (\mathrm{Cooperate}, \mathrm{Defect})$$

will never occur: if its opponent defects, then FairBot will find no proof of its opponent's cooperation, so FairBot will itself defect. This property is called being *unexploitable*.

At this point, a natural question arises: what happens when FairBot encounters a copy of itself? That is, what is $\mathrm{FairBot}_k(\mathrm{FairBot}_k)$? Each FairBot will be searching for a proof that the other will cooperate. As such, one might expect to see a bottomless regression that will exhaust the proof bound B. ("The first FairBot must prove that the second FairBot must prove that the first FairBot must prove that. . . .") Thus, it seems like they will find no proof of cooperation, and hence defect.

However, this turns out not to be the case. Letting

$$p[k] := \big(\mathrm{FairBot}_k(\mathrm{FairBot}_k) = \mathrm{Cooperate}\big)\,,$$

it is a direct consequence of Theorem 4.2 that

$$\mathrm{FairBot}_k(\mathrm{FairBot}_k) = \mathrm{Cooperate}.$$

In fact a much stronger claim is true, as the next theorem will show. It will demonstrate a mutually cooperative program equilibrium (in the sense of Tennenholtz [10]) among a wide variety of (unequal) agents, provided only that they employ a certain principle of fairness, as follows.

5.1. ***G*-fairness.** Given a nonnegative increasing function G, we say that an agent (i.e., program) A_k taking a parameter $k \in \mathbb{N}$ is *G-fair* if for any opponent (i.e., program) Opp, we have

$$\vdash \Box_{k+G(\mathrm{LengthOf}(\mathrm{Opp}))}\big(\mathrm{Opp}(A_k) = C\big) \to \big(A_k(\mathrm{Opp}) = C\big)\,,$$

where LengthOf(Opp) is the character length of the opponent's source code.

In words, A_k is G-fair if finding a short proof of its opponent's cooperation is a sufficient condition for A_k to cooperate, when "short" is flexibly defined to be an increasing function of its opponent's complexity, i.e., $k + G(\mathrm{LengthOf}(\mathrm{Opp}))$. The agents $\mathrm{FairBot}_k$ 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:

[1] We must assume that Opp halts and returns either Cooperate or Defect in this case; note that $\mathrm{FairBot}_k$ always halts.

THEOREM 5.1 (Robust cooperation criterion). *Suppose that*

- $e(k)$, *the proof expansion function of our proof system as defined in Section* 4, *satisfies* $e(\mathcal{O}(\lg(k))) \prec k$, *and*
- $G(\ell)$ *is any nondecreasing function satisfying* $G(\ell) \succ e(\mathcal{O}(\ell))$.

Then, for any G-fair agents A_k *and* B_k, *there exists some* $r \gg 0$ *such that for all* $m, n > r$,

$$A_m(B_n) = B_n(A_m) = \text{Cooperate}.$$

5.2. Feasibility of bounds in Theorem 5.1. Before the proof, recall from Section 4.2 that we can achieve $e(k) \in \mathcal{O}(k)$ for automatic proof systems that are designed for easy verifiability, in which case $e(\mathcal{O}(\lg(k))) \prec k$, as needed.

PROOF OF THEOREM 5.1. The proof will make use of Theorem 4.2 at the very end, but first requires some additional proof-theoretic analysis. For brevity we let

$$a(k) := G(\text{LengthOf}(A_k)), \tag{5.1}$$

$$b(k) := G(\text{LengthOf}(B_k)), \tag{5.2}$$

$$\alpha[m,n] := \big(A_m(B_n) = \text{Cooperate}\big), \text{ and} \tag{5.3}$$

$$\beta[n,m] := \big(B_n(A_m) = \text{Cooperate}\big) \tag{5.4}$$

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

$$\vdash \Box_{m+b(n)}(\beta[n,m]) \to \alpha[m,n] \text{ and} \tag{5.5}$$

$$\vdash \Box_{n+a(m)}(\alpha[m,n]) \to \beta[n,m].$$

For later convenience, we also choose a nondecreasing computable function

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

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

$$a(k), b(k) > G(\lg(k)) \geq 6f(k). \tag{5.6}$$

Define an "eventual cooperation" formula:

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

Using Quantifier Distribution once on the definition of $p[k]$,

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

where $C' = C + 2f(k) + \lg(m)$. Applying Quantifier Distribution again,

$$\vdash (\forall k)\big(\Box_{f(k)} p[k] \to (\forall m > k)(\forall n > k)(\Box_{C''}(\alpha[m,n] \text{ and } \beta[n,m]))\big). \tag{5.7}$$

Where $C'' = 3C + 4f(k) + 2\lg(m) + \lg(n)$. Now, for m, n large and $> k$, we have

$$3C + \lg(n) < n \qquad \text{and by (5.6)},$$

$$4f(k) + 2\lg(m) < 6f(m) < a(m).$$

Adding these inequalities yields

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

so for some k_1, from (5.7) we derive

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

Similarly, we also have

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

$$\vdash (\forall k > k_2)\,\big(\Box_{f(k)}(p[k]) \to (\forall m > k)(\forall n > k)\big(\Box_{n+a(m)}(\alpha[m,n]) \text{ and } \Box_{m+b(n)}(\beta[n,m])\big)\big)\,. \quad (5.8)$$

Thus by (5.5),

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

$$\vdash (\forall k > k_2)\big(\Box_{f(k)}(p[k]) \to p[k]\big).$$

Therefore, by Theorem 4.2 (and the note following it), for some $\hat{k}$ we have

$$\vdash \left(\forall k > \hat{k}\right)(p[k]).$$

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

$$A_m(B_n) = B_n(A_m) = \text{Cooperate}. \qquad \dashv$$

Theorem 5.1 interesting for four reasons:

1. *It is surprising.* Löb's Theorem has not been applied much in the setting of game theory, and in fact at the time of writing, 100% of the dozens of mathematicians and computer scientists that the author has asked to guess the output of $\text{FairBot}_k(\text{FairBot}_k)$ have either guessed incorrectly (expecting the proof searches to enter an infinite regress and thus reach their bounds), or given an invalid argument for cooperation (such as "it would be better for the agents to cooperate, so they will").
2. *It is advantageous.* When k is large, FairBot_k outperforms the classical Nash/correlated equilibrium solution (Defect, Defect) to the Prisoner's Dilemma when facing itself (or any other G-fair agent) in a one-shot game with no iteration and no future reputation.
3. *It is unexploitable.* That is, the outcome (Cooperate, Defect) will never occur with a FairBot as player 1. If an opponent will defect against FairBot, FairBot will find no proof of the opponent's cooperation, so FairBot will also defect.
4. *It is robust.* Previous examples of cooperative program equilibria studied by Tennenholtz [10] and Fortnow [4] 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. By contrast, the G-fairness criterion relies only on the provability of the opponent's cooperation, rather than details of its implementation, and therefore establishes mutual cooperation between a broad class of agents.

§6. **Conclusion.** Theorem 4.2 represents a resource-bounded generalization of Löb's Theorem, which can be applied to algorithms that read and write proofs using bounded computational resources, such as formal verification software. Theorem 5.1 makes use of Theorem 4.2, and some additional proof-theoretic analysis, to demonstrate how algorithmic agents who have access to one another's source codes can *inexploitably* achieve cooperative outcomes that out-perform classical Nash equilibria and correlated equilibria. Moreover, the condition for cooperation in Theorem 5.1, called "G-fairness", is more robust than previously known unexploitable cooperative conditions, which depended on literal source-code equality (Tennenholtz, [10]).

As a direction for potential future investigation, it seems likely that other agents described in the purely logical (noncomputable) setting of Bárász, Christiano, Fallenstein et al. [1] and LaVictoire, Fallenstein, Yudkowsky et al. [5] 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.

Appendix A. String abbreviations in proofs. In Section 2.2, we required that our proof system allow the definition and use of abbreviations, because real-world proof systems (such as MetaMath (Megill, [6])) 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):

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

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

$$\begin{aligned}
Ind(A) : (\forall y_1)(\forall y_2)(\cdots)(\forall y_k)\,(&\\
&\big(A(0, \bar{y}) \text{ and } (\forall x)\big(A(x, \bar{y}) \to A(Sx, \bar{y})\big)\big) \to (\forall x)\big(A(x, \bar{y})\big)\big)\,.
\end{aligned}$$

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:

$$(x + y) + 0 = x + y. \tag{A.1}$$

By P3:

$$(x + y) + 0 = x + (y + 0). \tag{A.2}$$

By FOL (substitution):

$$(x + y) + z = x + (y + z) \to S((x + y) + z) = S(x + (y + z)). \tag{A.3}$$

By P4:

$$S((x + y) + z) = (x + y) + Sz. \tag{A.4}$$

By FOL (substitution of A.4 in A.3):

$$(x + y) + z = x + (y + z) \to (x + y) + Sz = S(x + (y + z)). \tag{A.5}$$

By P4:

$$S(x + (y + z)) = x + S(y + z). \tag{A.6}$$

By P4:

$$S(y + z) = y + Sz. \tag{A.7}$$

By FOL (substitution of A.7 in A.6):

$$S(x + (y + z)) = x + (y + Sz). \tag{A.8}$$

By FOL (substitution of A.8 in A.5):

$$(x + y) + z = x + (y + z) \to (x + y) + Sz = x + (y + Sz). \tag{A.9}$$

By $Ind(A(z))$ from A.9:

$$(\forall x)(\forall y)(\forall z)\big((x + y) + z = x + (y + z)\big). \tag{A.10}$$

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 (Megill, [6]). This restriction is not needed for the proofs in this article, 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:

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

By P3:

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

Define abbreviation:

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

By FOL (substitution):

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

By P4:

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

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

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

By P4:

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

By P4:

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

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

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

By FOL (substitution of A.19 in A.16):

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

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

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

REFERENCES

[1] M. Bárász, P. Christiano, B. Fallenstein, M. Herreshoff, P. LaVictoire, and E. Yudkowsky, *Robust cooperation in the prisoner's dilemma*, arXiv preprint, 2014, arXiv:1401.5577.

[2] G. Boolos, ***The Logic of Provability***, Cambridge University Press, New York, 1993.

[3] R. Cori and D. Lascar, ***Mathematical Logic: A Course with Exercises, Part II***, Oxford University Press, New York, 2001.

[4] L. Fortnow, *Program equilibria and discounted computation time*, ***TARK '09: 12th Conference on Theoretical Aspects of Rationality and Knowledge***, ACM Press, 2009, pp. 128–133.

[5] P. LaVictoire, B. Fallenstein, E. Yudkowsky, M. Barasz, P. Christiano, and M. Herreshoff, *Program equilibrium in the prisoner's dilemma via Löb's theorem*, ***Multiagent Interaction without Prior Coordination: Papers from the AAAI-14 Workshop***, AAAI Publications, 2014.

[6] N. Megill, ***Metamath***: ***A Computer Language for Pure Mathematics***, Lulu Press, Morrisville, NC, 2007.

[7] P. Pudlák, *On the lengths of proofs of finistic consistency statements in first order theories*, ***Logic Colloquium '84*** (J. B. Paris, A. J. Wilkie, and G. M. Wilmers, editors), North-Holland, Amsterdam, 1986, pp. 165–196.

[8] ———, *The lengths of proofs*, ***Handbook of Proof Theory*** (S. R. Buss, editor), Studies in Logic and the Foundations of Mathematics, vol. 137, Elsevier, Amsterdam, 1998, pp. 547–637.

[9] P. Tarau, *Bijective size-proportionate Gödel numberings for term algebras*, unpublished manuscript, 2013.

[10] M. Tennenholtz, *Program equilibrium*. ***Games and Economic Behavior***, vol. 49 (2004), no. 2, pp. 363–373.

[11] S.-C. Tsai, J.-C. Chang, and R.-J. Chen, *A space-efficient Gödel numbering with Chinese remainder theorem*, ***Proceedings of the 19th Workshop on Combinatorial Mathematics and Computation Theory***, 2002, pp. 192–195.

MACHINE INTELLIGENCE RESEARCH INSTITUTE
2030 ADDISON STREET, BERKELEY, CA 94704, USA
E-mail: critch@intelligence.org
URL: http://intelligence.org/