Logical Induction and AI-Safety

Computer Science Final Project
Quest University Canada

Maximilian Kahn

November 20, 2018

1 Motivation

As advances in AI lead to widespread use, it has become increasingly important to be able to understand the underlying mechanisms that constitute the AI decision-process. The processes that underpin the current state of AI, deep neural networks, are notoriously opaque (which is to say that we have a very little idea what actually happens during the training process and why machine learning algorithms work so well). [1] Thus, if humanity wishes to ensure its own safety against AI, it is important to have a formal understanding of how AI will generate knowledge and make decisions. In the paper *Logical Induction*, Garrabrant et al. do just this; they formulate a computable algorithm (called the logical inductor) that is able to assign probabilities to logical statements (in a given formal language) and refine those probabilities over time [2]. For example, if the formal language is Peano arithmetic, a logical inductor will assign a probability (that the statement is true) to every arithmetical statement including the twin prime conjecture, the Riemann hypothesis, and Fermat's last theorem. To a human, the fact that there is a 10% chance that the 20,000th digit in the expansion of π is a 6 is intuitive because we have already done the reasoning that there are 10 options and so there is a 1/10 chance that the outcome is a 6. However, an AI does not initially hold this belief. The logical inductor algorithm provides a formal framework for us to understand how an AI could come to assign a 10% probability that the 20,000th digit in the expansion of π is a 6 or that there is a 100% probability that $\sqrt{169} = 13$. Not only does the logical inductor begin with reasonable values for both statements but since the π-statement does not require empirical observation, the logical inductor also learns as time passes to assign a value of 1 or 0 to the π-statement. One could imagine that the logical inductor begins with the initial assumption of 10% but once it computes the 20,000th digit in the expansion of π and finds that it is in fact a 7, then perhaps it assigns a value of 0.01 to the π-statement (note that it does not assign a value of 0 automatically because it is possible that there was a mistake in its computation (imagine a human writing out the digits of π up to 20,000th digit). However, in the limit as the value 7 keeps popping up, the logical inductor can be confident to assign a probability of 0 to the π-statement.

2 The Formal Language of the Logical Inductor

Garrabrant et al. define a formal language that underpins the logical inductor algorithm, however, there is so much terminology that outlining their formal language necessitates its own section. Each of their definitions are listed and then an explanation of the terminology is given,

In this formal language, ϕ, ψ, and χ are used to represent well-formed formulas (these will also synonymously be referred to as *sentences*) in some language of propositional logic $\mathcal{L}$. Then let $\mathcal{S}$ be the set of well-formed formulas (i.e. sentences) in $\mathcal{L}$ and let $\mathcal{T}$ be a set of axioms that can be used to prove theorems in $\mathcal{L}$.

In the course of explaining the components that constitute a logical inductor, the *logical induction criterion* outlined below will become more clear. However, it is important to note before embarking through the rest of the paper that one can come to an intuitive understanding of how the logical inductor works by imagining that the mechanism by which the logical inductor adjusts its beliefs (i.e. probabilities) in logical statements is analogous to how prices are adjusted in a stock market.

Consider the scenario of a stock having varying prices on two stock exchanges A and B. A trader could make arbitrarily large amounts of money by buying the stock for the lower price on exchange A and then selling it for the higher price on exchange B. This process is known as *arbitrage*. However, as demand for the stock goes up on exchange A, the price will correct itself so in actuality no one can cash in indefinitely and we can be confident that the price on exchange A converges to the price on exchange B (and vice versa).

2.0.1 The Logical Induction Criterion

"A market $\overline{\mathbb{P}}$ is said to satisfy the ***logical induction criterion*** *relative to a deductive process $\overline{\mathcal{D}}$ if there is no efficiently computable trader $\overline{\mathrm{T}}$ that exploits $\overline{\mathbb{P}}$ relative to $\overline{\mathcal{D}}$. A market $\overline{\mathbb{P}}$ meeting this criterion is called a* ***logical inductor over*** *$\overline{\mathcal{D}}$."*

To frame this within the stock exchange analogy, we would say that arbitrage is not possible for any trader in a market that satisfies the logical induction criterion.

2.1 Markets

2.1.1 Valuation

"A ***valuation*** *is any function $\mathbb{V} : \mathcal{S} \to [0, 1]$. We refer to $\mathbb{V}(\phi)$ as the value of ϕ according to $\mathbb{V}$. A valuation is called rational if its image is in $\mathbb{Q}$."*

A valuation takes as its input a logical statement and outputs a number between 0 and 1 which we call the valuation of that logical statement. We call a valuation rational if its range is contained in the set of rational numbers.

2.1.2 Pricing

"A ***pricing*** *$\mathbb{P} : \mathcal{S} \to \mathbb{Q} \cap [0, 1]$ is any computable rational valuation. If $\mathbb{P}(\phi) = p$ we say that the price of a ϕ-share according to $\mathbb{P}$ is p, where the intended interpretation is that a ϕ-share is worth 1 if ϕ is true."*

Notice that the pricing function is built upon the valuation function. Indeed, the pricing function is actually just any (computable) rational valuation! The notation $\mathbb{Q} \cap [0, 1]$ simply means that the output of the pricing function is both a rational number (i.e. is in the set of rational numbers, $\mathbb{Q}$) and is between 0 and 1.

As an example of how the pricing function could be used, imagine the logical statement ϕ, where ϕ represents "$2 + 2 = 4$", we could then apply $\mathbb{P}$ to ϕ to obtain $\mathbb{P}(\phi) = 1$. If there is uncertainty as to the truthfulness of the logical statement then the pricing function will assign a value between 0 and 1 which reflects the current best estimate of how likely the statement is to be true.

2.1.3 Market

"A ***market*** *$\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, ...)$ is a computable sequence of pricings $\mathbb{P}_i : \mathcal{S} \to \mathbb{Q} \cap [0, 1]$."*

A market is simply a list of pricings that can change day by day - where we can obtain the "price" or probability of a logical statement by applying the pricing function to a logical statement. So something we could say is that on day n the pricing function assigned to the logical statement, ϕ_1, the probability 0.75, or equivalently, on day n, $\mathbb{P}_1(\phi_1) = 0.75$.

Note the emphasis on *computable sequence* as opposed to simply *sequence*. There are only a countably infinite number of irrational numbers which are computable, while there are an uncountably infinite number of irrational numbers which are not computable. Since Garrabrant et al. are trying to devise a computable algorithm, we don't want to be working with irrational values (which may end up being not computable). So it is important then that we specify that the algorithm will only deal with rational numbers, or, more precisely, that the pricing function will only output rational values between 0 and 1.

2.1.4 Belief State

*"A **belief state** $\mathbb{P} : \mathcal{S} \rightarrow \mathbb{Q} \cap [0, 1]$ is a computable rational valuation with finite support, where $\mathbb{P}(\phi)$ is interpreted as the probability of ϕ (which is 0 for all but finitely many ϕ)."*

Notice that the belief state function is essentially identical to the pricing function (indeed, they are used synonymously). The authors seem to put in this definition to remind us that the semantics we are using (i.e. the stock exchange scenario) is simply one interpretation of the logical inductor. Another interpretation of the number the pricing function spits out is the *probability* of ϕ. *Finite support* simply means that the belief state function only outputs a nonzero value for finitely many logical statements. This suggests that there are an infinite number of logical statements that are not true while there are a finite number of logical statements that are possibly true or are true.

2.1.5 Computable Belief Sequence

*"A **computable belief sequence** $\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, ...)$ is a computable sequence of belief states, interpreted as a reasoner's explicit beliefs about logic as they are refined over time."*

Just as the belief state function is identical to the pricing function, so is the computable belief sequence to the notion of a market. Here, we are to interpret $\overline{\mathbb{P}}$ as a sequence of belief states, which is to say, a list of probabilities, that a given set of logical statements are true. One could see $\overline{\mathbb{P}}$ as the memory centre of the algorithm.

2.2 Deduction processes

2.2.1 Deductive process

*"A **deductive process** $\overline{\mathcal{D}} : \mathbb{N}^+ \rightarrow Fin(\mathcal{S})$ is a computable nested sequence $\mathcal{D}_1 \subseteq \mathcal{D}_2 \subseteq \mathcal{D}_3 ...$ of finite sets of sentences. We write $\mathcal{D}_\infty$ for the union $\bigcup_n \mathcal{D}_n$."*

The authors note here that the above description of a deductive process is a "barren" one. The notation $\mathbb{N}^+ \rightarrow Fin(\mathcal{S})$ indicates that the deductive process, $\overline{\mathcal{D}}$, maps the set of positive natural numbers (that is, [1, 2, 3, 4,...]) to a nested sequence of finite sets of sentences (where $\mathcal{D}_i$ is the i^{th} finite set of sentences) such that $\mathcal{D}_1$ is a subset of $\mathcal{D}_2$, $\mathcal{D}_2$ is a subset of $\mathcal{D}_3$ and so on. The union summation symbol $\bigcup_n \mathcal{D}_n$ means to combine the elements into a super-set which contains every element from each of the sets being combined and where duplicates are only put into the super-set once.

We can imagine the deductive process as being a group of the world's brightest mathematical minds working tirelessly day and night to prove the truthfulness of theorems in our formal language. For example, after a certain amount of time, we would expect that $\overline{\mathcal{D}}$ would tell us whether things like the Riemann hypothesis are true or not.

2.2.2 World

*"A **world** is any truth assignment $\mathbb{W} : \mathcal{S} \rightarrow \mathbb{B}$. If $\mathbb{W}(\phi) = 1$ we say that ϕ is **true in** $\mathbb{W}$. If $\mathbb{W}(\phi) = 0$ we say that ϕ is **false in** $\mathbb{W}$. We write $\mathcal{W}$ for the set of all worlds."*

Notice that the world function is similar to the valuation function except that logical statements are not mapped to all real numbers between 0 and 1; in the case of the world function the mapping is to either 0 or 1. Just like the notion of a market, $\overline{\mathbb{P}}$, which is a sequence of the prices associated with each ϕ in a formal language, $\mathcal{W}$ is a sequence of Boolean values associated with the truth-value of each ϕ in a formal language.

Importantly, logically uncertain reasoners cannot immediately tell whether a truth claim is inconsistent or not and so must refine its beliefs over time to do away with inconsistencies. The notion of "consistency" is defined below.

2.2.3 Propositional Consistency

*"A world $\mathbb{W}$ is called **propositionally consistent**, abbreviated **p.c.**, if for all $\phi \in \mathcal{S}$, $\mathbb{W}(\phi)$ is determined by Boolean algebra from the truth values that $\mathbb{W}$ assigns to the prime formulas of ϕ. In other words, $\mathbb{W}$ is p.c. if $\mathbb{W}(\phi \wedge \psi) = \mathbb{W}(\phi) \wedge \mathbb{W}(\psi), \mathbb{W}(\phi \vee \psi) = \mathbb{W}(\phi) \vee \mathbb{W}(\psi)$, and so on. Given a set of sentences $\mathcal{D}$, we define $\mathcal{PC}(\mathcal{D})$ to be the set of all p.c. worlds where $\mathbb{W}(\phi) = 1$ for all $\phi \in \mathcal{D}$. We refer to $\mathcal{PC}(\mathcal{D})$ as the set of worlds **propositionally consistent with** $\mathcal{D}$.*
*Given a set of sentences $\mathcal{T}$, we will refer to $\mathcal{PC}(\mathcal{T})$ as the set of worlds **consistent with** $\mathcal{T}$, because in this case $\mathcal{PC}(\mathcal{T})$ is equal to the set of all worlds $\mathbb{W}$ such that*

$$\mathcal{T} \cup \{\phi | \mathbb{W}(\phi) = 1\} \cup \{\neg\phi | \mathbb{W}(\phi) = 0\} \nvdash \perp$$

Note that a limited reasoner won't be able to tell whether a given world $\mathbb{W}$ is in $\mathcal{PC}(\mathcal{T})$. A reasoner can computably check whether a restriction of $\mathbb{W}$ to a finite domain is propositionally consistent with a finite set of sentences, but that's about it. Roughly speaking, the definition of exploitation (below) will say that a good reasoner should perform well when measured on day n by worlds propositionally consistent with $\mathcal{D}_n$, and we ourselves will be interested in deductive processes that pin down a particular theory $\mathcal{T}$ by propositional consistency."

In other words, a world, $\mathbb{W}$, is defined to be p.c. if it obeys the properties of Boolean algebra. For example, let ϕ: "$\sqrt{100} = 10$" and ψ: "$\sqrt{36} = 6$", given that $\mathbb{W}(\phi) = 1$ and $\mathbb{W}(\psi) = 1$, then if $\mathbb{W}(\phi \wedge \psi) = \mathbb{W}(\phi) \wedge \mathbb{W}(\psi), \mathbb{W}(\phi \vee \psi) = \mathbb{W}(\phi) \vee \mathbb{W}(\psi)$, and so on are true, we can conclude that $\mathbb{W}$ is p.c. Given the set of sentences $\mathcal{D}$, the terminology, $\mathcal{PC}(\mathcal{D})$ refers to the set of all worlds where $\mathbb{W}(\phi) = 1$ for all ϕ in $\mathcal{D}$.

Now let's look at $\mathcal{PC}(\mathcal{T})$ which is explained as the set of worlds consistent with $\mathcal{T}$ where $\mathcal{T}$ is a set of sentences (recall from earlier that $\mathcal{T}$ is defined to be the axioms of a given formal language). So $\mathcal{PC}(\mathcal{T})$ is equal to the set of all worlds $\mathbb{W}$ such that the union of the sets - $\mathcal{T}$, the set of worlds such that $\mathbb{W}(\phi) = 1$, and the set of worlds such that $\neg\phi | \mathbb{W}(\phi) = 0$, does not contain contradictions.

Finally, a limited reasoner cannot know whether a given world is in the set $\mathcal{PC}(\mathcal{T})$ presumably because they are limited finite time/computational resources. However, the authors do point out that a reasoner could still check that a finite number of logical statements are consistent in $\mathbb{W}$.

2.2.4 $\mathcal{T}$-Complete

*"Given a theory $\mathcal{T}$, we say that a deductive process $\overline{\mathcal{D}}$ is $\mathcal{T}$**-Complete** if*

$$\mathcal{PC}(\mathcal{D}_\infty) = \mathcal{PC}(\mathcal{T})."$$

Recall that the deductive process, $\mathcal{D}_\infty$ is defined as $\bigcup_n \mathcal{D}_n$. The authors give the example of letting $\mathcal{D}_n$ be the set of all theorems in Peano arithmetic which are provable in at most n characters. Then the deductive process, $\overline{\mathcal{D}}$, is said to be PA-complete and a reasoner who has access to $\overline{\mathcal{D}}$ is said to be someone who on day n knows the truth-values associated with all theorems which are at most n characters long. Examples of theorems they may know about include the irrationality of the square root of two (Pythagoras) and the four-square theorem (Langrange).

2.3 Efficient Computability

2.3.1 Efficiently Computable

*"An infinite sequence $\overline{\text{x}}$ is called **efficiently computable**, abbreviated **e.c.**, if there is a computable function f that outputs x_n on input n, with runtime polynomial in n (i.e. in the length of n written in unary)."*

Garrabrant et al. call an infinite sequence, $\overline{\text{x}}$ efficiently computable if there is a computable function, f that outputs x_n on input n with a polynomial runtime. Polynomial runtime refers to the fact that the number of computational steps involved in the calculation scales polynomially with the input. (As a side note, unary refers to the numeric system where each number is represented by the number of ones it represents, for example 2 in unary is "11", and 5 in unary is "11111".)

2.4 Traders

The authors describe a trader as a function which is able to examine the market, $\overline{\mathbb{P}}$, every day and generates a trading strategy based on the market prices of each logical sentence being "traded". For example, the "market order": $4\phi - 5\psi$, can be interpreted as "buy 4 shares of ϕ and sell 5 shares of ψ".

2.4.1 Valuation Feature

*"A valuation **feature** $\alpha : [0,1]^{\mathcal{S}\times\mathbb{N}^+} \to \mathbb{R}$ is a continuous function from valuation sequences to real numbers such that $\alpha(\mathbb{V})$ depends only on the initial sequence $\mathbb{V}_{\leq n}$ for some $n \in \mathbb{N}^+$ called the rank of the feature rank(α). For any $m \geq n$, we define $\alpha(\mathbb{V}_{\geq m})$ in the natural way. We will often deal with features that have range in [0,1]; we call these [0,1]-features. We write $\mathcal{F}$ for the set of all features, $\mathcal{F}_n$ for the set of valuation features of rank $\leq n$, and define an **$\mathcal{F}$-progression** $\overline{\alpha}$ to be a sequence of features such that $\alpha_n \in \mathcal{F}_n$."*

A valuation feature is a function that maps values between the set of numbers between 0 and 1 to the set of real numbers. The valuation feature, α operates on sequences of valuations such that it only depends on the first valuation sequence (which is notated as $\mathbb{V}_{\leq n}$ for some n in the set of natural numbers (not counting zero). The *rank* of the valuation feature is defined to be the value of n.

Then the authors define $\mathcal{F}$ to be the set of all features (which will look like a list of real numbers) and $\mathcal{F}_n$ to be the set of all features of rank n or less. Finally, they define $\overline{\alpha}$ to be a sequence of features such that the n^{th} element in $\overline{\alpha}$ is a member of the set $\mathcal{F}_n$ (which in term is a subset of $\mathcal{F}$).

2.4.2 Price Feature

*"For each $\phi \in \mathcal{S}$ and $n \in \mathbb{N}^+$, we define a **price feature** $\phi^{*n} \in \mathcal{F}_n$.*

$$\phi^{*n}(\overline{\mathbb{V}}) := \mathbb{V}_n(\phi)$$

We call these "price features" because they will almost always be applied to a market $\overline{\mathbb{P}}$, in which case ϕ^{*n} gives the price $\mathbb{P}_n(\phi)$ of ϕ on day n as a function of $\overline{\mathbb{P}}$"

Notice how we are building on the pricing function we encountered earlier. However, here the price feature typically takes the market as input and is therefore able to output the price of a given logical statement as a function of the market prices.

2.4.3 Expressible Feature

*"An **expressible feature** $\xi \in \mathcal{F}$ is a valuation feature expressible by an algebraic expression built from price features ϕ^{*n} for each $n \in \mathbb{N}^+$ and $\phi \in \mathcal{S}$, rational numbers, addition, multiplication, $max(-,-)$, and a "safe reciprocation" function $max(1,-)^{-1}$. We write $\mathcal{EF}$ for the set of all expressible features, $\mathcal{EF}_n$ for the set of expressible feature of rank$\leq n$ and define an **$\mathcal{EF}$-progression** to be a sequence ξ such that $\xi_n \in \mathcal{EF}_n$."*

It seems that the point of defining the *expressible feature* is to have a formal definition of what can be expressed as a feature. Expressible features can be built from price features, as well as rational numbers, basic arithmetical operations and the functions: $max(-,-)$ and $max(1,-)^{-1}$. The authors give the example of the expressible feature $\xi := max(0, \phi^{*6} - \psi^{*7})$. ξ checks whether the price of ϕ is higher on day 6 than the price of ψ is on day 7, if the price is higher, ξ returns the difference, otherwise it returns 0. If say, $\mathbb{P}_6(\phi) = 0.7$ and $\mathbb{P}_7(\phi) = 0.3$ then $\xi(\overline{\mathbb{P}}) = 0.4$. Hopefully it is clear that expressible features return some useful information about the relative prices of logical statements across time and that one can construct expressible features from the aforementioned list of components (arithmetical operations, etc.).

2.4.4 Trading Strategy

*"A **trading strategy for day** n, also called an **n-strategy**, is an affine combination of the form*

$$T = c + \xi_1\phi_1 + ... + \xi_k\phi_k$$

where $\phi_1, ..., \phi_k$ are sentences, $\xi_1, ..., \xi_k$ are expressible features of rank $\leq n$, and

$$c = -\sum_i \xi_i {\phi_i}^{*n}$$

is a "cash term" recording the net cash flow when executing a transaction that buys ξ_i shares of ϕ_i for each i at the prevailing market price. (Buying negative shares is called "selling".) We define $T[1]$ to be c, and $T[\phi]$ to be the coefficient of ϕ in T, which is 0 if $\phi \notin (\phi_1, ..., \phi_k)$.
An n-strategy T can be encoded by the tuples $(\phi_1, ..., \phi_k)and(\xi_1, ..., \xi_k)$ because the c term is determined by them. Explicitly, by linearity we have

$$T = \xi_1 \cdot (\phi_1 - {\phi_1}^{*n}) + \cdots + \xi_k \cdot (\phi_k - {\phi_k}^{*n})$$

which means any n-strategy can be written as a linear combination of $(\phi_1 - {\phi_1}^{*n})$ terms, each of which means "buy one share of ϕ_i at the prevailing price."

An affine combination is simply a linear combination and we can effectively ignore this term since it is self-evident in the first equation. In order to understand the first equation, recall that the linear combination $4\phi - 5\psi$ could be interpreted as the market order: "buy 4 shares of ϕ and sell 5 shares of ψ. Now if ϕ = 30¢ and ϕ = 20¢ then the while trading strategy would have the net value $4 \cdot$ 30¢ $-5 \cdot$ 20¢ = 20¢. Therefore, we could write out the net sum earned using trading strategy T as 20¢. This is the value of c above and note that c is -1 times the sum since negative values are associated with selling shares while positive values are associated with buying shares. So the proposed trading strategy in the new format would be written as: T = 20¢ $+4 \cdot$ 30¢ $-5 \cdot$ 20¢.

Also, note that the coefficients in front of the shares (which informs how many shares should be bought and sold) need not be a number. This makes sense, because we don't want to have the same trading strategy every day. Instead we want our trading strategy to be a function of the market prices. In order to take into account market prices in the trading strategy, the coefficients are actually expressible features which can accomplish that exact task.

The authors give the example of

$$\left[(\neg\neg\phi)^{*5} - \phi^{*5}\right] \cdot (\phi - \phi^{*5}) + \left[\phi^{*5} - (\neg\neg\phi)^{*5}\right] \cdot \left(\neg\neg\phi - (\neg\neg\phi)^{*5}\right)$$

where this trading strategy compares the price of ϕ to the price of $\neg\neg\phi$ on day five. If $\neg\neg\phi$ costs more, then the first expression in the large square brackets is a positive number which indicates that the strategy is suggesting to buy a certain number of ϕ-shares and sell $\neg\neg\phi$-shares. In essence this is arbitrage, where the trader is able to exploit the uneven prices of ϕ and $\neg\neg\phi$ to make money.

2.4.5 Trader

*"A **trader** $\overline{\mathrm{T}}$ is a sequence $(T_1, T_2, ...)$ where each T_n is a trading strategy for day n."*

A trader can be imagined as someone who checks their calendar to see what the date is (the date being the number n in this universe). Then they are given a limited amount of time to work on their computer programs which help them analyze the market. Finally, they come up with a trading strategy for day n. Garrabrant et al. mention that we are often interested in the set of efficiently computable traders (i.e. traders with a polynomial runtime), this makes sense as traders that take longer than a day to construct a trading strategy would get nowhere!

2.4.6 Affine Combination

*"An $\mathcal{F}$-**combination** $A : \mathcal{S} \cup \{1\} \to \mathcal{F}_n$ is an affine expression of the form*

$$A := c + \alpha_1\phi_1 + ... + \alpha_k\phi_k$$

where $(\phi_1, ..., \phi_k)$ are sentences and $(c, \alpha_1, ..., \alpha_k)$ are in $\mathcal{F}$. We define $\mathbb{R}$-combinations, $\mathbb{Q}$-combinations, and $\mathcal{EF}$-combinations analogously.
*We write A[1] for the trailing coefficient c, and $A[\phi]$ for the coefficient of ϕ, which is 0 if $\phi \notin (\phi_1, ..., \phi_k)$. The **rank** of A is defined to be the maximum rank among all its coefficients. Given any valuation $\mathbb{V}$, we abuse notation in the usual way and define the **value** of A (according to $\mathbb{V}$) linearly by:*

$$\mathbb{V}(A) := c + \alpha_1\mathbb{V}(\phi_1) + ... + \mathbb{V}(\alpha_k\phi_k)$$

*An $\mathcal{F}$-**combination progression** is a sequence $\overline{\mathrm{A}}$ of affine combinations where A_n has rank $\leq n$. An $\mathcal{EF}$-**combination** progression is defined similarly."*

The authors note that a trade is an example of an $\mathcal{F}$-combination. Similarly, the holdings of a trader can be encoded as the linear combination $T(\overline{\mathbb{P}})$. Note that $T(\overline{\mathbb{P}})$ is an example of a $\mathbb{Q}$-combination (recall that markets only store rational values).

2.5 Exploitation

To illustrate exploitation, Garrabrant et al. ask us to consider a market $\overline{\mathbb{P}}$ where $\mathbb{P}_n(\text{"}1+1=2\text{"}) = 0.5$ for all n, and where a trader buys one share of "$1+1=2$" every day (Now imagine there is a reasoner who buys and sells shares based on the prevailing market prices and is obligated to pay \$1 to holders of ϕ-shares if and when the deductive process (recall the group of mathematicians), $\overline{\mathcal{D}}$, says ϕ. The idea then is that the trader can make an unbounded amount of money off of the reasoner by buying a share of "$1+1=2$" (after $\overline{\mathcal{D}}$ says "$1+1=2$") every day. The trader can make an unbounded amount of money because while the trader pays the reasoner 50¢ for a share of "$1+1=2$", the reasoner must pay \$1 to the trader because the statement is confirmed to be true.

2.5.1 Exploitation

*"A trader $\overline{\mathrm{T}}$ is said to **exploit** a valuation sequence $\overline{\mathbb{V}}$ relative to a deductive process $\overline{\mathrm{D}}$ if the set of values*

$$\left\{\mathbb{W}\left(\sum_{i\leq n} T_i(\overline{\mathbb{V}})\right)\middle| n \in \mathbb{N}^+, \mathbb{W} \in \mathcal{PC}(\mathcal{D}_n)\right\}$$

is bounded below, but not bounded above."

Above is the formal definition of what it generally means to exploit a valuation sequence. Where we can interpret $\mathbb{W}\big(\sum_{i\leq n} T_i(\overline{\mathbb{P}})\big)$ as meaning the net holdings of the trader after trading on the market $\overline{\mathbb{P}}$. The syntax here is a little confusing but the idea is that a share ϕ has a value of \$1 if ϕ is true in $\mathbb{W}$ and \$0 otherwise. Then the whole expression can be interpreted (replacing $\overline{\mathbb{V}}$ for $\overline{\mathbb{P}}$) as meaning the net holdings of the trader across all time (because n takes on an infinite number of values) and for all worlds that are propositionally consistent in $\overline{\mathcal{D}}$ is exploitative if it is bounded from below but not above (i.e. in the initial example, the trader is able to make 50¢ off of the reasoner and essentially exploit the market).

2.6 Main Result

Now the moment we have all been waiting for, the main result, folks!

2.6.1 Main Result

"For any deductive process process $\overline{\mathcal{D}}$, there exists a computable belief sequence $\overline{\mathbb{P}}$ satisfying the logical induction criterion relative to $\overline{\mathcal{D}}$."

The intuition for why this result holds true is to imagine the computable belief sequence as being a stock market. If the market prices aren't readjusted as a result of supply-and-demand (i.e. if consistency between logical statements is not achieved) then a trader could exploit the market and arbitrage inconsistent logical statements. Therefore, the market prices must adjust themselves so that in the limit, the market cannot be exploited. It turns out logical inductors do indeed have this property of logical consistency in the limit as well as a variety of other astonishing properties.

3 Some Astonishing Properties of Logical Inductors

1. **Convergence and Coherence**: Which is to say that in the limit, a logical inductor has views which are entirely logically consistent.
2. **Timely Learning**: A logical inductor is able to assign high probabilities to theorems which are likely to be true and low probabilities to theorems which are likely to be false - and does this in a timely manner. Also, it assigns reasonable values to theorems which accurately reflect their truth-values way before the theorems are actually proven or disproven.
3. **Calibration and Unbiaseness**: Logical inductors are well-calibrated meaning that if a statement is true, in the limit the number assigned to that statement should equal 1. (And 0 if the statement is false.) Furthermore, given all the statements with a probability of 80% (for example), a logical inductor is well-calibrated if 80% of those statements turn out to be true. Given good feedback, a logical inductor will not be biased.
4. **Learning Statistical Patterns**: The logical inductor learns statistical patterns. So for example if something looks pseudorandom, it learns how to treat these cases well (such as assigning an initial value of 10% that the 100th digit in the expansion of π is a 7.
5. **Learning Logical Relationships**: Logical inductors learn to handle logical constraints well.
6. **Non-Dogmatism**: The logical inductor will never assign a value of 1 to something that cannot be proved or a value of 0 to something that cannot be disproved.
7. **Conditionals**: If we give to the logical inductor new axioms to follow, it does a good job incorporating those new axioms (that is say, it continues to do logical induction).
8. **Expectations**: Logical inductors are able to assign reasonable values to logically uncertain statements (such as "the Goldbach Conjecture is true").
9. **Trust in Consistency**: A logical inductor learns to trust consistency.
10. **Reasoning about Halting**: The authors note here that if there is an efficient method for generating programs that halt, a logical inductor will be able to learn in a timely manner that those programs will halt; if there is an efficient method for generating programs that do not halt, a logical inductor will be able to learn in a timely manner to not expect those programs to halt anytime soon.
11. **Introspection**: Logical inductors are able to introspect on their learning. That is, they are able to figure out how well they know the things that they know (i.e. they "know what they know").
12. **Self-Trust**: From the authors: "Logical inductors trust in their future beliefs."

4 Conclusion

The logical inductor framework provides a formal underpinning for understanding how AI will come to perform good reasoning about logical statements when there is uncertainty surrounding the truth values of those logical statements. In actuality this paper is theoretically very interesting but does not offer anything we would actually want to build (a result of the logical induction algorithm being extremely inefficient). Nonetheless, this kind of work is important if we are to move forward to an age where humans and AI can coexist (peacefully) together. The fact that logical inductors possess properties like introspection the ability to reason about halting demonstrates that this line of research shows great promise for the advancement of AI and AI-safety.

References

[1] Miles, R. (2018, October 04). AI & Logical Induction - Computerphile. Retrieved October 20, 2018, from https://www.youtube.com/watch?v=gDqkCxYYDGk

[2] Garrabrant, S., Benson-Tilsen, T., Critch, A., Soares, N., & Taylor, J. (2016). Logical induction. arXiv preprint arXiv:1609.03543.