Option markets as prediction markets for un-falsifiable, un-verifiable sentences

Abhimanyu Pallavi Sudhir

7 May 2023

Abstract

Prediction markets are useful for estimating probabilities of fixed-term questions (i.e. questions whose true answer will be revealed at a given time) – this includes questions about the values of real-world events (i.e. statistical uncertainty), and questions about the values of primitive recursive functions (i.e. logical or algorithmic uncertainty). However, they cannot be directly applied to questions without a fixed resolution criterion (real-world applications of prediction markets to such questions often amount to predicting not whether a sentence is true, but whether it will be proven) – such questions could be represented by countable unions or intersections of more basic events, or as First-Order-Logic sentences on the Arithmetical Hierarchy (or even beyond FOL, as hyperarithmetical sentences). In this paper, we suggest modeling such events with *perpetual options*, so that their probabilities are reflected by the prices of these options. Our work has immediate implications for prediction market design and logical uncertainty, but also broader implications for philosophy and mathematical logic.

1 Introduction

Prediction markets are an attractive framework for modeling probabilities of events – in a sense, they are more general than regular probability theory, because they handle incomplete beliefs and computational costs, and can be used even when there is no meaningful σ-algebra of events to speak of, i.e. for logical or algorithmic uncertainty [1–4].

However, there is an important sense in which prediction markets are less general than σ-algebras: while they can efficiently assign probabilities to finite unions and intersections of events (known as "combinatorial markets") [5, 6], they do not help us define probabilities of countable unions and intersections of events, which standard probability theory does:

$$\Pr\left[\bigcup_{n=1}^{\infty} X_n\right] = \sup_N \Pr\left[\bigcup_{n=1}^{N} X_n\right]$$
$$\Pr\left[\bigcap_{n=1}^{\infty} X_n\right] = \inf_N \Pr\left[\bigcap_{n=1}^{N} X_n\right]$$

Such events may be described in the language of First-Order Logic: a $\Sigma_0 = \Pi_0$ sentence is a sentence about primitive recursion functions, with only bounded quantifiers; a Σ_{n+1} sentence is one of the form $\exists x \in \mathbb{N}^k, \pi(x)$ where π is a Π_n formula; a Π_{n+1} sentence is one of the form $\forall x \in \mathbb{N}^k, \sigma(x)$ where σ is a Σ_n formula.

Example 1.1. Σ_1 sentences are the *verifiable* sentences ("this event will occur", "this program halts", "this program returns 37"); Π_1 sentences are the *falsifiable* sentences ("this rule will always hold true"). Other examples: "there are a finite number of Mersenne primes" (Σ_2), "every man who will ever be born will be mortal" (Π_2); sentences about limits, i.e. "such sequence diverges to $+\infty$" (Π_3)

Remark. The meaningfulness of such sentences can be understood in philosophical terms: *finitism* is the philosophy that rejects any sentence that is not primitive recursive. By assigning measures to countable unions and intersections, probability theory rejects finitism – we seek to do the same with prediction markets.

One approach to extend prediction markets to non-primitive-recursive sentences, in the context of logical uncertainty, is the framework of Garrabrant induction [4], which fixes a price of \$1 for $\phi + \neg\phi$ even if they are both unprovable (i.e. stocks of $\phi + \neg\phi$ are counted in the trader's budget, and a non-exploitable logical inductor may not lose an infinite amount of $\phi + \neg\phi$ stock), so that even unprovable sentences must be given non-zero prices that ultimately depend on considerations of algorithmic information theory.

However, this solution is still inextricably connected to a prover, which makes it difficult to generalize beyond the context of logical uncertainty, and it is not clear that the probabilities assigned (or the "priority given") to unprovable sentences are aligned with our desire to predict physical events that are correlated with them (e.g. shouldn't our probability for Goodstein's theorem, or the consistency of Peano Arithmetic, really approach 1?). Ideally we would like our market to be a "stronger system" than a particular formal theory, which only learns to trust that formal theory (i.e. contains it as a trader and allocates it a large budget) if it proves itself. There would still be sentences independent of the market (e.g. "the market price of this sentence on Day 5 is less than 0.5") but the market should be able to hold confident beliefs about sentences independent of a particular formal theory.

We propose an alternate approach: to model first-order logic sentences as *options*.

2 The framework

2.1 Setting: finite processes and combinatorial stocks

We will first describe the setting we are working with, in a way that is general enough that it can be applied to both algorithmic uncertainty and statistical uncertainty.

We use the term *finite process* to refer to a blackbox that reveals a value out of some specified finite set s at some specified finite time t, so that this value can be called upon by any program at t. The exact nature of such a process is left as a black-box: it could be a program with known source code, or a physical process of unknown mechanism.

```
▷ Not to be actually implemented, essentially a “state of the world” type
class FinProc
    $s : \text{finset}(\mathbb{N})$                                    ▷ set of possible values
    $t : \mathbb{N}$                                                    ▷ maturity date
    $v(\tau) : \texttt{Maybe}(s) := \begin{cases} \dots & \tau \geq \texttt{self}.t \\ \texttt{None} & \text{else} \end{cases}$     ▷ observe its value
```

We suppose that the finite processes we are interested in can be computably enumerated, e.g. by a stock ticker symbol. The most basic relationships between the finite processes we are interested in reasoning about are finite logical combinations between them, i.e. questions like $X = 1 \wedge Y = 4 \wedge Z = 4$ which can be represented by terms $\hat{X} : \text{finset}\,(\text{FinProc})$ and inherits properties $\hat{X}.s = \prod_{X \in \hat{X}} X.s$, $\hat{X}.t = \max_{X \in \hat{X}} X.t$, $\hat{X}.v(t) = (X.v(t))_{X \in \hat{X}}$.

As in [5, 6], it is easy to trade stocks on the outcomes of finite logical combinations of such finite processes, i.e. “combinatorial stocks” – denote the type of combinatorial stock portfolios as

$$\text{PF} = (\hat{X} : \text{finset}\,(\text{FinProc})) \to_{\circ} \left(\hat{X}.s \to \mathbb{Q}\right)$$

[notation: we denote dependent types like $f : (a : A) \to B(a)$, and $\to_{\circ}$ denotes a function with finite support] E.g. if $\mathbf{q} : \text{PF}$ then $\mathbf{q}(X, Y, Z)$ denotes a stock distribution on the eight (assuming $|s| = 2$ for all the finite processes) possible values of (X, Y, Z).

```
class Agent
    $\mathbf{q} : \text{PF}$                                   ▷ Inventory of combinatorial stocks
    $c : \mathbb{Q}$                                          ▷ Cash inventory
    order : $(t : \mathbb{N}) \to \text{PF}$          ▷ The algorithm implemented by the agent
```

We may have some family of agents that we are interested in – let Enumerator(t) be an enumeration of these agents, corresponding to which agent interfaces with the market at each point in time. There may be distinct Agentinstances with the same algorithm, but also the same instance appears multiple times in the enumeration, for the duration that is its lifespan. In the interest of fairness, each agent should be given ample opportunities to interface with the market.

We can then define a rewarder algorithm, running in the background, for all combinatorial stocks, by rewarding holders of stocks when their time has come. This is what really gives “meaning” to each asset, and the bit that needs to be generalized.

```
function REWARDER
    while $t \in \mathbb{N}$ do
        $\alpha \leftarrow$ ENUMERATOR($t$)
        for $\hat{X} \in \operatorname{supp}(\alpha.\mathbf{q})$ do          ▷ for all stocks $\alpha$ has a stake in
            if $t \geq \hat{X}.t$ then                               ▷ if they're matured
                $\alpha.\mathbf{q}(\hat{X}) \leftarrow \mathbf{0}$                                ▷ take them away
                $\alpha.c \leftarrow \alpha.c + \alpha.\mathbf{q}(\hat{X})(\hat{X}.v(t))$                ▷ and pay reward
```

Now we can define a market for all combinatorial stocks. Let $P(\mathbf{q} : \mathrm{PF})$ be the “costing function”, that gives the cash that a market-maker will accept to issue a total of $\mathbf{q}$ outstanding stocks – this alone determines the market. For example, logarithmic market scoring [6] is defined by the costing function:

$$P(\mathbf{q}) = \sum_{\hat{X} \in \operatorname{supp}(\mathbf{q})} \lambda_{\hat{X}} \sum_{x \in \hat{X}.s} \exp\left(\mathbf{q}(\hat{X}, x)/\lambda_{\hat{X}}\right)$$

Where $\lambda_{\hat{X}}$ the “market-maker subsidy” for the market on $\hat{X}$ (if the finite processes can be computably enumerated, $\lambda_{\hat{X}}$ can be pre-specified so it sums to a finite value over all $\hat{X}$) – see appendix for further details. Then simply:

```
function MARKET
    $\mathbf{q}, c \leftarrow 0$    ▷ Account of outstanding stocks, market-maker's cash reserves
    while $t \in \mathbb{N}$ do
        $\alpha \leftarrow$ ENUMERATOR($t$)
        $\Delta\mathbf{q} \leftarrow \alpha(t)$
        $\Delta c \leftarrow P(\mathbf{q} + \Delta\mathbf{q}) - P(\mathbf{q})$
        if $\Delta c < \alpha.c$ then                                    ▷ Check if in budget
            $\mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
            $c \leftarrow c + \Delta c$
            $\alpha.\mathbf{q} \leftarrow \alpha.\mathbf{q} + \Delta\mathbf{q}$
            $\alpha.c \leftarrow \alpha.c - \Delta c$
        $\mathbf{p} = \nabla P(\mathbf{q})$                              ▷ update current market prices
```

Of course, as stated, the market really incentivizes agents to make orders on unboundedly larger supports, causing unboundedly larger computational costs – this can be fixed by charging transaction fees, but we will ignore this detail for now and assume that the market's computational costs are zero.

Example 2.1 (Primitive sentences as finite processes)**.** Since we can computably enumerate the primitive recursive functions, we can also use computably enumerate primitive recursive function-input pairs with a Cantor pairing function, denote this enumeration as $(C_i.f, C_i.n)$. Then consider the enumeration of finite processes X_i given by

$$\begin{aligned} X_i.s &= \{\text{False}, \text{True}\} \\ X_i.t &= i \\ X_i.v(\tau) &= \begin{cases} C_i.f(C_i.n) == 0 & \tau \geq \texttt{self}.t \\ \texttt{None} & \text{else} \end{cases} \end{aligned}$$

I.e. at each time i, we decide if the ith primitive computation equals to zero.

Example 2.2 (Observations as finite proceesses)**.** Let $Y_t : \Omega \to \{0, 1\}$ be random variables representing a stream of observations at each time t, generated by some joint distribution $\mathbf{P}(y_1, \dots)$ (for example, it could be the universal measure). Then:

$$\begin{aligned} X_i.s &= \{0, 1\} \\ X_i.t &= i \\ X_i.v(\tau) &= \begin{cases} Y_i(\omega) & \tau \geq \texttt{self}.t \\ \texttt{None} & \text{else} \end{cases} \end{aligned}$$

2.2 The Option hierarchy

We are now ready to describe our main contribution: allowing traders to bet on express beliefs about facts *not* resolved by a finite process, by modeling such facts as options on combinatorial stocks, options on those options, etc. Intuitively, a Σ_{n+1} sentence $\exists x, \pi(x)$ can be thought of as an infinite combination $\pi(1) \lor \pi(2) \lor \dots$, and a Π_{n+1} sentence $\forall x, \sigma(x)$ can be thought of as an infinite combination $\sigma(1) \land \sigma(2) \land \dots$ Unlike for a finite combination however, our "maturity date" may no longer be finite, and so standard prediction markets do not work. For Σ_1 sentences (resp. Π_1 sentences) you can still get around the problem with perpetual prediction markets by taking (resp. giving) an upfront payment because the questions are still computationally verifiable (resp. computationally falsifiable), but for higher sentences, this too fails. Instead, we might want:

- A Σ_{n+1} sentence $\exists x, \pi(x)$ is associated with a zero-strike-price perpetual call option on the underlying asset $\exists t \leq x, \pi(x)$ (i.e. the right to take any finite number of assets of the form $\pi(x)$)
- A Π_n sentence is \$1 minus its negation, i.e. the position of the option-writer of a Σ_n option.

We needn't stop at finite n either: we can also define an "ω-level" option, i.e. whose underlying assets are an ascending sequence of n-level options, and so on defining a "χ"-level" asset for any computable ordinal χ (which represent sentences called "hyperarithmetical"). Naturally, this means recursion.

type $\textsc{Option} = \mathrm{DNF}\,(\textsc{FinProc}) \oplus (\mathbb{N} \to \textsc{Option})$

(where $\mathrm{DNF}\,(\textsc{FinProc})$ represents the set of all disjoint normal forms over a finite number of finite processes, $\oplus$ is the union of types) How to interpret this: an option is either simply a stock (in the outcome of some finite processes), or the *option to sell any of the underlying assets at strike price \$1.* It should be seen as representing a Σ sentence of some level (while a Π sentence is represented by \$1 minus an option, i.e. the short position on an option). All this "interpretation" will be given legitimacy by the rewarder function (i.e. the exchange that carries out the operations given in the contract).

Unlike mere finset ($\textsc{FinProc}$), an object $\hat{X} : \textsc{Option}$ does not have propeties $\hat{X}.t$ or $\hat{X}.v$ (it does not have a definitely revealed value). $\hat{X}.s$ is now always $\{\mathrm{False}, \mathrm{True}\}$.

$$\mathrm{PF} = (\hat{X} : \textsc{Option}) \to_{\circ} (\{\mathrm{False}, \mathrm{True}\} \to \mathbb{Q})$$

Agents can now have their inventory $\mathbf{q}$ and $\textsc{order}$ algorithm defined in terms of the new PF (i.e. they can trade options). We may define the ordinal "rank" of an option in the obvious way: the rank of a stock (a DNF of finite processes) is 0, while the rank of an option with underlying assets $A : \mathbb{N} \to \textsc{Option}$ is $\sup_x (1 + \operatorname{rank} A(x))$.

Although we have defined options in this generality, the most important case for us (corresponding exactly to the first-order-logic sentences) is simply that of finite-rank options. Intuitively: writing a Σ_{n+1} sentence as $\exists x, \neg\sigma(x)$ (where σ is Σ_n), its corresponding option is over assets of the form $\sigma(x_1) \wedge \cdots \wedge \sigma(x_i)$ (which, as a finite conjunction of Σ_n sentences is itself Σ_n) for every finite tuple of integers $(x_1, \dots x_i)$.

Example 2.3 (First-order logic sentences as finite-rank options)**.** Denote as $\textsc{Option.just}$ and $\textsc{Option}.\exists$ the two built-in constructors for $\textsc{Option}$ (i.e. wrappers that take terms of type $\mathrm{DNF}\,(\textsc{FinProc})$ and $\mathbb{N} \to \textsc{Option}$ respectively and return an $\textsc{Option}$); let $\mathrm{pnformat}\,()$ be a function that formats a given valid Σ_n sentence into the form $\exists x_1 \neg\exists x_2 \dots \neg\exists x_n P(x_1, \dots x_n)$ with P in Disjoint Normal Form.

Then we construct the map $\textsc{Option.of}\,[]$, from Σ sentences to options, inductively as follows:

- Base case Σ_0 sentence $\mathtt{f(n)} = \mathtt{0}$ (f primitive recursive), from Ex 2.1: $\textsc{Option.of}\,[\mathtt{f(n)} = \mathtt{0}] = \textsc{Option.just}\left[X_{\langle f,n\rangle}\right]$ where $\langle \cdot, \cdot \rangle$ is an enumeration of primitive computations.
- Inductive case Σ_{n+1} sentence $\exists x, \neg\sigma(x)$ ($\sigma(x)$ is for each x a Σ_n sentence): $\textsc{Option.of}\,[\exists x, \neg\sigma(x)] = \textsc{Option}.\exists\,[x \mapsto \textsc{Option.of}\,[\mathrm{pnformat}\,(\sigma(1) \wedge \cdots \wedge \sigma(x))]]$.

It is clear how to define the semantics (i.e. to make meaningful) the position of the option-buyer – simply include a term corresponding to exercising an option in the REWARDER algorithm. It may be less clear how to correctly reward the option-writer, i.e. how the market-maker should optimally exercise *their option*, bought from the option-writer. The only natural way to do this is for the market-maker to sell the option off to the

https://arxiv.org/pdf/2212.06888.pdf https://www.hec.edu/sites/default/files/documents/paper

References

[1] Caspar Oesterheld and Vincent Conitzer. "Decision Scoring Rules." In: *WINE*. 2020, p. 468.

[2] Caspar Oesterheld. *Futarchy implements evidential decision theory*. The Universe from an Intentional Stance. Dec. 18, 2017. URL: `https://casparoesterheld.com/2017/12/18/futarchy-implements-evidential-decision-theory/` (visited on 04/02/2023).

[3] Caspar Oesterheld, Abram Demski, and Vincent Conitzer. "A theory of bounded inductive rationality". In: 2021.

[4] Scott Garrabrant et al. "Logical Induction". In: *CoRR* abs/1609.03543 (2016). arXiv: `1609.03543`. URL: `http://arxiv.org/abs/1609.03543`.

[5] Robin Hanson. "Combinatorial Information Market Design". In: *Information Systems Frontiers* 5.1 (Jan. 1, 2003), pp. 107–119. ISSN: 1572-9419. DOI: `10.1023/A:1022058209073`. URL: `https://doi.org/10.1023/A:1022058209073` (visited on 05/05/2023).

[6] Robin Hanson. "Logarithmic Market Scoring Rules for Modular Combinatorial Information Aggregation". In: *The Journal of Prediction Markets* 1.1 (Jan. 2002), pp. 3–15. DOI: `10.5750/jpm.v1i1.417`.

Appendix

Market scoring rules

In general for a costing function P the instantaneous prices are given by $\nabla P(\mathbf{q})$ (which can be calculated as if $\mathbf{q}$ were an infinite-dimensional vector with only finitely many non-zero values, i.e.

$$\nabla P : \mathrm{PF} \to \left((\hat{X} : \mathrm{finset}\,(\textsc{FinProc})) \to \left(\hat{X}.s \to \mathbb{R}\right)\right)$$

So a necessary condition for an acceptable costing function is that $\nabla P(\mathbf{q}') = \nabla P(\mathbf{q})$ only when $\mathbf{q}' - \mathbf{q}$ amounts to a cash term (i.e. an equal distribution over mutually exclusive stocks), and that in this case $P(\mathbf{q}') - P(\mathbf{q}) = \mathbf{q}' - \mathbf{q}$. In that case, the costing function leads to a scoring rule for report $\mathbf{r}$ given by $\theta(\mathbf{r}) - P \circ \theta(\mathbf{r})$ where θ is any right-inverse of ∇P, i.e. $\theta(\mathbf{r})$ is a portfolio that leads to a market price of $\mathbf{r}$. Of course, $\mathbf{r}$ is expected to differ from the market maker's prior position on only finitely many assets.

For example, logarithmic market scoring [6] is defined by the costing function:

$$P(\mathbf{q}) = \sum_{\hat{X} \in \operatorname{supp}(\mathbf{q})} \lambda_{\hat{X}} \log \sum_{x \in \hat{X}.s} \exp\left(\mathbf{q}(\hat{X}, x)/\lambda_{\hat{X}}\right)$$

Where $\lambda_{\hat{X}}$ the "market-maker subsidy" for the market on $\hat{X}$ (if the finite processes can be computably enumerated, $\lambda_{\hat{X}}$ can be pre-specified so it sums to a finite value over all $\hat{X}$). In fact it represents the price per bit of information on the value of $\hat{X}$. One can check that prices are then given by:

$$\nabla P(\mathbf{q}, \hat{X}, x) = \frac{\exp\left(\mathbf{q}(\hat{X}, x)/\lambda_{\hat{X}}\right)}{\sum_{x \in \hat{X}.s} \exp\left(\mathbf{q}(\hat{X}, x)/\lambda_{\hat{X}}\right)}$$

One may check that the resulting scoring rule (the total market payout made in each possible outcome, for moving prices to $\mathbf{r}$) is $\mathbf{s}(\mathbf{r}) = \left[\lambda_{\hat{X}} \log \mathbf{r}(\hat{X}, x)\right]_{\hat{X} \in \operatorname{finset}(\textsc{FinProc}), x \in \hat{X}.s}$, the expectation of which, under belief $\mathbf{p}$, is:

$$s(\mathbf{p}, \mathbf{r}) = \sum_{\hat{X}} \sum_{x \in \hat{X}.s} \lambda_{\hat{X}} \mathbf{p}(\hat{X}, x) \log \mathbf{r}(\hat{X}, x) = -\sum_{\hat{X}} \lambda_{\hat{X}} H(\mathbf{p}_{\hat{X}}, \mathbf{r}_{\hat{X}})$$

Thus the expected profit from making a report $\mathbf{r}'$ when current market prices are $\mathbf{r}$:

$$\Delta s(\mathbf{p}, \mathbf{r}, \mathbf{r}') = s(\mathbf{p}, \mathbf{r}') - s(\mathbf{p}, \mathbf{r}) = \sum_{\hat{X}} \lambda_{\hat{X}} \left[H(\mathbf{p}_{\hat{X}}, \mathbf{r}_{\hat{X}}) - H(\mathbf{p}_{\hat{X}}, \mathbf{r}'_{\hat{X}})\right]$$

Where H is cross-entropy. We would *like* to say that $\Delta s(\mathbf{p}, \mathbf{r}, \mathbf{r}')$ is maximized when $\mathbf{r}' = \mathbf{p}$, but the report $\mathbf{r}'$ has finite support while $\mathbf{p}$ is not, so there is no "best possible report" to make. But we can see that the profit is decreasing in the cross-entropy $H(\mathbf{p}_{\hat{X}}, \mathbf{r}'_{\hat{X}})$, i.e. the closer your report to your true belief, the better.

More precisely, one may calculate the value of acquiring some new information (say the value of some random variable Y) as follows.

Suppose making a report of $\mathbf{r}$ is associated with some transaction cost $C(\operatorname{supp}(\mathbf{r}))$ such that $\lim_{|\operatorname{supp}(\mathbf{r})| \to \infty} C(\operatorname{supp}(\mathbf{r})) = \infty$. Then of course the agent makes the report that maximizes $\max_{\mathbf{r}} [s(\mathbf{p}, \mathbf{r}) - C(\operatorname{supp}(\mathbf{r}))]$. For some $S \subseteq \operatorname{finset}(\textsc{FinProc})$, denoting $\delta = \lambda(S') = \sum_{\hat{X} \notin S} \lambda_{\hat{X}}$ and $\varepsilon = C(S)$:

$$\begin{aligned}
\max_{\mathbf{r}} \left[s(\mathbf{p},\mathbf{r}) - C(\operatorname{supp}(\mathbf{r}))\right] &\le -\sum_{\hat{X}} \lambda_{\hat{X}} H(\mathbf{p}_{\hat{X}}) \\
\max_{\mathbf{r}} \left[s(\mathbf{p},\mathbf{r}) - C(\operatorname{supp}(\mathbf{r}))\right] &\ge s(\mathbf{p},\mathbf{p}|_S) - C(\mathbf{p}|_S) \\
&\ge -\sum_{\hat{X}} \lambda_{\hat{X}} H(\mathbf{p}_{\hat{X}}) - \lambda(S')\log 2 - C(S)
\end{aligned}$$

Similarly the expected score from the report after acquiring information Y is $\mathbf{E}_Y\left[\max_{\mathbf{r}}\left[s(\mathbf{p}_{\hat{X}=x|Y=y},\mathbf{r}) - C(\operatorname{supp}(\mathbf{r}))\right]\right]$. Since the term inside the expectation is bounded within $\lambda(S')\log 2 - C(S)$ of $-\sum_{\hat{X}} \lambda_{\hat{X}} H(\mathbf{p}_{\hat{X}}|Y=y)$, the whole term is bounded within the same constant $\lambda(S')\log 2 - C(S)$ of its expectation under Y, which is the conditional entropy:

$$-\sum_{\hat{X}} \lambda_{\hat{X}} H(\hat{X}|Y) - \lambda(S')\log(2) - C(S) \le \cdots \le -\sum_{\hat{X}} \lambda_{\hat{X}} H(\hat{X}|Y)$$

And so the value gained from aquiring Y is within $\lambda(S')\log 2 - C(S)$ of the mutual information $\sum_{\hat{X}} \lambda_{\hat{X}} I(\hat{X};Y)$. Thus by making the transaction cost sufficiently slow-growing, the price paid by the market maker for one bit of information on $\hat{X}$ can be made arbitrarily close to $\lambda_{\hat{X}}$.