

Combinatorial Arbitrage in Prediction Markets*

Hongjian Shi[†]

Daojing Zhai[‡]

First version: September 15, 2026

This version: October 10, 2026

Abstract

We develop a framework to identify and quantify arbitrage across logically related prediction-market contracts. We prove that prices outside the convex hull of admissible payoff vectors admit arbitrage and identify optimal arbitrage portfolios through integer programming. Using venue metadata and semantic analysis of resolution criteria, we recover 1.9 million logical relations and compile them into constraint systems covering more than 255,000 contracts. We replay full-depth Polymarket order books to simulate an arbitrageur executing these portfolios against resting orders. Among settled portfolios, guaranteed profit after spreads and fees equals 2.2% of committed capital, while profit at resolution reaches 3.2%. We find that arbitrage opportunities persist longer and attract less observed participation where relation discovery and execution are more costly.

Keywords: Prediction markets; combinatorial arbitrage; market microstructure; large language models; limit order books.

JEL codes: C61; C63; G12; G13; G14.

*We thank the Yale Center for Research Computing for providing computational resources and support.

[†]Department of Mathematics, Technical University of Munich. Email: hongjian.shi@tum.de.

[‡]Baruch College, City University of New York. Email: daojing.zhai@baruch.cuny.edu.

1 Introduction

Prediction markets are emerging as a new asset class. Like conventional financial assets, prediction contracts support price discovery, hedging, and speculation. Their market structure, however, creates a particular challenge for consistent pricing. Exchanges such as Polymarket and Kalshi list many contingent claims that resolve on the same underlying events, so logical relations tie their pay-offs together. Each contract trades in its own order book, and the exchanges do not generally enforce the price restrictions implied by these relations. This fragmentation can create *combinatorial arbitrage*: a portfolio of related contracts that guarantees a profit under every admissible joint resolution.

This paper develops a unified framework to identify these opportunities and to measure their executable size and profit. The framework has three parts. First, we mathematically characterize when the prices of logically related contracts admit arbitrage and find the portfolio with the largest guaranteed profit by integer programming. Second, we recover the logical relations among contracts from venue metadata and from resolution text, using structured parsers and large language models (LLMs). Third, we collect Polymarket’s full-depth order books and simulate an arbitrageur who trades these portfolios against displayed depth.

More specifically, we formalize the arbitrage opportunity by representing each admissible joint resolution as a binary payoff vector. Quoted prices are arbitrage-free if and only if they lie in the coherent set, the convex hull of these payoff vectors. When prices lie outside this set, the largest guaranteed profit per unit position equals their ℓ_1 distance from it. With many related contracts, the payoff vectors are too numerous to list. We therefore encode logical relations as a linear constraint system and solve for the optimal portfolio by integer programming. Quoted prices give the direction of the trade but not its size. We then scan the displayed depth at each price level of the order book to determine how much the arbitrageur can trade after spreads and taker fees.

The empirical analysis covers Polymarket from 23 June to 3 August 2026 and combines three data sources: about 745,000 contracts and their associated resolution texts retrieved from Polymarket’s market catalog, 73.22 billion order-book messages, and 227 million on-chain fill events. The catalog tells us how contracts relate, the order books measure what an arbitrageur can trade, and the fills show what traders did. We recover the logical relations in three layers. *Venue-asserted* relations come from the groups of mutually exclusive contracts that Polymarket designates itself and from the rules of these groups. *Venue-derived* relations follow from the match, contract type, and line that the venue reports for sports contracts. *Semantic-inferred* relations are read from contract text, by structured parsers or by large language models. In these three layers, we recover 1.9 million relations and reduce them to about 570,000 relations covering more than 255,000 contracts; we compile each layer into a separate constraint system and evaluate interacting relations jointly within it. An invalid relation can create a false profit guarantee, whereas omitting a valid relation can only hide an opportunity. We therefore screen relations using checks appropriate to each source before admitting them to the corresponding system.

We report three findings. First, combinatorial arbitrage survives spreads and fees. We detect 491,939 arbitrage windows, the intervals over which a portfolio has positive guaranteed profit at the best quotes, and the arbitrageur can trade in about 125,000 of them. In the windows whose contracts have all resolved, the arbitrageur accumulates \$2.50 million of guaranteed profit on \$113.7 million of capital, a return of 2.2%. Profit at resolution is \$3.63 million, a return of 3.2%. The first execution in each window alone yields \$1.10 million in guaranteed profit. Venue-asserted and venue-derived relations contribute 34% and 59% of the guaranteed profit, and semantic-inferred relations the remaining 7%. These profits are robust to the execution assumptions. With a 1-second order delay, the filled orders retain 79% of the profit at resolution. Returns on capital also change little when re-posted depth does not count as new liquidity and when the arbitrageur fills only part of the displayed depth.

Second, opportunities persist longer where the logical relation is harder to find. Among windows that last at least 0.3 seconds, the median duration is 2.9 seconds for venue-asserted relations, 6.3 seconds for venue-derived relations, and about 31 seconds for semantic-inferred relations. The ordering follows the work each layer requires: the venue publishes the first layer, the second layer must be reconstructed from match fields, and the third layer requires reading and comparing resolution text. The differences remain after controlling for portfolio characteristics, the component’s daily number of arbitrage windows, and day and hour fixed effects: relative to venue-asserted windows, venue-derived windows last about 2 times as long and semantic-inferred windows about 10 times as long. Each additional leg in a portfolio is also associated with a longer duration. The layers differ in their contracts and execution mechanisms as well as in the information they require, so the evidence is consistent with both discovery and execution costs limiting arbitrage.

Third, actual trading follows the arbitrage direction more often when the venue makes the relation explicit. In 44% of the windows identified by venue-asserted relations, at least one trader trades every leg of the portfolio in the arbitrage direction. For windows identified by venue-derived and semantic-inferred relations, the percentage declines to 15% and 9%. These trades arrive quickly: the median lag from the opening of a window is about 3 seconds. Trading is also concentrated among a small number of wallets: 854 addresses make these trades, and the 10 most active addresses account for 78% of the windows with such a trade. The semantic-inferred layer—and in particular its LLM component, whose relations require the most work to identify—turns out to be indeed the one in which opportunities last longest and the fewest are taken.

Related literature. First, the paper contributes to research on prediction markets. This literature studies how prices aggregate information (Wolfers and Zitzewitz, 2004), how price discovery is shared across venues (Ng et al., 2026), how informed traders contribute to forecast accuracy (Gomez-Cram et al., 2026), and how adverse selection affects liquidity provision (Bartlett and O’Hara, 2026). A smaller set of papers examines prices across related contracts. Sonnemann et al. (2013) document framing effects across equivalent partitions, and Rothschild and Pennock (2014) document misalignment among identical and logically related contracts and test liquidity through field trades.

Saguillo et al. (2025) use language models to identify dependencies between Polymarket markets and measure arbitrage within markets and between pairs of markets with equivalent payoffs. To our knowledge, this paper is the first to evaluate logical relations of different types jointly in one constraint system and to measure the executable size and profit of the resulting portfolios on full-depth order books.

Second, the paper relates to the finance literature on violations of the law of one price (Lamont and Thaler, 2003). Familiar cases include spreads between Treasury cash and derivative markets (Fleckenstein and Longstaff, 2020; Zhai, 2026) and deviations from covered interest parity (Du, Tepper, and Verdelhan, 2018). These studies attribute price differences to market segmentation, funding costs, and balance-sheet constraints, and models of financially constrained arbitrageurs explain why the differences persist (Gromb and Vayanos, 2002). They take the payoff relation as known. Prediction markets differ in two ways. First, the relations are unknown and numerous: they must be discovered among hundreds of thousands of contracts. Second, the relations are richer than equivalence: they include mutual exclusivity, nesting, partition, and conjunction. Both features make it challenging to identify arbitrage opportunities.

Third, the paper connects to the literature on combinatorial market design. This literature studies how a market maker can keep prices coherent across related claims (Hanson, 2003; Abernethy, Chen, and Wortman Vaughan, 2013). Computing coherent prices is hard when the claims are many (Chen et al., 2008), and Dudík, Lahaie, and Pennock (2012) and Kroer et al. (2016) make it tractable with constraint generation and integer programming. Recent work recovers the relations between contracts from text: Gebele and Matthes (2026) identify equivalent or subset-related contract pairs across venues, and Capponi, Gliozzo, and Zhu (2025) discover semantic relations for statistical trading. We differ in two ways. First, we take the trader’s side: prices have already formed in separate order books, and no market maker keeps them coherent. Second, we connect relation recovery to arbitrage: the recovered relations become the constraints of the arbitrage problem, and we evaluate the resulting portfolios against full-depth order books.

Finally, the paper contributes to the literature on limits to arbitrage. Mispricing can persist because arbitrageurs face noise-trader risk (De Long et al., 1990), constrained capital (Shleifer and Vishny, 1997), and holding and transaction costs (Pontiff, 2006). Moskowitz (2021) uses sports betting markets to test asset pricing theories and finds momentum returns that fail to overcome transaction costs, which prevents arbitrage from eliminating them. The opportunities we measure remain after spreads and fees, so transaction costs alone do not explain them. Our evidence points to the cost of discovering the relation, an information cost in the sense of Grossman and Stiglitz (1980): prices adjust more slowly when information requires more complicated analysis (Cohen and Lou, 2012). Our findings are consistent with this evidence: opportunities persist longest, and the fewest are taken, where the relation must be inferred from contract text. Execution adds a second cost. The legs of a portfolio trade in separate order books, so the arbitrageur bears the risk that one leg fills and another does not.

Road map. Section 2 describes the prediction market structure and the data. Section 3 develops the framework, from quoted prices to finite-depth order books. Section 4 recovers the logical relation system in three layers. Section 5 measures the opportunities, their persistence, and the trading around them. Section 6 concludes the paper. Appendix A collects the proofs and derivations behind the framework. Appendix B documents how the data are collected, how the relation system is constructed, and how the opportunities are measured.

2 Prediction Market Structure and Data

2.1 Market Structure and the Source of Combinatorial Arbitrage

Polymarket and Kalshi saw their combined monthly notional trading volume rise from under \$5 billion in mid-2025 to above \$50 billion in July 2026.¹ This growth has drawn retail investors alongside Wall Street hedge funds and professional market makers, while prediction contracts are increasingly functioning as standardized financial instruments for hedging event risk, discovering prices, and aggregating dispersed beliefs.

A prediction-market contract is a fully collateralized binary claim through which investors express a view on an event governed by a prespecified resolution rule. Each contract creates a YES and a NO outcome token backed jointly by one dollar: one dollar *splits* into one YES token and one NO token, the pair *merges* back into the dollar at any time, and at resolution the winning token pays one dollar while the other pays zero. Before resolution, the contract trades between zero and one in a central limit order book, and the price of the YES token is commonly interpreted as the market-implied probability of a YES resolution.

The two tokens are complementary representations of the same claim. Although the venue exposes an order book under each token identifier, the two views describe the same economic order book: a NO bid at b^N is a YES ask at $1 - b^N$, while a NO ask at a^N is a YES bid at $1 - a^N$. The matching engine crosses complementary orders by splitting a dollar into the token pair or merging the pair, so buying NO is equivalent to selling YES, and selling NO is equivalent to buying YES.² The fee structure of Polymarket also respects the same complementarity: whether a taker buys or sells, the taker fee follows the same formula, proportional to the product of the share price and its complement.

One distinctive feature of prediction markets is that a single real-world event can support multiple contracts. A football match, for example, may generate contracts on the winner, the exact score, the total number of goals, and combinations of these outcomes. Because these contracts depend on the same realized match outcome, their terminal payoffs must satisfy the restrictions imposed by their logical relations. This common resolution is not reflected in the trading architecture: each contract has its own order book, and the matching engine neither enforces coherence across

¹See [Pew Research Center, September 23, 2026](#).

²Throughout the paper, we therefore express all positions and quotes in YES-equivalent units and work with a single canonical YES-side order book.

order books nor offers an atomic multi-leg order spanning them. Prices can therefore appear reasonable contract by contract even though no single belief can rationalize them jointly, creating combinatorial arbitrage opportunities. Venues provide little direct enforcement of these restrictions, so restoring coherence across order books is left to arbitrageurs who trade against the mispricing.³

2.2 Data: Polymarket at Full Depth

To examine arbitrage opportunities in the prediction market, we assemble a full-depth record of Polymarket along three dimensions: which contracts are listed (market catalog), how each contract is quoted (full-depth order books), and what trades take place (real transactions).⁴ The record covers 23 June to 3 August 2026, and over these 42 days we observe roughly 745,000 binary contracts, 73.22 billion order-book messages, and 227 million on-chain fill events resolving to 143.53 million observed trades that carry \$5.20 billion of volume. Table 1 summarizes the three parts of the dataset.

Market catalog. We retrieve the venue’s listing catalog and reconstruct its hierarchy from event pages down to binary conditions and outcome tokens. For each contract we observe its identifiers, fee schedule, negative-risk designation, and the displayed resolution text preserved in our catalog snapshot. The hierarchy tells us which contracts can share an underlying state, and the criteria text is the raw material from which Section 4 recovers their precise logical relations. Relations are audited against this preserved catalog text rather than against the listing-time `QuestionInitialized` record on chain. The catalog covers roughly 745,000 market records on 178,000 event pages and identifies 20,866 negative-risk groups (Table 1). Appendix B.1.1 details its construction.

Full-depth order books. Polymarket provides no retrospective depth feed, so we operated a live collector throughout the sample, recording venue-wide book snapshots and price-level updates with exchange and receipt timestamps. Full depth makes execution measurable. Across valid two-sided quote states, the pooled median spread is 8.13 cents, and median depth is 40.20 shares at the best bid and 29.46 shares at the best ask. Orders exceeding the quantity at the best quote execute at additional price levels. Best quotes alone therefore do not determine their execution costs. The full-depth order-book data contain 73.22 billion messages, reduced to about 3.6 billion valid two-sided quote states for venue-wide detection; executable size is always measured on the full order books reconstructed for each candidate window. Appendices B.1.2 and B.1.3 detail the collection and the reconstruction.

Real transactions. We decode every `OrderFilled` settlement event emitted by Polymarket’s two exchange contracts on Polygon over the window. We refer to these records as *On-Chain Fills*.

³The one exception is Polymarket’s **negative-risk** mechanism (Kalshi’s **collateral-return** facility plays a similar role): for venue-designated groups of mutually exclusive contracts, one dollar of collateral *mints* one YES share of every contract in the group (a split followed by a *conversion* of the NO share), so selling the whole group is cheap to execute; buying it still takes one order per contract. Section 5.3 shows that arbitrage opportunities inside these groups close most quickly.

⁴We focus on Polymarket because all three dimensions are publicly observable there: quotes stream at full depth, the listing-time resolution question is recoverable on chain, and every trade settles on the Polygon blockchain. On centrally cleared venues such as Kalshi, settlement occurs inside the exchange’s clearinghouse, so the published trades cannot be linked to individual accounts.

Table 1: Summary statistics of the Polymarket record

	Sports	Crypto	Politics	Finance	Weather	Other	All
<i>Panel A: Market catalog (daily)</i>							
Active contracts	50,603	8,366	4,077	2,727	2,061	1,865	69,698
Active event pages	5,033	6,042	1,238	497	230	314	13,355
<i>Panel B: Full-depth order books (daily)</i>							
Quote states	30,471,793	12,805,233	11,245,638	9,180,315	13,542,637	8,163,812	85,409,429
Active order books	90,388	15,524	7,449	5,149	3,388	3,335	125,233
Bid-ask spread (cents)	31.60	4.73	8.21	14.12	3.98	12.67	9.41
<i>Panel C: Real transactions (daily)</i>							
Trading volume (USDC)	89,037,081	18,559,072	8,095,243	1,046,705	1,991,597	573,129	123,738,023
Observed trades	787,617	2,274,957	115,208	27,588	192,080	21,825	3,417,376
Trader addresses	50,140	26,811	18,760	5,408	7,382	4,269	94,555

Notes: Daily means over 23 June–3 August 2026 (42 days). Other pools technology, culture and untagged contracts. Active contracts and event pages are those with a quoted book that day; the spread is the within-day median over valid two-sided quote states; volume counts each observed trade once.

Each fill reports the traded token, side, price, quantity, fee, maker and taker identifiers, transaction hash and log index, and block timestamp, and the token identifiers link each fill back to the catalog and the reconstructed order books. These records show whether actual trades follow the detected portfolios. Section 5.4 matches fills to their contracts and trading directions, while Sections 5.1–5.3 measure opportunities at displayed quotes. The transaction data contain 227 million `OrderFilled` events, which resolve to 143.53 million observed trades carrying \$5.20 billion of volume among 677,000 distinct trader addresses (Table 1). Appendix B.1.4 details the retrieval.

3 A Framework for Combinatorial Arbitrage

Building on the institutional primitives in Section 2, this section develops a framework for combinatorial arbitrage in three steps. Section 3.1 characterizes arbitrage as a failure of price coherence and measures its profit as a distance. Section 3.2 compiles the logical relations into a constraint system and solves for the optimal portfolio by integer programming. Section 3.3 then replaces quoted prices with order books, where executable size and guaranteed profit depend on spreads, depth, and fees.

3.1 Characterizing Arbitrage through Price Coherence

We begin with a single quoted price per contract, isolating the payoff restrictions imposed by logical relations. The no-arbitrage characterization follows the Dutch-book argument of [de Finetti \(1937\)](#).

No-arbitrage characterization. Let Ω be a finite state space, with $\omega \in \Omega$ denoting a state of the world. A finite collection of binary YES contracts, indexed by $i \in I$, trades at quoted prices $\mathbf{p} = (p_i)_{i \in I} \in [0, 1]^I$, and the payoff map $\phi : \Omega \rightarrow \{0, 1\}^I$ records the amount $\phi_i(\omega)$ that one share of contract i pays at resolution in state ω . We maintain binary settlement in the modeled state space.

Section 2.1 establishes the YES–NO payoff equivalence, so we express all positions and quotes in YES-equivalent units. A *portfolio* $\mathbf{y} \in \mathbb{R}^I$ has initial cost $\mathbf{y}^\top \mathbf{p}$ and terminal payoff $\mathbf{y}^\top \boldsymbol{\phi}(\omega)$; negative components of $\mathbf{y}$ are short positions, and a negative initial cost means cash is received at inception. The portfolio is held until resolution, at which point its terminal payoff is realized.⁵ Definition 1 and Proposition 1 formalize arbitrage and price coherence. Proposition 2 characterizes the largest guaranteed profit and an optimal portfolio.

Definition 1 (Arbitrage). A portfolio $\mathbf{y} \in \mathbb{R}^I$ is an *arbitrage* at prices $\mathbf{p}$ if its *guaranteed profit*, the minimum profit across states, is strictly positive:

$$\min_{\omega \in \Omega} \mathbf{y}^\top \boldsymbol{\phi}(\omega) - \mathbf{y}^\top \mathbf{p} > 0, \quad (1)$$

that is, if its *worst-case terminal payoff* $\min_{\omega} \mathbf{y}^\top \boldsymbol{\phi}(\omega)$ exceeds its initial cost.

Proposition 1 (No arbitrage $\iff$ coherent prices). Let $M := \text{conv}\{\boldsymbol{\phi}(\omega) : \omega \in \Omega\}$ denote the convex hull of the payoff vectors across all states. No arbitrage exists at prices $\mathbf{p}$ if and only if $\mathbf{p} \in M$.

Proposition 2 (Guaranteed profit as ℓ_1 distance). Each portfolio’s *guaranteed profit* is its *worst-case profit*; under the unit-position normalization $\|\mathbf{y}\|_\infty \leq 1$, the largest profit any portfolio can guarantee equals the ℓ_1 distance from the quote vector to the coherent set,

$$\max_{\|\mathbf{y}\|_\infty \leq 1} \min_{\omega \in \Omega} \{\mathbf{y}^\top \boldsymbol{\phi}(\omega) - \mathbf{y}^\top \mathbf{p}\} = \min_{\boldsymbol{\mu} \in M} \|\boldsymbol{\mu} - \mathbf{p}\|_1, \quad (2)$$

and an optimal portfolio buys every contract priced below the minimizing coherent price vector $\boldsymbol{\mu}^*$ and sells every contract priced above it.

Proofs of both propositions are in Appendix A.1. The two propositions carry the framework’s economic content. M collects the expected payoff vectors generated by all beliefs over states: each point of M is $\mathbb{E}_q[\boldsymbol{\phi}]$ for some belief q , so $\mathbf{p} \in M$ means that one probability distribution prices every contract at its expected payoff, and the quotes reflect a single coherent belief. If $\mathbf{p}$ falls outside M , no belief rationalizes the quotes and an arbitrage exists; the best portfolio guarantees exactly the ℓ_1 distance from the quotes to the coherent set, and the separating direction $\mathbf{y}$ is the trade itself.

Figure 1 illustrates two contract pairs of different types drawn from the data. Panel (a) shows two mutually exclusive (not collectively exhaustive) contracts “Exact Score: New Zealand 0–2 Belgium?” and “New Zealand vs. Belgium: Over 2.5 goals”: a 0–2 final score has two goals and cannot coincide with more than 2.5 goals in total, so the only admissible payoff vectors are $(0, 0)$, $(1, 0)$, and $(0, 1)$, whose convex hull is the blue triangle. The fee-adjusted bid vector $(0.1756, 0.7850)$ lies inside it; five minutes later, the fee-adjusted bid vector $(0.2247, 0.8973)$ lies outside the no-arbitrage region at an ℓ_1 distance of 0.1220 from it, so selling one share of each contract guarantees \$0.1220 after fees.

⁵The empirical guarantees are conditional on binary settlement, so Section 5.1 evaluates only windows whose filled legs resolve to zero or one. In the quoted-price benchmark, the portfolio can be unwound before resolution once quotes return to the coherent set. With finite-depth order books, however, future liquidation spreads, depth, and fees are unknown when the portfolio is formed, so early-exit profit cannot be guaranteed ex ante. Appendix A.3 considers this possibility; throughout the empirical analysis, we therefore hold portfolios to resolution.

(a) New Zealand 0–2 Belgium vs. over 2.5 goals (b) Cumberland–Heidelberg: over 3.5 vs. 2.5 goals

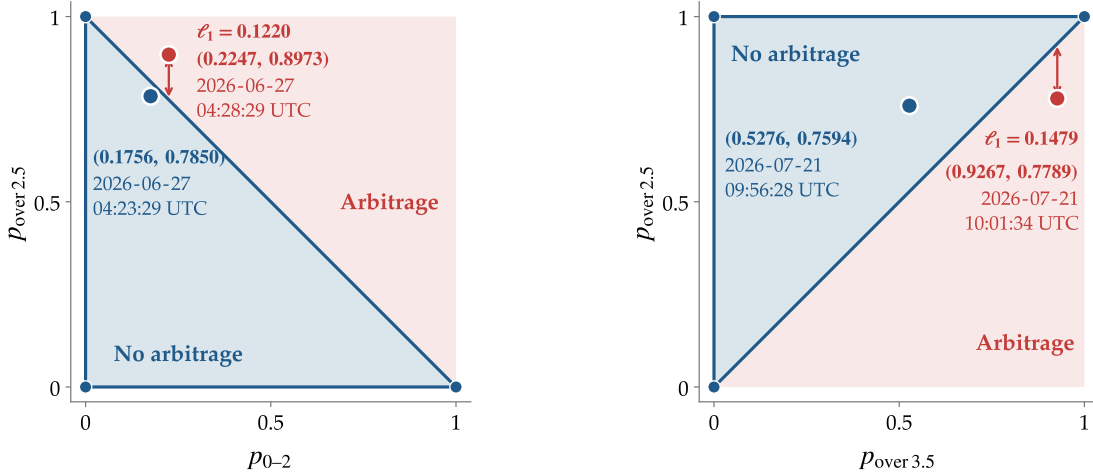


Figure 1: Price coherence in two football contract pairs, June and July 2026. Panels (a) and (b) show the New Zealand–Belgium exact score 0–2 versus over 2.5 goals and the Cumberland–Heidelberg over 3.5 goals versus over 2.5 goals; arrows mark the ℓ_1 distance to coherence.

Panel (b) shows two nested contracts from an Australia Cup match, “**Cumberland United vs. Heidelberg United: Over 3.5 goals**” and “**Cumberland United vs. Heidelberg United: Over 2.5 goals**”: more than 3.5 goals are also more than 2.5 goals, so the admissible payoff vectors (0,0), (0,1), and (1,1) place the coherent set above the diagonal. The fee-adjusted bid for the over-3.5 contract and ask for the over-2.5 contract form the execution quote vector: (0.5276,0.7594) is coherent, whereas five minutes later (0.9267,0.7789) lies 0.1479 below the diagonal, so selling the over-3.5 contract and buying the over-2.5 contract guarantees \$0.1479 after fees. Figure A.1 in Appendix A.4 draws the same geometry for partition and conjunction relations.

3.2 Detecting Combinatorial Arbitrage via Integer Programming

When only two or three contracts are involved, the corresponding arbitrage can be easily identified geometrically, as illustrated in Figure 1 for two contracts and Figure A.1 for three. As the number of related contracts grows, however, this geometric characterization becomes less tractable. We therefore formulate the problem as an integer program (Schrijver, 1986; Nemhauser and Wolsey, 1988).

The program must first determine a given portfolio’s lowest payoff across all payoff vectors admitted by the logical relation system. Let $z \in \{0,1\}^I$ denote a joint payoff vector, where $z_i = 1$ exactly when contract i resolves YES; there are up to $2^{|I|}$ possible binary payoff vectors. Each logical relation excludes some vectors and compiles into the linear constraint rows of Table 2. Stacking the constraint rows into $Fz \leq g$ yields the admissible payoff set $\Phi := \{z \in \{0,1\}^I : Fz \leq g\}$, which coincides with $\{\phi(\omega) : \omega \in \Omega\}$ when the compiled relations exactly characterize the payoff set induced by the state space, so the coherent set of Section 3.1 is $M = \text{conv}(\Phi)$, the integer hull of the relaxed polytope $\{z \in [0,1]^I : Fz \leq g\}$. Properly speaking, the constraint system is the triple (I, F, g) , but when no confusion can arise we write (F, g) , or simply F , for it.

Table 2: Logical relations, their constraint rows, and the licensed trades

Relation	Meaning	Constraints	Violation at quotes	Licensed trade
Mutual exclusivity of A and B	At most one of the two contracts pays	$z_A + z_B \leq 1$ ⁶	$\tilde{b}_A + \tilde{b}_B > 1$	Sell one share of A and one of B
Partition of $A_1, \dots, A_n$	Exactly one of the n contracts pays	$z_{A_1} + \dots + z_{A_n} \leq 1$ $z_{A_1} + \dots + z_{A_n} \geq 1$	$\sum_i \tilde{b}_{A_i} > 1$ $\sum_i \tilde{a}_{A_i} < 1$	Sell one share of every contract Buy one share of every contract
Nesting A implies B	Whenever A pays, B pays	$z_A \leq z_B$	$\tilde{b}_A > \tilde{a}_B$	Sell one share of A , buy one share of B
Conjunction $C = A \wedge B$	C pays exactly when both A and B pay	$z_C \leq z_A$ $z_C \leq z_B$ $z_C \geq z_A + z_B - 1$	$\tilde{b}_C > \tilde{a}_A$ (or $\tilde{a}_B$) $\tilde{a}_C < \tilde{b}_A + \tilde{b}_B - 1$	Sell C , buy A (or B) Buy C , sell A and B

Notes: $z_i \in \{0, 1\}$ is the payoff of contract i , and $\tilde{b}_i$ and $\tilde{a}_i$ are its fee-adjusted best bid and best ask.

The system in fact changes over time: the admissible payoff set is $\Phi_t := \{z \in \{0, 1\}^{I_t} : F_t z \leq g_t\}$ at time t , where I_t is the set of contracts listed at t and $F_t z \leq g_t$ stacks the rows compiled from the relations in effect at t , that is, the relations whose contracts are all listed at t . We take the time-aggregated system (I_*, F_*, g_*) to be a reduced system such that, for every t , $I_t \subseteq I_*$ and the rows of $F_* z \leq g_*$ in effect at t form a subset of the rows of $F_t z \leq g_t$ that implies the remaining rows for every binary z , so that these rows alone define Φ_t . Every proposition is stated for a general (I, F, g) and therefore holds at each t . Detection at t uses (I_t, F_t, g_t) , and Section 4 describes the aggregate F_* .

Given the compiled constraint system (I, F, g) , the largest guaranteed profit can be written as $\max_{\|y\|_\infty \leq 1} \{(\min_{z \in \{0, 1\}^I : Fz \leq g} y^\top z) - y^\top p\}$. Arbitrage detection then proceeds in two nested steps.

Inner problem. Fix a proposed portfolio y . Among all admissible payoff vectors, the integer program finds the one under which that portfolio pays least:

$$V(y) := \min_z y^\top z \quad \text{subject to } Fz \leq g, \quad z \in \{0, 1\}^I. \quad (3)$$

$V(y)$ equals the portfolio's worst-case terminal payoff: because payoffs depend on the state only through its payoff vector, minimizing over the finite set Φ recovers exactly the minimum over Ω in Definition 1, attained at a vertex of M . The portfolio is an arbitrage if and only if $V(y) > y^\top p$.

Outer problem. The portfolio y is not fixed in advance. We now optimize over the portfolio itself, searching over normalized portfolios for the one with the largest guaranteed profit:

$$\pi^* = \max_{\|y\|_\infty \leq 1} \{V(y) - y^\top p\}. \quad (4)$$

A positive π^* identifies an arbitrage, the optimizer y^* gives the trade, and π^* equals the ℓ_1 distance of Proposition 2.

⁶For binary payoffs z_i , $i \in I$, mutual exclusivity of every pair (pairwise exclusivity) is equivalent to mutual exclusivity of the whole set ($\sum_{i \in I} z_i \leq 1$); see the proof in Appendix A.1.

Solving the nested problem. To see how the two problems fit together, let η denote a portfolio's claimed worst-case terminal payoff. Each admissible payoff vector z imposes the inequality $y^\top z \geq \eta$; imposing this inequality for every $z \in \Phi$ and maximizing $\eta - y^\top p$ reproduces the outer problem in (4). Direct evaluation would therefore require enumerating all admissible payoff vectors, whose number can grow exponentially with the number of related contracts.⁷

Algorithm 1 may help avoid full enumeration: we start with a small working set Φ_{work} of admissible payoff vectors, initialized here as a singleton. The MASTER step chooses the portfolio and its claimed payoff (y, η) that maximize $\eta - y^\top p$, subject to $y^\top z \geq \eta$ for every $z \in \Phi_{\text{work}}$. The ORACLE step then solves the inner problem in (3) and identifies the admissible payoff vector under which the current portfolio pays least. If that vector pays less than η , it disproves the claim and enters the working set; otherwise the claimed payoff holds over all of Φ , and the current portfolio solves the full problem. Each failed claim adds a new vector from the finite set Φ , ensuring finite termination; the worst case may still require every vector. Algorithm 1 is thus the cutting-plane method of Kelley (1960) applied to the concave piecewise-linear function V in (3), with the integer program in (3) acting as the separation oracle in the sense of Grötschel, Lovász, and Schrijver (1988).

Algorithm 1 Cutting-plane arbitrage detector

Require: quote vector p ; constraints (F, g) ; any admissible payoff vector z_0

```

1:  $\Phi_{\text{work}} \leftarrow \{z_0\}$ 
2: loop
3:    $(y, \eta) \leftarrow \text{MASTER}(p, \Phi_{\text{work}})$  ▷ best claimed payoff under known payoff vectors
4:    $z_{\text{new}} \leftarrow \text{ORACLE}(y, F, g)$  ▷ worst-case payoff vector
5:   if  $y^\top z_{\text{new}} \geq \eta$  then
6:     return  $y$  and  $\pi = \eta - y^\top p$  ▷ claimed payoff certified
7:   end if
8:    $\Phi_{\text{work}} \leftarrow \Phi_{\text{work}} \cup \{z_{\text{new}}\}$  ▷ add violated constraint
9: end loop
```

In Appendix A.2, we derive the linear-programming reformulation behind the MASTER step and describe the caching that carries discovered payoff vectors across quote states.

3.3 Measuring Executable Arbitrage in Finite-Depth Order Books

The quoted-price benchmark abstracts from two features of real execution. First, execution costs depend on bid–ask spreads, depth at each price level, and taker fees. Second, the unit-position normalization identifies the optimal portfolio direction but not its executable size. To address both limitations, we replace the linear quoted-price cost $y^\top p$ with the actual execution cost, and we keep the payoff side of Sections 3.1 and 3.2 unchanged. We then answer three questions in turn: when an executable arbitrage exists, how large it is, and how Algorithm 1 computes it.

⁷Most detected opportunities involve only a few contracts, but the detector is evaluated at every relevant best-quote update. A five-contract connected component has $2^5 = 32$ possible binary payoff vectors; at the scale of about 3.6 billion best-quote states in our sample, repeating even this small enumeration would entail about 115 billion payoff-vector evaluations. Even small payoff sets can therefore be expensive to enumerate repeatedly across the full sample.

The execution cost of one contract. For contract i , we arrange its bids and asks with their quantities into a price ladder with A_i ask levels and B_i bid levels, such that the fee-adjusted bid prices $\tilde{b}_{i\ell}$ and ask prices $\tilde{a}_{i\ell}$ are ordered from lowest to highest $\tilde{b}_{iB_i} \leq \dots \leq \tilde{b}_{i2} \leq \tilde{b}_{i1} \leq \tilde{a}_{i1} \leq \tilde{a}_{i2} \leq \dots \leq \tilde{a}_{iA_i}$:

$$\underbrace{(\tilde{b}_{iB_i}, q_{iB_i}^b), \dots, (\tilde{b}_{i2}, q_{i2}^b), (\tilde{b}_{i1}, q_{i1}^b)}_{\text{selling more walks down the bids}}, \quad \underbrace{(\tilde{a}_{i1}, q_{i1}^a), (\tilde{a}_{i2}, q_{i2}^a), \dots, (\tilde{a}_{iA_i}, q_{iA_i}^a)}_{\text{buying more walks up the asks}}. \quad (5)$$

Let $K_i(s)$ denote the total cost of changing the position in contract i by s shares, with $s > 0$ for buying and $s < 0$ for selling. We obtain $K_i(s)$ by moving stepwise up or down the ladder: as an order reaches deeper price levels, additional shares are bought at higher asks or sold at lower bids. For example, a purchase of s shares with $q_{i1}^a + q_{i2}^a < s \leq q_{i1}^a + q_{i2}^a + q_{i3}^a$ costs $K_i(s) = \tilde{a}_{i1}q_{i1}^a + \tilde{a}_{i2}q_{i2}^a + \tilde{a}_{i3}(s - q_{i1}^a - q_{i2}^a)$. A sale of $|s|$ shares walks down the bids in the same way, and its cost is equal to minus the proceeds, so $K_i(s) < 0$. Trading nothing costs nothing, so $K_i(0) = 0$. A trade beyond the displayed depth cannot be executed, so $K_i(s) = +\infty$ for s outside $[-Q_i^b, Q_i^a]$, where $Q_i^a := \sum_{\ell=1}^{A_i} q_{i\ell}^a$ and $Q_i^b := \sum_{\ell=1}^{B_i} q_{i\ell}^b$ are the displayed depths on the ask and bid sides.

Executable portfolios and their profit. The execution cost of a portfolio $\mathbf{y}$ is the sum of the costs of its legs, $K(\mathbf{y}) := \sum_{i \in I} K_i(y_i)$, so $K(\mathbf{0}) = 0$. A portfolio can be executed exactly when its cost is finite, so the *executable set* is

$$D := \text{dom } K = \{\mathbf{y} \in \mathbb{R}^I : K(\mathbf{y}) < +\infty\} = \prod_{i \in I} [-Q_i^b, Q_i^a], \quad (6)$$

a box whose sides are the displayed depths. For a portfolio $\mathbf{y} \in D$, the executable profit under payoff vector $\mathbf{z} \in \Phi$ is $\mathbf{y}^\top \mathbf{z} - K(\mathbf{y})$, and its guaranteed executable profit is

$$\min_{\mathbf{z} \in \Phi} \{\mathbf{y}^\top \mathbf{z} - K(\mathbf{y})\} = \min_{\mathbf{z} \in \Phi} \mathbf{y}^\top \mathbf{z} - K(\mathbf{y}). \quad (7)$$

The payoff side is unchanged, whereas $K(\mathbf{y})$ records the prices, depth, and fees consumed along the order book. In parallel with Definition 1, a portfolio $\mathbf{y} \in D$ is an *executable arbitrage* if its guaranteed executable profit is strictly positive.

Proposition 3 extends the single-price benchmark of Proposition 1 by incorporating trading costs. No executable arbitrage exists if and only if a single belief can value every contract within its fee-adjusted bid–ask spread. Spreads and fees therefore allow some incoherence among quoted prices without creating a profitable trade: arbitrage requires discrepancies that persist at actual execution prices (Jouini and Kallal, 1995; Pennanen, 2011). Appendix A.1 provides the proof and recovers Propositions 1 and 2 as the special case of the linear quoted-price cost.

Proposition 3 (Executable arbitrage). *Over the executable set D , the maximum guaranteed executable profit*

$$\Pi^* = \max_{\mathbf{y} \in D} \min_{\mathbf{z} \in \Phi} \{\mathbf{y}^\top \mathbf{z} - K(\mathbf{y})\} \quad (8)$$

is nonnegative and is attained at an optimal executable portfolio $\mathbf{y}^$. An executable arbitrage exists, that is, $\Pi^* > 0$, if and only if no coherent price vector lies within every fee-adjusted spread:*

$$M \cap \prod_{i \in I} [\tilde{b}_{i1}, \tilde{a}_{i1}] = \emptyset. \quad (9)$$

While the best quotes $\tilde{b}_{i1}$ and $\tilde{a}_{i1}$ determine whether an arbitrage exists, the full order book determines its optimal scale and total guaranteed profit. As a portfolio expands, higher purchase prices and lower sale proceeds erode the gain from additional trades. A large price discrepancy supported by little depth can therefore offer less profit than a smaller discrepancy in a deeper market.

Algorithm 1 carries over with the same payoff constraints and worst-case oracle; the MASTER step now uses the order-book cost K and the executable set D and, because K is piecewise linear, remains a linear program. The cap on the number of legs in Section 5.1 is the exception: the portfolios that satisfy it do not form a convex set, so the MASTER step becomes a mixed-integer linear program. Appendix A.2 provides computational details.

Section 5.1 measures executable arbitrage in two steps before settlement: the detector uses the best quotes to identify arbitrage windows, and the trade simulation reoptimizes quantities on the detected legs within each window using the full-depth order book. This separates the detection of inconsistent prices from the measurement of the profit available at displayed quantities.

4 Recovering the Logical Relation System at Scale

Section 3 takes the constraint system (F, g) as given. Prediction-market venues, however, list contracts individually and do not publish the complete logical structure. Section 4.1 first defines the Type I and Type II errors that can arise when recovering these relations and shows why their effects on arbitrage detection are asymmetric. Section 4.2 then explains how we recover the logical structure in three layers.

4.1 Scale and Relation Errors

In Section 3, we implicitly assumed a complete and valid logical relation system. We now examine two possible errors in recovering this system. First, N contracts generate $N(N - 1)/2$ possible pairs and up to 2^N subsets for multivariate search. Exhaustive search is therefore infeasible, so valid relations may remain unidentified. Second, a proposed relation may itself be incorrect. We call the admission of an invalid relation a Type I error and the omission of a valid relation a Type II error.

A Type I error directly changes the objects defined in Section 3. Adding a false constraint to $Fz \leq g$ can exclude a payoff vector that is feasible under the true relation system, producing an incorrectly small payoff set $\tilde{\Phi} \subsetneq \Phi$ and coherent set $\tilde{M} = \text{conv}(\tilde{\Phi})$. The detector then minimizes over the wrong subset, so the calculated worst-case payoff can exceed the true worst-case payoff. The reported guaranteed profit and the portfolio weights y^* chosen to maximize it may both be wrong; in an excluded feasible state, the same portfolio can lose money. Repeated small apparent gains can therefore conceal a large loss when that state occurs—the “picking up pennies in front of a steamroller” problem discussed by Stulz (2007, p. 182).

A Type II error instead leaves out a valid constraint and therefore expands the payoff set. Proposition 4 shows that this error has a more limited consequence. Appendix A.1 gives the proof.

Proposition 4 (False omission is conservative). *Let Φ be the payoff set generated by the full set of valid logical relations, and let $\Phi^- \supseteq \Phi$ be the payoff set obtained by omitting valid relations without adding invalid ones. For any execution-cost function K , executable set $D = \text{dom } K$, and portfolio $\mathbf{y} \in D$,*

$$\min_{\mathbf{z} \in \Phi^-} \{\mathbf{y}^\top \mathbf{z} - K(\mathbf{y})\} \leq \min_{\mathbf{z} \in \Phi} \{\mathbf{y}^\top \mathbf{z} - K(\mathbf{y})\}. \quad (10)$$

The maximal guaranteed profit computed under Φ^- therefore cannot exceed that computed under Φ . Any portfolio classified as an arbitrage under Φ^- remains an arbitrage under the true payoff set Φ .

Type II errors are not costless, though: they reduce detection power and the associated profit opportunities. The proposition shows only that relations left unidentified because of scale cannot manufacture a guaranteed profit that does not exist. We therefore seek broad coverage while requiring stronger evidence before admitting a relation. The next subsection describes how three complementary layers pursue both goals.

4.2 Recovering Relations in Three Layers

We build the relation system in three layers, as shown in Figure 2. First, we obtain *venue-asserted* relations from the venue’s collateral mechanism and the rules of the groups it designates. Second, for sports contracts, we construct *venue-derived* relations by mapping typed fields—the match, contract type, and line—into a common score state. Third, we extract *semantic-inferred* relations from natural-language contract text. Each layer uses evidence suited to its source and applies its checks before a relation enters the system.

Layer 1: Venue-asserted relations. Polymarket places contracts that it treats as mutually exclusive in the same negative-risk group, identified by a common `negRiskMarketID`. We check these mutual-exclusivity relations against the venue’s collateral mechanism. For example, group `0x143fa1...cc00` contains three contracts for the Algeria–Austria World Cup match: “**Algeria wins**”, “**the match ends in a draw**”, and “**Austria wins**”. Exactly one of these three contracts resolves YES, so we encode the group as the partition restriction $z_{\text{Alg}} + z_{\text{Draw}} + z_{\text{Aut}} = 1$. The 20,866 negative-risk groups in our sample yield 20,118 partition and 748 mutual-exclusivity restrictions.⁸

Layer 2: Venue-derived relations. In some contract classes, restrictions are not encoded directly by the venue but follow deterministically from its typed metadata. For sports contracts, Polymarket reports the match, contract type, and line. We map these fields into a common score state that determines each contract’s payoff. This mapping reveals the logical relations among contracts on the same match. The venue-derived layer produces more relations than either of the other two layers (Table 4). We retain only relations verified by the deterministic score-state mapping.

⁸Negative-risk group membership alone is not sufficient to guarantee collective exhaustiveness: the venue rules out more than one YES outcome but not an all-NO outcome. We therefore read the complete contract rules of every group. For 20,118 of the 20,866 groups the rules establish collective exhaustiveness, and then we impose equality; at resolution, every settled complete basket in these groups paid one dollar. For example, in the Algeria–Austria group, the draw contract would resolve to YES if the match were cancelled entirely with no make-up game, so cancellation would not create a fourth outcome. The remaining 748 groups, whose rules contradict exactly-one settlement or which we could not assess, retain only the mutual-exclusivity inequality. Appendix A.4.2 shows how this difference affects buying and selling a group.

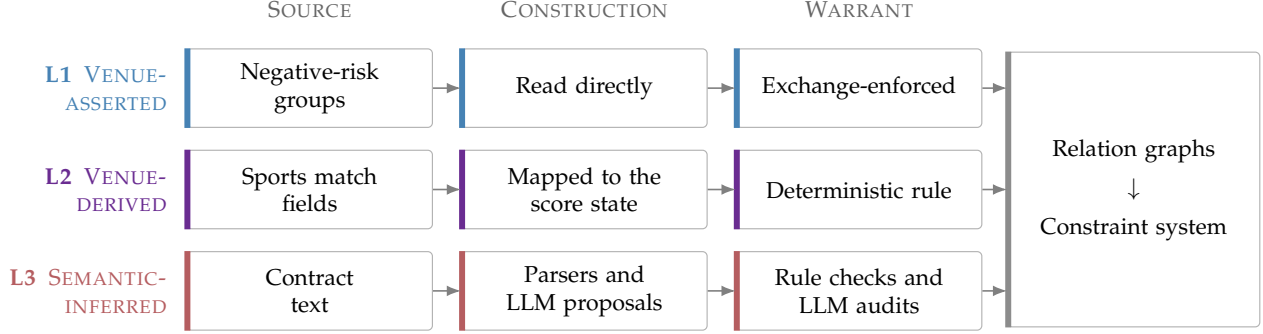


Figure 2: Three relation layers and their checks.

Layer 3: Semantic-inferred relations. For relations that are not encoded in Layers 1 or 2, we use the natural-language contract text. Some contracts contain standardized numerical thresholds or deadlines that can be parsed mechanically. For example, “Bitcoin above \$65,000 on July 28, 5AM ET?” implies “Bitcoin above \$64,800 on July 28, 5AM ET?”. Our parsers first match the underlying event, measured quantity, unit, horizon, resolution source, and operative deadline. They then infer only the relations implied by those matched fields.

LLM proposals and auditing. Our parsers cover standardized patterns, but other relations require interpreting free-form resolution criteria. We use large language models (LLMs) to search for these relations. Because comparing every contract pair is infeasible, we first pre-screen the pairs. We rank contract pairs by weighted word overlap across their event titles, questions, and group titles, giving more weight to distinctive words and phrases. This pre-screen cuts 277 billion possible pairs to 704,450 candidate pairs and 13,894 candidate groups. Appendix B.2.1 describes the retrieval algorithm and group construction.

We use five leading open-weight LLMs available in early September 2026 as *proposal models*: Qwen3-32B (Yang et al., 2025), Qwen3-235B-A22B (Qwen Team, 2025), Llama-4-Maverick (Meta AI, 2025), DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), and GLM-5.2 (Z.ai, 2026). For each candidate, the proposal models receive the relevant contract metadata and natural-language text. Each model must return a relation type and direction, exact textual evidence, and an explanation, or abstain when the evidence is insufficient. In addition, because proposal models are error-prone, every LLM proposal is reviewed by two *audit models* from different developers: Claude Fable 5.1 in Claude Code at max reasoning effort and GPT-5.6 Sol (later supplemented by GPT-6 Astra) in Codex at max or ultra reasoning effort. Both are frontier LLMs available as of September 2026 and are deployed as agents with the reasoning-effort settings specified above. Each auditor independently checks the contract text and dates, searches for a counterexample, and determines whether a relation exists and, if so, what type of relation the evidence supports. We retain a proposal only when both auditors agree on its type and direction; all other proposals are removed. Appendix B.2 reports the prompts and audit examples.

Table 3: Audit-model agreement by proposal model

Proposal source	Parameters (billions)		Pairwise relations		Multivariate units	
	Active	Total	Proposed	Confirmed	Proposed	Confirmed
Llama-4-Maverick	17	400	99,361	18,651 (18.8%)	8,818	349 (4.0%)
Qwen3-32B	32.8	32.8	90,398	23,084 (25.5%)	7,871	283 (3.6%)
DeepSeek-V4-Flash	13	284	128,084	49,729 (38.8%)	9,298	702 (7.6%)
Qwen3-235B-A22B	22	235	135,202	58,963 (43.6%)	6,007	488 (8.1%)
GLM-5.2	40	744	137,055	88,571 (64.6%)	2,011	437 (21.7%)
Union set	–	–	239,653	102,288 (42.7%)	23,519	1,153 (4.9%)

Notes: Confirmed means that both auditors confirm the model’s original relation.

In Table 3, the percentage of proposals that both auditors confirm varies widely across the five proposal models and tracks model size only loosely. GLM-5.2 has the highest confirmation rate: it proposes 137,055 pairwise and 2,011 multivariate relations, of which 64.6% and 21.7%, respectively, are confirmed by both audit models.

The relation system. Table 4 reports the resulting system. Panel A counts the 1,942,133 relations the three layers recover, including the ones implied by others; Panel B counts the 573,024 relations of the reduced system, of which 5,268 appear in two layers; these relations compile into 603,242 rows of F_* . Each relation compiles into the rows of Table 2, and the last column of Table 4 counts these rows by layer. Layer 2 contributes the most because the score-state mapping is deterministic and applies to every contract carrying the venue’s typed sports fields, the largest contract class in our sample. Layer 3 contributes 118,792 relations after deduplication across the dual-audited LLM outputs of Table 3 and the outputs of the numeric and temporal parsers.

Coverage of semantic-inferred relations need not be complete. Candidate retrieval, model abstention, and limited computation can all omit valid relations, creating Type II errors. Such omissions reduce detection power, but they cannot manufacture an opportunity that is not there. More accurate models and greater computation could search a wider candidate set and reduce these omissions further. By Proposition 4, provided the retained relations are valid, the guaranteed profit computed on an incomplete system is a lower bound on the guaranteed profit under the complete one, so every portfolio we certify remains an arbitrage once the missing relations are restored.

Figure 3 shows how the three layers contribute differently across event classes and logical relation types. Layer 1 is mostly partitions: 20,118 of the 20,866 negative-risk groups in our sample are compiled as exhaustive partitions. Layer 2 compares the payoffs of contracts on the same match—exact scores, moneylines, spreads, totals, both-teams-to-score, and team totals—across all possible scores. It is this common underlying score state that enables a single layer to recover all four logical relation types. Within Layer 2, nesting is the most common type, with 246,848 relations. Layer 3 reaches the classes largely left uncovered by the other two layers and is dominated by nesting relations among threshold contracts in crypto and finance.

Table 4: Recovered and reduced relations by layer

Relation layer	Mutual exclusivity	Partition	Nesting	Conjunction	Other	Relations	Rows
<i>Panel A: Recovered relations in F</i>							
Layer 1: Venue-asserted	748	20,118	0	0	0	20,866	40,984
Layer 2: Venue-derived	870,389	5,254	480,310	2,370	0	1,358,323	1,368,317
Layer 3: Semantic-inferred	25,448	25	537,381	0	90	562,944	563,050
Total	896,585	25,397	1,017,691	2,370	90	1,942,133	1,972,351
<i>Panel B: Reduced relations in F_*</i>							
Layer 1: Venue-asserted	748	20,118	0	0	0	20,866	40,984
Layer 2: Venue-derived	178,894	5,254	246,848	2,370	0	433,366	443,360
Layer 3: Semantic-inferred	14,292	25	104,385	0	90	118,792	118,898
Total	193,934	25,397	351,233	2,370	90	573,024	603,242

Notes: Panel A counts recovered relations, with identical constraints merged within each layer. Panel B reports counts for the reduced system. Relations shared by different layers are counted in each layer. Other counts constraint rows not in the logical relation types of Table 2.

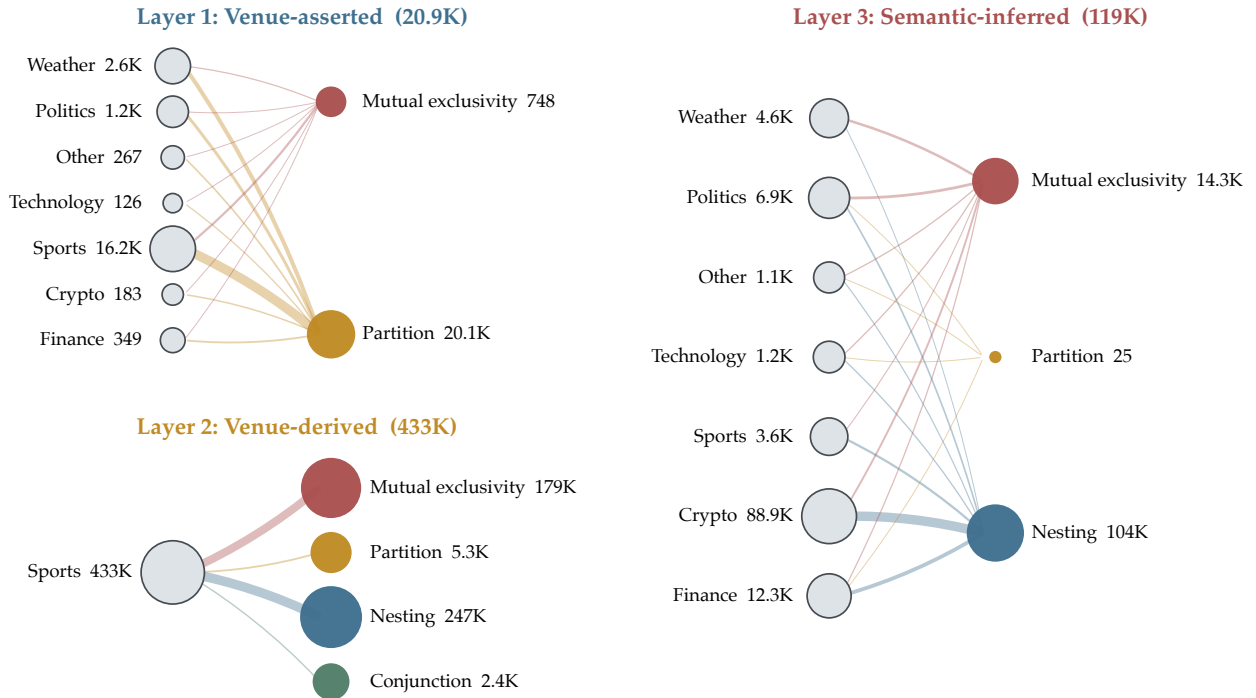


Figure 3: Reduced relations in F_* by layer, event class, and logical relation type. Each panel maps one layer’s reduced relations from event class to logical relation type; node diameter is linear in the logarithm of the count, and the labels give the counts.

The overall relation system is highly concentrated in two contract classes: sports accounts for 453,190 relations and crypto for 89,088, together representing 94.6% of the total, while no other class exceeds 3%. Two forces produce this imbalance. Sports and crypto are the venue’s largest classes, accounting for 67.5% and 22.4% of its listed contracts. Moreover, their contracts are typically expressed as typed lines and numeric thresholds, so their relations can be recovered by score mappings or structured parsers without individual LLM evaluation.

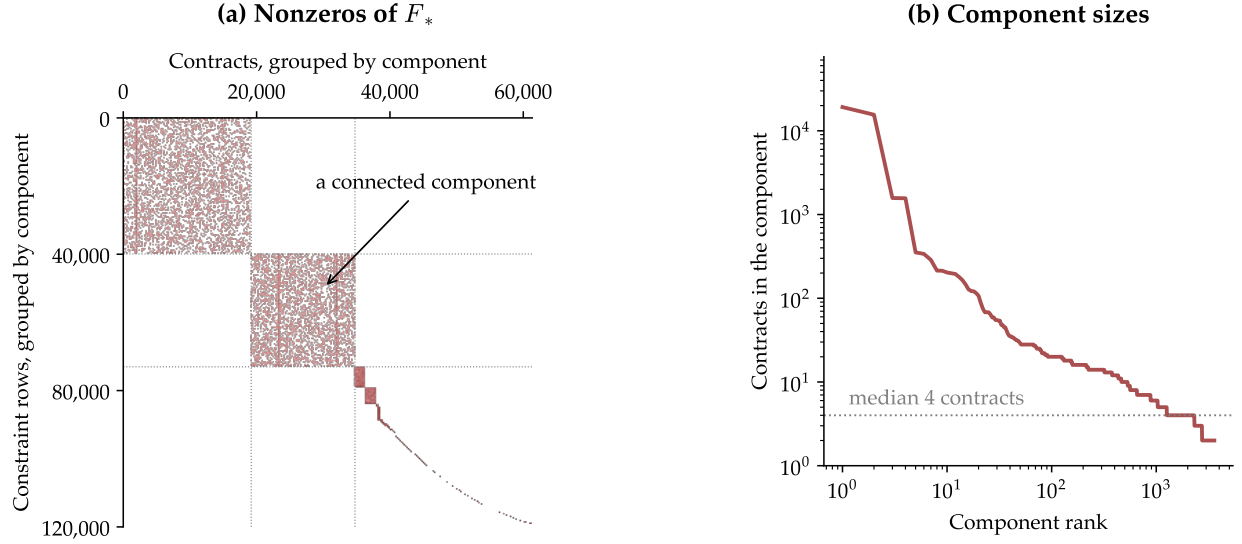


Figure 4: Connected components of the semantic layer’s constraint matrix F_* . Panel (a) shows the nonzeros of the semantic-inferred layer’s F_* on a $1,200 \times 1,200$ grid of cells, each covering about 100 rows and 50 contracts, with rows and columns ordered by connected component from largest to smallest and each cell shaded by its number of nonzeros on a log scale; panel (b) shows the number of contracts in each component against its rank.

Connected components. Contracts linked by a chain of relations form a *connected component*: the smallest set that must be priced jointly, because no constraint reaches across its boundary. We group the rows of each layer’s system by the contracts they share. For the semantic-inferred layer, whose F_* has 118,898 rows over 61,349 contract columns, the system separates into 3,591 connected components, with a median size of 4 contracts and a maximum of 19,188. Figure 4 shows the matrix with rows and columns permuted so that each component occupies a diagonal block.

The two largest consist of Bitcoin and Ethereum price-threshold contracts, with 19,188 and 15,566 contracts. These components are large because two types of relations chain together almost all contracts on the same coin: “Bitcoin hits \$65,000 on July 28” implies “Bitcoin hits \$64,000 on July 28”, and the former contract, by temporal nesting, also implies “Bitcoin hits \$65,000 in July” while being implied by “Bitcoin above \$65,000 on July 28, 5AM ET?”. Thus, within one horizon every threshold is tied to every lower threshold for reaching and to every higher threshold for dipping, while each contract is tied to the reach or dip contracts of every containing horizon, causing 19,188 of the 19,843 Bitcoin and 15,566 of the 16,119 Ethereum price-threshold contracts listed during the sample to fall into a single component each. The blocks are shaded by the number of nonzeros per cell; within a block the contracts are in no particular order, so the nonzeros spread over the block. The remaining 3,589 components form the thin diagonal; 87% of them hold ten contracts or fewer.

The venue-asserted and venue-derived layers do not enter this matrix: we compile each layer into its own constraint system, and Section 5.1 detects arbitrage separately within each, so that profits supported by different layers stay distinct and an invalid relation in one layer cannot create a false guarantee in another. Appendix B.3 details the detection procedure for each layer.

5 The Anatomy of the Arbitrage Opportunity

We now combine the logical relation system of Section 4 with the full-depth order books to measure combinatorial arbitrage. Section 5.1 explains how we detect arbitrage windows, measure their executable size, and compare their guaranteed profit with their profit at resolution. Section 5.2 reports the resulting opportunities and examines their sensitivity to execution assumptions. Section 5.3 examines the conditions under which arbitrage opportunities persist longer, and Section 5.4 uses on-chain fills to examine who takes these opportunities.

5.1 Measurement: Detection, Execution, and Settlement

We measure arbitrage in three steps. Best quotes identify arbitrage windows, full-depth order books determine the portfolios that can be executed, and contract resolutions determine their profit at resolution. Appendix B.4 details the data construction and checks.

Step 1: detect arbitrage windows using only the best quotes. Figure 4 shows how related contracts form connected components, with no constraints across components. We can therefore run the detector separately on each component of each layer’s system. A component can be large, but at any given instant only some of its contracts are quoted, so only the relations among them that are in effect at that instant apply. We call the quoted contracts and these constraints the component’s *active block*.

We reconstruct each order book on the exchange clock, applying all messages timestamped at or before the instant being evaluated.⁹ Whenever a best quote in the active block changes, we solve the arbitrage problem in Section 3.2 using current bids and asks after taker fees. An *arbitrage window* opens when the detector finds a portfolio with positive guaranteed profit at the best quotes. It remains open while successive evaluations return positive profit and closes at the first evaluation that does not.¹⁰ This step identifies 491,939 arbitrage windows.

Step 2: simulate arbitrage trades against full-depth order books. Full-depth simulation is needed only for the arbitrage windows identified in Step 1: if the best quotes are arbitrage-free, then deeper quotes, which are weakly less favorable, cannot yield arbitrage (Proposition 3). Within an arbitrage window, best quotes identify a profitable portfolio direction, but do not determine how much can be traded. To determine executable size, we have to incorporate the depth available at each price level. Figure 5(a) illustrates an example with two contracts. We therefore replay the live order-book messages in exchange-time order to reconstruct the full-depth books throughout each arbitrage window. At every update, we reoptimize quantities on the detected legs, retaining their trading directions, and simulate trades against available depth to determine executable size and guaranteed profit.

⁹We run data collectors that receive order-book messages from Polymarket over multiple network connections. In practice, delays differ across connections and can reach several minutes, so we align messages by exchange timestamps rather than their receipt times. To check reconstruction accuracy, we also compare each book’s best bid and ask with the exchange’s reported quotes, which agree in 97.2% of 13.3 billion comparisons.

¹⁰We cap the active block at 32 contracts in the venue-asserted layer, 48 in the venue-derived layer, and 16 in the semantic-inferred layer. We cap portfolios at 5 legs because coordinating trades across more order books can be difficult in practice. The 5-leg cap exempts sales in venue-asserted groups, but not purchases (Appendix A.4.2 explains why).

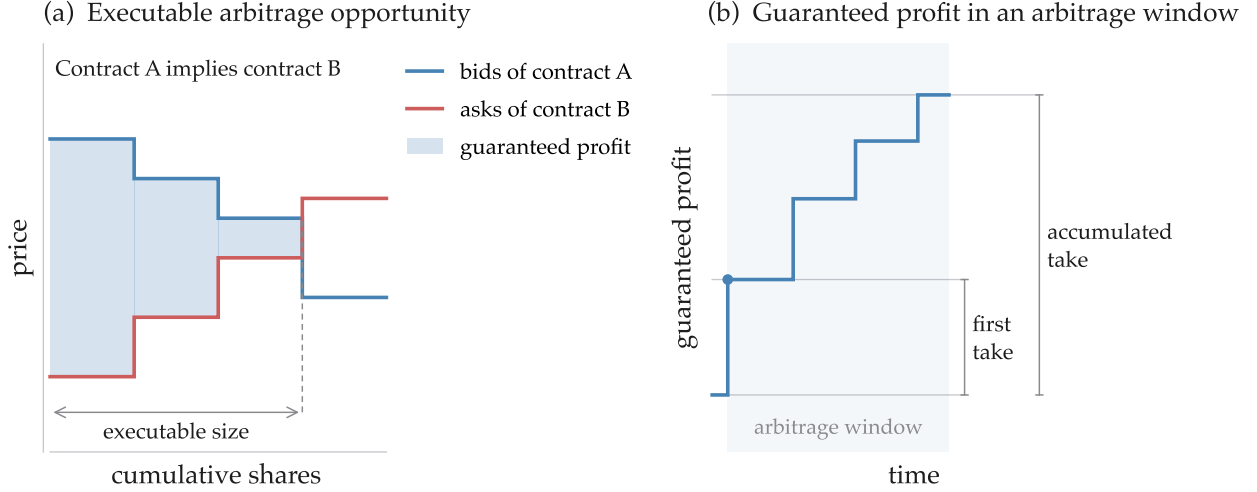


Figure 5: Simulated arbitrage trades: panel (a) illustrates guaranteed profit across price levels, and panel (b) distinguishes the first take from the accumulated take. The illustration abstracts from fees.

Because simulated executions do not alter the live order-book messages, previously consumed liquidity can reappear in subsequent reconstructed books. To prevent double-counting, we maintain a shared liquidity ledger and deduct previously consumed depth before each reoptimization. A window is called an *executed window* if at least one trade is executed within it in our simulation. Of the 491,939 windows initially detected as arbitrage opportunities, we retain 125,095 executed windows after excising 97 whose profit guarantees relied on the six invalid relations identified in the ex-post audit conducted on September 22, 2026 (see “Corrective excision” in Appendix B.3).

We report two measures of guaranteed profit within an executed window. The *first take* is the guaranteed profit of the first execution in each executed window. The *accumulated take* is the sum of guaranteed profit across all executions within the window. The profit available after the first execution depends on how subsequent quotes respond to the arbitrageur’s trades. We cannot infer these counterfactual responses from the observed data. Conditional on execution at displayed quotes, we use the first take as a lower bound, assuming no further execution within the window, and the accumulated take as an upper bound, assuming subsequent quotes remain as observed. Figure 5(b) illustrates the two measures within one window.

Step 3: test arbitrage windows at resolution. We evaluate executed portfolios using their contracts’ resolutions. We classify a window as *settled* if every filled leg had resolved to zero or one by the time the paper was written.¹¹ We define the portfolio’s *profit at resolution* as its terminal payoff less the cash outlay and fees paid at execution, and its return as this profit divided by committed capital. Capital is the cash paid for purchases plus one dollar of collateral per share sold on each leg. The negative-risk mechanism specifically lowers the collateral of the sold legs in venue-asserted portfolios: one dollar mints a basket of one YES share per contract in the group, so these legs cost one dollar per basket.

¹¹Windows with an open or void leg are classified as unsettled. Void contracts pay half a dollar to each side.

5.2 Guaranteed Profit versus Profit at Resolution

Table 5: Arbitrage windows, guaranteed profit, and profit at resolution by relation layer

Panel A: All arbitrage windows

	Windows			Percentage	
	Detected	Executed	Settled	Executed/Detected	Settled/Executed
Venue-asserted	234,605	64,681	62,860	27.6%	97.2%
Venue-derived	99,159	31,317	31,243	31.6%	99.8%
Semantic-inferred	158,175	29,097	18,273	18.4%	62.8%
All	491,939	125,095	112,376	25.4%	89.8%

Panel B: Accumulated take, settled windows

	Profit and capital (in thousands of dollars)			Return on capital	
	Guaranteed profit	Profit at resolution	Capital	Guaranteed	At resolution
Venue-asserted	852	971	22,600	3.8%	4.3%
Venue-derived	1,467	2,291	79,027	1.9%	2.9%
Semantic-inferred	179	364	12,089	1.5%	3.0%
All	2,498	3,626	113,716	2.2%	3.2%

Panel C: First take, settled windows

	Profit and capital (in thousands of dollars)			Return on capital	
	Guaranteed profit	Profit at resolution	Capital	Guaranteed	At resolution
Venue-asserted	375	408	9,759	3.8%	4.2%
Venue-derived	695	815	29,564	2.4%	2.8%
Semantic-inferred	35	75	1,240	2.8%	6.0%
All	1,104	1,297	40,563	2.7%	3.2%

Notes: The sample runs from 23 June to 3 August 2026. Capital sums the amounts committed across executions and can count the same dollar more than once.

Table 5 summarizes the scale of executable arbitrage. Overall, 25.4% of the detected windows have an execution, and 89.8% of executed windows have settled on every filled leg. The percentage settled is lower for semantic-inferred relations, at 62.8%, only because their contracts resolve later. Within the settled sample, the accumulated take is \$2.50 million in profit on \$113.7 million of capital, a guaranteed return of 2.2%. The first take totals \$1.10 million of this profit on \$40.6 million of capital; the remaining \$1.39 million is guaranteed profit from later executions within the same windows. At resolution, the executions in the accumulated take pay \$3.63 million, a return of 3.2%.¹²

The guaranteed profit of the accumulated take comes mostly from venue-asserted and venue-derived relations, which contribute 34% and 59%, respectively. Semantic-inferred relations account for the remaining 7%. The contribution of later executions also differs across layers. They are particularly important in venue-derived relations, where the accumulated take is about 2 times the first take: repeated quote arrivals allow the arbitrageur to execute again before the window closes. Much of the profit comes from high-return portfolios. The median window guarantees 0.5% on capital, but half of the accumulated take comes from windows with guaranteed returns above 6.5% (Figure 6(a)). The first take is also concentrated in high-return windows (Figure 6(b)).

¹²In our sample, no settled window in any layer has a profit at resolution below its guaranteed profit. This is consistent with the validity of identified logical relations.

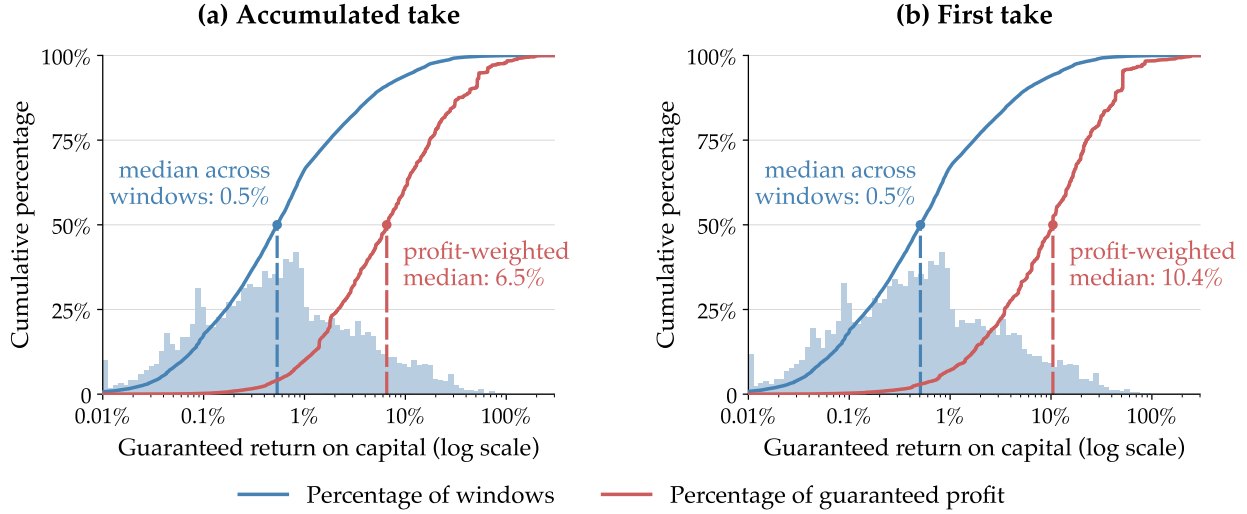


Figure 6: The distribution of executable arbitrage in settled windows. Panels (a) and (b) show the guaranteed return on capital of the accumulated take and of the first take on log scales. The blue curve gives the cumulative percentage of windows, the red curve their cumulative percentage of guaranteed profit, and the histogram counts windows in log bins.

The economic value of these returns also depends on how long capital is committed. For venue-asserted (Layer 1) portfolios that sell the group, the sold legs come from minted baskets rather than purchases, and the sale proceeds exceed the minting cost and fees, so capital need not remain committed until resolution once the legs have been sold. Venue-asserted portfolios that buy the group, and venue-derived (Layer 2) and semantic-inferred (Layer 3) portfolios have no equivalent conversion. If held to resolution, they tie up capital until the latest leg resolves. We measure the holding period from the execution to the resolution of the latest leg, using the venue’s recorded close time, and count sold venue-asserted baskets at 1 day because the conversion returns their collateral at once. In the settled sample, the window-level median is about 3 hours for venue-derived portfolios and 7 days for semantic-inferred portfolios. Venue-derived contracts generally resolve at the end of a match, whereas semantic-inferred relations also link elections, company valuations, and month-end rankings whose resolutions can be weeks or months away. The capital in Table 5 sums capital committed across executions, so the same dollar may be counted multiple times as it is redeployed. Under this holding-period convention, the capital-weighted average holding period is 1.6 days for the accumulated take and 0.8 days for the first take. Estimated capital outstanding peaks at \$11.1 million and \$3.5 million, respectively.

We next assess the robustness of the measured profits to alternative assumptions about liquidity and execution. The three checks address re-posted quotes, partial participation, and order delay. Table 6 reports the results.

Robustness check 1: cancel and re-post. The first concern is whether replenished depth represents new liquidity. A maker may cancel and re-post an earlier quote (Hasbrouck and Saar, 2013). If the arbitrageur has already consumed that liquidity, counting its return as new can overstate profits. We therefore retain the peak displayed depth at each price level as a reference after its last observation,

Table 6: Robustness checks for arbitrage opportunities

Panel A: liquidity settings, settled windows						
Setting	Number of		Percentage of baseline		Return on capital	
	All settled windows	Executions	Guaranteed profit		Guaranteed	At resolution
Baseline	112,376	391,383	100.0%		2.2%	3.2%
Robustness check 1: cancel and re-post						
suppressed within 10 s	105,628	366,458	100.0%		2.3%	3.2%
suppressed within 60 s	103,010	344,745	100.5%		2.4%	3.3%
suppressed within 120 s	102,013	334,317	100.7%		2.4%	3.4%
Robustness check 2: partial participation						
90%	112,226	391,067	92.4%		2.3%	3.2%
70%	111,822	390,236	75.3%		2.4%	3.4%
50%	111,102	388,737	56.5%		2.5%	3.5%
Panel B: order delay, settled orders						
Delay	Settled orders	Percentage filled		Profit at resolution		Orders with a loss
		Orders in full	Quantity	Thousand \$	Percentage of baseline	
Robustness check 3: order delay						
100 ms	391,936	71.6%	92.4%	3,315	91.4%	14.2%
250 ms	392,047	65.1%	89.5%	3,270	90.2%	17.3%
500 ms	392,160	59.9%	86.9%	3,058	84.4%	19.6%
750 ms	392,221	57.0%	85.3%	2,990	82.5%	20.9%
1,000 ms	392,312	54.3%	84.0%	2,874	79.3%	22.1%

Notes: The baseline is Table 5. Panel A reports the accumulated take.

with retention periods of 10 seconds, 1 minute, and 2 minutes. Depth that disappears and returns within this period does not count as new liquidity; only the amount above the reference does. The first block of Table 6 shows that the accumulated take remains within 1% of the baseline. The take can exceed the baseline because all windows draw on one liquidity ledger, so the rule changes which window takes a quote. In the baseline, a bid that a maker cancels and re-posts every few seconds is taken at every re-post, and each take also consumes the best quotes on the portfolio's other legs; the rule the re-post counts again only after the horizon, and those quotes remain for later windows. For example, in the remarkable Argentina–Cape Verde match on 3 July 2026, consider the contract on “**Argentina wins**”. The baseline sold 230,629 shares of this contract in 234 executions before 23:56:15 UTC, and the window opening at that time filled 355 shares, earning \$0.99. Under the 10-second rule, the earlier windows sold 145,916 shares in 96 executions, preserving liquidity for the later window, which sold 72,703 shares in one of its 14 executions and earned \$3,529 in total.

Robustness check 2: partial participation. Another concern is that competing traders may consume part of the displayed depth before the arbitrageur can trade. We therefore let the arbitrageur fill only the best-priced 90%, 70%, or 50% of the depth available on each leg at each execution and book the remainder as taken by other traders. The second block of Table 6 shows that, at 50% participation, the accumulated take retains 57% of its baseline value, while the number of settled windows is almost unchanged. The retained share exceeds the participation share because the arbitrageur keeps the best-priced depth, so the guaranteed return on capital rises from 2.2% to 2.5%.

Robustness check 3: order delay. Sometimes quotes disappear while an arbitrageur identifies an opportunity and submits orders. As a robustness check, we resubmit every order of the baseline after a delay of 100 milliseconds to 1 second and fill it against the order book at that time, with the liquidity ledger as in the baseline. A portfolio whose legs fill unequally no longer has a guaranteed profit, so Panel B of Table 6 reports what the delayed orders fill and what they pay at resolution. With a 1-second delay, 54% of the orders fill in full and the orders fill 84% of the submitted quantity; the settled orders pay 79% of the profit at resolution that the same orders pay in the baseline, and 22% of them lose money at resolution. In the first two checks, returns on capital change little, and no settled window has a profit at resolution below its guaranteed profit.

5.3 Persistence of Arbitrage Opportunities

Arbitrage opportunities should persist longer when their underlying payoff relations are harder to identify. Since venue-derived and semantic-inferred relations generally require more interpretation than venue-asserted relations, we next compare persistence across the three layers to examine this prediction. We measure persistence by the elapsed time from an executed window’s opening to its close, as defined in Section 5.1. Windows that last less than 0.3 seconds account for 47% of executed windows but only 8% of the accumulated take. Table 7 reports persistence for all executed windows and for executed windows that last at least 0.3 seconds.

We first focus on executed windows that last at least 0.3 seconds. The right half of Panel A shows that median duration rises from 2.9 seconds for venue-asserted relations to 6.3 seconds for venue-derived relations and about 31 seconds for semantic-inferred relations. The percentage of windows that last at least 10 seconds rises from 32% to 44% and 62%. The right half of Panel B shows that, in the same sample, these differences remain after controlling for portfolio characteristics, the component’s daily number of detected windows, and day and hour fixed effects. In Column (6), relative to venue-asserted windows, venue-derived windows are 8 percentage points more likely to last at least 10 seconds, parser windows 28 percentage points, and LLM windows 37 percentage points. Exponentiating the log-duration coefficients in (8) implies durations about 2, 7, and 15 times, respectively, as long as those of venue-asserted windows, conditional on the included controls. Each additional leg is associated with a 0.5-percentage-point higher probability of lasting at least 10 seconds and a 2% longer duration. These patterns are consistent with discovery and execution costs limiting arbitrage. The comparisons are conditional associations, since the layers also differ in their contracts, liquidity, and conversion mechanisms.

We next consider all executed windows. The left half of Panel A shows an increase from venue-asserted to venue-derived and then to semantic-inferred windows in the 75th percentile of duration, and the percentage of windows that last at least 10 seconds doubles from venue-asserted windows to the other two layers. Median duration also increases from venue-asserted to venue-derived and then to LLM windows. The left half of Panel B reports the same regressions for the full sample; the estimated differences across layers retain their signs, ordering, and statistical significance.

Table 7: Persistence of arbitrage opportunities by layer

Panel A: survival by layer

Layer	All executed windows				Executed windows of at least 0.3 seconds			
	Windows	Median (s)	p75 (s)	Lasts ≥ 10 s	Windows	Median (s)	p75 (s)	Lasts ≥ 10 s
Venue-asserted	64,681	0.2	2.0	15%	29,070	2.9	10.7	32%
Venue-derived	31,317	1.5	20.6	31%	22,407	6.3	45.7	44%
Semantic-inferred	29,097	0.2	28.2	30%	14,287	30.8	223.8	62%
Parsers	21,744	0.1	17.0	27%	9,489	32.0	214.6	62%
LLMs	7,353	2.9	89.9	40%	4,798	29.0	270.7	61%
All	125,095	0.4	6.8	22%	65,764	5.7	44.2	43%

Panel B: layer and the speed of correction

	All executed windows				Executed windows of at least 0.3 seconds			
	Lasts ≥ 10 s		log duration		Lasts ≥ 10 s		log duration	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Venue-derived	0.108*** (0.012)	0.085*** (0.011)	1.497*** (0.107)	1.233*** (0.108)	0.088*** (0.013)	0.077*** (0.012)	0.669*** (0.072)	0.583*** (0.069)
Semantic-inferred	0.277*** (0.023)		2.452*** (0.215)		0.324*** (0.022)		2.333*** (0.111)	
Parsers		0.199*** (0.035)		1.540*** (0.301)		0.275*** (0.034)		1.960*** (0.174)
LLMs		0.352*** (0.023)		3.333*** (0.289)		0.372*** (0.021)		2.692*** (0.113)
Number of legs	0.006*** (0.001)	0.006*** (0.001)	0.046*** (0.007)	0.038*** (0.007)	0.005*** (0.001)	0.005*** (0.001)	0.022*** (0.006)	0.018*** (0.006)
Controls and FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Observations	125,095	125,095	125,095	125,095	65,764	65,764	65,764	65,764
Within R^2	0.114	0.119	0.168	0.178	0.107	0.109	0.179	0.183

Notes: Duration is the exchange time from a window's opening to its close. Panel B uses a linear probability model for Lasts ≥ 10 s; the venue-asserted layer is the omitted category. Controls are the log first take, log capital committed, log number of windows the component opened that day, and the portfolio's direction (purchases only, or purchases and sales; sales only is omitted), with day and hour fixed effects (FE). Standard errors in parentheses are clustered by leg structure; *, **, and *** denote significance at the 10%, 5%, and 1% levels.

5.4 Who Takes the Arbitrage

The preceding analysis measures what an arbitrageur could earn at displayed quotes. We now use on-chain fills to examine whether actual trades follow the detected portfolios. For each executed window, we collect fills on its contracts from 5 seconds before it opens to 30 seconds after it closes. A taker trades in the portfolio's direction when it buys YES or sells NO on a purchased leg, or sells YES or buys NO on a sold leg. A window has a *matched trade* when the same taker address has trades on every leg of the portfolio, with each trade in the portfolio's direction on that leg. Table 8 (Panel A) reports the percentage of executed windows with a matched trade (same direction). We provide a placebo benchmark for two-sided trading by repeating the match with every leg's direction reversed. Windows matched in both directions enter both counts; the two matches may involve different taker addresses. These matches establish directional trading activity, but do not by themselves establish execution of the portfolio at the prices and quantities used in our calculations.

Table 8: Matched trades in arbitrage windows

Panel A: matched trades by layer

Layer	Number of executed windows				Median lag until every leg trades (seconds)
	All executed	With any address trading every leg			
		Same direction (matched trades)	Opposite direction	Both directions	
Venue-asserted	64,681	28,507 (44.1%)	4,366 (6.8%)	3,631 (5.6%)	2.7
Venue-derived	31,317	4,554 (14.5%)	1,776 (5.7%)	1,371 (4.4%)	1.0
Semantic-inferred	29,097	2,574 (8.8%)	88 (0.3%)	71 (0.2%)	3.1
All	125,095	35,635 (28.5%)	6,230 (5.0%)	5,073 (4.1%)	2.6

Panel B: wallet addresses by layer

Layer	Number of distinct addresses	Number of executed windows with a matched trade	
		By any address	By top 10
Venue-asserted	666	28,507	24,344 (85.4%)
Venue-derived	390	4,554	3,399 (74.6%)
Semantic-inferred	151	2,574	2,272 (88.3%)
All	854	35,635	27,620 (77.5%)

Notes: A matched trade requires one taker address to trade every leg in the portfolio's direction. Panel A percentages use all executed windows; Panel B percentages use windows matched by any address. Median lag runs from opening until every leg has a same-direction fill. Top 10 addresses are ranked by matched windows within each layer and, for the two All rows, across layers.

The excess over the placebo is largest in venue-asserted groups. In 44% of these windows there is a matched trade, compared with about 7% in the placebo. Venue-derived relations show a much smaller difference: about 15% of windows have a matched trade, compared with 6% in the placebo. About 30% of these same-direction matched venue-derived windows also have an opposite-direction match. The smaller excess over the placebo is consistent with ordinary two-sided betting flow contributing substantially to the matches. For semantic-inferred relations, the matching rate is lower but still exceeds the placebo. Only 9% have a matched trade, compared with less than 1% in the placebo.

Participation is also concentrated (Panel B of Table 8): across all layers, about 850 addresses make matched trades, and the 10 most active trade every leg in about 78% of the matched windows. Concentration is high within each layer as well: the 10 most active addresses in each layer account for 85% of matched windows in venue-asserted relations, 75% in venue-derived relations, and 88% in semantic-inferred relations.

These trading patterns help interpret the persistence results. Where the venue publishes the relation and, for sales, offers basket conversion, trades follow the arbitrage direction more often and opportunities disappear quickly. Among opportunities lasting at least 0.3 seconds, those based on semantic-inferred relations persist longest, yet few have one address trading every leg. The duration regressions also associate additional legs with longer windows, consistent with the difficulty of assembling trades across separate books. The evidence is thus consistent with both discovery and execution costs limiting arbitrage, even when displayed prices offer a profit after spreads and fees.

6 Conclusion

We develop a framework for combinatorial arbitrage among logically related prediction-market contracts, linking restrictions on their joint payoffs to executable portfolios. We recover relations from venue mechanisms, typed contract fields, and natural-language resolution criteria, and compile them into a common no-arbitrage program. Full-depth order books then determine how much can be traded and the profit guaranteed after spreads and fees, conditional on valid relations and the modeled payoff states.

The Polymarket data reveal substantial executable opportunities. Among settled windows, an arbitrageur accumulates \$2.50 million in guaranteed profit, assuming execution at displayed quotes and that subsequent quotes remain as observed. Restricting the arbitrageur to the first execution in each window yields \$1.10 million. Much of the accumulated take comes from a small proportion of high-return windows. Persistence also differs across relation layers. Among windows lasting at least 0.3 seconds, median duration rises from 2.9 seconds for venue-asserted relations to about 31 seconds for semantic-inferred relations. These differences remain after controlling for portfolio characteristics and time effects.

The results suggest that finding the payoff relation is an important part of arbitrage. Opportunities based on groups the venue publishes disappear quickly and more often have one address trading every leg. Relations recovered from contract text are associated with longer opportunities and less observed participation. Because the layers also differ in liquidity and execution mechanisms, these comparisons do not isolate discovery costs from execution costs. They show how both can matter in markets where related claims trade in separate order books.

References

- Abernethy, Jacob, Yiling Chen, and Jennifer Wortman Vaughan, 2013, Efficient market making via convex optimization, and a connection to online learning, *ACM Transactions on Economics and Computation* 1.
- Bartlett, Robert, and Maureen O’Hara, 2026, Adverse selection in prediction markets: Evidence from Kalshi, Stanford Law School working paper.
- Capponi, Agostino, Alfio Gliozzo, and Brian Zhu, 2025, Agentic AI for clustering, relationship discovery, and semantic trading in prediction markets, *arXiv preprint* arXiv:2512.02436v2.
- Chen, Yiling, Lance Fortnow, Nicolas Lambert, David M. Pennock, and Jennifer Wortman, 2008, Complexity of combinatorial market makers, in *Proceedings of the 9th ACM Conference on Electronic Commerce (EC ’08)*, 190–199.
- Cohen, Lauren, and Dong Lou, 2012, Complicated firms, *Journal of Financial Economics* 104, 383–400.

- de Finetti, Bruno, 1937, La prévision: ses lois logiques, ses sources subjectives, *Annales de l'Institut Henri Poincaré* 7, 1–68.
- De Long, J. Bradford, Andrei Shleifer, Lawrence H. Summers, and Robert J. Waldmann, 1990, Noise trader risk in financial markets, *Journal of Political Economy* 98, 703–738.
- DeepSeek-AI, et al., 2026, DeepSeek-V4: Towards highly efficient million-token context intelligence, *arXiv preprint* arXiv:2606.19348.
- Du, Wenxin, Alexander Tepper, and Adrien Verdelhan, 2018, Deviations from covered interest rate parity, *Journal of Finance* 73, 915–957.
- Dudík, Miroslav, Sébastien Lahaie, and David M. Pennock, 2012, A tractable combinatorial market maker using constraint generation, in *Proceedings of the 13th ACM Conference on Electronic Commerce*, 459–476.
- Fleckenstein, Matthias, and Francis A. Longstaff, 2020, Renting balance sheet space: Intermediary balance sheet rental costs and the valuation of derivatives, *Review of Financial Studies* 33, 5051–5091.
- Gebele, Jonas, and Florian Matthes, 2026, Semantic non-fungibility and violations of the law of one price in prediction markets, *arXiv preprint* arXiv:2601.01706.
- Gomez-Cram, Roberto, Yunhan Guo, Theis Ingerslev Jensen, and Howard Kung, 2026, Prediction market accuracy: Crowd wisdom or informed minority?, Working paper, SSRN 6617059.
- Gromb, Denis, and Dimitri Vayanos, 2002, Equilibrium and welfare in markets with financially constrained arbitrageurs, *Journal of Financial Economics* 66, 361–407.
- Grossman, Sanford J., and Joseph E. Stiglitz, 1980, On the impossibility of informationally efficient markets, *American Economic Review* 70, 393–408.
- Grötschel, Martin, László Lovász, and Alexander Schrijver, 1988, *Geometric Algorithms and Combinatorial Optimization* (Springer, Berlin).
- Hanson, Robin, 2003, Combinatorial information market design, *Information Systems Frontiers* 5, 107–119.
- Hasbrouck, Joel, and Gideon Saar, 2013, Low-latency trading, *Journal of Financial Markets* 16, 646–679.
- Jouini, Elyes, and Hedi Kallal, 1995, Martingales and arbitrage in securities markets with transaction costs, *Journal of Economic Theory* 66, 178–197.
- Kelley, James E., Jr., 1960, The cutting-plane method for solving convex programs, *Journal of the Society for Industrial and Applied Mathematics* 8, 703–712.
- Kroer, Christian, Miroslav Dudík, Sébastien Lahaie, and Sivaraman Balakrishnan, 2016, Arbitrage-free combinatorial market making via integer programming, in *Proceedings of the 17th ACM Conference on Economics and Computation (EC '16)*, 161–178.

- Lamont, Owen A., and Richard H. Thaler, 2003, Anomalies: The law of one price in financial markets, *Journal of Economic Perspectives* 17, 191–202.
- Meta AI, 2025, Llama 4 model card, Meta Llama Models.
- Moskowitz, Tobias J., 2021, Asset pricing and sports betting, *Journal of Finance* 76, 3153–3209.
- Nemhauser, George L., and Laurence A. Wolsey, 1988, *Integer and Combinatorial Optimization* (Wiley, New York).
- Ng, Hunter, Lin Peng, Yubo Tao, and Dexin Zhou, 2026, Price discovery and trading in modern prediction markets, Working paper, SSRN 5331995.
- Pennanen, Teemu, 2011, Arbitrage and deflators in illiquid markets, *Finance and Stochastics* 15, 57–83.
- Pontiff, Jeffrey, 2006, Costly arbitrage and the myth of idiosyncratic risk, *Journal of Accounting and Economics* 42, 35–52.
- Qwen Team, 2025, Qwen3-235B-A22B-Instruct-2507-FP8 model card, Hugging Face.
- Rockafellar, R. Tyrrell, 1970, *Convex Analysis* (Princeton University Press, Princeton, NJ).
- Rothschild, David, and David M. Pennock, 2014, The extent of price misalignment in prediction markets, *Algorithmic Finance* 3, 3–20.
- Saguillo, Oriol, Vahid Ghafouri, Lucianna Kiffer, and Guillermo Suárez-Tangil, 2025, Unravelling the probabilistic forest: Arbitrage in prediction markets, in *7th Conference on Advances in Financial Technologies (AFT 2025)*, volume 354 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 27:1–27:24.
- Schrijver, Alexander, 1986, *Theory of Linear and Integer Programming* (Wiley, Chichester).
- Shleifer, Andrei, and Robert W. Vishny, 1997, The limits of arbitrage, *Journal of Finance* 52, 35–55.
- Sonnemann, Ulrich, Colin F. Camerer, Craig R. Fox, and Thomas Langer, 2013, How psychological framing affects economic market prices in the lab and field, *Proceedings of the National Academy of Sciences* 110, 11779–11784.
- Stulz, René M., 2007, Hedge funds: Past, present, and future, *Journal of Economic Perspectives* 21, 175–194.
- Wolfers, Justin, and Eric Zitzewitz, 2004, Prediction markets, *Journal of Economic Perspectives* 18, 107–126.
- Yang, An, et al., 2025, Qwen3 technical report, *arXiv preprint* arXiv:2505.09388.
- Z.ai, 2026, GLM-5.2 model card, Hugging Face.
- Zhai, Daojing, 2026, When Treasuries crowd the debt market: Treasury market inelasticity and the basis trade, Working paper, SSRN 6226558.

Appendix A Arbitrage Theory and Detection

A.1 Proofs

Proof of Footnote 6. The claim is that two sets P and Q are equal for every $S \subseteq I$, where

$$P := \{z \in \{0,1\}^I : z_i + z_j \leq 1 \text{ for all } i < j \text{ in } S\}, \quad Q := \{z \in \{0,1\}^I : \sum_{i \in S} z_i \leq 1\}. \quad (\text{A.1})$$

The inclusion $Q \subseteq P$ is immediate: if $\sum_{i \in S} z_i \leq 1$, then $z_i + z_j \leq \sum_{k \in S} z_k \leq 1$ for every pair $i < j$, since every $z_k \geq 0$. For the reverse inclusion, suppose, for contradiction, that some $z \in P$ has $\sum_{i \in S} z_i \geq 2$. Since each z_k is 0 or 1, at least two coordinates $i \neq j$ in S equal one, so $z_i + z_j = 2 > 1$, contradicting $z \in P$. This step requires integrality: on $[0,1]^I$ the two sets P and Q differ, as $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is a counterexample for $|S| = 3$. $\square$

Proof of Proposition 1. If $p \in M$ then $p = \mathbb{E}_q[\phi]$ for some belief q . Hence the initial cost of every portfolio equals its expected terminal payoff, $y^\top p = \mathbb{E}_q[y^\top \phi]$, so its expected profit under q is zero. Its profit therefore cannot be strictly positive in every state, and (1) has no solution. If $p \notin M$, then since M is compact and convex, the separating-hyperplane theorem gives a direction y with $y^\top p < \min_{\mu \in M} y^\top \mu = \min_{\omega} y^\top \phi(\omega)$. The portfolio's worst-case terminal payoff therefore exceeds its initial cost, so its guaranteed profit is strictly positive: an arbitrage exists. $\square$

Proof of Proposition 2. A linear function attains the same minimum over a finite point set and over its convex hull, so the worst-case terminal payoff satisfies $\min_{\omega} y^\top \phi(\omega) = \min_{\mu \in M} y^\top \mu$. Hence

$$\begin{aligned} \max_{\|y\|_{\infty} \leq 1} \min_{\omega \in \Omega} \{y^\top \phi(\omega) - y^\top p\} &= \max_{\|y\|_{\infty} \leq 1} \min_{\mu \in M} y^\top (\mu - p) \\ &= \min_{\mu \in M} \max_{\|y\|_{\infty} \leq 1} y^\top (\mu - p) = \min_{\mu \in M} \|\mu - p\|_1, \end{aligned} \quad (\text{A.2})$$

where the exchange of maximum and minimum follows from linear-programming duality on compact convex sets, and coordinatewise maximization gives the ℓ_1 norm. Let y^* be a maximin optimizer and μ^* a minimizing coherent price vector. The minimax equality implies that they form a saddle point, so $y_i^* = +1$ where $p_i < \mu_i^*$ and $y_i^* = -1$ where $p_i > \mu_i^*$. When $p_i = \mu_i^*$, the corresponding position is chosen jointly with the other positions so that $y^{*\top}(\mu - \mu^*) \geq 0$ for every $\mu \in M$; it need not be zero. $\square$

Proof of Proposition 3. Proposition 3 is a special case of the following proposition:

Proposition 3*. Let $K : \mathbb{R}^I \rightarrow \mathbb{R} \cup \{+\infty\}$ be a proper closed convex function, with $K(0) = 0$ and a compact domain $D = \text{dom } K$. Let $K^*(\mu) := \sup_{y \in \mathbb{R}^I} \{\mu^\top y - K(y)\}$ denote its convex conjugate, and let $\partial K(0) := \{\mu \in \mathbb{R}^I : \mu^\top y \leq K(y) \text{ for every } y \in \mathbb{R}^I\}$ denote its subdifferential at zero. Then

$$\Pi^* := \max_{y \in D} \min_{z \in \Phi} \{y^\top z - K(y)\}$$

is nonnegative, is attained, and takes the value of $\min_{\mu \in M} K^*(\mu)$. Moreover, $\Pi^* > 0$ if and only if $M \cap \partial K(0) = \emptyset$.

We first use Proposition 3* to prove Proposition 3, and prove Proposition 3* itself later. The order-book cost function is $K(\mathbf{y}) = \sum_{i \in I} K_i(y_i)$, where K_i can be formally written as

$$K_i(y_i) := \begin{cases} \int_0^{y_i} c_i(s) ds & -Q_i^b \leq y_i \leq Q_i^a, \\ +\infty & \text{otherwise.} \end{cases} \quad (\text{A.3})$$

Here $c_i(s)$ denotes the price of the s -th share traded, and is a step function:

$$c_i(s) := \begin{cases} \tilde{a}_{i\ell} & \text{if } Q_{i,\ell-1}^a < s \leq Q_{i\ell}^a \quad (\text{buying}), \\ \tilde{b}_{i\ell} & \text{if } -Q_{i\ell}^b \leq s < -Q_{i,\ell-1}^b \quad (\text{selling}), \end{cases} \quad (\text{A.4})$$

where $Q_{i\ell}^a := \sum_{k=1}^{\ell} q_{ik}^a$ and $Q_{i\ell}^b := \sum_{k=1}^{\ell} q_{ik}^b$ are the depths up to the ℓ -th level of asks and bids, respectively.

By (A.3) and (A.4), K is proper, closed and convex, vanishes at zero, and has the compact domain (6), and thus Proposition 3* applies. Because K is a sum of functions of separate coordinates, $\partial K(\mathbf{0})$ is the product of the sets $\partial K_i(0)$. Each K_i is piecewise linear, and at zero its right-hand and left-hand derivatives are $(K_i)'_+(0) = \tilde{a}_{i1}$ and $(K_i)'_-(0) = \tilde{b}_{i1}$, so $\partial K_i(0) = [\tilde{b}_{i1}, \tilde{a}_{i1}]$ (Rockafellar, 1970, p. 229). The condition $M \cap \partial K(\mathbf{0}) = \emptyset$ is therefore condition (9). $\square$

Proof of Proposition 3.* Fix a portfolio $\mathbf{y} \in D$. Since $K(\mathbf{y})$ is its initial cost and does not depend on the realized state,

$$\min_{\mathbf{z} \in \Phi} \{\mathbf{y}^\top \mathbf{z} - K(\mathbf{y})\} = \min_{\mathbf{z} \in \Phi} \mathbf{y}^\top \mathbf{z} - K(\mathbf{y}) = \min_{\boldsymbol{\mu} \in M} \{\boldsymbol{\mu}^\top \mathbf{y} - K(\mathbf{y})\}. \quad (\text{A.5})$$

The second equality uses $M = \text{conv}(\Phi)$: a linear functional has the same minimum over the payoff vectors and over their convex hull. Hence

$$\Pi^* = \max_{\mathbf{y} \in D} \min_{\boldsymbol{\mu} \in M} \{\boldsymbol{\mu}^\top \mathbf{y} - K(\mathbf{y})\}. \quad (\text{A.6})$$

Because K is closed, this objective is upper semicontinuous in $\mathbf{y}$ on the compact set D , so its maximum is attained. The portfolio $\mathbf{y} = \mathbf{0}$ lies in D and has value $-K(\mathbf{0}) = 0$, so $\Pi^* \geq 0$.

Now M and D are compact and convex, and on $M \times D$ the function

$$h(\boldsymbol{\mu}, \mathbf{y}) := \boldsymbol{\mu}^\top \mathbf{y} - K(\mathbf{y}) \quad (\text{A.7})$$

is continuous and affine in $\boldsymbol{\mu}$, and upper semicontinuous and concave in $\mathbf{y}$ because K is closed and convex. The minimax theorem therefore allows the order of optimization to be exchanged:

$$\Pi^* = \min_{\boldsymbol{\mu} \in M} \max_{\mathbf{y} \in D} \{\boldsymbol{\mu}^\top \mathbf{y} - K(\mathbf{y})\} = \min_{\boldsymbol{\mu} \in M} K^*(\boldsymbol{\mu}), \quad (\text{A.8})$$

where the last equality holds because $K(\mathbf{y}) = +\infty$ outside D .

It remains to characterize when $\Pi^* = 0$. Since $K(\mathbf{0}) = 0$, every $\boldsymbol{\mu}$ satisfies $K^*(\boldsymbol{\mu}) \geq \boldsymbol{\mu}^\top \mathbf{0} - K(\mathbf{0}) = 0$, and equality holds exactly when $\boldsymbol{\mu}^\top \mathbf{y} \leq K(\mathbf{y})$ for every $\mathbf{y}$, that is, when $\boldsymbol{\mu} \in \partial K(\mathbf{0})$. Because D is compact, K^* is finite and convex, hence continuous, so its minimum over the compact set M is attained. Therefore $\Pi^* = 0$ if and only if some $\boldsymbol{\mu} \in M$ has $K^*(\boldsymbol{\mu}) = 0$, that is, if and only if $M \cap \partial K(\mathbf{0}) \neq \emptyset$. $\square$

Remark 1 (The quoted-price benchmark as a special case). For a coherent valuation $\mu \in M$, $K^*(\mu)$ is the largest gain from establishing a portfolio at the displayed order book and valuing it at μ , and Proposition 3* states that Π^* is this gain under the coherent valuation that makes it smallest. The linear quoted-price model of Section 3.1 is the special case with cost

$$K^{\text{quoted}}(\mathbf{y}) := \mathbf{y}^\top \mathbf{p} + \iota_{\{\|\mathbf{y}\|_\infty \leq 1\}}(\mathbf{y}), \quad (\text{A.9})$$

where ι_C is 0 on C and $+\infty$ outside it. This K^{quoted} is proper, closed and convex, vanishes at zero, and has the compact domain $\{\|\mathbf{y}\|_\infty \leq 1\}$, so Proposition 3* applies. Its conjugate is

$$(K^{\text{quoted}})^*(\mu) = \sup_{\|\mathbf{y}\|_\infty \leq 1} (\mu - \mathbf{p})^\top \mathbf{y} = \|\mu - \mathbf{p}\|_1, \quad (\text{A.10})$$

so $\Pi^* = \min_{\mu \in M} \|\mu - \mathbf{p}\|_1$, which is Proposition 2; and its subdifferential at zero is

$$\partial K^{\text{quoted}}(\mathbf{0}) = \{\mu : (\mu - \mathbf{p})^\top \mathbf{y} \leq 0 \text{ for all } \|\mathbf{y}\|_\infty \leq 1\} = \{\mathbf{p}\}, \quad (\text{A.11})$$

so $\Pi^* > 0$ if and only if $\mathbf{p} \notin M$, which is Proposition 1.

Proof of Proposition 4. Since $\Phi \subseteq \Phi^-$, minimizing the same net-payoff function over Φ^- can only give a weakly smaller value than minimizing it over Φ . Maximizing both sides over the common executable set D preserves the inequality. Strict positivity for a particular portfolio under Φ^- therefore implies strict positivity under Φ . $\square$

A.2 The Optimal Portfolio and the Cutting-Plane Detector

Recall from Proposition 1 that an arbitrage exists exactly when $\mathbf{p}$ lies outside the convex hull M , and from Proposition 2 that, since $V(\mathbf{y}) = \min_{\mu \in M} \mathbf{y}^\top \mu$ is the worst-case terminal payoff, the normalized portfolio with the largest guaranteed profit attains

$$\max_{\|\mathbf{y}\|_\infty \leq 1} [V(\mathbf{y}) - \mathbf{y}^\top \mathbf{p}] = \min_{\mu \in M} \|\mu - \mathbf{p}\|_1. \quad (\text{A.12})$$

The normalization $\|\mathbf{y}\|_\infty \leq 1$ fixes only the scale: in the linear quoted-price model, an arbitrageur could otherwise increase a profitable position without bound. When the distance is zero, $\mathbf{p} \in M$, so no arbitrage exists and some belief q rationalizes the quoted prices. When the distance is positive, an arbitrageur buys contracts priced below the minimizing coherent price vector and sells those priced above it. This characterization is the quoted-price analogue of Proposition 2.4 in Kroer et al. (2016), with the ℓ_1 distance in place of their Bregman divergence.

Computing the optimum in (A.12) by enumeration can be infeasible because the convex hull M can have exponentially many extreme points. To avoid this enumeration, we introduce an auxiliary variable η as a lower bound on the portfolio's terminal payoff. The problem (A.12) is then equivalent to the linear program $\max_{\mathbf{y}, \eta} \eta - \mathbf{y}^\top \mathbf{p}$ subject to $\mathbf{y}^\top \mathbf{z} \geq \eta$ for every $\mathbf{z} \in \Phi$ and $\|\mathbf{y}\|_\infty \leq 1$. This reformulation is the MASTER step of Algorithm 1, which enforces the constraint only on the working set of discovered payoff vectors and lets the separation oracle supply violated rows (Kelley, 1960; Grötschel, Lovász, and Schrijver, 1988).

With the order-book cost of Section 3.3 in place of the linear cost and the unit box, the MASTER step remains a linear program, because K is piecewise linear: one variable per price level, bounded by the quantity displayed at that level, carries that level's price and fee, and the convexity of each K_i makes the optimum fill the levels in ladder order. The 5-leg cap of Section 5.1 adds one binary indicator $x_i \in \{0, 1\}$ per contract, linked to the position by $-Q_i^b x_i \leq y_i \leq Q_i^a x_i$, so that $x_i = 0$ forces $y_i = 0$, and bounds the sum of the indicators by five, $\sum_{i \in I} x_i \leq 5$; the MASTER step then becomes a mixed-integer linear program (MILP). Where the cap applies, the integer variables enter only the detection of Step 1 in Section 5.1, which chooses the legs; the reoptimization of Step 2 keeps the detected legs and their directions, so it is the linear program above over the full-depth ladders.

The admissible payoff vectors depend only on the constraint system (F, g) , not on prices, so previously discovered vectors can be reused while that system remains unchanged. At each new quote state, we first solve the MASTER step using these vectors. A zero maximum rules out arbitrage; a positive maximum requires the ORACLE step, the integer program (3), to verify the proposed portfolio against the full payoff set. Each violated payoff constraint adds a previously unseen vector, so the procedure terminates after finitely many iterations.

A.3 Early Liquidation

The quoted-price benchmark also permits the arbitrageur to exit before resolution: for the optimal portfolio of Section 3.1, any future coherent price vector $p_\tau \in M$ yields

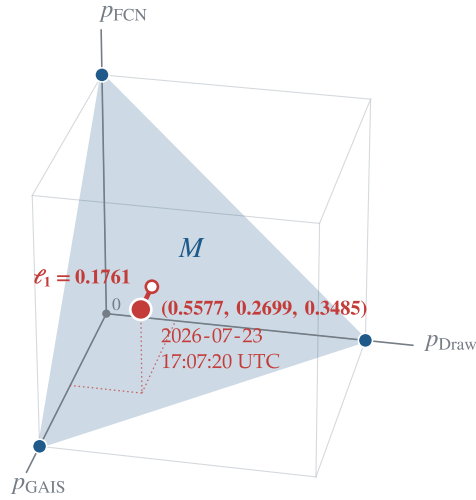
$$y^{\star\top}(p_\tau - p) \geq \min_{\mu \in M} y^{\star\top}(\mu - p), \quad (\text{A.13})$$

so an arbitrageur who liquidates at these prices earns at least the profit guaranteed at resolution. Extending this conclusion to a finite-depth order book requires the future liquidation order book. If $K_0(y)$ is the entry cost and K_τ the execution-cost function of the time- τ order book, closing at τ produces liquidation profit $-K_0(y) - K_\tau(-y)$, which depends on future spreads, depth, and fees that are unknown at entry. Resolution avoids this uncertainty because the contracts pay directly according to $z \in \Phi$, requiring no further market transaction. We therefore treat resolution as the guaranteed exit; earlier liquidation realizes the arbitrage sooner only when the future order book supports it.

A.4 Additional Logical Relations

A.4.1 Partition and conjunction

(a) GAIS–Nordsjælland Moneyline



(b) Norway–France Both Teams to Score

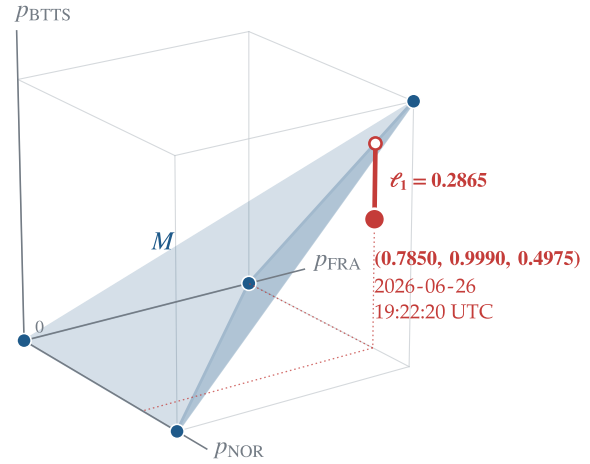


Figure A.1: Price coherence for partition and conjunction relations. Panel (a) shows the GAIS–Nordsjælland win and draw contracts; panel (b) shows the Norway–France both-teams-to-score relation. Red segments mark the ℓ_1 distance to coherence.

Panel (a) of Figure A.1 considers the YES contracts “GAIS win,” “Draw,” and “FC Nordsjælland win,” with Polymarket market IDs 2974928, 2974929, and 2974930. The contracts form an exhaustive partition, so their joint payoff vector can take the values $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, corresponding to a GAIS win, a draw, and a Nordsjælland win. Their convex hull is the blue triangular coherent set M . At 17:07:20 UTC on 23 July 2026, the fee-adjusted bid vector $(0.5577, 0.2699, 0.3485)$ lies outside M at ℓ_1 distance 0.1761, the red segment along the GAIS axis: an arbitrageur who sells one share of each contract receives \$1.1761 initially and pays at most one dollar at resolution, guaranteeing \$0.1761 after fees.

Panel (b) considers the contracts “Norway Over 0.5 goals,” “France Over 0.5 goals,” and “Both Teams to Score,” with Polymarket market IDs 2481818, 2481821, and 2323839. The third contract is the conjunction of the first two, so their joint payoff vector can take the values $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, and $(1, 1, 1)$, whose convex hull is the blue tetrahedral coherent set M . At 19:22:20 UTC on 26 June 2026, the fee-adjusted bids for the two team totals and the fee-adjusted ask for the both-teams-to-score contract form the execution quote vector $(0.7850, 0.9990, 0.4975)$, which violates the Fréchet lower bound $0.7850 + 0.9990 - 1 = 0.7840$ on the third coordinate; the red segment marks the ℓ_1 distance 0.2865 to M . At the best quotes, an arbitrageur can guarantee \$0.2865 by selling one share of each of the first two contracts and buying one share of the both-teams-to-score contract. Trading against the 32 shares displayed yields total guaranteed profit of \$9.17 after fees.

A.4.2 Two sides of partitions in negative-risk groups

Polymarket describes negative risk as “a mechanism for multi-outcome events where only one outcome can win,” and its adapter converts a NO share of any contract into one YES share of every other contract in the same negative-risk group; it does not convert a YES share into NO shares of the others. The asymmetry follows from what the adapter guarantees: at most one contract resolves YES, not that exactly one does. A NO share therefore pays at least as much as the other contracts’ YES shares together in every admissible state, with the surplus when no contract wins, so it can be exchanged for them; the reverse exchange would pay out more than it takes in when no contract wins, so it is not offered.¹³

A seller of the basket can therefore proceed as follows: split one dollar into one YES and one NO share of any contract, and then convert the NO share into one YES share of every other contract to mint the basket of one YES share of every contract in the group. The seller can sell these shares and collect the proceeds at once, whereas a buyer of the basket collects its payoff only at resolution. In practice the seller needs only the few members whose bids carry the overpricing to secure a profit: once the bids of two to five liquid contracts sum to more than one dollar, selling those alone returns the dollar and locks in those bids minus one, and every further member sold, or kept and won, only adds to it. A buyer, by contrast, is covered only after buying every member, one leg per contract, and only in a group whose rules make exactly one contract win; selling needs only that at most one wins, so the seller’s profit is guaranteed in every negative-risk group, not only in the partitions.

In the Algeria–Austria group (three contracts: “Algeria wins”, “draw”, and “Austria wins”, of which exactly one is YES) of Section 4.2, the two moments below, both in the first half of the match on 28 June 2026 (kick-off 02:00 UTC), make the asymmetry concrete. Purely to keep the arithmetic simple, we use bids and asks before taker fees instead of the fee-adjusted ones used elsewhere in the paper. Figure A.3 nonetheless includes these fees.¹⁴

¹³The forward exchange is safe for the adapter, because in no admissible state does it pay out more than it takes in, and the reverse exchange is not. Both claims follow from the two resolution cases of the NO share: if the NO share pays zero, its YES counterpart pays one, forcing all the other YES shares to pay zero; if it pays one, the other YES shares sum to at most one, and to zero when no contract wins, which leaves the surplus of one dollar. The reverse exchange would take in the other contracts’ YES shares and hand out one NO share; when no contract wins, the YES shares taken in all pay zero, while the NO share handed out pays one dollar.

¹⁴Polymarket charges taker fees: a taker who trades C shares at price p pays $rCp(1 - p)$, where r is the contract’s fee rate. The sports rate was still 0.03 on 28 June 2026; Polymarket raised it to 0.05 on 10 July 2026 (Polymarket, “Changelog”). By contrast, two kinds of operation change a position without a trade on the order book. The first is splitting and merging: splitting turns one dollar into one YES token and one NO token of the same contract, and merging turns one YES token and one NO token back into one dollar (Polymarket, “Manage Positions”). The second is conversion in a negative-risk group: one NO share of one contract is turned into one YES share of every other contract in the group (Polymarket, “Negative Risk Markets”). Neither operation carried a fee during our sample period, as three pieces of evidence show. First, Polymarket’s fee schedule charges a fee only to the taker of a trade, and it lists no fee for splitting, merging or converting (Polymarket, “Fees”). Second, the contract that performs the conversion allows a conversion fee, which is fixed when a group is created (NegRiskAdapter.sol); we read this fee from the contract on Polygon for each of the 20,866 negative-risk groups in our sample, and it is zero for every one of them. Third, in on-chain transactions sampled across the period, every split, merge and conversion paid out exactly the amounts stated above.

The sell side. At 02:44:45 UTC the bids of the three YES contracts were 0.16, 0.31 and 0.57, which sum to 1.04, more than one dollar.

- With the mechanism: split one dollar on “Algeria wins” into one YES and one NO of it, and convert that NO into one YES each of “draw” and “Austria wins”; one dollar now holds the three YES. Sell all three YES at the bids for 1.04 and net 0.04; the dollar comes back on the spot.
- The same result can be reached without touching the YES shares: buy the three NO directly at their asks of 0.84, 0.69 and 0.43, paying 1.96 in all, and convert the three NO into two dollars in one call, on the spot. The net gain is also 0.04, and again nothing is held.
- Without the mechanism: buy the three NO for 1.96 and hold them to resolution, where they pay two dollars. The gain is the same 0.04, but the 1.96 dollars stay locked until the match ends. What the negative-risk mechanism does is to return those 1.96 dollars early. Splitting one dollar on each contract and selling the three YES at the bids comes to exactly the same thing: it leaves the same three NO in hand and the same 1.96 dollars locked.

The buy side. At 02:28:24 UTC the asks of the three YES contracts were 0.20, 0.36 and 0.35, which sum to 0.91, less than one dollar; the bids at the same moment were 0.08, 0.32 and 0.33.

- Buy the three YES for 0.91 and hold the basket. At resolution exactly one of them pays one dollar, so the gain is 0.09, but the 0.91 dollars stay locked until the match ends.
- Selling a basket of NO instead leads to the same position: split one dollar on each of the three contracts, paying three dollars for three YES and three NO, and sell the three NO at their bids of 0.80, 0.64 and 0.65 for 2.09 in total. The net cash outlay is still 0.91, and the holding is still the basket of YES: it is the same trade under another name.
- The only way to cash out early is to sell the three YES back at their bids, which brings only 0.73, a loss of 0.18; conversion cannot help, because it consumes NO and there is no NO in hand. So the buy side’s 0.09 waits for resolution.

Figure A.2 draws the two moments: each token’s ask and bid, the NO prices mirroring the YES prices, and the sums against the coherent values 1 and 2.

Figure A.3 shows the same two moments with fees included. At 02:44:45 UTC, as Algeria’s equalizer reached the books, the fee-adjusted bids summed to 1.0222; the replay minted 68 baskets for \$68, sold them for \$69.51 and earned \$1.51 at once. At 02:28:24 UTC, as Austria scored, the fee-adjusted asks summed to 0.9285; the replay bought 136 shares of each contract for \$126.28 and earned \$9.72 at resolution.

(a) Sell side, 02:44:45 UTC: bids sum to $1.04 > 1$				(b) Buy side, 02:28:24 UTC: asks sum to $0.91 < 1$			
	YES token		NO token		YES token		NO token
A Algeria	YES	0.18 ask 0.16 bid	NO 0.84 ask 0.82 bid	A Algeria	YES	0.20 ask 0.08 bid	NO 0.92 ask 0.80 bid
B Draw	YES	0.55 ask 0.31 bid	NO 0.69 ask 0.45 bid	B Draw	YES	0.36 ask 0.32 bid	NO 0.68 ask 0.64 bid
C Austria	YES	0.58 ask 0.57 bid	NO 0.43 ask 0.42 bid	C Austria	YES	0.35 ask 0.33 bid	NO 0.67 ask 0.65 bid
coherent: $YES_A + YES_B + YES_C = 1$, $NO_A + NO_B + NO_C = 3 - 1 = 2$ sell YES at the bids: $0.16 + 0.31 + 0.57 = 1.04$ (mint for \$1 first) NO side, same trade: buy at $0.84 + 0.69 + 0.43 = 1.96 < 2$, convert				coherent: $YES_A + YES_B + YES_C = 1$, $NO_A + NO_B + NO_C = 3 - 1 = 2$ buy YES at the asks: $0.20 + 0.36 + 0.35 = 0.91$, hold to resolution NO side, same trade: split, sell at $0.80 + 0.64 + 0.65 = 2.09 > 2$			

Figure A.2: The quotes (fees left out) at two moments of the Algeria–Austria World Cup match on 28 June 2026, 02:44:45 UTC in panel (a) and 02:28:24 UTC in panel (b). Each contract’s YES and NO tokens branch into an ask (buy) and a bid (sell); a NO ask is one minus the YES bid and a NO bid one minus the YES ask. The prices the text uses are in red, the others in gray. Coherent prices sum to one over the YES tokens and to two over the NO tokens. (a) The bids sum to 1.04: sell the three YES after minting, or buy the three NO for 1.96 and convert. (b) The asks sum to 0.91: buy the three YES and hold, or split and sell the three NO for 2.09, the same trade.

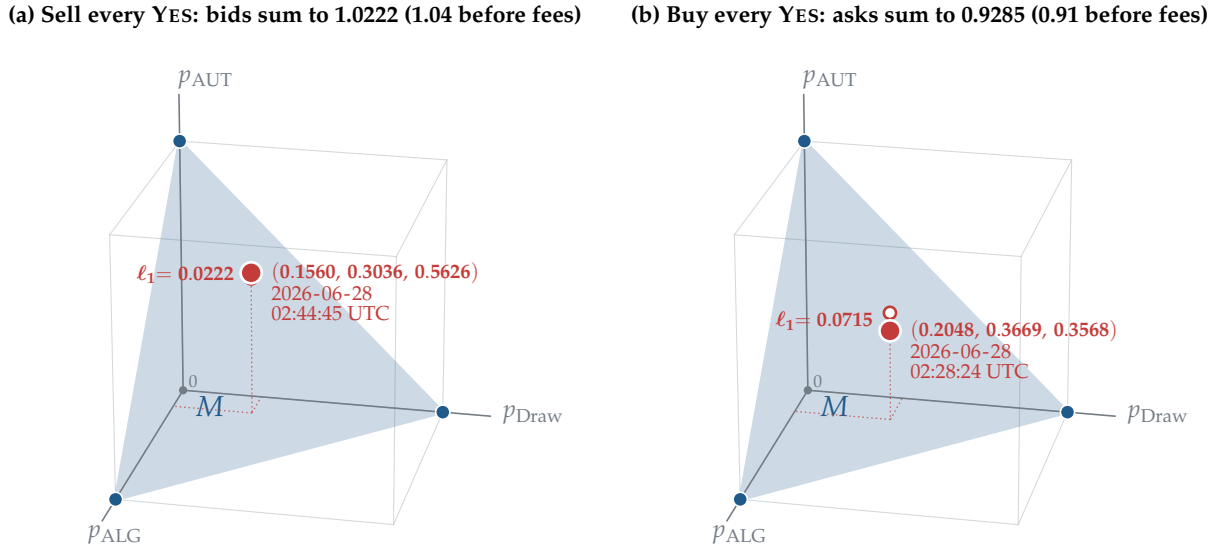


Figure A.3: Two sides of a partition: the Algeria–Austria match-result contracts on 28 June 2026, with the coherent set M in blue and the red segment marking the ℓ_1 distance. (a) At 02:44:45 UTC the fee-adjusted bids sum to 1.0222 (1.04 before fees), above M : minting a basket for one dollar and selling every YES earns 0.0222 per basket at once. (b) At 02:28:24 UTC the fee-adjusted asks sum to 0.9285 (0.91 before fees), below M : buying every YES earns 0.0715 per basket at resolution.

Appendix B Data, Relation Extraction, and Layered Detection

B.1 Data Construction

B.1.1 Metadata and Resolution Criteria

We retrieve the market catalog from Polymarket’s Gamma API, following its pagination to collect the event, condition, and outcome-token records. We retain the raw responses before constructing the dataset. The catalog supplies the event hierarchy, condition and outcome-token identifiers, trading status, fee parameters, negative-risk identifiers, and the displayed resolution text. Because displayed catalog text can change after listing, it is not by itself a listing-time record of the rules under which a contract was introduced.

Polymarket’s UMA adapters, both regular and negative-risk, emit `QuestionInitialized` events whose ancillary data record the resolution question when a condition is initialized, so the listing-time rule is recoverable on chain through the question and condition identifiers. The relation audit reads the catalog text preserved at retrieval. We do not compare that text with the listing-time record, so the checks assess relations under the displayed rules at retrieval.

B.1.2 Live Full-Depth Order-Book Collection

Polymarket does not archive historical full-depth order books, so we collect its live order-book messages. Contracts can begin and finish trading between refreshes of the contract list. To capture these contracts, we first retrieve the list of contracts accepting orders and subscribe to the messages of every outcome token, spreading the tokens over as many WebSocket connections as needed, with at most 600 on each to avoid exceeding the token limits per connection that Polymarket’s server accepted. We then monitor `new_market` messages and add newly listed contracts within about 30 seconds. Every three hours, we refresh the contract list, remove contracts no longer accepting orders, and rebuild the connections. A book message reports bid and ask quantities at every displayed price level; a subsequent `price_change` message replaces the quantity at a specified level. Reconnecting supplies a new snapshot. After verifying that updates for the YES and NO tokens are complementary, we retain the canonical YES-side book described in Section 2.1.

B.1.3 Offline Order-Book Reconstruction

For detection, we initialize each order book from its latest valid snapshot and apply subsequent price-level updates in exchange-timestamp order. We apply all messages sharing a timestamp before evaluating the books and retain both synchronized best quotes and the full-depth messages. Changes in quotes or contract eligibility determine when we evaluate each connected component. Appendix B.4 explains the ordering and checks for gaps.

For each detected arbitrage window, we reconstruct the full-depth books from opening to closing, starting from the latest valid snapshots within the preceding six hours. The arbitrageur buys at the asks and sells at the bids, pays the contract-specific taker fee at each price level, and reoptimizes quantities over the depth remaining after earlier executions. The arbitrageur cannot reuse consumed depth, even if an unchanged snapshot still displays it. These books supply the execution-cost function in Section 3.3.

B.1.4 On-Chain Fills

We decode every `OrderFilled` event of the regular and negative-risk CTF Exchange V2 contracts over the window into outcome token, side, price, quantity, fee, maker and taker identifiers, transaction hash and log index, and block timestamp. The token identifiers link each fill to the catalog and the reconstructed order books, and shared transaction hashes identify fills settled in the same transaction. Recorded maker and taker fields identify addresses rather than ultimate economic owners; Section 5.4 matches fills to the detected portfolios by contract, direction, and time.

B.2 LLM Relation Extraction

B.2.1 Candidate Retrieval

We represent each contract by its event title, question, and group title. Words and adjacent word pairs receive inverse-document-frequency weights, and we rank contract pairs by cosine similarity or weighted containment. The search runs separately within four broad topic groups. Contracts with the venue’s typed sports fields are handled by venue-derived relations; other contracts concerning sports remain eligible for semantic inference. For each contract, we keep up to eight candidates from other events and three from the same event above a fixed similarity threshold. We exclude pairs within the same negative-risk group and pairs with nonoverlapping trading intervals. Candidate groups contain between three and twelve contracts, drawn either from all contracts listed under an event or from contracts linked by the pairwise search.

B.2.2 Prompt Templates

The two prompts beginning on the next page are reproduced verbatim. The first requests pairwise relations and the second requests multivariate relations; both treat the model’s output as a proposal requiring further review. The audits reported in Section 4.2 use the catalog text preserved at retrieval.

```

1 SYSTEM_PROMPT = """"You extract candidate logical relations between prediction-market YES outcomes.
2 Judge from the supplied as-of-retrieval question and resolution text only. A relation must hold in every
3 possible world allowed by the rules; matching topics or one realized outcome is not enough.
4
5 Labels:
6 - equivalent: A and B resolve Yes in exactly the same worlds.
7 - a_implies_b: whenever A resolves Yes, B must resolve Yes, but not necessarily conversely.
8 - b_implies_a: whenever B resolves Yes, A must resolve Yes, but not necessarily conversely.
9 - disjoint: A and B cannot both resolve Yes.
10 - overlap_uncertain: related outcomes, but no listed relation is certified by the supplied rules.
11 - unrelated: not the same underlying proposition or no meaningful logical connection.
12
13 Be conservative about deadline, measurement source, jurisdiction, office/control versus election win,
14 temporary versus permanent events, and market-creation lower bounds. Evidence strings must be exact
15 substrings copied from the supplied A or B fields. Model output is only a candidate for manual audit.""

```

```

1 SYSTEM_PROMPT = """"You extract candidate multivariate logical relations among prediction-market YES outcomes.
2 Judge only from the supplied as-of-retrieval question and resolution text. A relation must hold in every
3 possible world allowed by those rules. The result is a candidate for listing-time manual audit, not authority.
4
5 Return zero or more irreducibly group-level relations:
6 - at_most_one: no two listed members can both resolve Yes; sum(members) <= 1.
7 - at_least_one: at least one listed member must resolve Yes; sum(members) >= 1.
8 - exactly_one: exactly one listed member resolves Yes; sum(members) = 1.
9 - complete_partition_of_parent: mutually exclusive children completely partition parent;
10   sum(members) = parent.
11 - incomplete_partition_of_parent: mutually exclusive children are only a subset of parent;
12   sum(members) <= parent. Do not use merely because each child implies parent unless mutual
13   exclusivity is also certified.
14 - union_equivalent: parent resolves Yes iff at least one member resolves Yes; members may overlap.
15 - intersection_equivalent: parent resolves Yes iff every member resolves Yes.
16
17 Do not emit ordinary pairwise implication/equivalence/disjoint; the pairwise branch handles those.
18 Every condition id must come from the supplied group. Parent must be empty for at_most_one,
19 at_least_one, and exactly_one; otherwise it must be a supplied condition not repeated among members.
20 Use the smallest sufficient member set. Missing markets matter: when group_complete_view is false,
21 do not infer exhaustiveness merely because the displayed list looks complete. An exhaustive relation
22 may still be emitted if the supplied rule text itself proves it. Evidence quotes must be exact
23 substrings from each involved market's supplied fields. Be conservative about deadlines, measurement
24 sources, fallback outcomes, other/none categories, cancellation rules, and changed resolution rules.""

```

B.2.3 Illustrative Model Outputs and Audit Responses

The following three examples show how the proposal models classify relations and how the audit models assess them. We reproduce here the substantive fields of their pairwise proposal outputs and audit verdicts. The examples cover sports, politics, and finance and include both retained and rejected proposals. In each case, the two audit models independently read the resolution criteria and assess the proposed relation. The proposal model's label, confidence, and explanation do not serve as evidence. All three examples apply the same rule: we retain a proposal only when both audit models independently support the same relation from the contract criteria. A shared topic does not establish a logical relation, while an admissible counterexample suffices to reject it.

Sports: Qwen3-235B-A22B (retained). The input pairs contract A, “Will the announcers say ‘Goal’ 20+ times during the Czechia vs Mexico FIFA World Cup Match?,” with contract B, the otherwise identical 40+ threshold.

```

1 relation_type: b_implies_a
2 confidence: high
3 a_evidence: ["Will the announcers say 'Goal' 20+ times during the Czechia vs Mexico FIFA World Cup Match?"]
4 b_evidence: ["Will the announcers say 'Goal' 40+ times during the Czechia vs Mexico FIFA World Cup Match?"]
5 rationale: "Both markets refer to the same event (Czechia vs Mexico World Cup match on June 24, 2026) and
  ↳ the same term ('Goal'). If the announcers say 'Goal' 40+ times (B), then they must have said it 20+
  ↳ times (A). So B implies A. However, A does not imply B (could be 25 times). Therefore, the correct
  ↳ relation is b_implies_a."
```

The final audit outcome is as follows.

Claude Fable 5.1 audit: b_implies_a, high confidence.

GPT-5.6 Sol audit: b_implies_a, high confidence.

Decision: Both verify the same event, term, broadcast, and measurement window; the 40+ threshold is nested within the 20+ threshold. Retained.

Politics: Llama-4-Maverick (rejected). The input pairs contract A, “Will Donald Trump visit Virginia in 2026?,” with contract B, “Will Donald Trump visit Michigan in 2026?”

```

1 relation_type: disjoint
2 confidence: high
3 a_evidence: ["Virginia", "2026"]
4 b_evidence: ["Michigan", "2026"]
5 rationale: "The markets are about visiting different states."
```

The final audit outcome is as follows.

Claude Fable 5.1 audit: No relation, high confidence.

GPT-5.6 Sol audit: No relation, high confidence.

Decision: Visiting one state does not preclude visiting the other; $A = B = 1$ is feasible. Rejected.

Finance: GLM-5.2 (retained). The input pairs contract A, “Will Databricks’ valuation hit (HIGH) \$160B by December 31?,” with contract B, “Will Databricks’ valuation hit (HIGH) \$190B by July 31?” The output shown is the third independent GLM-5.2 evaluation of this pair.

```

1 relation_type: b_implies_a
2 confidence: high
3 a_evidence: ["Will Databricks' valuation hit (HIGH) $160B by December 31?"]
4 b_evidence: ["Will Databricks' valuation hit (HIGH) $190B by July 31?"]
5 rationale: "If Databricks' valuation reaches $190B by July 31, it necessarily reaches or exceeds $160B by
  ↳ December 31 (since July 31 is before December 31). The converse does not hold."
```

The final audit outcome is as follows.

Claude Fable 5.1 audit: b_implies_a, high confidence.

GPT-6 Astra audit: b_implies_a, high confidence.

Decision: Both verify the same Nasdaq Private Market series, a higher threshold for B, and a narrower window for B. Retained.

B.3 Detecting in Three Layers

Table B.1: Arbitrage detection by relation layer

Layer	Admissible payoff set	What detection solves	Size of the system
Venue-asserted	Group restrictions $\sum_{i \in G} z_i \leq 1$, with equality for the 20,118 groups treated as exhaustive	Sell when $\sum_{i \in G} \tilde{b}_i > 1$ over the quoted members; buy when $\sum_{i \in G} \tilde{a}_i < 1$ only for exhaustive groups with every member quoted	20,866 groups
Venue-derived	Binary payoff vectors satisfying restrictions derived from the common score state of a match	Enumerate feasible payoff vectors within active blocks and maximize guaranteed profit at best quotes, subject to the leg cap	3,291 matches
Semantic-inferred	$\Phi = \{z \in \{0, 1\}^I : Fz \leq g\}$, with rows from parsed and model-inferred relations	Enumerate feasible payoff vectors within active blocks and maximize guaranteed profit at best quotes, subject to the leg cap	118,898 rows, 61,349 contracts, 3,591 components

Notes: $\tilde{b}_i$ and $\tilde{a}_i$ are fee-adjusted best bids and asks, as in Table 2.

What each detector solves. We detect arbitrage separately within each relation layer. Each layer supplies its own payoff restrictions, which determine the worst-case payoff of a proposed portfolio. Table B.1 summarizes these restrictions and how we use them in detection.

Venue-asserted and venue-derived relations. A negative-risk group of n contracts admits $n + 1$ payoff vectors: one for each contract resolving YES and one with every contract resolving NO. For the 20,118 groups compiled as exhaustive partitions, we exclude the all-zero vector. These restrictions yield the bid and ask sum tests in Table B.1. Selling uses whichever members are quoted, whereas buying the basket requires a quote on every member of an exhaustive group. For venue-derived relations, we first map contracts on the same match to a score grid containing between 121 and 1,681 states. We use these states to derive implication, exclusion, partition, and conjunction restrictions. Detection then minimizes over the binary payoff vectors admitted by the compiled restrictions, using the same restrictions in the full-depth calculation. This set can include payoff vectors that no score state produces, so its minimum payoff can be lower than the minimum over the score grid.

Compiling the semantic layer. Semantic-inferred relations link contracts across events without a common observed variable, such as a final score, that determines all their payoffs. The compiled rows describe a payoff set in $\{0, 1\}^{61,349}$ without requiring us to list every vector. Algorithm 1 provides a general solution by adding payoff vectors as needed. Our implementation instead enumerates the feasible vectors within each active block and reuses them across quote states while the constraints remain unchanged. For a given block and constraint system, this gives the same worst-case payoff as the integer oracle. We do not evaluate active blocks containing more than 16 contracts in the semantic-inferred layer; the corresponding caps are 32 for venue-asserted and 48 for venue-derived relations. Doubling the semantic-inferred cap to 32 contracts changes detection little:

the detector finds 161,019 arbitrage windows instead of 158,175, and 157,862 windows are identical under both caps. The 5-leg cap separately restricts venue-derived and semantic-inferred portfolios; sales in venue-asserted groups are exempt, as described in Section 5.1.

Three separate systems. We keep the three systems separate so that a portfolio’s guaranteed profit remains attributable to the relations that support it. Combining them would allow portfolios to rely on restrictions from several sources. As Section 4.1 shows, an invalid restriction can create a false guarantee, whereas an omitted valid restriction can only reduce the calculated guaranteed profit. We therefore assess each layer using its own evidence: the venue’s collateral mechanism and group rules, typed metadata, or contract text.

Corrective excision. An audit on September 22, 2026 identified six parser-generated relations that incorrectly imposed collective exhaustiveness. The contracts permitted cases in which every member resolves NO, including no offering, no prison time, and a rate increase between two contracts’ cutoff dates. As an ex-post correction to the completed replay, we relax the affected equality constraints to impose only mutual exclusivity and recalculate the guaranteed profit of the saved executions in the 316 windows of the affected components. We excise 9,172 executions whose guarantees become negative, including the first take of 97 windows. This reduces the first take by \$2,201 and the accumulated take by \$17,404, all within the semantic-inferred layer. Eighty-two of these windows had settled at a profit, illustrating why profitable settlement does not validate an ex-ante guarantee. We do not rerun the liquidity ledger or make depth consumed by excluded trades available to other trades.

Some semantic-inferred relations overlap with or extend venue-asserted relations. Of the 118,898 rows in the semantic-inferred system, 133 involve only contracts within one negative-risk group. Of these, 106 repeat the venue’s mutual-exclusivity restriction, while 13 partitions, two rows each, impose exactly-one and one row imposes an implication between members. The latter restrictions are stronger than group membership alone; the 13 partitions coincide with exhaustiveness that Layer 1 already imposes on these groups. Another 27,276 rows, or 22.9% of the system, include a group member: 11,453 connect two or more groups, and 15,823 connect a group member with an ungrouped contract. Keeping the systems separate excludes portfolios that require combining these relations with a group’s mutual-exclusivity restriction. No row includes a contract with typed sports fields, since those contracts are handled by score mappings and excluded from semantic inference.

We follow one arbitrageur across all three relation layers in Section 5. The arbitrageur fills displayed depth only once and records consumed quantities in a shared liquidity ledger. This depth is unavailable for later trades, including those in other windows, layers, or days. If the same portfolio becomes profitable again, the arbitrageur can trade only against depth not already consumed. In the baseline, the arbitrageur can trade a quote re-posted at a previously consumed price as new depth; Section 5.2 examines this assumption. The 491,939 detected windows contain 39,257 distinct portfolios, of which 26 appear in more than one layer. These portfolios account for 1,461 windows, including 350 executed windows with a total first take of \$340.

B.4 From Raw Messages to Settlement: Measurement Details

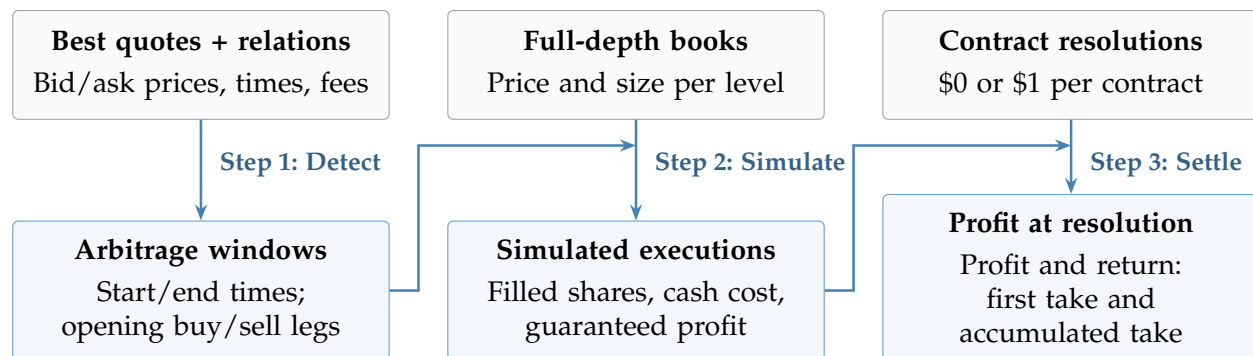


Figure B.1: Inputs (top) and outputs (bottom) of the three measurement steps. Within each window, the arbitrageur trades against the available depth; contract resolutions then determine profit. Best quotes and full-depth books use the same raw messages.

This subsection expands the three steps of Section 5.1. Figure B.1 shows their inputs and outputs. Table B.2 reports their key statistics. The paragraphs below describe their procedures.

Step 1: detect arbitrage windows from best quotes. We reconstruct best quotes from raw snapshots and price-level updates. We store the messages in a common format, preserving exchange and receipt timestamps, source order, token identifiers, and quantities without sampling. We also retain the raw archive. We order messages by exchange timestamp, including delayed messages received in later files, to align quotes across connections. Messages with the same timestamp retain their source-file, row, and within-message order. We remove exact duplicates and apply all messages sharing a timestamp before evaluating the books, so partial updates do not create artificial windows.

A snapshot replaces both sides of the book. A price-level update reports the new resting quantity, rather than a change in quantity; zero removes the level. We evaluate a book only after receiving a snapshot. After applying each timestamp batch, we compare the reconstructed best quotes with those reported in the final price-level update. If they disagree, we exclude the book until agreement resumes or a new snapshot restores it. We record these gaps and separately count missing reported quotes. This check verifies best quotes, but cannot establish the completeness of depth at other price levels. Both detection and full-depth reconstruction use the same exchange timestamps.

Detection uses quoted contracts and valid constraints for which all required contracts are available. Active blocks exceeding the respective layer caps of 32, 48, and 16 contracts are recorded and excluded from evaluation. Portfolios are capped at 5 legs, except for sales in venue-asserted groups (Section 5.1). We evaluate every relevant update, including changes in fees, trading eligibility, and the validity of relations. A previous solution is reused only when the objective, constraints, variable domains, bounds, and solver settings are unchanged; quantities or fees can change the problem even when prices do not move. We continue checking book and window states when reusing a solution and retry failed optimizations. For each window, we save the opening portfolio, supporting relations, profit and capital measures, and the half-open interval $[t_{\text{open}}, t_{\text{close}})$.

Table B.2: Data reconstruction, execution, and settlement

Statistic	Result
<i>Step 1: reconstruction and detection</i>	
Order-book messages (billions)	73.22
Reconstructed best-quote comparisons (billions)	13.3
Agreement with venue-reported best quotes	97.2%
Arbitrage windows detected	491,939
<i>Step 2: simulated arbitrage trades</i>	
Percentage of detected windows with an execution	25.4%
Executions	462,151
Percentage of executed windows first filled after opening	12.2%
Opening-to-first-execution delay, 99th percentile (seconds)	231.9
<i>Step 3: settlement</i>	
Percentage of executed windows settled	89.8%
Executions in settled windows	391,383
Settled windows earning less than guaranteed profit	0

Notes: First-execution delays use exchange timestamps and cover executed windows. The final row checks both the first take and the accumulated take.

Step 2: simulate arbitrage trades against full-depth order books. Within each detected window, the arbitrageur trades the opening portfolio’s legs in their original directions and reoptimizes quantities at each eligible update. We retain the window’s original opening and closing times, including when the arbitrageur has exhausted its available depth. We initialize each required book from the latest valid snapshot within the preceding six hours. Without such a snapshot, or during a recorded gap, the arbitrageur cannot trade that contract. The arbitrageur buys at the asks and sells at the bids, paying the applicable taker fee at each price level. If a leg is unavailable or depleted, its quantity is set to zero; the arbitrageur may still trade a profitable feasible subset.

The arbitrageur records consumed depth by token, book side, and price in a shared liquidity ledger across days, windows, and relation layers. Let $O_{\ell,t}$ be the displayed quantity at price level ℓ , R_{ℓ,t^-} the quantity remaining after previous executions, and $q_{\ell,t}$ the quantity the arbitrageur trades at time t . After an order-book update, the arbitrageur can trade the available quantity $R_{\ell,t}^{\text{available}}$; after execution, the remaining quantity is $R_{\ell,t}$:

$$R_{\ell,t}^{\text{available}} = \max\{0, R_{\ell,t^-} + O_{\ell,t} - O_{\ell,t^-}\}, \quad R_{\ell,t} = R_{\ell,t}^{\text{available}} - q_{\ell,t}.$$

For example, an arbitrageur who has filled 100 displayed shares cannot fill them again when another snapshot shows the same 100 shares. If displayed depth rises to 130, the arbitrageur can fill 30 additional shares. When opportunities overlap, the arbitrageur trades first in the window with the highest guaranteed profit at displayed depth, breaking ties by window identifier. Each trade reduces the depth available for subsequent trades. This sequential allocation need not maximize profit jointly across windows. We process windows sharing liquidity together in exchange-time order, preserving the allocation when calculations run in parallel.

The first take is the guaranteed profit from the arbitrageur’s first execution, which may occur after the window opens; the accumulated take sums guaranteed profit from all executions in the window. Windows without an execution remain in the count of detected windows. Price-level data do not identify individual orders, so cancellation followed by re-posting can supply new depth in the baseline. The robustness checks vary the treatment of re-posted depth, the participation rate, and execution latency separately. With execution latency, the arbitrageur waits the specified delay after the relevant quote information becomes available and then trades against the order book at execution time. For every execution, we save the quantities filled at each price level, cash outlay, fees, guaranteed profit, and committed capital.

Step 3: test arbitrage windows at resolution. We match the arbitrageur’s simulated executions to contract resolutions. Before calculating profit, we check that all outputs are present without duplication, use the same inputs and settings, and preserve every detected window. We also verify that execution-level amounts sum to window totals. We assign resolution payoffs only to legs with positive filled quantities. The saved resolution data link each contract to the token listed first in the catalog, which we use as its canonical YES side even when its displayed name differs. We use explicit resolution or winner records to determine payoffs, not a closed market’s last price. Missing, unresolved, and void outcomes remain distinct and are never replaced by zero. A window is settled only when every filled leg of every execution has a binary payoff. We also record whether the first execution’s legs have settled.

For each settled execution, profit at resolution is $\mathbf{y}^\top \mathbf{z}^{\text{resolved}} - K(\mathbf{y})$, using the same filled shares, cash flows, and fees that produced its guaranteed profit. Comparisons of the first take and the accumulated take use the same settled windows and the capital committed to their respective executions. For venue-asserted negative-risk portfolios, the arbitrageur needs one dollar of collateral for each minted basket; the number of baskets equals the largest quantity sold on any leg. For other portfolios, the arbitrageur needs one dollar for each share sold on each leg. Both measures also include cash paid for purchases. We compare profit at resolution with guaranteed profit and record any shortfall. Section 5.4 separately compares the simulated portfolios with actual trades.

Reproducibility checks. For each calculation, we record the code, software and solver versions, input checksums, settings, and output counts. The saved resolution data allow us to repeat the analysis at the same cutoff date. We check that raw and reformatted messages produce the same books, that reusing unchanged optimization problems preserves the detected windows, and that the arbitrageur cannot reuse depth across days or equivalent token representations. We further check that delayed first executions are recorded correctly and ensure that windows with unresolved or void filled legs are excluded from settlement comparisons.