An Axiomatic Framework for No-Arbitrage Relationships in Financial Derivatives Markets*

Steffen Schuldenzucker

First version: March 16, 2016
This version: May 26, 2016

Abstract No-arbitrage relationships are statements about prices of financial derivative contracts that follow purely from the assumption that no market participant can make a risk-free profit. They are a fundamental tool of modern finance and basis to all modern market models. The ever-growing complexity of financial derivatives impairs the effectiveness of conventional approaches based on expected payments for understanding these relationships. In this paper, we introduce the *Logic Portfolio Theory (LPT)*, a new framework in typed first-order logic with higher-order functions that allows users to prove no-arbitrage relationships based on the syntactic structure of contracts. We first show that LPT is rich enough to replace informal or stochastic arguments by proving the well-known *put-call parity* and *Merton's theorem* inside the theory. This also yields the most general versions of these no-arbitrage relationships to date. We second show that LPT is general enough to encompass both a simple stochastic model and a purely cash flow oriented model.

1 Introduction

Financial markets are ubiquitous in economies around the world. Every day, billions of dollars in assets are traded, including pork bellies, company shares, currencies, and complex financial derivatives. To understand the processes that govern these complex markets, economists as well as practitioners use *market models*. These models cannot take into account all aspects of a real market, so simplifying assumptions are needed. An assumption fundamental to all modern financial market models is the *no-arbitrage* principle, i.e., that no market participant can make a profit without taking risk.

To understand what this means, consider two assets that are traded in a market: a holder of asset A receives \$1 in case exactly one month from now, the price of a certain stock has increased while a holder of asset B receives \$1 if it has gone down or remained equal. Such assets would be considered *derivatives* because payments depend on another *underlying* asset, the stock. If the sum of the prices of the two assets is below \$1, say \$0.80, then a trader could buy both, receive a guaranteed payment of \$1 one month later, and (ignoring interest) make a profit of \$0.20. It is widely accepted that such an *arbitrage opportunity* should not exist in efficient markets for longer than microseconds because it would be quickly exploited and prices would adjust by the rules of supply and demand until it is no longer profitable. We can thus conclude by the no-arbitrage principle that the sum of the two prices must be at least \$1. This is what we call a *no-arbitrage relationship* between the assets A and B.

Note that this relationship holds independently of the choice of a market model because we did not make any assumptions on, e.g., the probability distribution of the underlying stock. In fact, we did not use the word "probability" at all. At a higher level, if we find a way to prove formally that the price of one contract or a combination of contracts must always be less than or equal to that of another one purely due to the no-arbitrage assumption, then we can be certain that this relationship will hold in any market model. Vice versa, a complete theory of no-arbitrage relationships of financial contracts would yield a compact description of the properties common to all market models.

Such a theory needs to deal with a large class of financial derivatives. While the assets from our example were simple (the holder received a payment at a fixed

* Steffen Schuldenzucker: Department of Informatics, University of Zurich, Switzerland. Email: schuldenzucker@ifi.uzh.ch. The pronoun "we" is used to refer to the author. We would like to thank (in alphabetical order) Patrick Bahr, the members of the Computation and Economics Research Group at the University of Zurich, the members of the Domain-Specific Languages group at the HIPERFIT cluster, Jean-Marc Eber, Stefan Geschke, and Sven Seuken for helpful comments on this work. Any errors remain our own.

future point in time, contingent on some underlying value), the derivatives one meets in today's financial industry can be much less so. Especially those contracts that the counterparties make directly "*over the counter*" rather than through a stock exchange, can reach an impressive degree of complexity. Counting a few examples, financial contracts may contain clauses stating that under certain conditions

- one or both parties are granted the right to exit early from the contract or extend it, possibly paying a premium,
- the contract is renewed periodically based on current or historic market data or based on a choice of one or both counterparties, or that
- further payments are delayed until a trigger event occurs or one of the counterparties decides to take action.

Researchers have been lacking a basic *language* to talk about these complex derivatives and their no-arbitrage relationships. Classical nonstochastic approaches (Varian, 1987) could not even express concepts such as "choice". Methods inspired by prior-free stochastic analysis (Björk, 2004) do neither seem to capture well the (essentially nonstochastic) nature of arbitrage arguments because they inherit a focus on finding the *fair price* (in terms of expected payments) of a contract, rather than on the relationships between prices of different contracts. Complexity of the expression for the fair price grows quickly in the complexity of the contract and often additional assumptions are needed to ensure that a price exists in the first place. Probability is also a very complex theory, creating difficulties in applications such as automated reasoning.

In the present paper, we address the problem of designing a rigorous, general framework for arbitrage arguments. We focus on the structure of contracts rather than on their prices. No-arbitrage relationships can be proved by analyzing this structure syntactically. The fundamental rules that relate different contract patterns correspond to fundamental arbitrage strategies.

More in detail, we present *LPT*, the *Logic Portfolio Theory* of arbitrage-free financial derivatives markets. LPT is a theory in typed first-order logic with higher-order functions. The *language* of LPT (Section 2) provides means for expressing financial contracts formally in terms of their fundamental building blocks (payment, delay, choice, reference to market data, etc). It is modeled closely after the seminal work on the formalization of contracts by Peyton Jones and Eber (2005). The *axioms* of LPT describe the no-arbitrage relationships between different building blocks. Our main contribution is the axiomatic description of a partial order on contracts we call the *arbitrage relation*: $x \preceq_b y$ means that the no-arbitrage condition implies that the price of the contract x must be below or equal to the price of the contract y whenever conditions b hold (Section 3). We then demonstrate the expressive power of our framework: we first show how LPT can complement pricing and help find bounds on prices when exact values are not known (Section 4). We next show how two no-arbitrage relationships well-known in the literature can be proved formally within LPT: the put-call parity and Merton's theorem.[1] As we will see, LPT in fact allows us to receive variants of these theorems more general than shown in prior work (Section 5). Finally, we describe two different models of the theory: a simple probabilistic model and a new model based on cash flows (Section 6). This raises confidence that LPT can indeed describe the class of all arbitrage-free financial market models.

Object-oriented frameworks (Arnold et al., 1995; Eggenschwiler and Gamma, 1992) and domain-specific languages (Frankau et al., 2009; Harrington et al., 2014) have been used by banks since the early 90s for flexible pricing of derivatives. Peyton Jones et al. (2000) were the first to describe a domain-specific language that could describe the *structure* of a financial contract rather than its *payout*. Despite this tradition, we are, to the best of our knowledge, the first to define an axiomatic system to reason about the price relationships in a formal financial contract language. Berthold (2011, 2013) provided an algorithm to compute a "canonical form" of a contract in an extension of Peyton Jones and Eber's (2005) language, which, however, fails to derive already basic identities that are true under no-arbitrage assumptions. Bahr et al. (2015) defined a simple language of financial contracts together with a semantic equivalence relation. This relation deliberately does not ignore *irrational choices* such as not exercising an option that leads to a risk-free profit and hence has fewer equivalences than can be shown via arbitrage arguments. Many of the mentioned authors also described a denotional semantics for their languages. LPT extends this concept by allowing several different semantics as models of the theory. A second branch of related research is formed by general business contract languages (Hvitved (2012) provides a survey); the generality of these approaches however

[1] Cf. (Hull, 2012) for an informal introduction and a discussion of no-arbitrage relationships in various stochastic models. An informal introduction is also provided at the beginning of Section 5.

prohibits a notion of arbitrage-freeness, making them unsuitable for the purpose of this work.

LPT defines a unique algebra of financial contracts, the elementary operations of which are the contracts' fundamental building blocks. Its rich language allows to express a large class of contracts and to combine them in unlimited ways. The theorems obtained are general and follow purely from the no-arbitrage principle without reference to any particular market model. Additional assumptions beyond the no-arbitrage principle can be expressed and reasoned about inside the theory to obtain stronger conclusions. Since LPT is a regular first-order theory (with higher-order functions), proofs can be carried out using the standard mathematical tools and intuition and standard model theory can be applied. To the best of our knowledge, we are the first to suggest a rigorous, logic-based framework to study arbitrage at a syntactic level.

1.1 Preliminaries

LPT is presented as a theory in typed first-order logic with higher-order functions[2] (Manzano, 1996). It therefore consists of two parts: first, the *language*, naming the *types* (i.e., different kinds of objects the theory talks about) and the elementary *operations* (functions to construct new objects from existing objects). Second, the *axioms*, specifying how the operations are related. Proofs are then constructed from the axioms in the usual fashion. Recap that a *model* of a theory defines a set for each type and a function for each operation such that the axioms hold. For example, the theory of vector spaces defines types `Vector` and `Scalar` with operations such as $(+)$ and $(\cdot)$ and the vector space axioms. Models of this theory are the vector spaces.

Our goal is to define LPT in such a way that the types encompass any kind of data contracts refer to together with the `Contract` type itself, the operations define the essential building blocks of contracts, and the axioms describe the relationships between contracts under no-arbitrage assumptions, so that the models are exactly the arbitrage-free market models. We can then write down and reason about contracts without explicit reference to any particular model. We would like to stress that we do not define what a financial contract "really is": the approach is completely symbolic.

We write $x : a$ to indicate that object x should have type a. a is a *type variable* here, i.e., it can stand for any type. A function with argument x is introduced by "$\lambda\, x.$". For example, $\lambda\, x.\ x+1$ describes the function that adds 1 to its argument. Following the tradition of (Peyton Jones and Eber, 2005), we write function application by juxtaposition, i.e. we write $f\ x\ y$ instead of $f(x, y)$.

[2] A translation into regular first-order logic is possible, too; cf. Appendix A for details.

2 The Language of LPT

In this section, we describe the language of LPT. We first describe our framework for *observables* (elements of market data) and then go on to describe how financial *contracts* are expressed. The language presented here is a modification of the contract language by Peyton Jones and Eber (2005).

We assume that elementary types such as `Real` (real numbers), $\texttt{Real}^+$ (non-negative real numbers), and `Bool` (boolean values `True` and `False`) are given externally, together with constant symbols such as $0 : \texttt{Real}^+$ or $\texttt{True} : \texttt{Bool}$ and operations like $(+) : \texttt{Real} \times \texttt{Real} \to \texttt{Real}$ or $\texttt{if}\ \ldots\ \texttt{then}\ \ldots\ \texttt{else}\ \ldots : \texttt{Bool} \times a \times a \to a$. We assume that axioms are provided to have these types match their expected meaning. In addition, let a type `Time` (points in time) be given together with a linear ordering. The axioms of LPT do not enforce a specific layout for the `Time` type: models could implement `Time` as $\{1, \ldots, T\}$, $\mathbb{N}$, or $\mathbb{R}$.

2.1 Observables: Formalizing Market Data

Derivatives typically refer to elements of market data in their definition. For example, the assets A and B described in the introduction referred to a stock price. In more abstract terms, contracts can refer to time-varying quantities the past of which is known, but the future is uncertain, and which we call *observables*.

If a is a type, then the type of observables taking values in a is denoted by `Obs` a. For example, `Obs Real` is the type of numeric time-varying quantities. Stock prices and interest rates cannot be negative and are thus of type $\texttt{Obs}\ \texttt{Real}^+$. `Obs Bool` denotes *market conditions* that are `True` or `False` at any point in time. `Obs Bool` will have a distinguished role in the framework.

A useful intuition for an observable is a "stream of values". For example, an object of type `Obs Real` can be interpreted as a stream of real numbers where market participants have access to the history of previous numbers, but of course not to the future. Note that this is neither the same as a single real number (i.e., an object of type `Real`) nor a function $\texttt{Time} \to \texttt{Real}$ (which would allow access to both the past and the

`return` : $a \to$ `Obs` a
For $x : a$, the observable `return` x : `Obs` a has the value x at any point in time.

$(\gg\!=)$: `Obs` $a \times (a \to$ `Obs` $b) \to$ `Obs` b
For an observable o : `Obs` a and a function $f : a \to$ `Obs` b, the value of the observable $o \gg\!= f$ at any point in time is obtained as follows: take the current value of o and call it x. x has type a. Then take the current value of $f\ x$: `Obs` b to obtain the final result of type b. $(\gg\!=)$ is called *bind*.

`now` : `Obs Time`
`now` always has as its value the current point in time.

`ever` : `Obs Bool` $\to$ `Obs Bool`
At any point in time, `ever` b is true if b is true or if it has ever been true since the beginning of time.

`always` : `Obs Bool` $\to$ `Obs Bool`
At any point in time, `always` b is true if b is true and it has always been true since the beginning of time.

Figure 1: Elementary operations on observables. a and b are arbitrary types.

future).

Consider Figure 1 for the elementary operations we introduce on observables together with their intended interpretations. The axioms for observables (found in Appendix B) aim to be minimal while formalizing these intended interpretations. They consist of the laws of the *monad* design pattern (Wadler, 1992) plus three additional axioms. The axioms for the temporal operators `now`, `ever`, and `always` consist of a translation of the modal logic *S4.3* (Blackburn et al., 2001, p. 189) plus additional axioms for `now`.

Lifts. We can *lift* operations on elementary types to observables. For example, we can lift the operator $(+)$: `Real` $\times$ `Real` $\to$ `Real` to

$$(+) : \quad \texttt{Obs Real} \times \texttt{Obs Real} \to \texttt{Obs Real}$$
$$o_1 + o_2 := o_1 \gg\!= \lambda x_1.\ o_2 \gg\!= \lambda x_2.\ \texttt{return}\ (x_1 + x_2).$$

The formal definition of $o_1 + o_2$ translates to English as follows: "the value of the observable $o_1 + o_2$ is at any point in time obtained by taking the current value of o_1 and calling it x_1, then taking the current value of o_2 and calling it x_2, then returning $x_1 + x_2$."

It is clear that the technique generalizes to functions of any type, so we can lift the operator $(<)$ to receive $(o_1 < o_2)$: `Obs Bool`, the observable that is `True` whenever the value of o_1 is below the value of o_2. We further receive lifts of boolean operators like $(\wedge)$, $(\neg)$ or implication $(\to)$ to `Obs Bool`.

Temporal Operators. Contracts may refer to not only the current value, but also to the history of an observable. For example, a contract might prescribe a payment in case an interest rate r : `Obs Real`$^+$ has been below 2% for at least 2 months. This condition can be expressed using the temporal operators `ever` and `always` as

$$\texttt{now} \gg\!= \lambda t.\ \texttt{always}\ ((\texttt{now} \geq t - 2\text{M}) \to r < 2\%).$$

In English, this means: "call the current point in time t. Then check if it has always been the case that at most 2 months before t, $r < 2\%$."

The transition from a to `Obs` a is similar to the method of temporalization (Finger and Gabbay, 1992), but the approach is different: while temporalization defines two distinct *layers* (an underlying logic and the temporal dimension), the theory of observables only defines a single class of types `Obs` a where the nesting order of `ever`, `always`, and $(\gg\!=)$ describes the times at which the values of observables are considered.

2.2 Building Contracts

A financial *contract* specifies the rights and obligations of its two counterparties, the *holder* and the *writer*. In the language of LPT, contracts are always described from the point of view of the holder. The situation of the writer is exactly reversed and is not modeled explicitly. The type of contracts is denoted by `Contract`.

Market participants *acquire*, i.e., enter into contracts. Acquisition is the only notion modeled in LPT because it is the most fundamental. For example, "buying" a contract x for price p can be expressed as acquiring the compound contract where the holder pays p and acquires x.

Consider Figure 2 for the elementary operations we introduce on contracts together with their intended interpretations. In comparison to (Peyton Jones and Eber, 2005), we add the powerful primitives $(\gg\!=)$ (for observables) and $(\rightsquigarrow)$ (for contracts), which in turn allows us to keep the remaining operations simple.

0 : `Contract`
The empty contract, stating no rights or obligations.

`one` : `Contract`
The contract `one` immediately pays a single unit of currency to the holder on acquisition.
For the sake of simplicity, we do not model different currencies, but the framework can be easily extended to allow for them.

$(+)$: `Contract` $\times$ `Contract` $\to$ `Contract`
Acquiring $x + y$ obliges the holder to immediately acquire both x and y. We also say that the holder acquires the *portfolio* consisting of x and y.

$(-)$: `Contract` $\to$ `Contract`
The contract $-x$ is like x, but with the writer and the holder sides interchanged. This means that the rights of x are obligations of $-x$ and all payments change signs. We write $x - y$ for $x + (-y)$.

$(\cdot)$: `Real`$^+$ $\times$ `Contract` $\to$ `Contract`
For α : `Real`$^+$, $\alpha \cdot x$ is like x where all payments are scaled by a factor of α.

$(\vee)$: `Contract` $\times$ `Contract` $\to$ `Contract`
A market participant acquiring $x \vee y$ must choose to acquire immediately one of x or y (but not both or none).

`when` : `Obs Bool` $\times$ `Contract` $\to$ `Contract`
Acquiring `when` b x obliges the holder to acquire x as soon as `ever` b becomes true. This means that if b has ever been true before, then x is acquired immediately and otherwise, x is acquired the first time b becomes true.
While this first time might not exist for pathological cases in continuous time, the considerations in this paper hold in any case.

`anytime` : `Contract` $\to$ `Contract`
The contract `anytime` x grants to the holder the right to acquire x at any time in the future, or never at all. This is called *exerting* the option. Typically, x is defined in such a way that it becomes worthless at a specified point in time, so that exertion is not reasonable indefinitely.

$(\rightsquigarrow)$: `Obs` $a \times (a \to$ `Contract`$) \to$ `Contract`
If a is a type, o : `Obs` a, and $f : a \to$ `Contract` is a function, then a market participant acquiring $o \rightsquigarrow f$ must immediately acquire $f\ \alpha$ where α is the value of o at the time of acquisition. $(\rightsquigarrow)$ can be seen as a `Contract`-variant of $(\gg\!=)$ and is called *contract bind*.

Figure 2: Elementary operations on contracts. Let x, y : `Contract`.

This is important to receive axioms of manageable complexity later. We also modify the interpretation of the `when` primitive, for the same reason.

Dynamic Payments. `Obs Real` embeds into `Contract` via the function

$$\begin{aligned}&\texttt{money} : \quad \texttt{Obs Real} \to \texttt{Contract}\\ &\texttt{money}\ o := o \rightsquigarrow \lambda\,\alpha.\ \texttt{if}\ (\alpha \geq 0)\\ &\qquad\qquad \texttt{then}\ (\alpha \cdot \texttt{one})\ \texttt{else}\ ((-\alpha) \cdot (-\texttt{one})).\end{aligned}$$

When a market participant acquires `money` o, the current value of the observable o, called α, is considered. If this value is positive, then the market participant receives an amount of α, otherwise she has to pay $-\alpha$.

Zero Coupon Bonds. Perhaps the simplest class of non-trivial contracts are so-called *zero-coupon bonds* (*ZCBs*): such a contract just grants the holder the right to receive a specified amount of money at a fixed time T. We define

$$\begin{aligned}&\texttt{zcb} : \quad \texttt{Real}^+ \times \texttt{Time} \to \texttt{Contract}\\ &\texttt{zcb}\ \alpha\ T := \texttt{when}\ (\texttt{now} = T)\ (\alpha \cdot \texttt{one})\end{aligned}$$

From now on, we just write α for $\alpha \cdot$ `one`.

European Options. For x : `Contract`, b : `Obs Bool`, and K : `Real`$^+$, we can define the *European call option* $\mathrm{C}_{x,K,b}$ with *underlying* x, *strike price* K, and *maturity* b. This contract gives the holder the right, but not the obligation, to buy the contract x for price K as soon as b becomes true. The *European put option* $\mathrm{P}_{x,K,b}$ gives the holder the right to sell x for K instead. Formally:

$$\begin{aligned}\mathrm{C}_{x,K,b} &:= \texttt{when}\ b\ ((x - K) \vee 0)\\ \mathrm{P}_{x,K,b} &:= \texttt{when}\ b\ ((K - x) \vee 0)\end{aligned}$$

Most of the finance literature would assume a fixed maturity, i.e., $b = (\texttt{now} = T)$ for some T : `Time`, and then go on to discuss different alternatives for the

underlying x separately, such as *stock options* (x is a stock price), *bond options* (x is a ZCB), or *compound options* (x is another option). LPT enables us to talk about all these financial instruments at a general level.

Conditional Acquisition. We define the helper function `cond` as follows: if b : `Obs Bool`, and x, y : `Contract`, then write

$$\texttt{cond}\ b\ x\ y := b \rightsquigarrow \lambda\beta.\ \texttt{if}\ \beta\ \texttt{then}\ x\ \texttt{else}\ y$$

for the contract that obliges the holder to acquire either x or y based on whether the condition b is true at the moment of acquisition. `cond` is an `if-then-else` construction for contracts.

Barrier Option. The following gives an example for a more exotic kind of option. Let o : `Obs Real`$^+$ and let K : `Real`$^+$ and b : `Obs Bool` like above. Let B : `Real`$^+$. The up-and-out *barrier option* is like a European call on $x :=$ `money` o, except for that the option becomes void if the underlying observable o touches the barrier B between acquisition and maturity. The barrier option can be expressed as

$$\begin{aligned}&\texttt{now} \rightsquigarrow \lambda t.\ \texttt{when}\ b\ \Big(\\ &\quad \texttt{cond}\ \big(\texttt{ever}\ (\texttt{now} \geq t \wedge o \geq B)\big)\\ &\qquad 0\ \big((x - K) \vee 0\big)\Big).\end{aligned}$$

3 The Axiomatic Description of the Arbitrage Relation

Having set up the language of observables and contracts, we now present our main contribution: the axiomatic description of the arbitrage relation. Introduce a new relational symbol $(\preceq_{\_})$ where for x, y : `Contract` and b : `Obs Bool` we write

$$x \preceq_b y$$

to mean that y is *preferable to x in arbitrage under conditions b*. This means that at any point in time where b is true there is a trading strategy by which a holder of $y - x$ arrives at a position where she does not run at a risk of losing money. We will see in Section 4 that this implies that whenever b holds, the price of x must below or equal to the price of y, if prices exist. The ordering $(\preceq_b)$ is *partial*: not any two contracts can be compared in general.

We define a few variants of the arbitrage relation. Let $\top$ = `return True` and $\bot$ = `return False`. For b, c : `Obs Bool` write $b \Rightarrow c$ to mean that $(b \to c) = \top$. Write

$x \approx_b y$ if $x \preceq_b y$ and $y \preceq_b x$. We say that x is *equivalent in arbitrage to y under conditions b*.

$x \prec_b y$ if $x \preceq_b y$ and for all c : `Obs Bool` with $c \Rightarrow b$ and $c \neq \bot$ we have $x \not\preceq_c y$. This means that y is preferable to x under b and no condition that is compatible with b can make them equivalent, i.e., b *guarantees* that they are not equivalent. We say that y is *strictly preferable in arbitrage to x under conditions b*.

$x \preceq y$ if $x \preceq_\top y$, i.e., y is preferable to x unconditionally. Define $(\approx)$ and $(\prec)$ equivalently.

Consider Figure 3 for the axioms we require for the arbitrage relation. The axioms can be seen as the fundamental arbitrage strategies. We describe some of them in detail.

Logic. Axioms 1 and 2 state that each relation $(\preceq_b)$ should be a partial order. Axioms 3–5 ensure that our notion of the b parameter as "under conditions b" is actually sensible: whatever holds under weak conditions also holds under stronger conditions (Axiom 3), we can perform case distinction on conditions (Axiom 4), and anything holds under conditions which are never true (Axiom 5).

Inverse to Portfolio Construction. Axiom 10 states that a portfolio consisting of both the holder and the writer positions in two copies of the same contract is equivalent to having no contract at all. The strategy to achieve 0 payout from $x - x$ is to reflect any choice a counterparty makes in the opposite contract and forward any incoming payments from one position to the other position.

Delay. To understand Axiom 25, consider a market participant who is holder in `when` $c\ y$ and writer in `when` $b\ x$. As time passes, one of the two events b or c will happen first and depending on that, the market participant would end up being either holder in x and writer in `when` $c\ y$ under conditions b (if b happens first) or holder in `when` $b\ x$ and writer in y under conditions c (if c happens first). The axiom requires that she can arrive at a risk-free position in both cases.

Equivalence of Currency and Numbers. Axiom 17 states a seemingly trivial property: receiving α dollars and β dollars is the same as receiving $\alpha + \beta$ dollars. Note however that the statement is *wrong* if one replaces `one` by a general contract x. That is because if x grants the holder a choice via $(\vee)$ or `anytime`, then the left-hand side of Axiom 17 would allow two independent choices while the right-hand side would

#	Group	Informal	Formal
1	Logic	Partial order	$x \preceq_b x$
2			If $x \preceq_b y \preceq_b z$, then $x \preceq_b z$
3		Conditions	If $b \Rightarrow c$ and $x \preceq_c y$, then $x \preceq_b y$
4			If $x \preceq_b y$ and $x \preceq_c y$, then $x \preceq_{b \vee c} y$
5			$x \preceq_\bot y$
6	Payment (`one`)	Positive value of currency	$\texttt{one} \succ 0$
7	Portfolio construction ($+$)	Associativity	$(x + y) + z \approx x + (y + z)$
8		Commutativity	$x + y \approx y + x$
9		Neutrality of 0	$x + 0 \approx x$
10		Inverse	$x - x \approx 0$
11		Monotonicity	If $x \preceq_b y$, then $x + z \preceq_b y + z$
12	Scaling ($\cdot$)	Scaling portfolios	$\alpha \cdot (x + y) \approx \alpha \cdot x + \alpha \cdot y$
13		Representation of $\texttt{Real}^+$-multiplication	$\alpha \cdot (\beta \cdot x) \approx (\alpha \cdot \beta) \cdot x$
14			$0 \cdot x \approx 0$
15			$1 \cdot x \approx x$
16		Monotonicity	If $x \preceq_b y$, then $\alpha \cdot x \preceq_b \alpha \cdot y$
17		Equivalence of currency and numbers	$\alpha \cdot \texttt{one} + \beta \cdot \texttt{one} \approx (\alpha + \beta) \cdot \texttt{one}$
19	Choice ($\vee$)	Reduction to x or y	$x \vee y \succeq x$ and $x \vee y \succeq y$
20		Minimality	If $z \succeq_b x$ and $z \succeq_b y$, then $z \succeq_b x \vee y$
21	Delay (`when`)	Compatibility with portfolio construction and scaling	$\texttt{when}\ b\ 0 \approx 0$
22			$\texttt{when}\ b\ (x + y) \approx \texttt{when}\ b\ x + \texttt{when}\ b\ y$
23			$\texttt{when}\ b\ (\alpha \cdot x) \approx \alpha \cdot \texttt{when}\ b\ x$
24		Reduction case	$\texttt{when}\ b\ x \approx_{\texttt{ever}\ b} x$
25		Non-reduction case	If $x \preceq_{d \wedge b} \texttt{when}\ c\ y$ and $\texttt{when}\ b\ x \preceq_{d \wedge c} y$, then $\texttt{when}\ b\ x \preceq_{d \wedge \neg \texttt{ever}\ b \wedge \neg \texttt{ever}\ c} \texttt{when}\ c\ y$
26	Time choice (`anytime`)	Reduction to exertion	$\texttt{anytime}\ x \succeq x$
27		Reduction to delayed choice	$\texttt{anytime}\ x \succeq \texttt{when}\ b\ (\texttt{anytime}\ x)$
28		Minimality	If $z \succeq_d x$ and $z \succeq_d \texttt{when}\ b\ z$ for any b, then $z \succeq_d \texttt{anytime}\ x$
29	Dynamics ($\rightsquigarrow$)	Binding `return`	$\texttt{return}\ \alpha \rightsquigarrow f \approx f\ \alpha$
30		Per-value analysis	If $f\ \alpha \preceq_{b \wedge o = \alpha \wedge p = \beta} g\ \beta$ for any α and β, then $o \rightsquigarrow f \preceq_b p \rightsquigarrow g$

Figure 3: Axioms for the arbitrage relation. Axioms are grouped conceptually.
Let $x, y, z : \texttt{Contract}$, $b, c : \texttt{Obs Bool}$, and $\alpha, \beta : \texttt{Real}^+$. Let $d : \texttt{Obs Bool}$ be of form $d = \texttt{ever}\ d'$.

allow only one, and having separate choices might be strictly preferable. In consequence, `Contract` does *not* in general form a `Real`-vector space with $(\cdot)$.

Choice and Time Choice. Axioms 19 and 20 characterize $x \vee y$ as the *join* of x and y: it is the unique minimal contract that is preferable to both x and y in arbitrage.

`anytime` x is characterized similarly as the minimal contract that allows exerting the option (Axiom 26) and postponing it until some future event (Axiom 27). Discarding the option is already contained in the axioms as postponing until $\bot$.

Discussion of Design Choices

We would like to point out some design choices. Axioms needed to be chosen strong enough such that one can show the statements that were seen to be true in prior work, yet they must not include any assumptions beyond arbitrage-freeness.

One challenge was the correct axiomatization of `when` in the non-reduction case (Axiom 25). An early candidate for an axiom was the "collapse rule"

$$\texttt{when}\ b\ (\texttt{when}\ c\ x) \approx \texttt{when}\ (\texttt{ever}\ b \wedge \texttt{ever}\ c)\ x.$$

This equation follows from the axioms and has a very elegant form, but it turned out to be too weak as an axiom: we need to take into account the exact timing as explained above. Isolating Axiom 25 as *the* core property of `when` is a result of many revisions of the framework.

Further attention has to be paid to the intricate details of the primitives `when` and `anytime`: for example, d in axioms 25 and 28 *must* have the property $d = \texttt{ever}\ d$: it is a condition that stays true over the lifetime of the contract if it is true at acquisition. At the same time, requiring this property for the b and c parameters in Axiom 25 would make this axiom too weak.

Having the condition b in $(\preceq_b)$ is not just a useful feature of the framework, but also crucially required to be able to argue about `when` and $(\rightsquigarrow)$ via axioms 24, 25, and 30. In Axiom 30, the condition $o = \alpha \wedge p = \beta$ allows proofs to make a case distinction over all possible values of o and p. The way the axiom is formulated automatically excludes pairs of values o and p cannot take: then the condition is $\bot$ and the precondition is trivially fulfilled.

Further important steps in the development of LPT were identifying the role of `Obs Bool` as conditions in the arbitrage relation, the axiomatization of `anytime` as the minimum of certain contracts and the introduction of the primitive $(\rightsquigarrow)$.

4 Prices

Contemporary market models and simulation methods are concerned primarily with the goal of computing the *fair price* of a contract while LPT focuses on the arbitrage relation and does not have any built-in notion of a price. Fortunately, users don't have to decide: LPT can be used to argue about prices as part of the theory.

We define the *price* of a contract as an equivalent in arbitrage amount of money:

Definition 4.1. If x : `Contract` and o : `Obs Real`, then o is called a *price for* x if

$$x \approx \texttt{money}\ o.$$

If b : `Obs Bool`, then o is called a *price for* x *under conditions* b if $x \approx_b \texttt{money}\ o$.

We do not claim that every contract has a price! In probabilistic models of LPT, many or all contracts have a price, while in the cash flow model (Section 6), very few do.

The following theorem constitutes a generalization of what is known as the *law of one price* in the finance literature. It states that prices of contracts, when they exist, must behave according to $(\preceq_\cdot)$.

Theorem 4.2 (Generalized Law of One Price)**.** *Let* b : `Obs Bool`. *Let* x, y : `Contract`. *Let* o, p : `Obs Real` *be prices for* x *and* y*, respectively, under conditions* b*. Then the following conditions are equivalent:*

1. $x \preceq_b y$.
2. $b \Rightarrow (o \leq p)$

The equivalence also holds for strict inequalities, i.e., if one replaces $(\preceq_b)$ *by* $(\prec_b)$ *and* $(\leq)$ *by* $(<)$.

The proof of the theorem can be found in Appendix C.

The theorem opens up an important application for LPT, namely computing bounds on prices: in a model of LPT, we might be interested in the fair price of a contract x, but it might be analytically difficult or computationally expensive to compute. If we find another contract y the price of which we know and we can show $x \preceq y$, then we know that the price of x can be at most the price of y. Depending on the application, this might be all we need.

As a special case, Theorem 4.2 implies the classical *law of one price*: if $x \approx y$, then $o = p$. That is,

equivalent contracts must trade for the same price. In particular, prices are unique if they exist.

A second theorem (omitted) shows that the operations 0, `one`, $(+)$, $(-)$, $(\vee)$, and $(\rightsquigarrow)$ on contracts translate into the operations `return` 0, `return` 1, $(+)$, $(-)$, max, and $(\gg\!=)$ on prices, respectively. Thus, the operations that introduce complexity in models are `when` and `anytime`.

5 Well-Known No-Arbitrage Relationships in LPT

We will now prove formally in LPT the two perhaps best known no-arbitrage relationships: the put-call parity for European options and Merton's theorem on the equivalence of European and American call options. By LPT's unique level of abstraction, we receive more general and stronger versions of these theorems than stated in prior work.

Traditionally, no-arbitrage relationships have been stated in terms of prices, but we can state them directly in terms of contracts. We know by Theorem 4.2 that statements about contracts carry to statements about prices for models that have them.

5.1 European Options, Put-Call Parity

Perhaps the best known no-arbitrage relationship is the put-call parity on European options. Recap from Section 2.2 that a European call $\mathrm{C}_{x,K,b}$ grants the holder the right to buy the underlying x for strike K at maturity b and a European put $\mathrm{P}_{x,K,b}$ grants the right to sell. The put-call parity states that if the underlying is a *dividend-free stock*, then the (price of the) portfolio consisting of a European put and its underlying equals the (price of the) portfolio consisting of a European call and a ZCB on the strike at maturity. We postpone the discussion of what exactly a "dividend-free stock" is supposed to be. Instead, LPT can help us derive a condition on x *equivalent* to the put-call parity: the following theorem shows that the put-call parity holds for a triple (x, K, b) iff `when` b $x \approx x$, i.e., iff a market participant is indifferent between acquiring x immediately or at maturity of the option.

Theorem 5.1 (Generalized Put-Call Parity)**.** *Let* x : `Contract`*,* b : `Obs Bool`*, and* K : $\texttt{Real}^+$*. The following are equivalent:*

1. $\mathrm{P}_{x,K,b} + x \approx \mathrm{C}_{x,K,b} + \texttt{when}\ b\ K$
2. $x \approx \texttt{when}\ b\ x$

Proof. (1) is equivalent to

$$\begin{aligned} x &\approx \mathrm{C}_{x,K,b} - \mathrm{P}_{x,K,b} + \texttt{when}\ b\ K \\ &= \texttt{when}\ b\ ((x-K) \vee 0) - \texttt{when}\ b\ ((K-x) \vee 0) \\ &\quad + \texttt{when}\ b\ K \\ &\approx \texttt{when}\ b\ (((x-K) \vee 0) - ((K-x) \vee 0) + K). \end{aligned}$$

The last line is by Axiom 22. It now suffices to show that the contract in parentheses on the last line is equivalent in arbitrage to x. The proof is given in Appendix D. □

In the literature, the put-call parity is typically considered for options with fixed maturities $b = (\texttt{now} = T)$, so the term *dividend-free stock* should refer to contracts where condition 2 is always fulfilled for these b:

Definition 5.2 (Dividend-Free Stock)**.** A contract x is called a *dividend-free stock* if $x \approx \texttt{when}\ (\texttt{now} = T)\ x$ for any T : `Time`.

The above condition does *not* state that there is no interest; it rather says that interest is already included in x. The following are examples for dividend-free stocks:

1. In a probabilistic model where contracts are compared based on their expected payout, we can consider an actual stock that does not include any payments to the owner (hence the name "dividend-free"), represented by its stock price o : $\texttt{Obs Real}^+$. In this case, the condition $x \approx \texttt{when}\ (\texttt{now} = T)\ x$ means that the expected growth of x is exactly the risk-free interest rate at which money can be borrowed or lent in the economy. This in turn means that o can be written as a *martingale process* plus a drift, which is indeed how stock prices are commonly modeled (Björk, 2004). It is thus no surprise that the put-call parity is best known for the case where the underlying is a stock.
2. A claim on an amount that grows deterministically (rather than in expectation) with the risk-free rate is a dividend-free stock.
3. If x is a ZCB, a European option with a fixed maturity, or any other contract that pays certain amounts of money (perhaps dependent on observables) at a point in time not earlier than T, then we receive the put-call parity for $b = (\texttt{now} = T)$. This consideration also yields a no-arbitrage relationship between financial products called *caps* and *floors* (essentially options on an interest rate) that is known as the *cap-floor parity*.

5.2 American Options, Merton's Theorem

We next turn to American options. These are like European options, but also allow exertion at any point *before* maturity. It is clear that American options are preferable to European options because they grant more rights to the holder. An at first sight surprising result that is sometimes attributed to Robert C. Merton states that under mild assumptions, they are in fact *equivalent* to European options, i.e., the right for early exertion does not give any advantage.

We first define some helper functions. For y : `Contract` and b : `Obs Bool` define

$$\texttt{american}\ b\ y := \texttt{anytime}\ (\texttt{cond}\ (\texttt{earlier}\ b)\ 0\ y)$$
$$\texttt{earlier}\ b := \texttt{now} \gg\!= \lambda t.\ \texttt{ever}\ (\texttt{now} < t \wedge b).$$

`american` b y is like `anytime` y, but exertion is only possible until and including the first occurrence of b. Now, for x : `Contract`, K : $\texttt{Real}^+$, and b : `Obs Bool`, define the *American call- and put options*

$$\mathrm{C}^{x,K,b} := \texttt{american}\ b\ ((x - K) \vee 0)$$
$$\mathrm{P}^{x,K,b} := \texttt{american}\ b\ ((K - x) \vee 0).$$

Fix an observable b : `Obs Bool` such that

$$\texttt{ever}\ b = \texttt{ever}\ (b \wedge \neg\texttt{earlier}\ b). \tag{1}$$

This means that there is always a "first" time b is true, and thus a "last" time where an American option can be exercised. The property is needed to exclude pathological cases if time is continuous.

Assume further *non-negative interest rates*. This means that receiving a fixed amount of money immediately is always preferable to receiving it later. In the language of LPT, this is expressed as `one` $\succeq$ `when` b `one` for all b : `Obs Bool` or, equivalently,

$$\texttt{one} \approx \texttt{anytime one}.$$

Under these assumptions, we receive a version of Merton's theorem for arbitrary contracts that satisfy the "$\preceq$" part of condition 2 from the put-call parity:

Theorem 5.3 (Generalized Merton's Theorem)**.** *Assume non-negative interest rates. Let* b : `Obs Bool` *satisfy equation* (1)*. Let* x : `Contract` *be such that* $x \preceq$ `when` $b\ x$*. Let* K : $\texttt{Real}^+$*. Then*

$$\mathrm{C}^{x,K,b} \preceq \mathrm{C}_{x,K,b}.$$

Proof Outline (full proof in Appendix D.2). Let $y = 0 \vee (x - K)$. We need to show that

$$\texttt{american}\ b\ y \preceq \texttt{when}\ b\ y. \tag{2}$$

We first show that $y \preceq$ `when` $b\ y$. This follows from the fact that $x \preceq$ `when` $b\ x$ by assumption and $-K \preceq -$`when` $b\ K$ by non-negative interest rates.

Now (2) follows because the early exertion right of `american` b does not give an advantage over `when` b because exertion at b is preferable anyways. Formally, we show that `american` $b\ y \preceq$ `american` b (`when` $b\ y$) $\approx$ `when` $b\ y$. □

Like the put-call parity, our version of Merton's theorem applies to the whole class of contracts LPT considers dividend-free stocks. Notice that for fixed maturities $b = (\texttt{now} = T)$, condition (1) is always fulfilled. Also recap from the previous section that the put-call parity is equivalent to $x \approx$ `when` $b\ x$. Thus, under non-negative interest rates and for b satisfying (1), the put-call parity implies Merton's theorem.

6 Models

LPT was designed with the goal of describing the relationships between contracts that hold in all arbitrage-free market models. In this section, we describe two such models of LPT.

6.1 Finite Tree Models

Finite tree models are probabilistic models where time is finite of form $\{1, \ldots, T\}$ and the set of possible states of nature at each point in time is also finite, so that the possible histories form a finite tree. Observables are implemented as stochastic processes on this tree.

`Contract` is implemented like `Obs Real`, i.e., every contract has a price and is in fact equal to its own price. The elementary operations correspond to computation of the expected payment from a contract. The operations 0, `one`, $(+)$, $(-)$, and $(\vee)$ are lifts of the corresponding operations on real numbers. `when` and `anytime` are defined recursively starting at the final time T. A contract y is considered preferable to a contract x under conditions b if its price is always greater or equal to the price of x in states of nature where b is `True`. Thus, this model describes *stochastic arbitrage*.

A special case is the CRR market model by Cox et al. (1979) where the tree is recombinant, which reduces the time needed to compute prices. Peyton Jones and Eber (2005) sketch a finite tree model as a possible implementation of their language, too.

The mere existence of a model implies:

Corollary 6.1. *LPT is consistent.*

6.2 The Cash Flow Model

The *cash flow model* is a nonstochastic model where a contract is identified with the set of cash flows it can generate. A contract y is only considered preferable to a contract x if choices in y can be made such that it generates the same or higher cash flows than x *at exactly the same points in time*. Consequently, this model only has the trivial prices (for contracts of form `money` o). Both finite and countably infinite time are supported.

In detail, we consider a (possibly infinite) tree of states of nature, but without an associated probability measure. Observables are implemented as processes on this tree. In contrast to the finite tree model, a contract is defined by a recursive structure more complex than an observable: every contract specifies a tree that represents choices to the holder and the writer about a) which payments are made at the time of acquisition and b) which contract is acquired in the following time step. The arbitrage relation is defined by comparing payments at *all* time steps. Thus, this model describes *deterministic arbitrage*. Care has to be taken to exclude *irrational choices* such as choosing 0 in the contract $0 \vee$ `one`.

A similar model was described by Gaillourdet (2011) for a language without the $(-)$ primitive, which greatly reduces expressiveness of the language and makes for a much simpler construction. Bahr et al. (2015) proposed another cash flow model that does however *not* exclude irrational choices. To the best of our knowledge, we are the first to describe a cash flow model that takes both aspects into account and that supports infinite time. A detailed description of the cash flow model can be found in Appendix E.

7 Conclusion and Outlook

We have introduced *LPT*, the *Logic Portfolio Theory* of arbitrage-free financial derivatives markets. LPT allows users to express a large class of financial derivative contracts using a small set of elementary operations. The axioms of LPT relate contracts by the *arbitrage relation*, i.e., under which conditions one contract cannot sell cheaper than another since that would give rise to an arbitrage opportunity. Our approach is novel in that financial contracts, rather than prices, are first-class objects, allowing us to reason about general financial contracts at the level of their defining elementary operations.

We have shown that LPT is powerful enough to capture well-known no-arbitrage relationships and extend them to more general forms, that it is consistent, and that it is compatible with a classical and a new model of financial markets.

Models of LPT display a rich collection of interconnected mathematical structures, including a boolean algebra with modal operations (`Obs Bool`, $(\Rightarrow)$, `ever`, `always`), a filtration of partial orders $(\preceq_.)$, a monad (`Obs` a), as well as interesting structure-preserving maps (`money`, elementary contract operations). Future work should explore the consequences of these interactions for the model theory of LPT.

We envision that LPT could be used as the basis of a software product to help investment banks better understand the relationships between their contractual commitments or to transform contracts into equivalent forms to speed up the computation of prices. LexiFi (2004–2014) have demonstrated with MLFi, a commercial product based on (Peyton Jones and Eber, 2005), that a language-centered approach can be extremely beneficial in this domain.

References

B. R. T. Arnold, A. v. Deursen, and M. Res. An algebraic specification of a language for describing financial products. In M. Wirsing, editor, *ICSE-17 Workshop on Formal Methods Application in Software Engineering*. IEEE, April 1995.

P. Bahr, J. Berthold, and M. Elsman. Certified symbolic management of financial multi-party contracts. In *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming*, New York, NY, USA, Sept. 2015. ACM.

S. Berthold. Towards a formal language for privacy options. In *Privacy and Identity Management for Life*. Springer, 2011.

S. Berthold. The privacy option language. Technical report, 2013.

T. Björk. *Arbitrage theory in continuous time*. Oxford university press, 2004.

P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, New York, NY, USA, 2001.

J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3), 1979.

T. Eggenschwiler and E. Gamma. Et++swapsmanager: Using object technology in the financial engineering domain. In *Conference Proceedings on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '92, New York, NY, USA, 1992. ACM.

M. Finger and D. M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 1(3):203–233, 1992.

S. Frankau, D. Spinellis, N. Nassuphis, and C. Burgard. Commercial uses: Going functional on exotic trades. *J. Funct. Program.*, 19(1), Jan. 2009. ISSN 0956-7968.

J.-M. Gaillourdet. A software language approach to derivative contracts in finance. In *(CM)ˆ2 Young Researchers Symposium 2011*. Center for Mathematical and Computational Modelling (CM)ˆ2, Februrary 2011.

C. Harrington, N. Dahl, P. Sestoft, and D. R. Christiansen. Pension reserve computations on gpus. In *Proceedings of the 3rd ACM SIGPLAN Workshop on Functional High-performance Computing*, FHPC '14, New York, NY, USA, 2014. ACM.

J. C. Hull. *Options, Futures and Other Derivatives*. Pearson/Prentice Hall, Boston, MA, USA, 8th edition, 2012.

T. Hvitved. *Contract Formalisation and Modular Implementation of Domain-Specific Languages*. PhD thesis, Department of Computer Science, University of Copenhagen, 2012.

M. Manzano. *Extensions of First-Order Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 1996.

LexiFi. Mlfi: the modeling language for finance. `https://www.lexifi.com/product/technology/contract-description-language`, 2004–2014.

S. Peyton Jones and J.-M. Eber. How to write a financial contract. In J. Gibbons and O. de Moor, editors, *The Fun of Programming*, Cornerstones of Computing. Palgrave Macmillan, 6 2005.

S. Peyton Jones, J.-M. Eber, and J. Seward. Composing contracts: An adventure in financial engineering (functional pearl). In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*, ICFP '00, New York, NY, USA, 2000. ACM.

H. R. Varian. The arbitrage principle in financial economics. *Journal of Economic Perspectives*, 1(2), 1987.

P. Wadler. Monads for functional programming. In *Marktoberdorf Summer School on Program Design Calculi*, volume 118 of *NATO ASI Series F: Computer and systems sciences*. Springer, 1992. Also in J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, Springer Verlag, LNCS 925, 1995.

Appendix

A A First-Order Version of LPT

LPT is presented as a theory with higher-order functions, but we can also consider a translation LPT_1 into first-order logic with countably infinitely many symbols and axioms. This is possible because the axioms of LPT (Section 3 and Appendix B) have a particularly simple structure: variables of functional type are only quantified over universally and only at the top level. (note that quantification is implicit in the presentation of the axioms)

Language The language of LPT_1 is defined by a recursive process as follows:

- Start with the symbols of LPT that have a first-order type. These are all symbols except for $(\gg\!=)$ and $(\rightsquigarrow)$.
- For any two types a and b and any functional term $f : a \to \mathtt{Obs}\ b$ that can be formed by the symbols defined so far, add a symbol $(\gg\!=_f) : \mathtt{Obs}\ a \to \mathtt{Obs}\ b$. Proceed analogously for $(\rightsquigarrow)$.
- Repeat indefinitely. The set of symbols of LPT_1 is the union of the ω-many stages.

A term in the language of LPT that contains an application of a higher-order function can now be translated to a first-order term in the language of LPT_1 by recursively following the same process, using the fact that terms have finite length.

Axioms The axioms of LPT_1 are obtained as follows:

- All axioms of LPT that do not include a variable of functional type are axioms of LPT_1.
- For each axiom with a functional variable f and each functional term $\tilde{f}$ in the language of LPT of appropriate type, an axiom of LPT_1 arises by replacing f by the translation of $\tilde{f}$ into the language of LPT_1 in the axiom.

  For example, Axiom 29 becomes

  $$\left(\rightsquigarrow_{\hat{f}}\right)\ (\mathtt{return}\ \alpha) \approx \hat{f_\alpha}$$

  where a is some type, $\tilde{f} : a \to \mathtt{Con}$ is a functional term of LPT, $\hat{f}$ is the translation of $\tilde{f}$ into LPT_1, and $\hat{f_\alpha}$ is the translation of the term $\tilde{f}$ where the parameter of $\tilde{f}$ has been replaced by α.

One checks that all statements and proofs from this paper only use either universal or concrete functions as arguments to higher-order functions. Hence, all these proofs can be carried out in LPT_1.

B Axioms for Observables

B.1 Monad laws; observable laws

The laws in this section describe the behavior of the $(\gg\!=)$ and `return` primitives. They ensure that e.g. the commutativity of $(+)$ on numbers implies the commutativity of the lift of $(+)$ to observables.

We first require the standard *Monad laws*.[3]

For $o : \mathtt{Obs}\ a$, $x : a$, $f : a \to \mathtt{Obs}\ b$ and $g : b \to \mathtt{Obs}\ c$ we require the following:

$$o \gg\!= \mathtt{return} = o \qquad : \mathtt{Obs}\ a \tag{Mo1}$$

$$\mathtt{return}\ x \gg\!= f = f\ x \qquad : \mathtt{Obs}\ b \tag{Mo2}$$

$$(o \gg\!= f) \gg\!= g = o \gg\!= (\lambda\, x.\ f\ x \gg\!= g) \qquad : \mathtt{Obs}\ c \tag{Mo3}$$

These are easily justified from the intuition of a "stream of values".

[3] The laws can be found e.g. in (Wadler, 1992).

We need additional axioms to ensure that `Obs` does not have side effects. These concern only lifts, so we define functions for unary and binary lifts:

$$\begin{aligned} \texttt{lift}_1 :&\quad (a \to b) \times \texttt{Obs}\ a \to \texttt{Obs}\ b \\ \texttt{lift}_1 go &:= o \gg\!= \lambda x.\ \texttt{return}\ (g\ x) \end{aligned}$$

$$\begin{aligned} \texttt{lift}_2 :&\quad (a \times b \to c) \times \texttt{Obs}\ a \times \texttt{Obs}\ b \to \texttt{Obs}\ c \\ \texttt{lift}_2\ h\ o_1\ o_2 &:= o_1 \gg\!= \lambda x_1.\ o_2 \gg\!= \lambda x_2.\ \texttt{return}\ (h\ x_1\ x_2) \end{aligned}$$

We now require the following:

$$\texttt{lift}_2\ h\ o_1\ o_2 = \texttt{lift}_2\ (\lambda x\ y.\ h\ y\ x)\ o_2\ o_1 \tag{Ob1}$$

$$\texttt{lift}_2\ h\ o\ o = \texttt{lift}_1(\lambda x.\ h\ x\ x)o \tag{Ob2}$$

$$\texttt{lift}_1(\lambda y.\ x)o = \texttt{return}\ x \tag{Ob3}$$

The first axiom states that the order of evaluation does not matter for observables. The second states that observables yield the same value each time they are read (at the same point in time). Finally, the third axiom states that observables the values of which are not used can be omitted.

One can see that this second set of axioms implies that observables can generally be reordered arbitrarily in chains of $(\gg\!=)$.

To see that axioms Ob1–Ob3 do not follow from the monad laws, note that all axioms fail for the *State* monad (carrying mutable state, cf. (Wadler, 1992)).

B.2 `ever` and `always`

The primitives `ever`, `always`, and `now` give to the theory of observables a notion of time.

For `ever` and `always`, we notice a similarity between expressions of type `Obs Bool` and modal logic if we write $\Box = \texttt{always}$, $\Diamond = \texttt{ever}$, and use the $(\Rightarrow)$ relation on `Obs Bool` for implication. A modal logic that is sometimes used for the "ever"/"always" relation is *S4.3* (cf. (Blackburn et al., 2001, p. 189 ff.)), the translation of which to `Obs Bool` is requiring that the following observables be equal to $\top$:

$$\texttt{always}\ (b \to c) \to (\texttt{always}\ b \to \texttt{always}\ c) \tag{K}$$

$$\texttt{ever}\ b \leftrightarrow \neg\texttt{always}\ \neg b \tag{Dual}$$

$$\texttt{always}\ \top \tag{Gen}$$

$$\texttt{ever}\ (\texttt{ever}\ b) \to \texttt{ever}\ b \tag{4}$$

$$b \to \texttt{ever}\ b \tag{T}$$

$$\texttt{ever}\ b \wedge \texttt{ever}\ c \to (\texttt{ever}\ (b \wedge \texttt{ever}\ c) \vee \texttt{ever}\ (\texttt{ever}\ b \wedge c)) \tag{.3}$$

Here, (K), (Dual) and (Gen) are exactly the axioms for a *normal* modal logic, i.e., one that is realized through a "many worlds" interpretation (*Kripke frames*), and axioms (4), (T), and (.3) axiomatize transitivity, reflexivity and linearity into the past[4] of the visibility relation "past states of the world", respectively.

B.3 `now`

The following axioms state that the `now` observable essentially reflects the `Time` type. They are easily verified by intuition. Let t : `Time` and b : `Obs Bool`. Then

$$\texttt{ever}\ (\texttt{now} = t) = \texttt{now} \geq t \tag{3}$$

$$\texttt{ever}\ (\texttt{now} = t \wedge b) \Rightarrow \texttt{always}\ (\texttt{now} = t \to b) \tag{4}$$

$$(\texttt{now} \gg\!= \lambda t.\ \texttt{always}\ (\texttt{now} \leq t)) = \top. \tag{5}$$

[4] (.3) does *not* state that the visibility relation be linear: any two *past* points in time must be related, but two *future* states may be unrelated. A model where the former holds, but the latter does not is where the states of the world are the nodes of a tree.

The "⇒" direction of Axiom (3) states that `now` is monotonically increasing. To see that intuitively, let t be some previous value of `now`. Then `ever` $(\texttt{now} = t)$ is true, hence the current value of `now` is $\geq t$. The other direction states that any previous point in time as of the `Time` type did actually exist.

Axiom (4) essentially states that `now` is *strictly* increasing in time: any value of `now` fixes all possible conditions of type `Obs Bool`: nothing may change while the value of `now` stays the same. Another point of view is that if time is discrete, then `now` must have the highest granularity.

(5) is essentially a monadic variant of (3). It has to be stated explicitly due to formal restrictions.

C Proof of Theorem 4.2

We first show that the statement holds for constant amounts of money.

Lemma C.1. *Let $\alpha, \beta : \texttt{Real}^+$. The following statements are equivalent:*

1. $\alpha \cdot \texttt{one} \preceq_b \beta \cdot \texttt{one}$ *for some* $b \neq \bot$.
2. $\alpha \cdot \texttt{one} \preceq \beta \cdot \texttt{one}$.
3. $\alpha \leq \beta$.

The equivalence also holds for strict inequalities.

Proof. $(2 \Rightarrow 1)$ is trivial.

$(3 \Rightarrow 2)$: There is nothing to show for $\alpha = \beta$, so assume $\alpha < \beta$, i.e., $\beta - \alpha > 0$. As $((\beta - \alpha)\cdot)$ preserves "$\prec$" and $\texttt{one} \succ 0$ by Axiom 6, $(\beta - \alpha) \cdot \texttt{one} \succ (\beta - \alpha) \cdot 0 \approx 0$. Now

$$\begin{aligned} \alpha \cdot \texttt{one} &\approx \alpha \cdot \texttt{one} + 0 \\ &\prec \alpha \cdot \texttt{one} + (\beta - \alpha) \cdot \texttt{one} \\ &\approx (\alpha + \beta - \alpha) \cdot \texttt{one} \\ &= \beta \cdot \texttt{one} \end{aligned}$$

where the second relation is because $((\alpha \cdot \texttt{one})+)$ is an isomorphism $0 \prec (\beta - \alpha) \cdot \texttt{one}$ and the third relation is due to Axiom 17.

$(1 \Rightarrow 3)$: If $\alpha \not\leq \beta$, i.e., $\beta < \alpha$, then by $(3 \Rightarrow 2)$, $\beta \cdot \texttt{one} \prec \alpha \cdot \texttt{one}$. In particular, by definition of "$\prec$", $\alpha \cdot \texttt{one} \not\preceq_b \beta \cdot \texttt{one}$.

If $\alpha \not< \beta$, i.e., $\beta \leq \alpha$, then again by $(3 \Rightarrow 2)$, $\beta \cdot \texttt{one} \preceq \alpha \cdot \texttt{one}$. In particular, $\beta \cdot \texttt{one} \preceq_b \alpha \cdot \texttt{one}$, so $\alpha \cdot \texttt{one} \not\prec_b \beta \cdot \texttt{one}$. □

Corollary C.2. *Lemma C.1 still holds if one allows $\alpha, \beta : \texttt{Real}$ instead of only $\texttt{Real}^+$ where*

$$\alpha \cdot \texttt{one} := (-\alpha) \cdot (-\texttt{one}) = -((-\alpha) \cdot \texttt{one}) \quad \text{if } \alpha < 0.$$

Proof. The proof is a simple case distinction on the signs of α and β. The only interesting remaining case is $\alpha < 0 \leq \beta$ (or its symmetric variants). Then

$$\alpha \cdot \texttt{one} = -((-\alpha) \cdot \texttt{one}) \prec 0 \preceq \beta \cdot \texttt{one}$$

as required. □

Now we can generalize the result of corollary C.2 to observables. The following statement is equivalent to Theorem 4.2.

Proposition C.3 (Law of One Price, Generalized Version)**.** Let $o, p : \texttt{Obs Real}$. Then the following conditions are equivalent for any $b : \texttt{Obs Bool}$:

1. $\texttt{money}\ o \preceq_b \texttt{money}\ p$.
2. $b \Rightarrow (o \leq p)$

The equivalence also holds for strict inequalities, i.e., if one replaces $(\preceq_b)$ by $(\prec_b)$ and $(\leq)$ by $(<)$.

Proof. First consider the "$\leq$" variant:
we have

$$\begin{aligned} & \texttt{money}\ o \preceq_b \texttt{money}\ p \\ \Leftrightarrow\ & \forall \alpha, \beta : \alpha \cdot \texttt{one} \preceq_{b \wedge o = \alpha \wedge p = \beta} \beta \cdot \texttt{one}. \\ \Leftrightarrow\ & \forall \alpha, \beta : ((b \wedge o = \alpha \wedge p = \beta) = \bot)\ \text{or}\ \alpha \leq \beta \\ \Leftrightarrow\ & \forall \alpha, \beta : ((o, p) = (\alpha, \beta)) \Rightarrow (b \to \alpha \leq \beta) \end{aligned}$$

where the first equivalence is a consequence of Axiom 30, the second is due to lemma C.1 and the third is just transformation of boolean observables. Again via transformation of observables, one notices that the last line is equivalent to

$$\begin{aligned} \top &= ((o, p) \gg\!= \lambda\, (\alpha, \beta).\ b \to \alpha \leq \beta) \\ &= b \to o \leq p. \end{aligned}$$

Now show the "$<$" variant:
By the $(\leq)$ part, $d := (b \wedge o \geq p)$ is the unique $(\Rightarrow)$-maximal boolean observable such that $d \Rightarrow b$ and $\texttt{money}\ o \succeq_d \texttt{money}\ p$. So by definition of $(\prec_b)$ we have

$$\texttt{money}\ o \prec_b \texttt{money}\ p \ \Leftrightarrow\ d = \bot \ \Leftrightarrow\ b \Rightarrow o < p. \qquad \square$$

D Proofs from Section 5

D.1 Proof of Theorem 5.1

We start with a small lemma:

Lemma D.1. *For any $x, y, z : \texttt{Contract}$ we have*

$$x + (y \vee z) \ \approx\ (x + y) \vee (x + z).$$

Proof. We show that the LHS satisfies the universal property of $(x + y) \vee (x + z)$: we have $x + (y \vee z) \succeq x + y$ since $y \vee z \succeq y$ and by monotonicity of $(+)$ (Axiom 11). Likewise for z.
For minimality, assume that $w \succeq (x + y), (x + z)$. Then also $w - x \succeq y, z$ and so $w - x \succeq y \vee z$ and $w \succeq x + (y \vee z)$. $\square$

Note that the lemma does *not* hold if one exchanges $(\vee)$ and $(+)$.
We can now show the last remaining equivalence in the proof of Theorem 5.1.

Lemma D.2. *For any $x : \texttt{Contract}$ we have*

$$(((x - K) \vee 0) - ((K - x) \vee 0) + K) \ \approx\ x.$$

Note that the statement would be an simple case distinction if x and K were numbers and $(\vee)$ was taking the maximum. To show it for LPT's contracts, a few transformations are required.

Proof. By Lemma D.1, it follows that

$$\begin{aligned} (x - K) \vee 0 &\approx (x \vee K) - K \\ (K - x) \vee 0 &\approx (K \vee x) - x \end{aligned}$$

and so the above expression is equivalent in arbitrage to

$$\begin{aligned} & ((x \vee K) - K) - ((K \vee x) - x) + K \\ \approx\ & (x \vee K) - K - (K \vee x) + x + K \\ \approx\ & x. \end{aligned}$$

$\square$

D.2 Proof of Theorem 5.3

For $y : \texttt{Contract}$ and $b : \texttt{Obs Bool}$ let

$$y_b := \texttt{cond}\ (\texttt{earlier}\ b)\ 0\ y,$$

so $\texttt{american}\ b\ y = \texttt{anytime}\ y_b$.

Lemma D.3. *Let* $y : \texttt{Contract}$ *and* $b : \texttt{Obs Bool}$ *fulfilling equation* (1). *Then*

$$\texttt{when}\ b\ y_b \approx_{\neg\texttt{earlier}\ b} \texttt{when}\ b\ y.$$

Proof. Define $\texttt{first}\ b := b \wedge \neg\texttt{earlier}\ b$. As b fulfills equation (1), we have $\texttt{ever}\ b = \texttt{ever}\ (\texttt{first}\ b)$.

The statement is then equivalent to

$$\texttt{when}\ (\texttt{first}\ b)\ y_b \approx_{\neg\texttt{earlier}\ (\texttt{first}\ b)} \texttt{when}\ (\texttt{first}\ b)\ y.$$

To see this for the condition, note that $\texttt{earlier}\ b = \texttt{earlier}\ (\texttt{ever}\ b) = \texttt{earlier}\ (\texttt{ever}\ (\texttt{first}\ b)) = \texttt{earlier}\ (\texttt{first}\ b)$.

By monotonicity of `when` (which follows easily from the axioms), it now suffices to show

$$y_b \approx_{\texttt{first}\ b} y,$$

which is clear by definition of y_b since $\texttt{first}\ b \Rightarrow \neg\texttt{earlier}\ b$. □

Lemma D.4. *Let* $y : \texttt{Contract}$ *be such that* $0 \preceq y \preceq \texttt{when}\ b\ y$. *Then*

$$\texttt{american}\ b\ y \preceq \texttt{when}\ b\ y.$$

Proof. Perform case distinction on $\top = \texttt{earlier}\ b \vee \neg\texttt{earlier}\ b$:

For $\texttt{earlier}\ b$, notice that $y \succeq 0$ and hence also $\texttt{when}\ b\ y \succeq \texttt{when}\ b\ 0 \approx 0$. On the other hand, $y_b \approx_{\texttt{earlier}\ b} 0$ and hence also $\texttt{american}\ b\ y = \texttt{anytime}\ y_b \approx_{\texttt{earlier}\ b} 0$ (via monotonicity. We have $\texttt{earlier}\ b = \texttt{ever}\ (\texttt{earlier}\ b)$.)

So consider $\neg\texttt{earlier}\ b$. By lemma D.3 and definition of `american` it suffices to show that

$$\texttt{anytime}\ y_b \preceq \texttt{when}\ b\ y_b.$$

To that end, we use minimality of `anytime` (Axiom 28). One has to show the following:

1. $\texttt{when}\ b\ y_b \succeq y_b$
2. $\texttt{when}\ b\ y_b \succeq \texttt{when}\ c\ (\texttt{when}\ b\ y_b)$ for all $c : \texttt{Obs Bool}$.

2 follows by collapsing of `when`.

1 follows from the assumption as follows: Do case distinction on $\top = \texttt{ever}\ b \vee \neg\texttt{earlier}\ b$. For $\texttt{ever}\ b$, the statement is trivial, so consider $\neg\texttt{earlier}\ b$.

Since we have $y \succeq 0$, also $y \succeq y_b$ (cf. definition of y_b) and so

$$y_b \preceq y \preceq \texttt{when}\ b\ y \approx_{\neg\texttt{earlier}\ b} \texttt{when}\ b\ y_b$$

where the second relation is by assumption and the third is lemma D.3. □

Proof of Theorem 5.3. Apply lemma D.4 to $y = 0 \vee (x - K)$. It only remains to show that the preconditions to the lemma are satisfied, then the desired statement follows directly.

Non-negative interest rates imply that $\texttt{when}\ b\ \texttt{one} \preceq \texttt{one}$. Hence

$$x - K \preceq \texttt{when}\ b\ x - \texttt{when}\ b\ K \approx \texttt{when}\ b\ (x - K).$$

By $0 \approx \texttt{when}\ b\ 0$ one then receives

$$0 \vee (x - K) \preceq \texttt{when}\ b\ 0 \vee \texttt{when}\ b\ (x - K) \preceq \texttt{when}\ b\ (0 \vee (x - K))$$

where the last relation follows from the universal property of $(\vee)$ and the fact that $\texttt{when}\ b$ is a monotonic map (which follows easily from the axioms for `when`). □

E The cash flow model

We want a build an LPT model that has "as few as possible" arbitrage relations. To achieve this, we want to model a contract as a system of cashflows. Two contracts should only be considered equal if their cashflows are the same at any time, up to irrational choices.

For example, in this notion, we should receive

- $\texttt{one} + \texttt{when}\ (\texttt{now} = T)\ \texttt{one} \succ \texttt{one}$ because the LHS generates an additional (positive) cashflow and otherwise is equal to the RHS.
- $\texttt{one} \vee 0 \approx \texttt{one}$ because while the LHS allows for one cashflow more, no rational trader would take the 0 choice, and the two are otherwise equal.
- $\texttt{one}$ and $\texttt{when}\ (\texttt{now} = T)\ 1\texttt{one}$ are incomparable under conditions weaker than $\texttt{now} = T$ because the generated cashflows happen at different points in time. This is a different way of saying that this model does not have any interest rates.

E.1 Elementary types and observables

We use the standard model for the numeric types and model $\texttt{Time} = \mathbb{N}$.

We take the simplest model of observables. Let `St` be any set, the set of "states of the world". Let `Hist` be the set of finite `St`-sequences and let `FullHist` be the set of infinite `St`-sequences. Then model

$$\texttt{Obs}\ a := \texttt{Hist} \to a,$$

i.e., an observable of type a is a function from `Hist` to a. Note how the current point in time is implicit in the length of the history.

It should be easy to extend this very simple model of observables to stochastic processes if desired.

E.2 Boolean trees

For any set a let `AOTree` a be the set of trees of finite height, but with possibly infinitely many children per node, where the leaves are elements of a and the inner nodes are labeled with either "$\vee$" or "$\wedge$". An `AOTree` a is the syntax tree of an abstract boolean expression in a which uses only "$\vee$" and "$\wedge$".

The elements of `AOTree` a are called *choice trees* where a "$\vee$" branching means that the holder has a choice and a "$\wedge$" branching means that the writer has a choice.

E.3 Contracts

Now define `Contract` as the following recursive data type:

$$\texttt{Contract} := \texttt{Obs}\ (\texttt{AOTree}\ (\mathbb{R}, \texttt{Contract}))$$

At any point in time, when acquired, a contract yields a choice structure the elements of which are pairs (α, x') where α is a payment (seen from the point of view of the holder, where positive payments are incoming), and x' is a subsequent contract, to be acquired at the next time step.

Note that the definition is not very succinct: The `Contract` for the next time step (acquired at time ≥ 1) contains again a description for time step 0, which would not be required. This will not be a problem below.

Define the primitive building blocks for contracts as follows. We write "$\vee$" and "$\wedge$" for the syntax elements of `AOTree`. We write an additional pair of parentheses for the single-element tree. Some lifted operations on `AOTree`s are defined below.

$$
\begin{aligned}
0 &:= h \mapsto ((0,0)) \\
\texttt{one} &:= h \mapsto ((1,0)) \\
x + y &:= h \mapsto x(h) + y(h) \\
-x &:= h \mapsto -x(h) \\
\gamma \cdot x &:= h \mapsto \gamma \cdot x(h) \\
x \vee y &:= h \mapsto x(h) \vee y(h) \\
\texttt{when}\ b\ x &:= h \mapsto \begin{cases} x(h) & \text{if } (\texttt{ever}\ b)(h) = \texttt{True} \\ ((0, \texttt{when}\ b\ x)) & \text{else} \end{cases} \\
\texttt{anytime}\ x &:= h \mapsto x(h) \vee ((0, \texttt{anytime}\ x)) \\
o \rightsquigarrow f &:= o \gg\!= f
\end{aligned}
$$

Define operations on `AOTree` $(\mathbb{R}, \texttt{Contract})$ recursively as follows. Note that any recursion has to hit a primitive case (i.e. a leaf) at some point as `AOTree`s have finite height. We only define the operations for two children a and b here, but the definitions easily generalize.

$a + b$:

$$
\begin{aligned}
(a \vee b) + c &:= (a + c) \vee (b + c) \\
(a \wedge b) + c &:= (a + c) \wedge (b + c)
\end{aligned}
$$

Also add the two other versions symmetrically. One has to show that the result does not depend on the ordering of decomposition, which is true up to "$\approx$" as defined below.

The primitive case:

$$
(\alpha, x') + (\beta, y') := (\alpha + \beta, x' + y')
$$

$-a$:

$$
\begin{aligned}
-(a \vee b) &:= (-a) \wedge (-b) \\
-(a \wedge b) &:= (-a) \vee (-b) \\
-(\alpha, x') &:= (-\alpha, -x')
\end{aligned}
$$

Note how "$\vee$" and "$\wedge$" are flipped in the two compound cases. This is why we need `AOTree` instead of just a set.

$\gamma \cdot a$: Nothing interesting here:

$$
\begin{aligned}
\gamma \cdot (a \vee b) &:= (\gamma \cdot a \vee \gamma \cdot b) \\
\gamma \cdot (a \wedge b) &:= (\gamma \cdot a \wedge \gamma \cdot b) \\
\gamma \cdot (\alpha, x') &:= (\gamma \cdot \alpha, \gamma \cdot x')
\end{aligned}
$$

Note that the definitions of the contract primitives are all recursive again and should hence be taken as compact descriptions of certain infinite sequences/trees. Note that it suffices that we can compute any *prefix* in finite time to make the entire sequence well-defined.

E.4 The arbitrage relation

Define:

$$x \preceq_b y \;:\Leftrightarrow\; \forall h \in \texttt{Hist}:$$
$$\text{if } b(h) = \texttt{True}, \text{ then } x(h) \leq^h y(h)$$

Here, $x(h)$ and $y(h)$ are of form $\texttt{AOTree}\,(\mathbb{R}, \texttt{Contract})$ and the "$\leq^h$" relation is to be defined now. "$a \leq^h b$" should express that for any reasonable choice pattern of the holder and the writer, now and in the future, the cashflow produced by b will always be at least as favorable to the holder (comparing at each point in time separately) as that of a, assuming history h at acquisition. This also encodes the acquisition time.

Following the structure of `AOTree`, one defines recursively:

$$\begin{aligned} a \vee b \leq^h c &\quad :\Leftrightarrow a \leq^h c \text{ and } b \leq^h c \\ a \wedge b \leq^h c &\quad :\Leftrightarrow a \leq^h c \text{ or } \quad b \leq^h c \\ c \leq^h a \vee b &:\Leftrightarrow c \leq^h a \text{ or } \quad c \leq^h b \\ c \leq^h a \wedge b &:\Leftrightarrow c \leq^h a \text{ and } c \leq^h b \end{aligned}$$

These rules are easily justified intuitively. For example, to check $c \leq^h a \vee b$, one only has to check if there is *some* choice by which the holder can be better than c, whereas in $c \leq^h a \wedge b$, we have to consider *all* possible choices of the writer. Again, one needs to check that the order of deconstruction of the compound expressions does not matter, which is easily doable.

For $h \in \texttt{Hist}$ and $s \in \texttt{St}$ define hs to be the finite sequence consisting of h with s appended at the end. Now we would like to give the definition of the base case recursively by

$$(\alpha, x') \leq^h (\beta, y') :\Leftrightarrow \alpha \leq \beta \text{ and } \forall s \in \texttt{St} : x'(hs) \leq^{hs} y'(hs). \tag{6}$$

But this is not a definition! Intuitively, this recursive definition makes sense: We have to transform the choice structure into logical relations and then check any two combinations of base cases by first checking the payments (cash flows at that time) and performing a recursive check on the subsequent contracts and any possible following state of the world. In finite time, this procedure terminates at some point and we're done, but in infinite time, it is clear that the above really didn't define anything: Consider e.g. the simple check $0 \leq^h 0$ where the 0 contract is defined as above as an endless stream of 0s. In order to decide the base case, we'd have to check whether $0 \leq 0$ (which is true) and then whether $0 \leq^{hs} 0$ for any s. — But that's exactly what we want to know, just one time index later!

E.5 Fixing the arbitrage relation

The intuitive argument why $0 \approx 0$ should hold is that we will never hit a payment (in finite time) where they are not equal. So let's make this notion explicit!

Define for $u \in \mathbb{N}$ a new relation "$\leq_u^h$" that is defined like "$\leq^h$", except for when $|h| \geq u$: Then $a \leq_u^h b$ is always true for any a and b. This is a well-defined relation as the recursion will always reach history length u at some point and terminate.

Now re-define

$$a \leq^h b :\Leftrightarrow \forall u \in \mathbb{N} : a \leq_u^h b.$$

It is easy to check that this definition indeed fulfills equation (6). So we found an "implementation" of what can be understood as a recursive "specification".

An alternative point of view is that "$a \leq^h b$" in fact defines an abstract *machine* that receives a $H \in \texttt{FullHist}$ as an infinite input such that h is a prefix of H. In each step u, the machine checks whether we can refute the claim that "$a \leq^h b$" with the information available up to time u. If so, it enters the `FAILED` state (terminates) and otherwise, it remains in the `ACTIVE` state. We say that $a \leq^h b$ if the machine never terminates on any input, i.e. the claim cannot be refuted.

E.6 Checking the axioms

Once the machinery of the model is set up, showing the axioms is surprisingly simple: to show that an arbitrage relation holds, all one has to do is consider finite cases, i.e. one in a sense only has to do an induction step. All the proofs of the axioms in the model are just formalizations of the intuitive justifications that we gave in the paper.

For $\texttt{one} \succ 0$, one has to provide a counterexample to "$\texttt{one} \leq^h 0$", which is obviously given by the first time step.