LexiFi®

Structuring, Pricing, and Processing

Complex Financial Products with MLFi®

A language approach transcends prevalent methods for modeling sophisticated financial products

White Paper

Banks serving institutional and corporate clients often compete on their ability to provide high value-added solutions. The design of such solutions drives financial innovation and frequently translates into complex financial products. The challenge for wholesale banks is to deliver tailored solutions within the window of opportunity, while containing costs and risks.

Complex financial products present two significant business challenges:

Communication

Banks and their customers lack a concise, precise, and complete description standard for complex products. The main tools for communicating product information between market participants are natural language (e.g., English) descriptions embodied in term sheets and confirmations, which are supported by auxiliary documentation such as master agreements and ISDA* definitions.

Before a transaction is executed, the main written communication medium between parties is a term sheet, which defines the salient features of the proposed deal. Term sheets tend to be concise because of timing constraints: a complete and accurate English-language description of a complex derivative contract is typically ten to twenty pages long, if not more, takes days to write, and requires the joint contribution of finance and legal professionals. To paraphrase Shakespeare in Hamlet, “there are more things” in a complex financial product than can be described in a short English-language text. Term sheets are therefore concise but incomplete.

Even post-transaction legal documents, such as confirmations, can be incomplete. Market participants often rely on common assumptions and shared knowledge, neither of which is explicitly documented, to structure and eventually agree to a transaction. Such intangible assumptions might prove difficult to defend in court. Inside the bank, ambiguous contract specifications can be a major source of conflicts between marketers and traders. Moreover, incomplete product specifications force operations personnel to repeatedly query the front office to understand the behavior of executed transactions.

While the present communication system might be acceptable for standard products, the lack of a concise, precise, and complete description mechanism for complex products exposes banks and their customers to significant legal and operational risks.

LexiFi believes that a formal product description tool is needed to improve communication between parties and to help “debug” legally binding natural language documents.

(*) International Swaps and Derivatives Association.

Process Automation

Competitive intensity in capital markets often decreases as the volume of transactions in a new class of financial instrument exceeds a certain level. Beyond that threshold, banks that rely on manual processes generally face rising costs and operational risks. The group within the bank that initiated the innovation is often small—it sometimes comprises a single trader—and manages the trading process with the help of spreadsheets. As volumes rise above a few dozen transactions, additional manpower does not guarantee that customer inquiries are handled in a timely fashion, errors are avoided, and market and credit risks are properly measured. Consequently, banks risk losing business precisely at the point where competition recedes and margins become more attractive. Capital markets frequently exhibit such cases of “winner takes all” dynamics that allow a small group of leaders to capture a disproportionate share of volumes and profits. In order to compete in such markets, banks need to commodify new products as quickly as possible.

LexiFi believes that process automation is challenging to achieve for complex products mainly because existing systems tightly couple contract definitions and processes—e.g., pricing, operational management. The introduction of a new instrument requires that all processes be rewritten for this instrument. Conversely, a new process requires adaptations for each instrument, one by one.

This last point is quite problematic since the life span of data (i.e., contract definitions) often exceeds that of applications that retrieve and process this data: we can suppose that contract definitions created today will outlive many front office and back office applications. One requirement is therefore to periodically add or replace applications without altering contract data, regardless of its complexity.

Decoupling contract specifications and processes can be achieved by defining a shared financial product description standard that guarantees contract and process independence.

MLFi Concepts

Shared Contract Description Standard

LexiFi has developed the Modeling Language for Finance (MLFi®), a precise formalism that exhaustively describes capital market, credit, and investment products with a limited set of core constructs. Contracts are specified exactly with a library of about twenty basic combinators or “primitives.”

The theoretical foundation for the MLFi language results from a joint effort over many years between LexiFi and leading researchers in programming languages.

MLFi contracts may incorporate complex option clauses based on any type and combination of underlyings—e.g., interest rate, equity, foreign exchange, credit. Contracts may be defined initially without any notion of credit risk, and later extended to incorporate credit risk information.

MLFi is implemented as an extension of a strongly typed programming language that checks types, date consistency, and other elements during compilation. This means that errors in financial product design are detected at the earliest possible stage.

Semantics

The main technical difference between FpML, pricing-orientated payoff languages, and MLFi lies in their semantics—or lack thereof.

FpML

While FpML brings structure in descriptions of financial contracts, as opposed to a "flat" collection of (name, value) pairs, FpML lacks semantics: the "meaning(s)" of a contract cannot be determined by simply looking at an FpML product definition. One needs an associated text such as ISDA definitions to fully understand an FpML contract specification and to be able to specify its behavior in the context of valuation or operational management.

Object-oriented models that implement FpML attempt to bridge the semantics gap but have two inherent limitations: they are very difficult to design and, most importantly, the resulting model is not compositional (see below).

FpML describes contract types and lists the parameters needed to describe each contract category. FpML therefore obeys a standards logic that relies on a menu of contracts. While the menu attempts to be exhaustive, it ultimately cannot be.

Because it is not compositional, FpML focuses on interbank contracts and does not support complex, client-driven transactions.

Payoff Languages

Payoff languages define a product's meaning in the context of pricing. For example, in a Monte-Carlo pricing context, a product is a program—a "machine"—that takes a simulated market scenario as input and returns the list of cash flows induced by the scenario, on the correct dates. Products are mapped from inception into the world of valuation algorithms: cash flows, options, and other potential events are expressed in a way that is understandable by the stochastic machinery used to value financial contracts. This mapping implies a loss of information: for example, a payoff language will not distinguish between a cash-settled option and a physically-settled option, or between an option that is exercised automatically if it expires in-the-money and one that is exercised only at the discretion of one party.

A contract described with a payoff language is a *program* that can only do one thing—be executed—in order to describe the contract's behavior in the single context of pricing.

MLFi

MLFi describes what contracts "are" independently from what they "do," and the resulting specification is used to derive pricing, simulation, back office management, and processing capabilities, with the guarantee that the behavior across these environments is consistent.

MLFi precisely and exhaustively describes financial products, independently from the processes that may be applied to them: contract definitions represent "data" that numerous programs can analyze and translate into equivalent definitions. The original product representation is preserved and remains independent from process-specific translations.

The "data" is specified with about twenty basic combinators or "primitives" and processes are described by their action on each primitive. Consequently, programs

that manipulate data are relatively easy to write and adapt mechanically to any contract.

MLFi effectively conveys the full meaning of a contract because the language is *compositional*. What this means is that it suffices to understand a small set of core constructs in order to design and process sophisticated products. Analogously, in algebra, the expression "(1 + 2) x (3 + 4)" is easy to understand because we know what a number is and we understand the meaning of algebraic operators. MLFi does exactly the same thing with financial contracts, with elementary MLFi contracts replacing numbers and MLFi combinators replacing algebraic operators. Expressions that contain elementary contracts and combinators are also contracts, which may in turn be further combined with other contracts to create increasingly higher-level building blocks for designing financial products.

To summarize, a contract described with MLFi is an expression (a *value*) that can be analyzed and manipulated in many ways to convey different meanings: MLFi has potentially multiple, compositional semantics.

The table below summarizes the characteristics of the three approaches for representing financial contracts.

	FpML	**Payoff Languages**	**MLFi**
Contract Description	Structured list of parameters.	Program.	Expression.
Number of constructors	Potentially infinite, as new constructors are needed to define new products.	Finite or potentially infinite, depending on the implementation.	Finite list of primitives.
Data separated from processes	Yes. Describes what contracts "are" independently from what they "do." What contracts "do" cannot be easily derived from what they "are."	No. Describe what contracts "do" in a pricing context. Contract description is a program that can only be executed.	Yes. Describes what contracts "are" independently from what they "do." Contract description is a value that can be analyzed and manipulated in many ways. What contracts "do" is automatically derived from what they "are."
Semantics	**No** semantics.	**Single** valuation (denotational) semantics.	**Many** possible denotational semantics (e.g., valuation, future cash flows) and operational semantics (e.g., event processing). Programs that manipulate contract definitions are compositional.
Analogy with algebra This line describes the possible meaning(s) that may be given to algebraic expression "(1 + X) x (3 + Y)" under the different approaches.	Meaning cannot be determined independently by simply looking at a parametric definition of the algebraic expression.	Find the value of X and Y, using a model, to calculate the value of the expression.	Denotational semantics: Find the value of X and Y, using a model, to calculate the value of the expression. Calculate the number of symbols, the sum of even numbers, etc. Operational semantics: If we know that the value of X is 5 (in the world of

			financial contracts this could represent the value of a fixing), record the fixing information as part of the expression and apply simplification rules. The expression becomes "6 x (3 + Y)".

Business Considerations

FpML's reliance on network effects may delay its adoption

MLFi delivers intrinsic value whereas FpML relies on network effects to create value. MLFi may be used by individual business units or product areas as a stand-alone tool to design, price, and process products. In contrast, FpML is primarily a communication standard and brings relatively limited value to a single group: users derive value from FpML when different groups within a single bank or, preferably, when several banks use the standard to communicate with each other. FpML's reliance on network externalities to deliver value may delay its adoption.

Payoff languages create operational risk

In the past decade, many derivatives desks have developed payoff languages to price new products, sometimes with little consideration for the downstream impact of executing complex transactions. As a result, complex contracts are invariably redefined in several systems and/or processed manually. Multiple contract representations are an important source of operational risk.

MLFi offers a technical solution to apply sound practices in the structured product business. MLFi contract descriptions lend themselves to many uses and can be shared among applications: MLFi preserves the freedom to innovate and provides control over the entire structured product value chain.

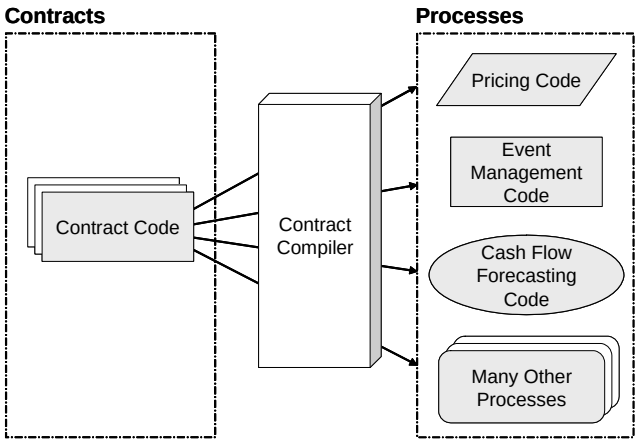
Contract and Process Independence

MLFi’s precise contract descriptions lend themselves to automated processing of various sorts. Processes are described by how they manipulate MLFi’s twenty basic combinators. As contracts are specified with the same twenty primitives, and the action of processes on each elementary combinator is known, processes adapt to any contract.

As we have seen in the table above, the ability for a single program to manipulate all data elements, without exception, has a parallel in algebra: in the same way one can apply a process to any algebraic expression—e.g., calculate the value, count the number of symbols—MLFi can apply a process to any contract.

MLFi’s semantics support an array of processes including pricing, event management, and cash flow forecasting, for all contract types, regardless of their complexity.

POSSIBLE MLFI SEMANTICS



Pricing

Users may link MLFi contract definitions with in-house valuation libraries, or rely on standard algorithms delivered by LexiFi, to calculate the value and risk parameters of a contract or portfolio.

Contract valuation proceeds in three steps:

- *Abstract Valuation Semantics*. An MLFi contract is first translated into an abstract value process, together with a handful of operations over these processes, which correspond directly to the mathematical and stochastic machinery used to value financial contracts. They include, for example, the process of assigning the value to an observable at maturity in order to calculate the payoff of a contract in each state of the world, or the process of working backward in time and discounting. A value process is independent of the financial model or the choice of discrete numerical method.

- *Model-Specific Optimization and Specialization*. Once a contract is expressed in the intermediate language of abstract value processes, meaning-preserving optimizing transformations may be applied before computing the value for the process.

  This approach is reminiscent of the way in which a compiler is typically structured. The program is first translated into a low-level but machine-independent intermediate language. Many optimizations are applied at this level. Then the program is further translated into the instruction set for the desired processor—e.g., Pentium®, Sparc®.

  Pricing code also needs to be refined in order to exploit the performance gains afforded by optimized algorithms and closed forms. MLFi introduces information about closed forms in model definitions and uses that information during compilation: MLFi is therefore able to recognize contracts that should be priced with an optimized algorithm or a closed form. A compilation option gives the choice of either using the optimized algorithm or closed form, or calling the full succession of default numerical calculation steps. This feature facilitates (i) the debugging of numerical and closed-form implementations and (ii) the concurrent deployment of simple models that admit a closed form, typically used for risk management, and more sophisticated models that do not admit a closed form, used for pricing.

- *Concrete Implementation*. In order for a computer to perform calculations, stochastic processes are implemented with a financial model and a discrete numerical method—typically, finite difference, lattice, or Monte Carlo.

  MLFi accelerates the task of linking a contract to an internally developed or commercially available pricing library. Valuing a contract amounts to calling low-level elements of a pricing library in the right order. For example, if one selects a lattice-based numerical method for valuing a contract, primitives of the pricing library may include a function that fills the maturity vector with the value of the payoff, and a discounting function that works back through the lattice, one step at a time, and fills the vector for the immediately preceding step. The specification of how a contract should drive a sequence of numerical calculation steps is traditionally hand-coded for each contract type. MLFi automates that process: once linked, a model implementation can be applied to an entire family of compatible structures. A structure is incompatible when it cannot be adequately valued by the model: for example, a single-currency Heath-Jarrow-Morton model cannot properly value a foreign exchange contract, and this will be detected at compile time. The MLFi compiler generates specialized pricing code, with the correct sequence of numerical calculation steps for each contract, that is then compiled and run natively to perform the valuation.

Contract Event Management

A Formal Operational Processing Model

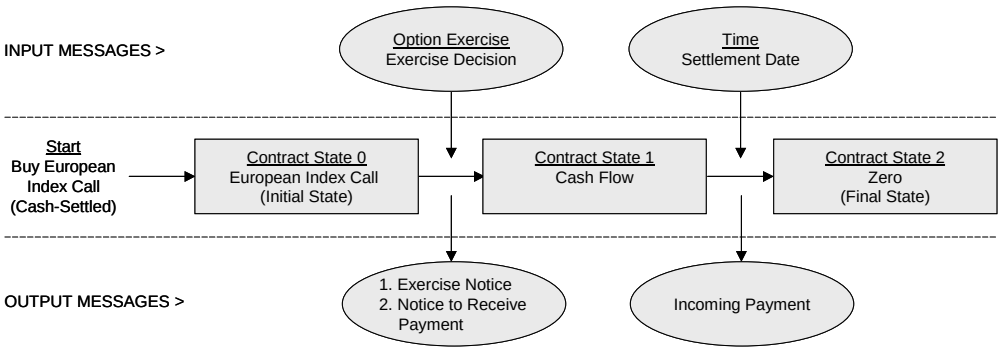
MLFi's core operational processing component is the contract Event Manager, which consistently processes events for all MLFi contracts, including:

- *Time-Related Events*. The passage of time may trigger payments such as coupons, dividends and redemptions, and other events that the Event Manager can handle automatically. The Event Manager may position the clock in the future or in the past.
- *Fixings*. Floating rate resets and other price or index observations are performed on the appropriate date. Contract specifications may include complex formulas of observable variables, including path-dependent formulas, for calculating quantities such as interest amounts, redemption amounts, or option payoffs.
- *Option Exercises*. The Event Manager provides tools for monitoring exercise opportunities and for processing exercise decisions.
- *Barrier Crossings*. The Event Manager records barrier crossings and processes relevant contracts. MLFi allows for very flexible definitions of "in" or "out" barriers. For example, the user may describe a one-year equity basket option that remains in existence as long as 6-month USD-LIBOR stays within a certain range during the first six months of the contract's life.

The Event Manager is designed in the spirit of a finite state system. A "finite automaton" consists of a finite set of internal configurations or "states" and a set of transitions from state to state that occur on input symbols chosen from an alphabet. Similarly, a contract starts its life in an initial state, and migrates from state to state upon the occurrence of messages chosen from an "alphabet" which represents the various types of events—time-related, fixings, option exercises, barrier crossings. For each type of message, there is exactly one transition out of each state. For example, if a physically-settled option is exercised (one type of message) the contract is transformed into a new contract that characterizes the underlying asset being delivered, whereas if the option is not exercised (another type of message) the contract is transformed into a "zero" contract, which carries

no obligation, in a final state. Contracts are effectively managed through event-controlled transformations. The Event Manager also optionally produces one or more output messages after each transition. In the previous example, the output message might have been an option exercise notice. The following transition diagram illustrates the successive definitions of a simple cash-settled equity index option contract (the option premium payment is omitted) as it is being processed.

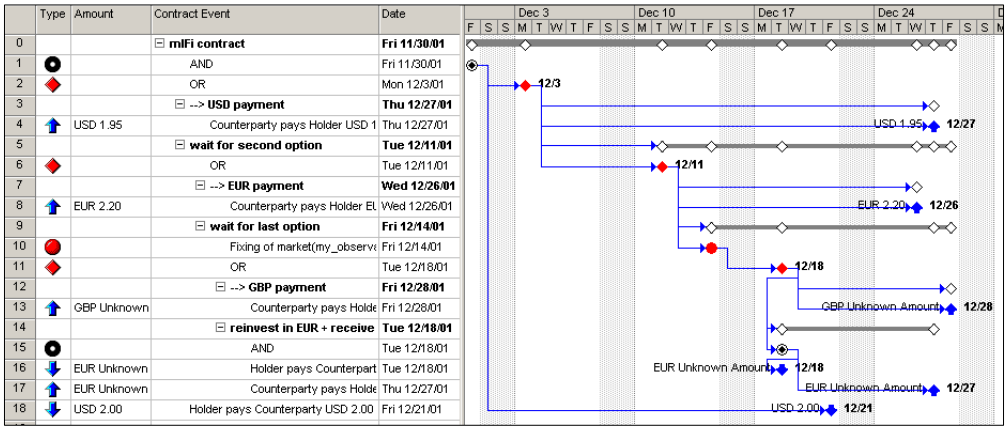
EVENT MANAGER TRANSITION DIAGRAM



Contract Execution Calendar

MLFi not only processes single events and transitions, but also provides tools to automatically visualize the potential proliferation of transitions and states out of a given state. The chart below illustrates possible sequences of states for a more complex contract. Note that transitions are labeled to facilitate contract tracking.

CONTRACT EXECUTION CALENDAR

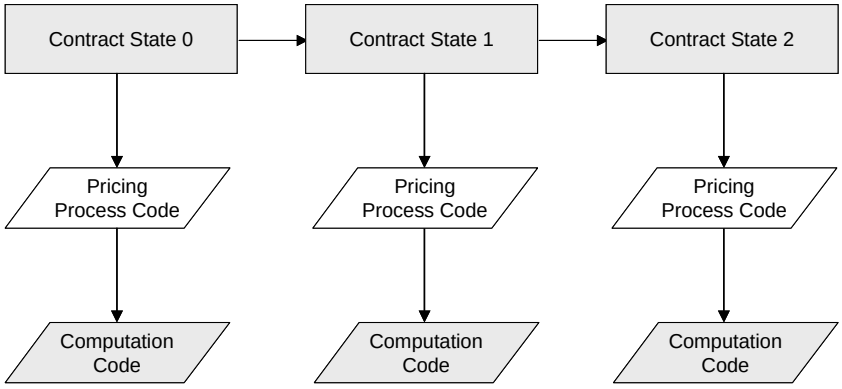


	Type	Amount	Contract Event	Date
0			**mlFi contract**	**Fri 11/30/01**
1			AND	Fri 11/30/01
2			OR	Mon 12/3/01
3			**--> USD payment**	**Thu 12/27/01**
4		USD 1.95	Counterparty pays Holder USD 1	Thu 12/27/01
5			**wait for second option**	**Tue 12/11/01**
6			OR	Tue 12/11/01
7			**--> EUR payment**	**Wed 12/26/01**
8		EUR 2.20	Counterparty pays Holder EU	Wed 12/26/01
9			**wait for last option**	**Fri 12/14/01**
10			Fixing of market(my_observe	Fri 12/14/01
11			OR	Tue 12/18/01
12			**--> GBP payment**	**Fri 12/28/01**
13		GBP Unknown	Counterparty pays Holde	Fri 12/28/01
14			**reinvest in EUR + receive**	**Tue 12/18/01**
15			AND	Tue 12/18/01
16		EUR Unknown	Holder pays Counterpart	Tue 12/18/01
17		EUR Unknown	Counterparty pays Holde	Thu 12/27/01
18		USD 2.00	Holder pays Counterparty USD 2.00	Fri 12/21/01

Contract State and Pricing Synchronization

MLFi's operational model guarantees that pricing code always mirrors the state of each contract. The combination of a formal operational model and pricing capabilities provides a unified framework for valuing complex products as they evolve in their life cycle (the state of a partly executed contract is still a contract, and as such can be used to drive a valuation engine), for simulating the value of a contract over a set of future dates, and for calculating value at risk and potential credit exposures on a portfolio of exotic products.

CONTRACT STATE AND PRICING SYNCHRONIZATION



The fact that LexiFi correctly reflects the evolving structure of financial contracts has two additional benefits:

- *Optimal Pricing*. MLFi updates pricing code as contracts transition from state to state. This provides an opportunity to use the most efficient pricing algorithm, including closed-form solutions when they exist, in each contract state. Consider the case of a Himalaya contract that pays the average performance on a twelve-stock basket. Each quarter the best performing stock is used to contribute to the average performance and then removed from the basket. Himalaya structures are often forced into existing front office or back office systems: a typical workaround consists in setting the weight of the best performing stock at the end of each quarter to zero. The consequence in terms of pricing is that the initial twelve-dimension Monte-Carlo simulation will be applied until the contract matures in three years. In contrast, MLFi would generate pricing code that accurately reflects the decreasing dimension of the valuation problem.
- *Correct Handling of Time-Related Events*. The impact of time-related events such as coupon payments on the behavior of a contract must be carefully reflected in traditional pricing algorithms. MLFi eliminates this important source of programming errors by generating valuation code that exactly mirrors the state of each residual contract.

THE IMPORTANCE OF VERSATILE COMBINATORS

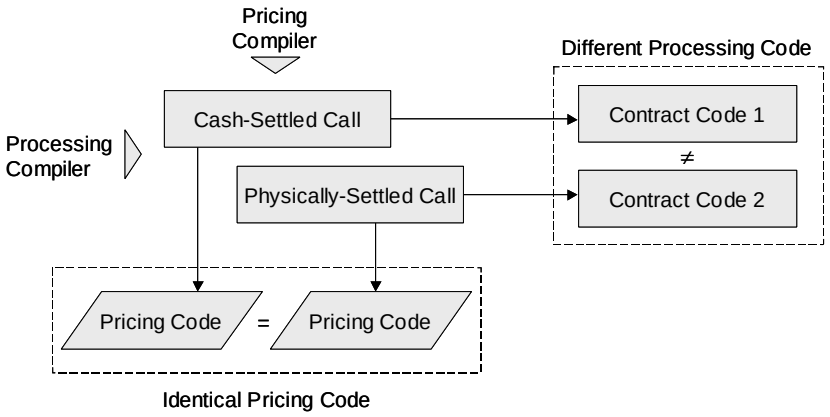
The same contract definition may drive a variety of processes, including valuation, event management, and cash flow forecasting.

However, the semantics are quite different: pricing and cash flow semantics map a contract into another domain while operational semantics gives rules about how a contract is mapped into another contract. Contract language combinators must therefore be carefully chosen in order to drive multiple processes.

The example below illustrates common limitations of contract languages designed primarily for pricing. Pricing-orientated languages often model a contractual choice with the **`max`** operator. The reason is that designers of pricing-orientated languages are often influenced by the underlying mathematics of valuation that express option payoffs with the **`max`** operator. However, the **`max`** operator, while adequate for pricing, does not correctly

describe the management process. For example, cash-settled options and physically-delivered options are generally priced with the same algorithm, but have different operational management procedures: upon exercise, the holder of a cash-settled option receives a cash payoff, whereas the holder of a physically-delivered call option receives either the underlying asset or nothing. **`max`** can describe the cash payoff, but not the physical settlement. MLFi substitutes the **`or`** combinator for the **`max`** combinator in order to correctly depict both valuation and management processes. Another limitation of pricing-orientated languages is that they generally do not keep a full history of events. In contrast, MLFi incorporates event-related information as uncertainty resolves.

IDENTICAL PRICING BUT DIFFERENT PROCESSING



Cash Flow Forecasting

Financial contracts often induce uncertain cash flows that depend on the realization of "observables"—quantities, such as 6-month USD-LIBOR, that are part of a contract's definition and whose value changes over time—and/or the option exercise strategy of the contract's holder.

Approximate values of future cash flows are required to support a number of processes, including risk measurement, where they are used to simplify the calculation of VaR and the implementation of stress-testing programs on large portfolios. For example, scenario analysis aims at identifying the financial impact of low probability but nevertheless plausible events that may not be captured by a statistical model. Scenario analysis may be implemented by making simplifying assumptions about the evolution of future market conditions—i.e., each scenario gives the potential value of any observable at any moment—and about exercise strategies—e.g., deep in-the-money options are assumed to be exercised, and far out-of-the money options are not.

MLFi may be used to implement a generic procedure that is applied to the contract language and that treats a portfolio as one large MLFi contract. In a first pass, a simplified model approximates the distribution of observable values, and option exercise probabilities. This allows, in a second pass, to reduce the portfolio to a set of (cash flow, probability) pairs and to obtain a clear, although approximate, vision of the portfolio's cash flow dispersion through time.

Integration

LexiFi provides several mechanisms for integrating MLFi-based applications with third-party systems:

- *Instantaneous SQL Database Integration*. LexiFi provides a simple, schema-independent way of persisting MLFi contracts in a relational database. This feature is designed to:
  - immediately automate operational management and reporting for new products, regardless of their complexity;
  - open MLFi to a broad range of users such as business users who rely on graphical query and reporting tools, and SQL programmers.

  MLFi contract definitions and their life cycle history are represented in a binary object that may be stored in a single table column.

  MLFi's contract manipulation capabilities are exposed as user-defined functions (UDFs), also called stored procedures, which enable users to query and modify any contract or portfolio using standard SQL syntax. LexiFi provides pre-packaged UDFs to handle common tasks such as adding a contract to a database, managing contract events, and reporting on a portfolio. A simple interface allows users to create UDFs for all functions exposed in MLFi's Finance library. In addition, access to the abstract syntax tree of MLFi contract definitions enables power users to develop custom functions for querying contracts.
- *XML Messages*. The Event Manager communicates via XML-formatted output messages. An external workflow engine may drive the Event Manager by generating the input messages it expects, and processing the output messages it returns.
- *Extensible Language*. MLFi is a full-fledged programming language that includes financial contract description features. MLFi is built as an extension of Caml, a strongly typed programming language developed by INRIA, The French National Institute for Research in Computer Science and Control. MLFi treats contracts as any other datatype, and, because it is a full-fledged language, may be extended natively or linked with external programs written in a programming language such as C or Java. For example, MLFi may invoke an external routine that monitors market data in order to alert traders as soon as a rate or price approaches a contractual barrier by less than a user-specified interval.
- *Integration Module for Microsoft.* LexiFi provides a simple and robust way of exploiting the full power of the MLFi language from Microsoft products, including Microsoft® Office, Microsoft Visual Basic®, and applications written using Microsoft Visual Studio®. By allowing MLFi code to be transparently embedded in Microsoft Office documents and applications, the user can integrate MLFi-generated tools with existing resources such as real-time and historical market data sources, pricing algorithms, and legacy systems, as well as exploit the convenience and ubiquity of the Microsoft platform for the development of new applications, which are partially or fully based on MLFi technology. LexiFi provides code samples and Microsoft Excel templates to accelerate the development of custom solutions.
- *C# Event Management API.* The C# API enables applications written in C# to drive MLFi's contract management system. All event processing tasks on a portfolio of contracts written in MLFi may be automatically performed by external applications.

- *Choice of Output*. Unlike traditional compilers that always translate source code into executable code, the MLFi compiler may translate MLFi code into a variety of outputs, including:
  - modified MLFi code—e.g., the Event Manager processes an event, and translates the initial contract into a new, simpler, contract;
  - source code in another programming language—e.g., (i) as MLFi product definitions contain information on the parameters needed to instantiate them, the user may dynamically generate input screens for each product type using the language of his choice; (ii) likewise, MLFi may output different flavors of valuation code to sequence the generation of prototype and production pricing code.
  - text file—e.g., (i) link the XML-formatted result of a calculation script with Microsoft Excel 2003 to create informative and consistent marketing documents or (ii) create vCalendar-formatted contract execution calendars to integrate with calendaring and scheduling applications.
- *Simple and Readable IT Model*. Transactions and their life cycle history are represented in a single document that applications may manipulate through a clear, well-documented API. There is no proprietary data or object model, with dozens of tables or UML diagrams, associated with MLFi.
- *Any PC or UNIX platform*. The MLFi Compiler runs under virtually any PC or Unix platform.

MLFi Applications

MLFi is a building block for developing a variety of high value-added transaction processing, decision support, and business integration solutions, including:

Structuring and Pricing of Complex Derivatives

Providing the right tools at the right time to salespeople and structurers requires a robust product assembly, pricing, and analysis infrastructure. MLFi leverages in-house valuation libraries to facilitate the development of structuring applications that the marketing team can use.

MLFi enables you to:

- *Accelerate the Development of Pricing Routines for New Products*. Shortening windows of opportunity are forcing banks to speed up the development of new valuation routines. However, existing modeling environments do not always facilitate pricing code reuse, and sometimes require new code even for minor product variations. This situation slows down the model development and deployment process, and creates a maintenance burden on the quantitative analysis team who needs to maintain and document dozens of programs that may differ only slightly. MLFi may serve as a universal contract description language that generates pricing code able to drive internally developed or commercially available pricing libraries. MLFi can drive both prototyping and production valuation libraries. For example, quantitative analysts may initially want to generate prototype code in response to a client inquiry, and benefit from the visualizations capabilities of modeling environments such as MATLAB®. After the transaction is consummated, MLFi may generate production code in C in order to integrate the new product in the trading system. The MLFi compiler's ability to output different flavors of valuation code eases the programming and maintenance workload of financial engineers, and allows them to spend more time on value-added tasks.
- *Develop Fast Portfolio Analytics and Pricing Grids*. Taking advantage of contract similarities and closed forms is key to speeding up portfolio analytics.

The MLFi Compiler treats a portfolio as one large contract for valuation purposes and identifies shared contract clauses to avoid duplicate calculations. In addition, MLFi detects contracts that should be priced with an optimal algorithm or admit a closed form for a given pricing model. This combination of features accelerates portfolio calculations and the generation of pricing grids for single trades or multi-legged trading strategies.

- *Place Structuring Tools in the Hands of Marketers*. Revenue growth in a structured products business depends in great part on how much time the sales force spends with customers. Unfortunately, structuring tools often lack elementary usability features and, as a result, remain the preserve of financial engineering "gurus." There is a requirement to build structuring tools that the potential user base can use. MLFi can form the basis of a graphical assembly tool that enables marketers to define products with increasingly higher-level components, and benefit from MLFi's pricing and operational semantics. MLFi constructs may be linked to in-house valuation models for performing pre-trade risk analyses. MLFi's operational semantics may also help to demonstrate to customers the potential behavior of a contract in the future. MLFi's event planning capabilities share educational qualities with theoretical option payoff diagrams in that they provide qualitative insights into the possible behavior of financial contracts.
- *Develop a Comprehensive Trading Strategy Simulator*. The structuring tool may be extended to include a full-featured trading strategy simulator. The goal of the simulator is to identify optimal trading strategies under a set of business, regulatory, tax, and other constraints. A trading strategy is applied mechanically for a large number of market scenarios. The simulator associates each state of the world with a portfolio and compares the portfolio's metrics with a user-specified set of constraints. Trading strategies would be defined compositionally with a dedicated strategy language that LexiFi may help develop. Armed with the ability to rapidly find optimal solutions to difficult problems, banks would deliver significant value to customers.

Operational Management of Complex Derivatives

The maintenance of complex derivatives contracts is rarely automated. Consequently, even a limited volume of structured transactions can have a significant downstream impact. Manual processing of complex contracts increases operating expenses and operational risk, and, in the end, prompts controllers to limit the flow of business.

MLFi may be used to develop operational management applications that enhance an existing processing infrastructure.

MLFi enables you to:

- *Systematically Detect and Process Events in a Portfolio of Complex Derivatives*. When dealing with complex products, the back office most frequently lacks tools to systematically detect and process events, and to automatically produce confirmations and other messages sent in response to those events. MLFi consistently processes time-related events, fixings, option exercises and barrier crossings for all contracts, and optionally generates messages whose content may be used to replace keywords in user-formatted templates. Events and messages may be driven by an external workflow engine and integrated into other applications. Process automation reduces operating expenses and operational risk.
- *Improve Communication Throughout the Firm*. The back office, risk management, and other departments within the bank sometimes have

difficulties in understanding the products that the front office is trading. MLFi may be used to educate operational and functional units about the definition and behavior of complex contracts. MLFi descriptions establish one consistent "definition of record" for each contract that the various departments may learn to read, but not necessarily to write, either by looking directly at the MLFi code or with the help of visualization tools: for example, a contract execution calendar (See "MLFi Concepts—Contract and Process Independence—Contract Event Management") helps understand the potential proliferation of contract transitions and states. Better transparency among stakeholders eliminates bottlenecks, reduces operational risk, and ultimately expands derivatives trading capacity.

- *Develop Self-Service, Customer-Orientated Applications for Complex Products*. Customers will increasingly demand to monitor and process executed transactions online as they do not always have the technical resources to implement systems that are sufficiently agile to keep pace with financial innovation. MLFi may be used to develop information delivery applications that provide visibility into the contract's life cycle, and processing capabilities—e.g., customers notify option exercise decisions online—both of which contribute to increase customer retention, reduce costs, and limit the potential for errors.

Product Repository

Data models and object models aiming to encapsulate the definitions and behavior of financial instruments were developed by banks and vendors in recent years in order to support a variety of departmental and firm-wide processes.

LexiFi believes that many of these initiatives did not meet expectations in terms of functionality, budget, or implementation time because the tools were not adapted. Data models are not compositional and are therefore inherently ill-suited to represent complex financial contracts. Even though major object-oriented languages are not compositional, they are potentially able to mimic the features of the contract sub-language included in MLFi. The problem with object-oriented languages lies in the implementation of pattern matching, a powerful mechanism that increases programmer productivity and program safety and performance. In addition, financial product data models and object models add a significant management burden on IT teams as they impose the maintenance of dozens of tables or UML diagrams. Instead, a precise and exhaustive contract description language lets finance professionals freely compose products and share contract definitions among people, processes, and systems.

MLFi enables you to:

- *Shield Applications from the Complexity of Financial Products*. A contract language may potentially be used for describing a variety of product-related data elements and processes. However, a language approach mainly adds value for dynamically adding new payoffs without modifying interfaces to pricing and risk management applications, and for automating event management. While a language may be used to describe market data, static data, and processes such as the generation of audit trails, it is probably advisable to "hard-code" those elements to prevent the contract language from becoming overly complex.
- *Create an Integrated Environment for Managing Derivatives*. A contract language and pricing model libraries may jointly form a fundamental system's layer for an entire derivatives business. This common layer would support many applications, even though the underlying businesses might be quite different: for example, an equity derivatives business may span a variety of

activities that operate normally quite independently, such as the electronic market-making of exchange-traded options (ETOs), pricing of vanilla OTC options, and structuring and pricing of tailored products. New integration requirements are emerging: for example, the ETO desk may need to communicate volatility surfaces to the structured products desk, and at the same time want to extend its own hedging algorithms to include vanilla or semi-exotic OTC products. MLFi may serve as an infrastructure component that helps create visibility and fluidity throughout a business.

- *Exploit Derivatives Market's "Winner Takes All" Dynamics*. We mentioned in the introduction that competitive intensity in derivatives markets often decreased as the volume of transactions in a new product exceeds a certain level, and a small group of leaders captures a disproportionate share of volumes and profits. MLFi allows banks to quickly create and commodify products in order to exploit the "winner takes all" dynamics of the derivatives market.

Enterprise Application Integration (EAI)

The goal of EAI is to efficiently exchange complex information through entire business processes across multiple and constantly evolving systems and platforms.

MLFi makes it easier for financial institutions to create connections between their various internal systems and to coordinate the transactions and processes that span those systems. MLFi enables interactions between such diverse systems as structuring, trading, back office, documentation processing, risk management and e-business applications, databases, and data warehouses.

MLFi enables you to:

- *Establish a Conformed Standard for Communicating between Systems, Employees, Customers, and Counterparties*. MLFi may serve as a pivot format that preserves the meaning and integrity of contracts during their entire life cycle. MLFi describes a contract's initial terms and conditions, and incorporates event-related information as uncertainty resolves, reflecting the passage of time and/or the occurrence of fixings, option exercises, and barrier crossings. Moreover, MLFi's reliance on elementary combinators for describing financial contracts enables the development of an infrastructure devoid of product restrictions.

- *Leverage Best-of-Breed Software and Develop Cutting-Edge Applications*. Front office, risk management, and back office applications do not evolve at the same pace. MLFi enables the development and deployment of loosely-coupled applications. MLFi promises to satisfy the combined goal of providing optimal trading tools for each group, and supporting consistent pricing, risk management and operational processing across systems.

- *Extend the Life of Legacy Systems*. Legacy systems developed over the years often have good features that are well aligned with the needs of the activities they support. However, other systems and departments are not always able to access their data. The definition of a common contract exchange standard may facilitate their integration with other systems and, as a result, give them a new life.

- *Expand Trading Capacity*. As explained above (See "MLFi Applications—Operational Management of Complex Derivatives") MLFi-based operational management tools may help mitigate the operational risk that constrains derivatives trading capacity. However, front office and back office systems for complex contracts do not function in isolation. This means that the operational

risk stemming from inter-application communications must also be addressed. For example, one possible scenario would be to extract contract definitions stored in front office systems, transform them into MLFi descriptions, and use these definitions to feed MLFi-based event tracking, documentation processing or other applications that could even be further integrated with downstream systems. MLFi may be used to develop both better interfaces and applications that contribute to expand derivatives trading capacity.

- *Prepare for the New Basel Capital Accord*. The New Basel Capital Accord proposes for the first time a measure for operational risk. The trading of complex financial contracts creates operational risk exposures for banks. Supervisory authorities will want to ensure that such exposures are properly measured in order to calculate a capital charge. In that respect, regulators are likely to scrutinize operational systems and inter-application communications in the same way they are currently auditing how market risk exposures are measured and managed. MLFi provides an opportunity to limit the capital charge for operational risk.

Data Warehousing Extract-Transform-Load (ETL)

The purpose of an ETL application is to extract data from one or more source systems, transform and map source data to a data warehouse and/or to one or more data marts, and load data into multi-dimensional presentation servers. ETL, also called data staging, is one of the most difficult pieces of a data warehouse project.

While a number of commercially available tools provide rich data modeling, extraction, transformation, loading, metadata management, and administration features to support the ETL process, these products lack critical functionality for a deployment in financial services.

MLFi enables you to:

- *Create Meaning-Preserving Extractions*. ETL tools are generally data driven and lose the semantics of financial contracts. A contract extracted from a source system is represented as a flat data projection from which the contract's exact meaning cannot be inferred without recourse to external documentation or algorithms. This results in a number of limitations:
    - *Inaccurate Description of Complex Contracts*. For example, data elements such as formula-based interest, redemption, and option payoffs can be extremely cumbersome to represent in a flat contract description. In the same vein, imagine conveying the full meaning of a Microsoft Excel® spreadsheet in a text file or database! MLFi enhances ETL tools by providing a live pivot format that preserves the full semantics of all extracted contracts.
    - *Lack of Contract State / Transformation Algorithm Synchronization*. The nature of financial contracts may depend on the occurrence of certain events: for instance, a simple option contract will be transformed into a firm position upon exercise. MLFi descriptions ensure that exported contract data is matched with the appropriate transformation algorithm and that the algorithm is optimal (See discussions on optimized algorithms and closed forms, and the Himalaya example in "MLFi Concepts—Contract and Process Independence—Pricing").
- *Rapidly Develop and Reuse Complex Transformations*. Another consequence of the lack of meaning-preserving extractions is that complex transformations—e.g., valuation, cash flow forecasts, risk measurement—need to be adapted for each product, one by one. In contrast, MLFi contract

definitions may drive calculation routines that work for an entire class of compatible contracts. This accelerates the development of transformation algorithms. In addition, MLFi's extensibility facilitates the reuse of sophisticated in-house valuation, exposure, and portfolio models that play a central role in the data staging process.

- *Obviate the Need for a Complex Data Warehouse Schema*. Many data warehouse implementations move data from operational systems to a centralized data warehouse, and then to subject-specific data marts. MLFi affords dramatic simplifications in the central data warehouse tier, which is often based on an entity-relationship data model. The design of a data model that can accommodate all contracts entered into by an institution is an inextricable problem mainly because of the diversity and complexity of financial products, and the rapid pace of innovation. LexiFi provides a simple, schema-independent way of persisting MLFi contract definitions in a SQL database: contracts and their lifecycle history are represented in a binary object that may be stored in a single table column. MLFi's contract descriptions encapsulate the most complex definitions: the central data warehouse schema only needs to describe the static data elements that are linked to MLFi contract definitions.
- *Leverage Source Systems' Application Logic.* It is often advantageous to leverage a source systems' application logic to extract relevant data as opposed to directly querying its physical database management system. This approach eliminates the need to duplicate proven data manipulation and financial functions, gives access to system management functions such as table locking and performance monitoring, and insulates the extract application from changes in the source systems' underlying data structures. The trend to progressively shield applications from the complexity of valuation models and contract descriptions (See "MLFi Applications—Product Repository") accelerates the sharing and reuse of algorithms used in source systems. MLFi's extensibility allows to fully benefit from that trend.
- *Facilitate Incremental Extractions and Loads*. It is generally desirable to minimize the size of extractions from production systems. Moreover, most data warehouses grow too large to completely replace their central fact tables in a single load window. Data warehouse architects find it much more efficient to incrementally extract and load the records that have been added or updated since the previous extraction or load. MLFi's operational semantics ensures that changes in a contract's life cycle are accurately reflected in each contract state. Contract state comparison is a powerful mechanism for detecting changes in existing contracts. In addition, the MLFi compiler can replay the history of transitions from state to state: this means that MLFi-based systems may be queried to provide an accurate snapshot of contracts descriptions at a particular point in time in order to perform historical extractions.
- *Perform Calculation Audits*. Finally, given the complexity of algorithms and data used to value financial contracts and to measure the risks of financial portfolios, the data staging application should enable users to check and validate intermediate results at each step of the data transformation process. LexiFi's compilation technology eases the linking of the same contract to multiple calculation libraries. This facilitates the comparison of transformation algorithms.

MLFi Benefits

Strengthen Customer Relationships

Faster Time-to-Market. The precise pricing of exact contract specifications and the timely automation of operational processes accelerate the introduction of new products and create new revenue opportunities without increasing costs or risks.

Better Service. Process automation also frees up operations personnel to devote more time to serve customers, mechanically improves service levels, and reduces error rates.

Faster time-to-market and better service result in improved client coverage and satisfaction.

Control Costs

Lower Investments. MLFi curtails the spiraling costs associated with the ongoing maintenance and upgrade of inadequate systems, and accelerates the delivery of highly customized solutions.

Lower Operating Expenses. In addition, the automation of multiple processes and easier communication with customers and counterparties, afforded by MLFi, enhance efficiency and reduce operating expenses.

Contain Operational Risk

MLFi promotes a controlled environment that reduces operational risk. For each complex contract, MLFi establishes one consistent "definition of record," which may be used by individual business units or product areas, company-wide functions, customers, and counterparties. By avoiding reliance on paper-based and manual processes, MLFi improves communication and increases transparency among stakeholders, which means fewer errors, exceptions, delays, and disputes.

LexiFi®

LexiFi provides software that enables financial institutions to structure, price, and process complex financial products.

LexiFi SAS
49 rue de Billancourt
F-92100 Boulogne-Billancourt
France

Phone: +33 1 47 43 90 00

info@lexifi.com www.lexifi.com

January 2005