

Automating the derivatives market: the need of a  
formal, exhaustive and compositional algebra  
allowing a uniform shareable description of the  
payoff of all kind of financial contracts

Jean-Marc Eber  
Founder, CEO  
LexiFi

September 5, 2019  
4th European Conference on Artificial Intelligence in Finance  
and Industry  
ZHAW, Winterthur, Switzerland

# Derivatives

- ▶ usual textbook example: a simple physically delivered European Call
- ▶ but there are much more complex financial contracts
- ▶ often, “simple” contracts contain optional clauses, for example “callable or convertible bonds”
- ▶ we restrict ourselves to **bilateral** financial contracts: one party, the holder, is “long”, the other is “short”

# What is (conceptually) difficult/easy to describe/implement

Easy:

- ▶ counterparty
- ▶ trade
- ▶ underlying reference
- ▶ ...

Difficult:

- ▶ contract “logic”
- ▶ contract life-cycle
- ▶ contract payoff

Easy: what can be mapped immediately into a relational database system

Difficult: what appears to have “infinite variability”

# Importance of a contract payoff description

Payoff:

- ▶ is a fundamental part of a financial contract description
- ▶ what are the “normal” rights and obligations associated to the holding of such a contract (typically receive or deliver money or physical goods, take decisions, . . . )?
- ▶ temporal and logical evolution of the contract, depending on “observables” and contract participants decisions
- ▶ market usage: informal, verbose description: mechanical treatment impossible, error prone, no industrialization further than “in-house” systems

**A generic and rigorous approach is needed**

# Limitations of a contract payoff description

Doesn't describe "everything"

- ▶ what happens when legal situation changes dramatically (example: Brexit)?
- ▶ what happens if a currency or equity disappears (Euro introduction)?
- ▶ there may be even rounding disputes/errors

**We should mitigate precisely what is part of the specification, what isn't**

# Standardizing financial contract payoffs ?

We often hear calls for standardizing financial payoffs

- ▶ this may work out for simple liquid sub-markets
- ▶ it is an illusion in general, as finance needs to adapt itself to ever changing demands and needs and market situations

**Don't standardize payoffs, but standardize a way to describe ever changing new payoffs**

## Many stakeholders, many use cases

A financial contract payoff:

- ▶ needs to be priced, and its risk managed accordingly (apply or “map” mathematical models, numerical procedures, . . . )
- ▶ needs to be explained, documented to a potential buyer
- ▶ needs to be executed over time, when uncertainty resolves (life-cycle management)
- ▶ needs to be accessible to regulators or law enforcement entities
- ▶ should be accessible to all kind of analytical tools: statistics, data analysis, AI, . . .

Fully expose payoff semantics!

Industrial fragmentation and specialization (cloud, exchange of documents, APIs, Blockchain, regulation, . . . ) makes a shareable rigorous description necessary

**Divergent needs make the design of a payoff specification formalism surprisingly difficult**

## Bad standardization: the “Menu approach”

“[...] version 1.0 of . . . . covers FX options and Swaps. Later versions will address other contract categories”

Translation:

- ▶ will never cover full spectrum
- ▶ will often be “late”
- ▶ will suffer from “resources exhaustion”

Iterative Standardization (ver. “1.0”, “1.5”, “2.0”, . . . ) is difficult, and must be designed for initially

Example: all earlier tentative structured products definition standardization efforts have failed!



## Goal

A contract payoff definition that can be read by a human being, efficiently processed by a computer, exchanged between market participants, and that satisfies three main goals:

- ▶ describe the rights and obligations of the parties both precisely and exhaustively avoiding future disputes
- ▶ lend itself to manipulations of various sorts, for example, for the purpose of pricing the contract and its credit risk, managing its clauses automatically, provide interactive simulation tools or producing cashflow forecasts
- ▶ reflect the evolution of the contract through time (life-cycle management)

## Avoiding future disputes: an old idea

“quando orientur controversiae, non magis disputatione opus erit inter duos philosophus, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus”

“[. . .] if controversies were to arise, there would be no more need of disputation between two philosophers than between two calculators. For it would suffice for them to take their pencils in their hands and to sit down at the abacus, and say to each other (and if they so wish also to a friend called to help): Let us calculate.”

Gottfried Wilhelm von Leibniz, “Dissertatio de Arte Combinatoria”, 1666

# Implementation cost amortization by genericity and global coverage

Support, once for all, all kind of payoffs, all underlyings, all markets etc.; therefore amortization over

- ▶ functionalities: benefits to many processes (front-office, back-office, regulation, marketing, . . . )
- ▶ time: designed to last for the foreseeable future (still usable/valid in 20 years?)
- ▶ space: potentially world-wide covering (currently “Accumulators” in Asia, “Autocalls” in Europe, etc.)

## Preferred methodology when suggesting a payoff description formalism

- ▶ self-contained (don't depend on other documents or rules)
- ▶ small (“minimalist”)
- ▶ precise: avoid divergent interpretations
- ▶ fundamental: focus on concepts (difficult), not on syntax (easy)
- ▶ implementable (better: show existing implementation)

## Our approach

For fulfilling all these requirements, we suggest that a payoff formalism should

- ▶ be a compositional algebra, defined with a limited number of basic combinators
- ▶ include lessons learned from theoretical computer science
- ▶ be as small as possible wrt. expressivity
- ▶ have a compositional semantics (only “understanding” all sub-expressions of an expression is needed for “understanding” an expression)
- ▶ not be considered as a “program” or “script”, but as a value, that can easily be analyzed, or even transformed
- ▶ be itself potentially subject to formal analysis (axiomatization, rewriting systems, machine-checked proofs, . . . )

Analogy with algebra:  $(1 + X) * (3 + Y)$

# Algebraic definition of a call (minimalist toy specification example)

The screenshot displays the LexiFi Apropos software interface. The title bar reads "Internal Contract Representation - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]". The top menu bar includes "File", "History", "Admin", "Navigation", and "Help". The top right corner shows the current contract name: "Simple European Call/Put".

The left sidebar contains a tree view of the software's structure:

- Contracts
  - Create with Instrument...
  - Create with Product Type...
  - Simple European Call/Put
    - Contract Information
    - Documents
    - Internal Contract Representation
    - Life Cycle Events
    - Manage
    - Meta Data
    - Parameters
    - Pricing
- Books
- Product Types
- Tools
  - Contracts
    - Contract Manager
    - Contract Reporting
    - Document Automation
  - Market Data
    - Static Data
  - Reporting
    - Grid Reporting

The main workspace is divided into two sections. The top section contains controls for the contract:

- Contract:
- Rendering:
- Options ...

The bottom section is a code editor displaying the algebraic definition of the contract in Mlfi syntax:

```
1 european_option "European" (cst 2019-01-12)
2   ~exercise:
3     (cash_flow (cst 2019-01-15) EUR (observe "EURO_STOXX_50" (cst 20
4     19-01-12) -.~ cst 4000.))
5   ~no_exercise:nothing
```

# More readable presentation of a call (simplified pretty-print for readability)

The screenshot shows the LexiFi Apropos application window. The title bar reads "Internal Contract Representation - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]". The menu bar includes "File", "History", "Admin", "Navigation", and "Help". The top right corner displays "Simple European Call/Put".

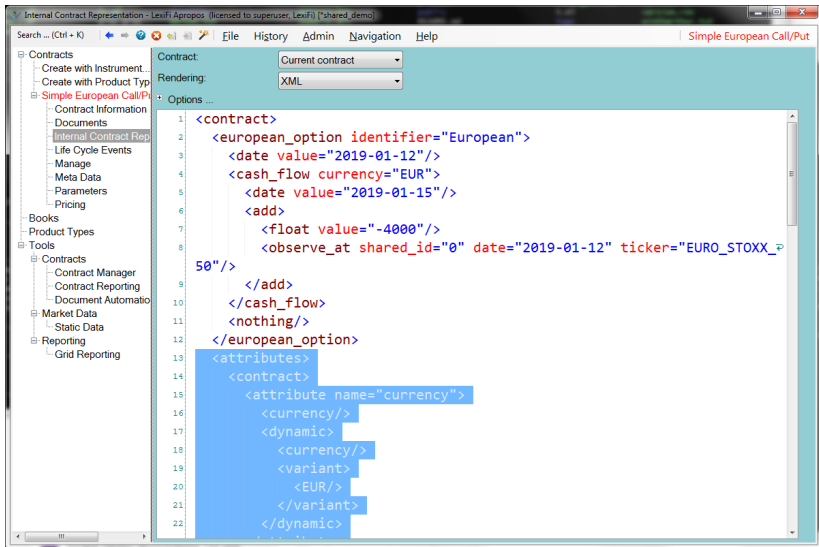
The left sidebar contains a tree view with the following structure:

- Contracts
  - Create with Instrument...
  - Create with Product Type...
  - Simple European Call/Put
    - Contract Information
    - Documents
    - Internal Contract Representation
    - Life Cycle Events
    - Manage
    - Meta Data
    - Parameters
    - Pricing
- Books
- Product Types
- Tools
  - Contracts
    - Contract Manager
    - Contract Reporting
    - Document Automation
  - Market Data
    - Static Data
  - Reporting
    - Grid Reporting

The main content area shows the contract details for "Simple European Call/Put". It includes a "Contract:" dropdown set to "Current contract" and a "Rendering:" dropdown set to "Pretty print". Below these are "Options ..." and a code editor displaying the following code:

```
1 european_option "European" t1
2 | Exercise ->
3   cash_flow t2 EUR (EURO_STOXX_50(t1) - 4000)
4 | No_exercise ->
5   nothing
6
7
8 payment fixing option other
9 t1 = 2019-01-12           x       x
10 t2 = 2019-01-15         x
```

# Call payoff rendered as an xml fragment



The screenshot shows the LexiFi Apropos interface. The left sidebar contains a tree view with categories like Contracts, Tools, Market Data, and Reporting. The main window displays the 'Simple European Call/Put' contract details. The 'Rendering' dropdown is set to 'XML'. The main area shows an XML fragment for a contract, with line numbers 1 through 22. The XML code is as follows:

```
1 <contract>
2   <european_option identifier="European">
3     <date value="2019-01-12"/>
4     <cash_flow currency="EUR">
5       <date value="2019-01-15"/>
6       <add>
7         <float value="-4000"/>
8         <observe_at shared_id="0" date="2019-01-12" ticker="EURO_STOXX_
9         50"/>
10      </add>
11    </cash_flow>
12    <nothing/>
13  </european_option>
14  <attributes>
15    <contract>
16      <attribute name="currency">
17        <currency/>
18        <dynamic>
19          <currency/>
20          <variant>
21            <EUR/>
22          </variant>
23        </dynamic>
```

Figure 3



## Life-cycle Management “for free”

Contract description transformation, similar to usual algebra

---

state 0	$(1+X)*(3+Y)$
Fixing	$X = 5$
state 1	$6*(3+Y)$
Fixing	$Y = 3$
state 2	30

---

- ▶ payoff description **simplifies** as uncertainty resolves
- ▶ this approach formalizes (and allows for implementation) life-cycle management.  $(1+X)*(3+Y)$  is the initial contract,  $6*(3+Y)$  current (simplified) contract,  $[X = 5; Y = 3]$  an audit trail of past observations etc.

Becomes a state-transition system, state being the “current” payoff description, transitions fired by external observations or events, transitions may have side effects (typically payments)

# Calendar of future events

The screenshot displays the LexiFi Apropos application interface. The title bar reads "Life Cycle Events - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]". The menu bar includes "Search ... (Ctrl + K)", "File", "History", "Admin", "Navigation", and "Help". The breadcrumb path is "Simple European Call/Put".

The left sidebar shows a tree view of the application's structure, with "Life Cycle Events" selected. The main content area is titled "Filters" and includes a dropdown for "Event type" set to "All", and several checked checkboxes: "Realized", "Future", "Maybe", "Out", and "Cancelled". Below the filters, there are links for "Apply fixings", "Manage options...", "Manage barriers...", "Execute deliveries...", "Cancel past events...", and "Remc".

A search bar is present, followed by a table of events. The table has columns for Status, Date, Event type, Value, Asset, and Details. Three items are shown:

Status	Date	Event type	Value	Asset	Details
Maybe	2019-0...	Fixing		EURO STOXX 50	
Future	2019-0...	Option			European, long party decides
Maybe	2019-0...	Receives	EURO STOXX 50(2019-01...	EUR	

Below the table, it states "3 of 3 items shown".

Figure 4

# Apply a fixing

Life Cycle Events - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]

Search ... (Ctrl + K) | File History Admin Navigation Help | Simple European Call/Put

Filters  
Event type: All | Realized  | Future  | Maybe  | Out  | Cancelled

Apply fixings | Manage fixings... | Manage options... | Manage barriers... | Execute deliveries... | [Cancel past events...](#) | [Remove past events](#)

Search:

Status	Date	Event type	Value	Asset	Details
Realized #1	2019-01-12	Fixing	4150	EURO STOXX 50	
Future	2019-01-12	Option			European, long party should exercise
Maybe	2019-01-15	Receives	150	EUR	

3 of 3 items shown

Figure 5

# Residual simplified contract: no reference to the underlying anymore

The screenshot displays the LexiFi Apropos interface for viewing the internal contract representation of a 'Simple European Call/Put' option. The window title is 'Internal Contract Representation - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]'. The top menu includes Search, File, History, Admin, Navigation, and Help. The left sidebar shows a tree view of the application's structure, with 'Internal Contract Rep' selected under the 'Simple European Call/Put' contract.

The main area shows the contract details and its internal representation. The 'Contract' dropdown is set to 'Current contract' and the 'Rendering' dropdown is set to 'Pretty print'. Below these are 'Options ...' and a code editor displaying the following internal representation:

```
1. european_option "European" t1
2. | Exercise ->
3. |   cash_flow t2 EUR 150
4. | No_exercise ->
5. |   nothing
6.
7. | payment fixing option other
8. t1 = 2019-01-12           x
9. t2 = 2019-01-15       x
```

Figure 6

## Industrial uses (examples)

- ▶ all LexiFi software stack built on top of this formalism since nearly two decades (continuously improved)
- ▶ LexiFi uses domain specific compilation techniques to generate highly efficient pricing code from “current contracts”, filtering out many pricing irrelevant informations
- ▶ analyze contract for semi-automatic “best pricing model” choice
- ▶ highly optimized contract life-cycle routines, embedding contract simplifications on the fly
- ▶ derive an interactive simulation tool for any contract payoff
- ▶ biggest LexiFi technology client is switching nearly all its payoffs to the algebraic representation (currently about 700.000 items)
- ▶ Bloomberg’s DLIB BLAN is built on top of this algebraic formalism

# Parameters of an "Autocall"

New Autocall - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]

Search ... (Ctrl + K) File History Admin Navigation Help

Contracts  
- Create with Instrument...  
- Create with Product Type  
New Autocall  
Books  
- Product Types  
Tools  
- Contracts  
- Contract Manager  
- Contract Reporting  
- Document Automatio  
- Market Data  
- Static Data  
- Reporting  
- Grid Reporting

Create Cancel Import Predefined Parameters Check/Preview Manage Saved Parameters

Denomination: EUR 1 000.000

Base dates: Import from Clipboard

Initial fixing date: 2018-01-12  
Issue date: Initial fixing date  
Final fixing date: Initial fixing date + 4 years  
Redemption date: Issue date + 4 years  
Barrier period start date: Initial fixing date  
Barrier period end date: Final fixing date

Barrier type: Final

Underlyings:

Strike percent of initial fixing:  100.00 %  
Barrier level percent of initial fixing:  60.00 %  
Autocall level percent of initial fixing: Global 100.00 %  
Coupon level percent of initial fixing: Global 100.00 %  
Lookback initial fixing:

Populate empty cells Clear initial fixing date and levels Import from Clipboard

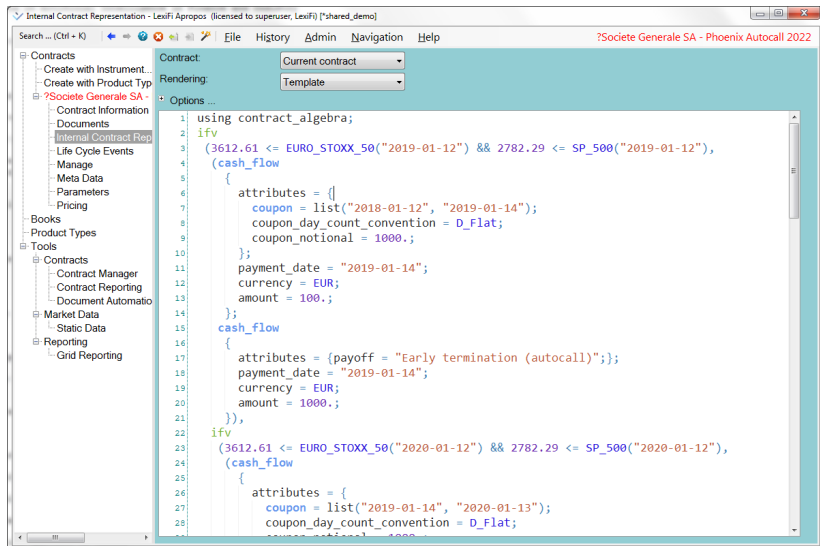
Underlying	Initial fixing date	Initial fixing	Strike level	Barrier level	Autocall level	Coupon level	Parity*
EURO_STOXX_50	2018-01-12	3 612.610					
SP_500	2018-01-12	2 782.290					
*							

\* When the currency of an Underlying differs from that of the Denomination, the expression in the Parity\* field is adjusted by the relea

Asian fixings:

Figure 7

# Autocall algebraic definition



The screenshot shows the LexiFi Apropos interface. The title bar reads "Internal Contract Representation - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]". The top menu bar includes "Search ... (Ctrl + K)", "File", "History", "Admin", "Navigation", and "Help". The top right corner displays "?Societe Generale SA - Phoenix Autocall 2022".

The left sidebar contains a tree view with categories: "Contracts", "Books", "Tools", "Market Data", and "Reporting". Under "Contracts", "Internal Contract Rep" is selected. Under "Tools", "Contracts" is expanded, showing "Contract Manager", "Contract Reporting", and "Document Automatio".

The main editor area shows the following code:

```
Contract: Current contract
Rendering: Template
Options ...
1 using contract_algebra;
2 ifv
3 (3612.61 <= EURO_STOXX_50("2019-01-12") && 2782.29 <= SP_500("2019-01-12"),
4 (cash_flow
5 {
6     attributes = {
7         coupon = list("2018-01-12", "2019-01-14");
8         coupon_day_count_convention = D_Flat;
9         coupon_notional = 1000.;
10    };
11    payment_date = "2019-01-14";
12    currency = EUR;
13    amount = 100.;
14 });
15 cash_flow
16 {
17     attributes = {payoff = "Early termination (autocall)"};
18     payment_date = "2019-01-14";
19     currency = EUR;
20     amount = 1000.;
21 });
22 ifv
23 (3612.61 <= EURO_STOXX_50("2020-01-12") && 2782.29 <= SP_500("2020-01-12"),
24 (cash_flow
25 {
26     attributes = {
27         coupon = list("2019-01-14", "2020-01-13");
28         coupon_day_count_convention = D_Flat;
29         coupon_notional = 1000.;
30    };
31 });
```

Figure 8

# Autocall calendar of future events

Life Cycle Events - LexiFi Apropos (licensed to superuser, LexiFi) [\*shared\_demo]

Search... (Ctrl+K) | File History Admin Navigation Help | ?Societe Generale SA - Phoenix Autocall 2022

Filters  
Event type: All | Realized  Future  Maybe  Out  Cancelled

Apply fixings: [Manage fixings...](#) | [Manage options...](#) | [Manage barriers...](#) | [Execute deliveries...](#) | [Cancel past events...](#) | [Remove past events](#)

Search:

Status	Date	Event type	Value	Asset	Details
Future	2019-01-12	Fixing		EURO STOXX...	
Future	2019-01-12	Fixing		SP 500	
Maybe	2019-01-14	Receives	100	EUR	coupon(2018-01-12, 2019-01-14)
Maybe	2019-01-14	Receives	1000	EUR	"Early termination (autocall)", Grp(...)
Maybe	2020-01-12	Fixing		EURO STOXX...	
Maybe	2020-01-12	Fixing		SP 500	
Maybe	2020-01-13	Receives	200	EUR	coupon(2019-01-14, 2020-01-13)
Maybe	2020-01-13	Receives	1000	EUR	"Early termination (autocall)", Grp(...)
Maybe	2021-01-12	Fixing		EURO STOXX...	
Maybe	2021-01-12	Fixing		SP 500	
Maybe	2021-01-12	Receives	300	EUR	coupon(2020-01-13, 2021-01-12)
Maybe	2021-01-12	Receives	1000	EUR	"Early termination (autocall)", Grp(...)
Maybe	2022-01-12	Fixing		EURO STOXX...	
Maybe	2022-01-12	Fixing		SP 500	
Maybe	2022-01-12	Barrier			Final Barrier: EURO STOXX 50 ...
Maybe	2022-01-12	Barrier			Final Barrier: SP 500 <= 1669.37
Maybe	2022-01-12	Receives	400	EUR	coupon(2021-01-12, 2022-01-12)
Maybe	2022-01-12	Receives	1000 * (1 + min(EURO STOXX 50(2022-...	EUR	Grp(1,certain)
Maybe	2022-01-12	Receives	1000	EUR	Grp(1,certain)
Maybe	2022-01-12	Receives	1000	EUR	"Early termination (autocall)", Grp(...)

20 of 20 items shown

Figure 9



# Generated graphical simulation tool

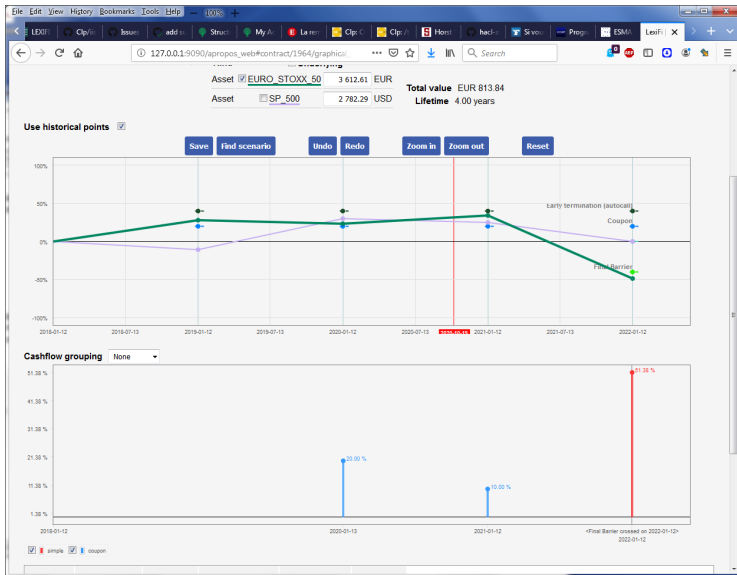


Figure 10

## Want to learn more and investigate?

Observe this field-proven technology at work:

- ▶ create and store some usual structured product
- ▶ investigate its algebraic definition
- ▶ manage the contract (fixings, barrier hittings, . . . ) up to maturity, see how algebraic definition simplifies
- ▶ simulate interactively the contract

---

Ask for free access to LexiFi's Technology Discovery Web Site by sending a message through <https://www.lexifi.com/#contact>.  
More information also on [www.lexifi.com](http://www.lexifi.com)

---