# 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)

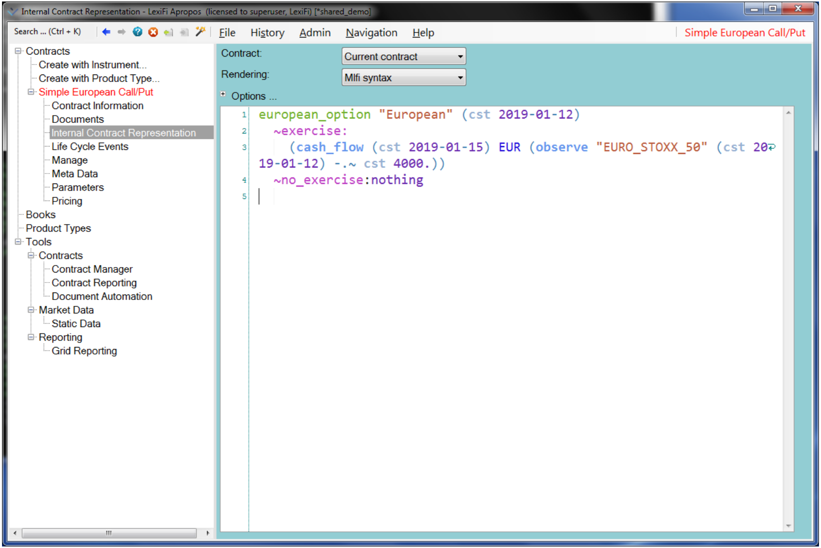
# Our approach

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)



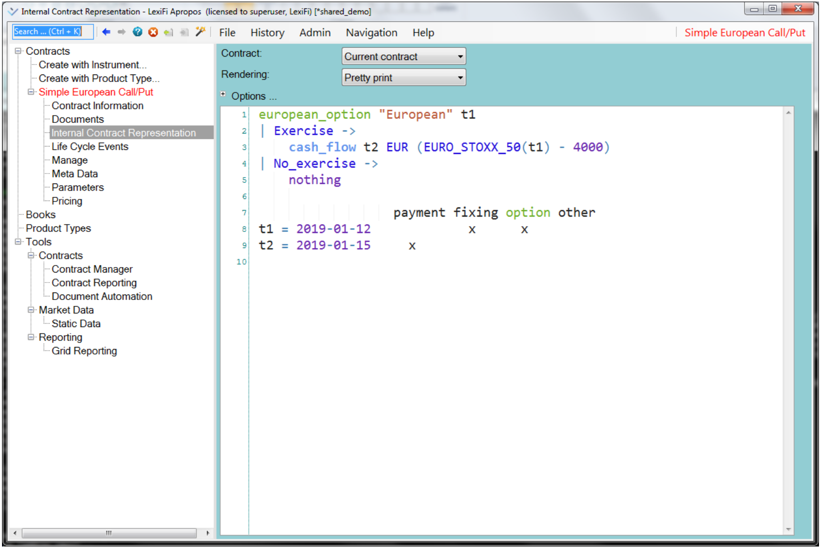
Contract: Current contract

Rendering: Mlfi syntax

```
european_option "European" (cst 2019-01-12)
  ~exercise:
    (cash_flow (cst 2019-01-15) EUR (observe "EURO_STOXX_50" (cst 2019-01-12) -.~ cst 4000.))
  ~no_exercise:nothing
```

Figure 1

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



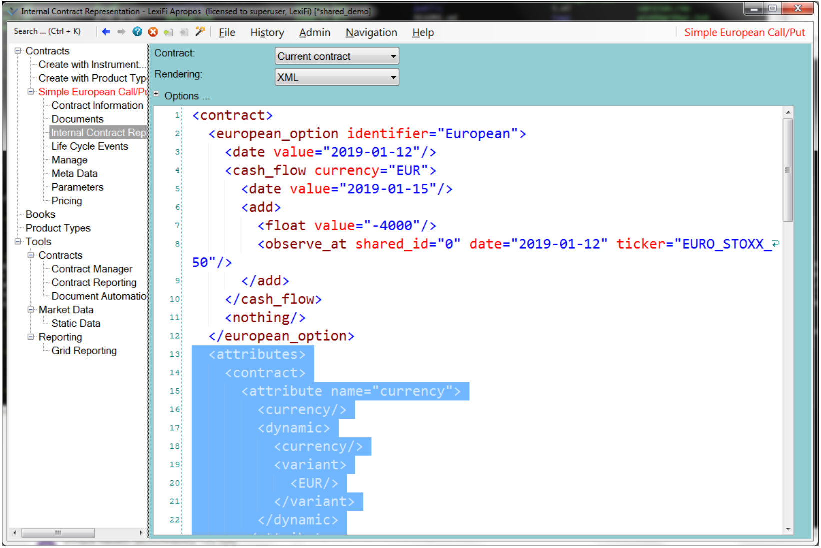
Contract: Current contract

Rendering: Pretty print

```
european_option "European" t1
| Exercise ->
    cash_flow t2 EUR (EURO_STOXX_50(t1) - 4000)
| No_exercise ->
    nothing

                  payment fixing option other
t1 = 2019-01-12            x      x
t2 = 2019-01-15     x
```

Figure 2

# Call payoff rendered as an xml fragment



Contract: Current contract

Rendering: XML

```
<contract>
  <european_option identifier="European">
    <date value="2019-01-12"/>
    <cash_flow currency="EUR">
      <date value="2019-01-15"/>
      <add>
        <float value="-4000"/>
        <observe_at shared_id="0" date="2019-01-12" ticker="EURO_STOXX_50"/>
      </add>
    </cash_flow>
    <nothing/>
  </european_option>
  <attributes>
    <contract>
      <attribute name="currency">
        <currency/>
        <dynamic>
          <currency/>
          <variant>
            <EUR/>
          </variant>
        </dynamic>
```

Figure 3

# Life-cycle Management “for free”

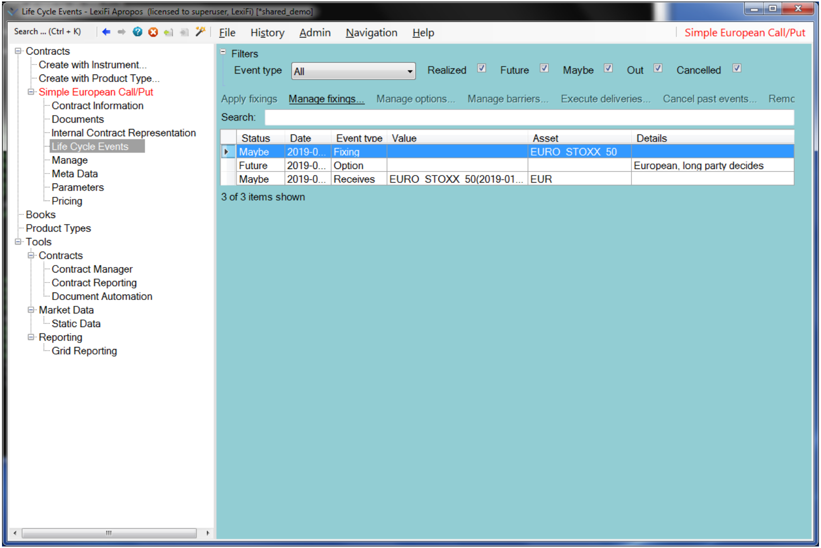
Contract description transformation, similar to usual algebra

| | |
|---|---|
| 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



Filters

Event type: All

Realized ☑ Future ☑ Maybe ☑ Out ☑ Cancelled ☑

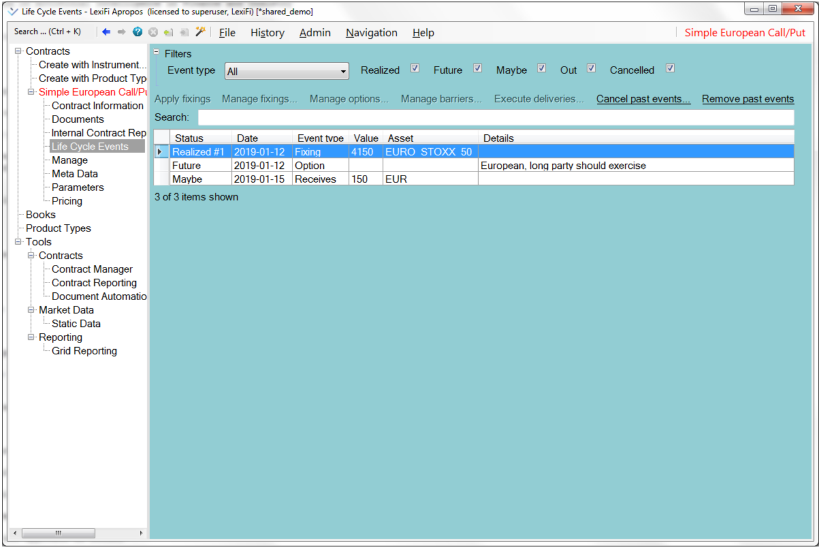
Search:

| 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 | |

3 of 3 items shown

Figure 4

# Apply a fixing



Event type: All

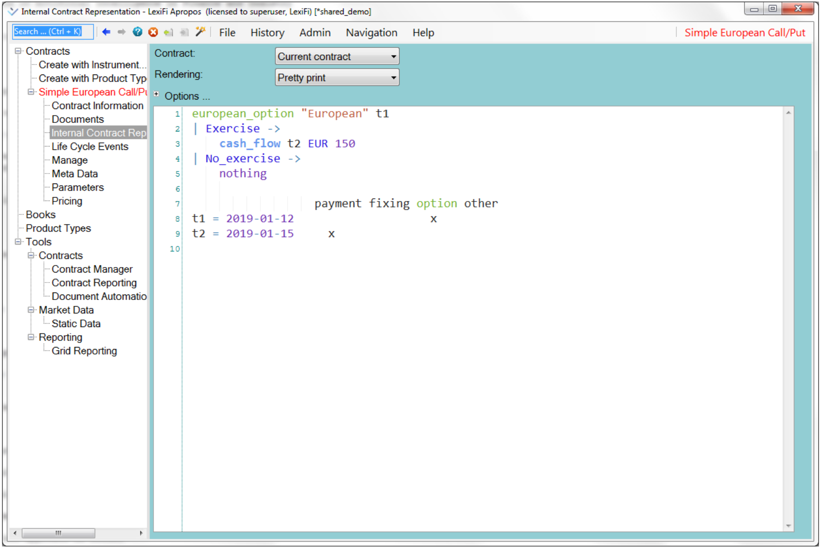
Realized ☑ Future ☑ Maybe ☑ Out ☑ Cancelled ☑

|  | 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



Contract: Current contract

Rendering: Pretty print

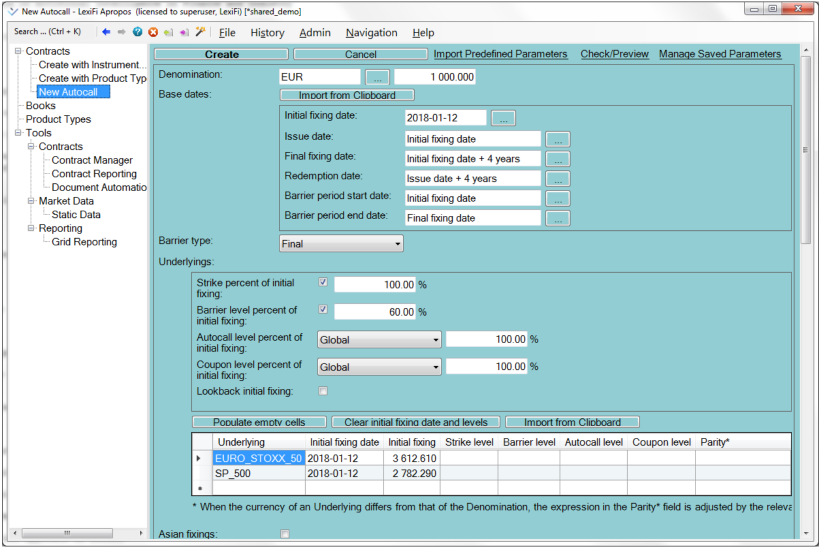
```
european_option "European" t1
| Exercise ->
    cash_flow t2 EUR 150
| No_exercise ->
    nothing

                  payment fixing option other
t1 = 2019-01-12                    x
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”



Denomination: EUR 1 000.000

Base dates:

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: □

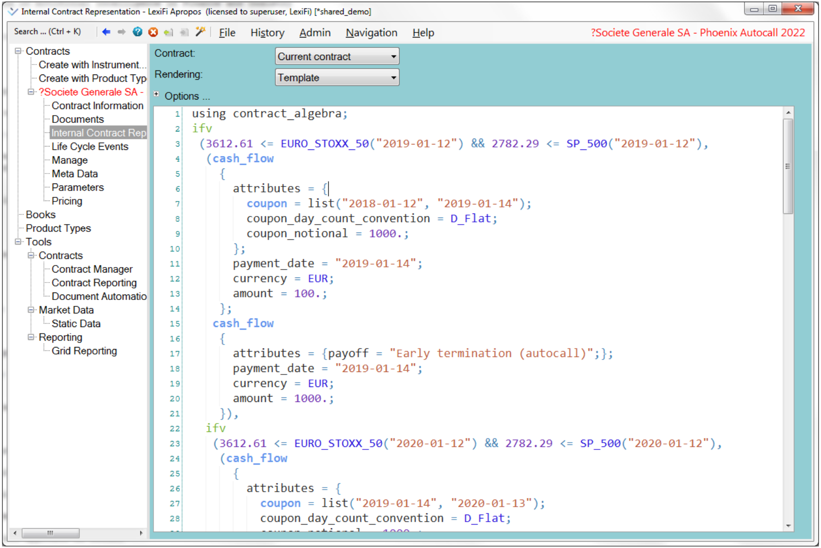
| Underlying | Initial fixing date | Initial fixing | Strike level | Barrier level | Autocall level | Coupon level | Parity* |
|---|---|---|---|---|---|---|---|
| EURO_STOXX_50 | 2018-01-12 | 3 612.610 | | | | | |
| SP_500 | 2018-01-12 | 2 782.290 | | | | | |

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

Asian fixings: □

Figure 7

# Autocall algebraic definition



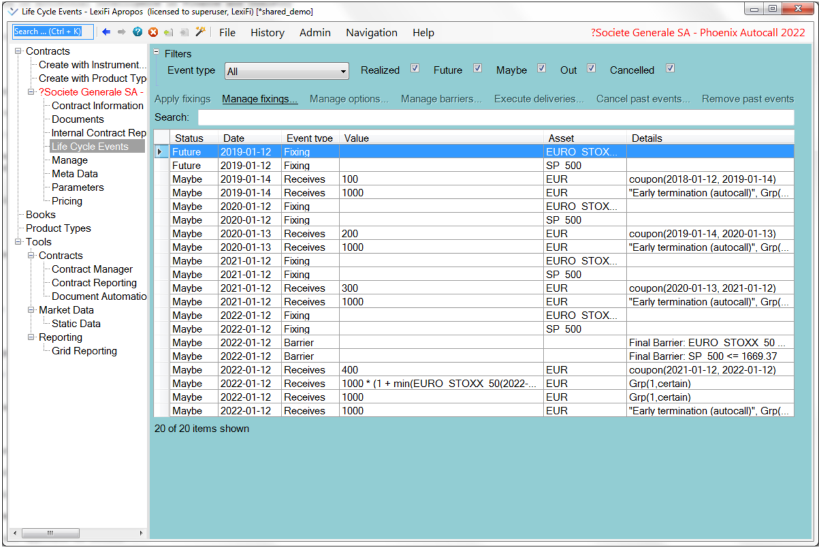
Contract: Current contract

Rendering: Template

```
using contract_algebra;
ifv
 (3612.61 <= EURO_STOXX_50("2019-01-12") && 2782.29 <= SP_500("2019-01-12"),
  (cash_flow
    {
      attributes = {
        coupon = list("2018-01-12", "2019-01-14");
        coupon_day_count_convention = D_Flat;
        coupon_notional = 1000.;
      };
      payment_date = "2019-01-14";
      currency = EUR;
      amount = 100.;
    };
   cash_flow
    {
      attributes = {payoff = "Early termination (autocall)";};
      payment_date = "2019-01-14";
      currency = EUR;
      amount = 1000.;
    }),
  ifv
   (3612.61 <= EURO_STOXX_50("2020-01-12") && 2782.29 <= SP_500("2020-01-12"),
    (cash_flow
      {
        attributes = {
          coupon = list("2019-01-14", "2020-01-13");
          coupon_day_count_convention = D_Flat;
          coupon_notional = 1000.
```

Figure 8

# Autocall calendar of future events



Event type: All

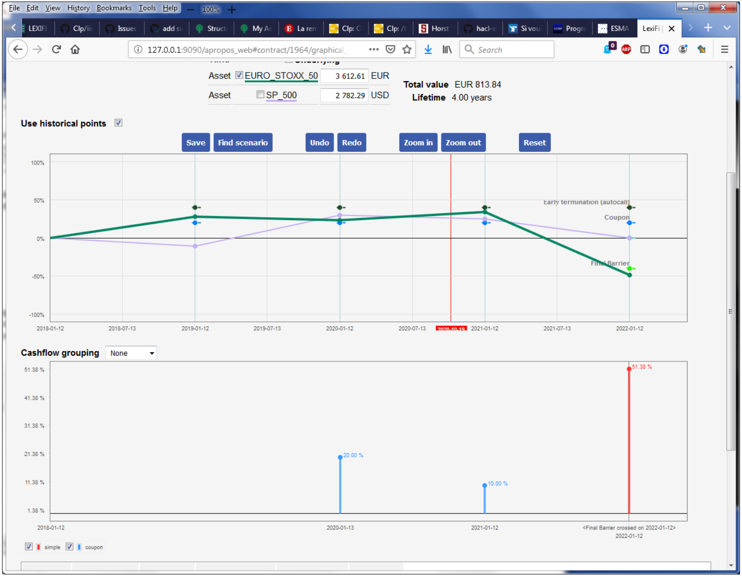
Realized ☑ Future ☑ Maybe ☑ Out ☑ Cancelled ☑

| Status | Date | Event type | Value | Asset | Details |
|---|---|---|---|---|---|
| Future | 2019-01-12 | Fixing | | EURO STOX... | |
| 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 STOX... | |
| 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 STOX... | |
| 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 STOX... | |
| 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



Asset ☑ EURO_STOXX_50 3 612.61 EUR

Asset □ SP_500 2 782.29 USD

**Total value** EUR 813.84

**Lifetime** 4.00 years

**Use historical points** ☑

**Cashflow grouping** None

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`