

MathOCL: a domain-specific language for financial modelling

MDENet Seedcorn project “Financial Model Validation using Model-driven Engineering”

Howard Haughton (HRS Ltd) & Yannis Zorgios (CLMS Ltd) &
Kevin Lano (AgileMDE Ltd)

Financial models

- Mathematical models that represent financial processes such as valuation & risk estimation.
- Regulators require institutions to periodically validate financial models to ensure they are operating as intended.
- We defined a domain-specific language (DSL) *MathOCL* for expressing financial models + associated tools for analysing models & transitioning to executable implementations.

Introduction

- Finance industry is significant in terms of employment & contribution to GDP.
- Also facilitates industrial & business activities through provision of investment & credit services.
- Mathematical theories and models, such as Black-Scholes option pricing model, underpin activities of finance institutions.
- Financial models used to price transactions, make business decisions or report financial results.

Financial model validation concepts

United States Federal Reserve definition:

“The term model refers to a quantitative method, system, or approach that applies statistical, economic, financial, or mathematical theories, techniques, and assumptions to process input data into quantitative estimates.”

Federal Reserve definition of model validation:

“Model validation is the set of processes and activities intended to verify that models are performing as expected, in line with their design objectives and business uses. Effective validation helps to ensure that models are sound, identifying potential limitations and assumptions and assessing their possible impact. All model components—inputs, processing, outputs, and reports—should be subject to validation.”

Model-driven Engineering

- MDE has been successful in specific application domains, e.g., in aerospace & automotive systems.
- But low uptake of MDE in finance sector, despite high importance of software correctness & quick time-to-market in finance.
- MDE notations such as OCL not familiar to finance practitioners, who use classical mathematics of continuous functions & stochastic processes.
- In order to apply MDE to financial model development we need to support domain-specific constructs.

MathOCL DSL

Extension of OCL supporting:

- Conventional mathematical notations for integration, \int_a^b , differentiation, ∂_x , statistical expectation $E[expr]$, summation (Σ), product (Π), powers (eg., e^x), etc.
- Definition of random variables following normal distribution + other significant statistical distributions.
- Algebraic simplification & symbolic evaluation.
- Equation solving & proof documentation.

<i>Construct</i>	<i>Explanation</i>
Define v	Introduce variable v
Define $v = expr$	Define v 's value as $expr$
Define $v = instr$	Define v 's value as result of $instr$
Define $v \sim D$	Define v as random variable from distribution D
Simplify $expr$	Simplify an expression
Solve $eqns$ for $vars$	Solve one/set of equations Quadratic, simple differential and multiple linear equations are supported.
Constraint on $v \mid expr$	Constrain v by $expr$
Prove $expr$ if $assm$	Prove $expr$ from $assm$

MathOCL DSL

- ANTLR grammar defined for MathOCL
- MathOCL syntax trees processed by *CSTL/CGTL* rewrite rules for algebraic simplification
- Interactive editing of MathOCL text to give instructions, eg., to factor an expression.

MathOCL Editor

File Edit Style Analysis Translate Help

∃	∅	•	∫	'	∂	Z
...	α	β	γ	δ	ε	ζ	θ	λ	μ	ν	π	ρ	σ	τ	χ	...

specification S
 Define X ~ LogNorm(0,1)
 Simplify E[2*X]

specification S Define X ~ LogNorm(0, 1)
 Simplify 2*e^{0.5}

Specification analysed & simplified

Example of random variable definition

MathOCL Editor

File Edit Style Analysis Translate Help

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int	'	∂	∞	\mathbb{N}
\mathbb{R}	α	β	γ	δ	ϵ	ζ	θ	λ	μ	ν	π	ρ	σ	τ	χ

specification S
Simplify $\lim_{x \rightarrow \infty} \sin(1/x)$

specification S Simplify 0

Example of limit evaluation

Case studies

- Computational solution of Thiele's equations for life insurance
- Realistic computational solution of bond pricing taking into account different day-count conventions
- Validation of computational procedure for pricing share-based options, the Cox, Ross, Rubinstein Binomial model (CRR), against analytic model of Black-Scholes.

MathOCL Editor

File Edit Style Analysis Translate Help

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int	'	∂	∞	\mathbb{N}	\mathbb{Z}
\mathbb{R}	α	β	γ	δ	ϵ	ζ	θ	λ	μ	ν	π	ρ	σ	τ	χ	ω

```

specification S
  Define r(x) = 0.01
  Solve Vdead' - r(x)*Vdead = 0 for Vdead

  Define mu_alive_dead(x) = x*0.001
  Define g(x) = r(x) + mu_alive_dead(x)

  Define b_alive_dead(x) = 1.0/(x + 25)
  Define f(x) = mu_alive_dead(x) * (b_alive_dead(x) + Vdead(x))

  Solve Valive' - g(x)*Valive + f(x) = 0 for Valive
  
```

```

pre: true
post: result = A2/J2(x);

operation mu_alive_dead(x : double) : double
pre: true
post: result = x*0.001;

operation g(x : double) : double
pre: true
post: result = r(x) + mu_alive_dead(x);

operation b_alive_dead(x : double) : double
pre: true
post: result = 1/(x + 25.0);

operation f(x : double) : double
pre: true
post: result = (mu_alive_dead(x)*(b_alive_dead(x) + Vdead(x)));

operation J3(x : double) : double
pre: true
post: result = MathLib.eValue()->pow(-((MathLib.indefiniteIntegral(lam
da x : double in g(x))->apply(x))));

operation Valive(x : double) : double
pre: true
post: result = (1/J3(x))*(MathLib.indefiniteIntegral(lambda x : double in
(f(x))*J3(x))->apply(x)) + A3/J3(x);
  
```

```

specification S Define r(x) = 0.01
Define J2(x) = e^{∫ -r(x) dx}
Define A2
Define Vdead(x) = A2/J2(x)

Define mu_alive_dead(x) = x*0.001
Define g(x) = r(x) + mu_alive_dead(x)
Define b_alive_dead(x) = 1/(x + 25.0)
Define f(x) = (mu_alive_dead(x)*(b_alive_dead(x) + Vdead(x)))
Define J3(x) = e^{∫ -g(x) dx}
Define A3
Define Valive(x) = (1/J3(x))^{∫ (f(x))*J3(x) dx} + A3/J3(x)
  
```

Translated to UML/OCL

Thiele's equations and solutions in MathOCL and UML/OCL

MathOCL Editor

File Edit Style Analysis Translate Help

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int	'	∂	∞	\mathbb{N}	\mathbb{Z}
\mathbb{R}	α	β	γ	δ	ϵ	ζ	θ	λ	μ	ν	π	ρ	σ	τ	χ	ω

```

specification S
  Define issu : Date = Date("2019/01/01")
  Define sett : Date = Date("2019/05/02")
  Define matur : Date = Date("2022/01/01")

  Define coup = 0.08
  Define yld = 0.06
  Define freq : ℕ = 2

  Define dp = bondPrice(yld, sett, matur, coup, "Actual/ActualCMA", freq)

  Define accint = accumulatedInterest(issu, sett, freq, coup, "Actual/ActualCMA", matur)

  Define cleanPrice = (dp - accint)

```

```

class S {
  attribute issu : OclDate := OclDate.newOclDate_String("2019/01/01");
  attribute sett : OclDate := OclDate.newOclDate_String("2019/05/02");
  attribute matur : OclDate := OclDate.newOclDate_String("2022/01/01");
  attribute coup : double := 0.08;
  attribute yld : double := 0.06;
  attribute freq : int := 2;
  attribute dp : double := FinanceLib.bondPrice(yld,sett,matur,coup,"Actual/ActualCMA",freq);
  attribute accint : double := FinanceLib.accumulatedInterest(issu,sett,freq,coup,"Actual/ActualCMA",matur);
  attribute cleanPrice : double := (dp - accint);

  operation issuDefinition() : OclDate
  pre: true
  post: result = OclDate.newOclDate_String("2019/01/01");

  operation settDefinition() : OclDate
  pre: true
  post: result = OclDate.newOclDate_String("2019/05/02");

  operation maturDefinition() : OclDate
  pre: true
  post: result = OclDate.newOclDate_String("2022/01/01");

  operation coupDefinition() : double
  pre: true
  post: result = 0.08;

  operation yldDefinition() : double
  pre: true
  post: result = 0.06;

  operation freqDefinition() : int
  pre: true
  post: result = 2;

  operation dpDefinition() : double
  pre: true
  post: result = FinanceLib.bondPrice(yld,sett,matur,coup,"Actual/ActualCMA",freq);
}

```

MathOCL model of bond valuation

MathOCL Editor

File Edit Style Analysis

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int
\mathbb{R}	α	β	γ	δ	ϵ	ζ	θ	λ	μ	ν	π

specification ReplicatingPortfolioAnalysis

```
Define S0 /* stock value at time 0 */
Define Ou /* Option value after stock price rise */
Define Od /* Option value after stock price drop */
```

```
Define t = 1 /* current time */
Define K = et
```

```
Define d /* down movement */
Constraint on d | d > 0 & d < 1
Define u = 1/d /* up movement */
```

```
Solve h1*S0*u + h2*K = Ou, h1*S0*d + h2*K = Od for h1, h2
```

MathOCL model of CRR validation

Analysis using MathOCL tools

- Top panel gives model in math notation. Tool provides automated keyword guidance + syntax checking.
- The model equation can be automatically solved (symbolically) using MathOCL tools, & interactively simplified to produce more explicit version
- Lower panel provides interactive simplification
- Right-hand panel gives UML/OCL translation, used for code generation.

MathOCL Editor

File Edit Style Analysis

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int	'
\mathbb{R}	α	β	γ	δ	ϵ	ζ	θ	λ	μ	ν	π	ρ

specification S

Solve $h1 \cdot S0 \cdot u + h2 \cdot K = Ou$, $h1 \cdot S0 \cdot d + h2 \cdot K = Od$ for $h1, h2$

specification S Define $h1 = \text{Cancel } K \text{ in } (K \cdot Od - (-Ou) \cdot K) / (S0 \cdot u \cdot K - K \cdot S0 \cdot d)$
Define $h2 = \text{Cancel } S0 \text{ in } (((-Ou) \cdot S0) \cdot d - (S0 \cdot u \cdot -Od)) / (S0 \cdot u \cdot K - K \cdot S0 \cdot d)$

MathOCL model: CRR equation solution and interactive instruction

Theorem-proving and proof-checking

- We are adding proof construction capabilities to MathOCL
- Basic proof checking can be carried out using MathOCL tools algebraic simplification, eg., to deduce $x < y$ from $e^x < e^y$.
- In CRR validation case, key argument is that as number N of periods in binomial tree for option term T , each period of length $Dt = T/N$, is increased, then option price estimate converges to Black-Scholes price.
- Steps + verification shown in following screen.

MathOCL Editor

File Edit Style Analysis Translate

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int	'	∂
\mathbb{R}	α	β	γ	δ	ϵ	ζ	θ	λ	μ	ν	π	ρ	σ

specification S

Define Dt

Define $u = e^{\sigma \cdot \sqrt{Dt}}$

Constraint on Dt | $0 < Dt \ \& \ Dt < 1$

Prove $\mu \cdot Dt < \sigma$ if $\mu \cdot Dt < \sigma \cdot \sqrt{Dt}$

Prove $\mu \cdot Dt < \sigma \cdot \sqrt{Dt}$ if $e^{\mu \cdot Dt} < e^{\sigma \cdot \sqrt{Dt}}$

Prove $e^{\mu \cdot Dt} < u$ if $e^{\mu \cdot Dt} - d < u - d$

specification S

Define Dt

Define $u = e^{\{g\{s\}\} \dagger Dt}$

Constraint on Dt | $0 < Dt \ \& \ Dt < 1$

Simplify $(g\{m\} \cdot Dt < g\{s\} \dagger Dt) \Rightarrow (g\{m\} \cdot Dt < g\{s\})$

Simplify true

Simplify true

CRR Proof steps in MathOCL; automated simplification/proof

Generation of code

- From simplified & explicit MathOCL specification containing only *Constraint*, *Simplify* and *Define* constructs, computational specification can be automatically generated in standard UML/OCL.
- From this UML/OCL specification, executable code in multiple languages can be generated using AgileUML toolset, including code in Python 3.9, C# and Java
- Mathematical and finance OCL libraries defined to support implementations
- Synthesis of Mamba3 code for zAppDev low-code platform.

```
Command Prompt - java MathApp

public double g(double x)
{
    double result = 0.0;

    result = this.r(x) + this.mu_alive_dead(x);
    return result;
}

public double b_alive_dead(double x)
{
    double result = 0.0;

    result = 1 / (x + 25.0);
    return result;
}

public double f(double x)
{
    double result = 0.0;

    result = ( this.mu_alive_dead(x) * (this.b_alive_dead(x) + this.Vdead(x)) );
    return result;
}

public double J3(double x)
{
    double result = 0.0;

    result = Math.Pow(MathLib.eValue(), -MathLib.indefiniteIntegral(x => (this.g(x))).Invoke(x));
    return result;
}

public double Valive(double x)
{
    double result = 0.0;

    result = ( 1 / this.J3(x) ) * MathLib.indefiniteIntegral(x => (( this.f(x) * this.J3(x) ))).Invoke(x) + A3 / this.
J3(x);
    return result;
}
}
```

Thiele's Equations C# code

MathOCL Editor

File Edit Style Analysis Translate

\exists	\forall	\in	\notin	\emptyset	\bullet	\approx	\rightarrow	Σ	Π	$\sqrt{\quad}$	\int
\mathbb{R}	α	β	ν	σ	ϵ	ζ	θ	\wedge	μ	ν	π

```

specification ReplicatingPortfolioAnalysis
  Define S0 /* stock value at time 0 */
  Define Ou /* Option value after stock price rise */
  Define Od /* Option value after stock price drop */

  Define t = 1 /* current time */
  Define K = e^t

public class ReplicatingPortfolioAnalysis
{
  private double S0; // internal
  private double Ou; // internal
  private double Od; // internal
  private double t; // internal
  private double K; // internal
  private double d; // internal
  private double u; // internal
  private double h1; // internal
  private double h2; // internal

  public double tDefinition()
  { double result = 0.0;
    result = 1;
    return result;
  }

  public double KDefinition()
  { double result = 0.0;
    result = Math.Pow(MathLib.eValue(), t);
    return result;
  }

  public double uDefinition()
  { double result = 0.0;
    result = 1 / d;
    return result;
  }

  public double h1Definition()
  { double result = 0.0;
    result = ( -Od - -Ou ) / ( S0 * u - S0 * d );
    return result;
  }
}

```

CRR C# code

```

OclDate
systemTime: long = 0
time: long = 0
year: int = 0
month: int = 0
day: int = 0
weekday: int = 0
hour: int = 0
minute: int = 0
second: int = 0
newOclDate(): OclDate
newOclDate String(...): OclDate
newOclDate Time(...): OclDate
newOclDate YMD(...): OclDate
newOclDate YMDHMS(...): OclDate
setTime(...): void
getTime(): long
dateBefore(...): boolean
dateAfter(...): boolean
compareToYMD(...): int
maxDateYMD(...): OclDate
yearDifference(...): long
monthDifference(...): long
differenceMonths(...): int
dayDifference(...): long
hourDifference(...): long
minuteDifference(...): long
secondDifference(...): long
getSystemTime(): long
setSystemTime(...): void
getYear(): int
getMonth(): int
getDate(): int
getDay(): int
getHour(): int
getHours(): int
getMinute(): int
getMinutes(): int
getSecond(): int
getSeconds(): int
addYears(...): OclDate
addMonths(...): OclDate
addMonthYMD(...): OclDate
subtractMonthYMD(...): OclDate
addDays(...): OclDate
addHours(...): OclDate
addMinutes(...): OclDate
addSeconds(...): OclDate
toString(): String
leapYear(...): boolean
isLeapYear(): boolean
monthDays(...): int
daysInMonth(): int
isEndOfMonth(): boolean
daysBetweenDates(...): int
    
```

```

MathLib
ix: int = 0
iy: int = 0
iz: int = 0
defaultTolerance: double = 0.0
hexdigit: Sequence(String) = Sequence{}
initialiseMathLib(): void
pi(): double
piValue(): double
e(): double
eValue(): double
setSeeds(...): void
setSeed(...): void
random(): double
random(): double
combinatorial(...): long
factorial(...): long
asinh(...): double
acosh(...): double
atanh(...): double
decimal2bits(...): String
decimal2binary(...): String
decimal2oct(...): String
decimal2octal(...): String
decimal2hex(...): String
bytes2integer(...): long
integer2bytes(...): Sequence(int)
integer2Nbytes(...): Sequence(int)
bitwiseAnd(...): int
bitwiseOr(...): int
bitwiseXor(...): int
bitwiseNot(...): int
toBitSequence(...): Sequence(boolean)
modInverse(...): long
modPow(...): long
doubleToLongBits(...): long
longBitsToDouble(...): double
roundN(...): double
truncateN(...): double
toFixedPoint(...): double
toFixedPointRound(...): double
isIntegerOverflow(...): boolean
mean(...): double
median(...): double
variance(...): double
standardDeviation(...): double
lcm(...): int
bisectionAsc(...): double
rowMult(...): Sequence(double)
matrixMultiplication(...): Sequence(Sequence(double))
differential(...): Function(double,double)
definiteIntegral(...): double
indefiniteIntegral(...): Function(double,double)
    
```

```

FinanceLib
discountDiscrete(...): double
netPresentValueDiscrete(...): double
presentValueDiscrete(...): double
irrDiscrete(...): double
straddleDates(...): Sequence(OclDate)
numberOfPeriods(...): int
sequenceOfPeriods(...): Sequence(int)
couponDates(...): Sequence(OclDate)
days360(...): int
numberOfMonths(...): Sequence(double)
calculateCouponPayments(...): Sequence(double)
bondCashFlows(...): Sequence(OclAny)
bondPrice(...): double
accInterest(...): double
accumulatedInterest(...): double
bondPriceClean(...): double
    
```

Mathematical and finance libraries

Related work

- Financial specification tools include Kapital system at J. P. Morgan
- There are also functional & declarative languages for specifying financial products
- MathOCL is novel because it includes modelling of financial *processes*, such as Monte-Carlo simulation
- It also uses conventional mathematical notation, instead of program-like syntax.

Conclusions

- We have carried out several case studies of financial specification using MathOCL in order to evaluate its effectiveness for financial model definition and analysis.
- Approach provides rigorous support for model construction & validation.
- Increased transparency of model construction & seamless translation to code.
- Potential cost savings from enhanced productivity and time saved using MDE/MathOCL versus manual development.