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

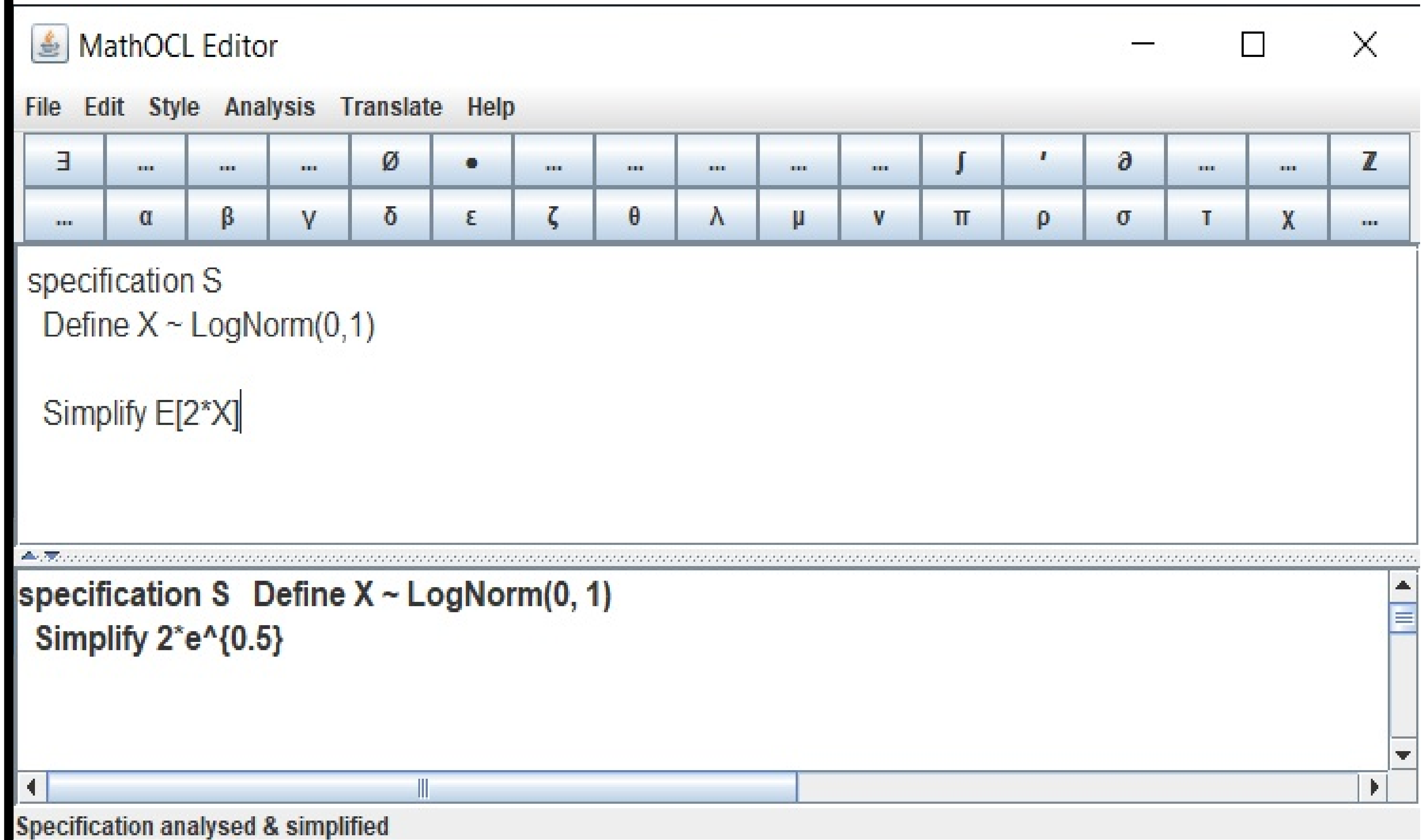
Extension of OCL supporting:

- 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.

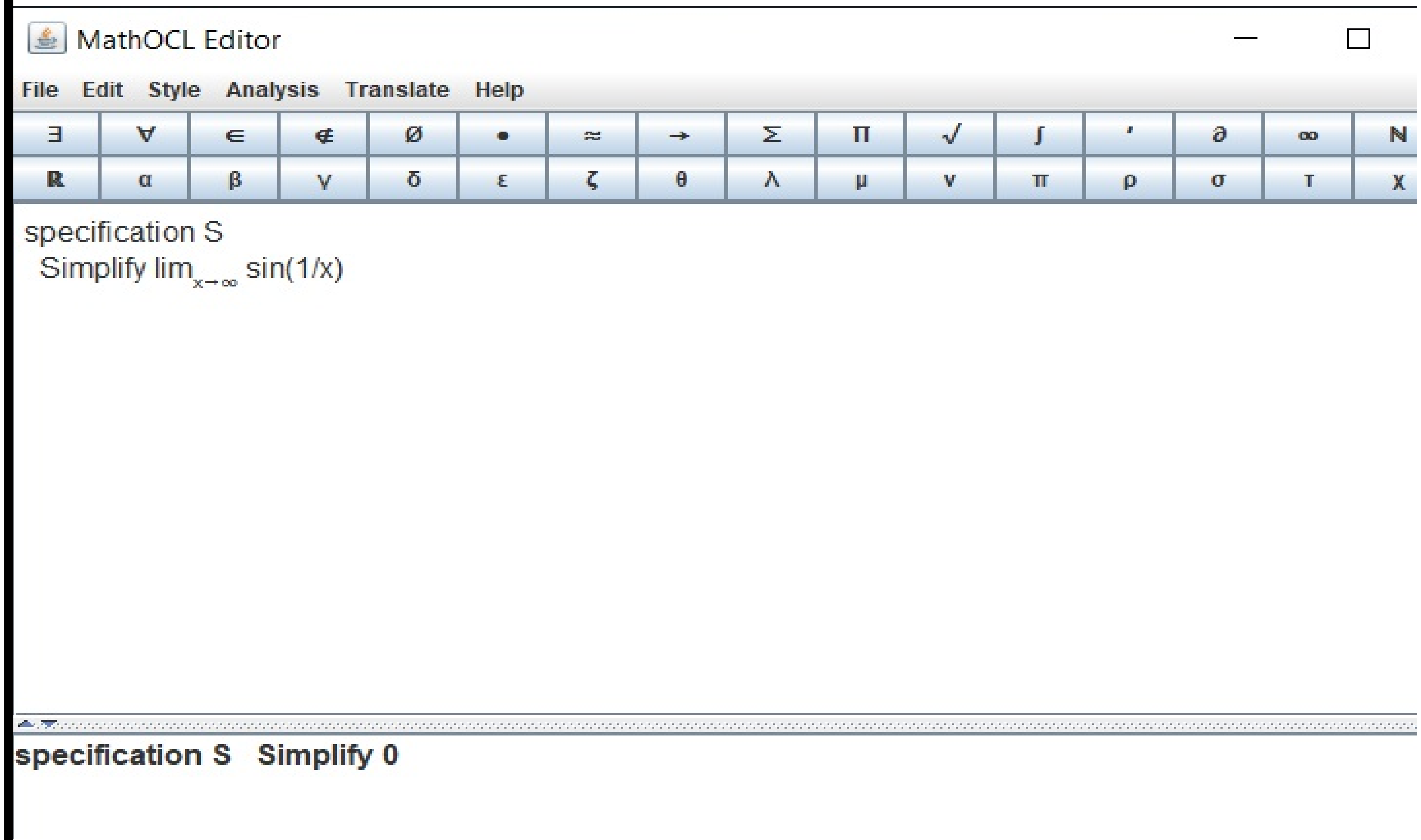
Construct	*Explanation*
Define v	Introduce variable v
Define $v = expr$	Define v's value as $expr$
Define $v = instr$	Define v's value
	as result of $instr$
Define $v \sim D$	Define v as random
	variable from distribution D
Simplify $expr$	Simplify an expression
Solve $eqns$ for $vars$	Solve one/set 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.



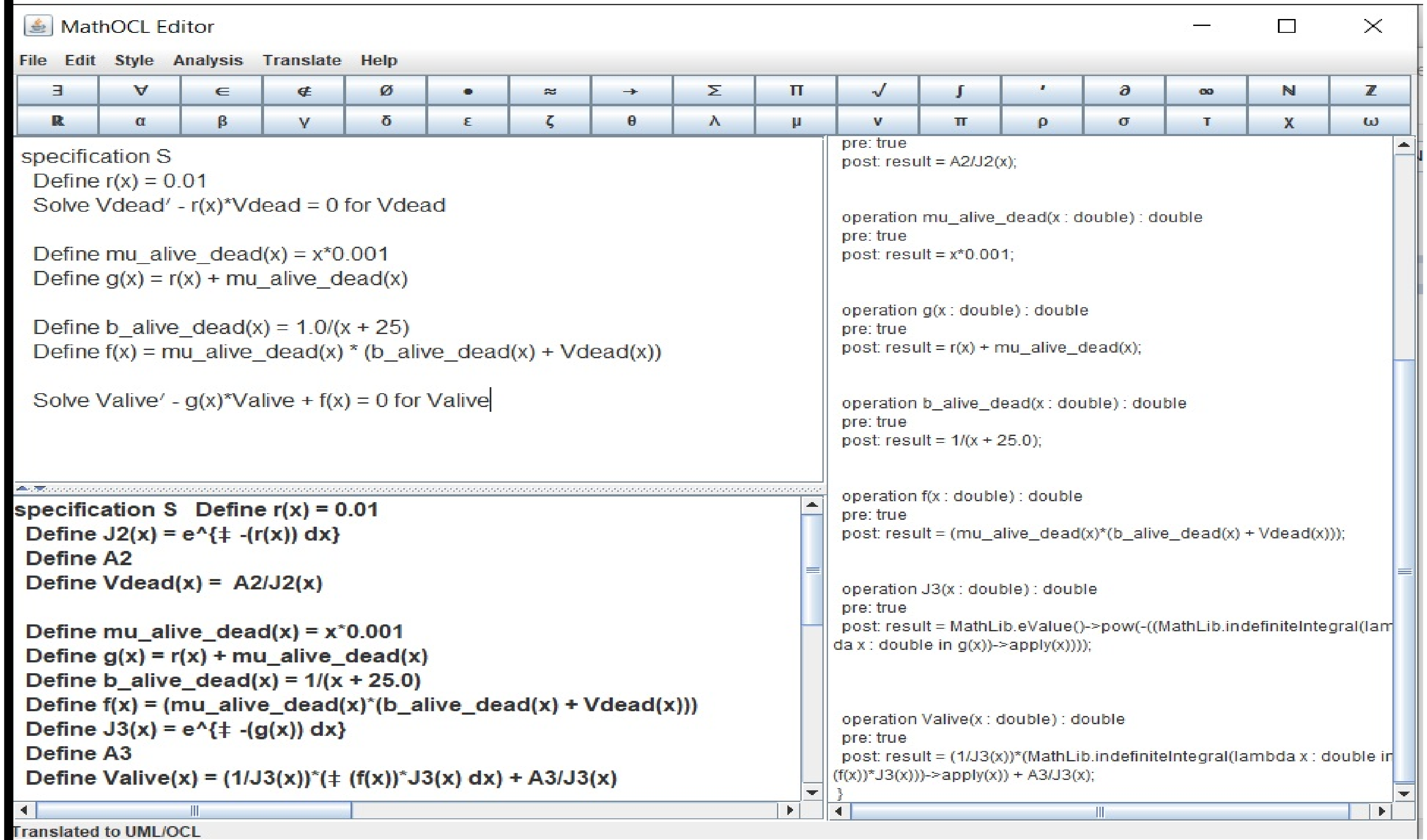
Example of random variable definition



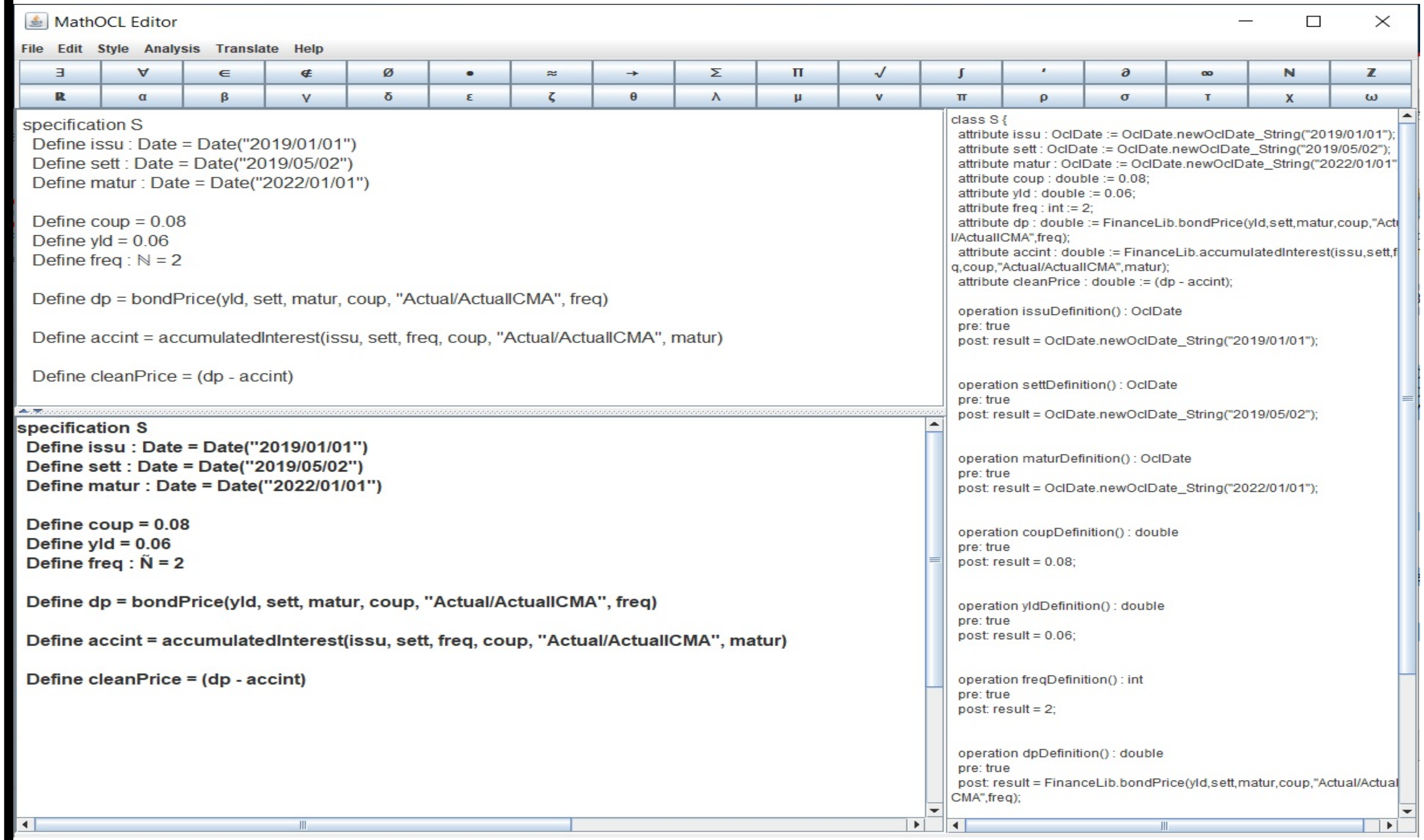
Example of limit evaluation

Case studies

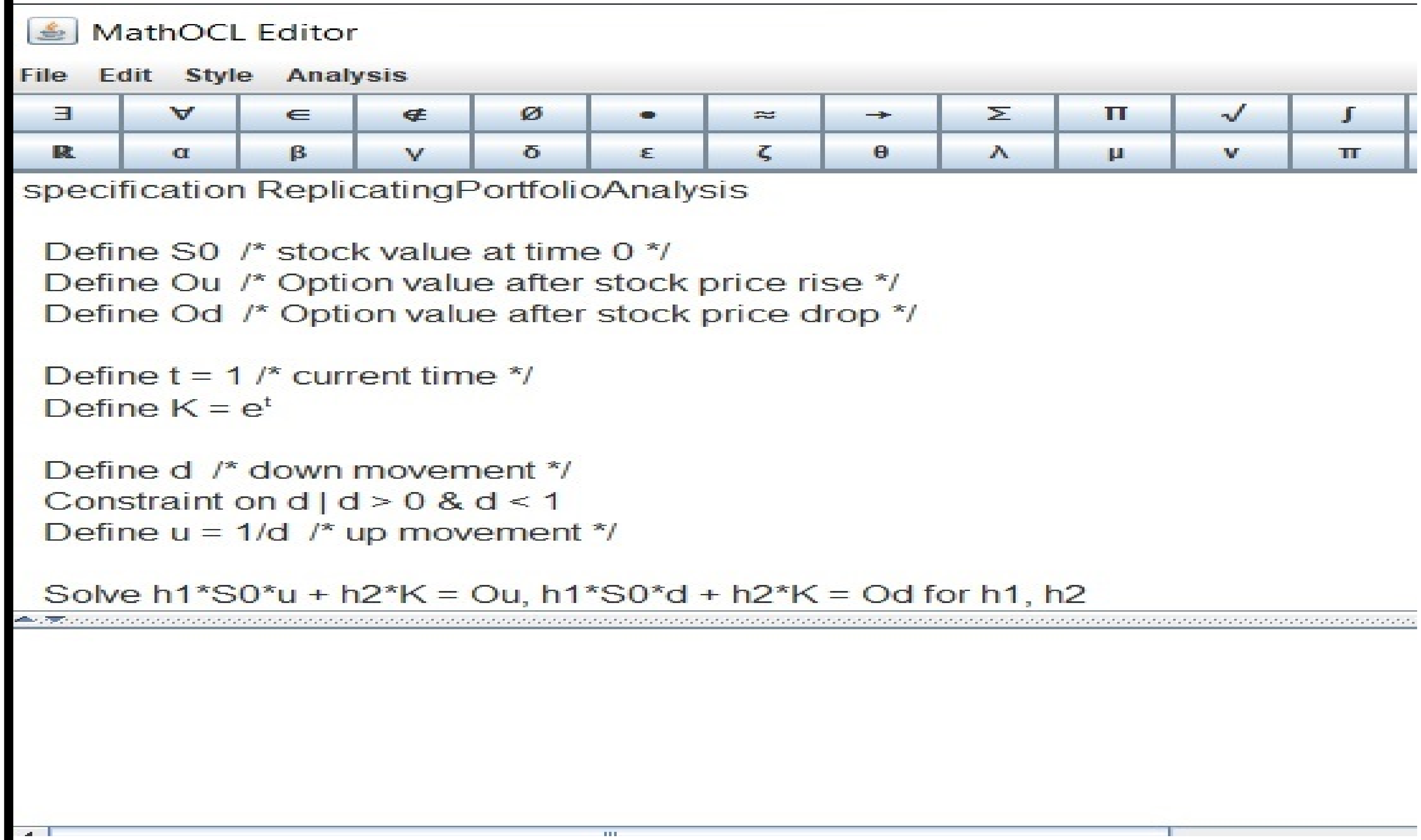
- Computational solution of Thiele's equations for life insurance
- Realistic computational solution of bond pricing taking into account different day-count conventions
- Validation of computational procedure for pricing share-based options, the Cox, Ross, Rubinstein Binomial model (CRR), against analytic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



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



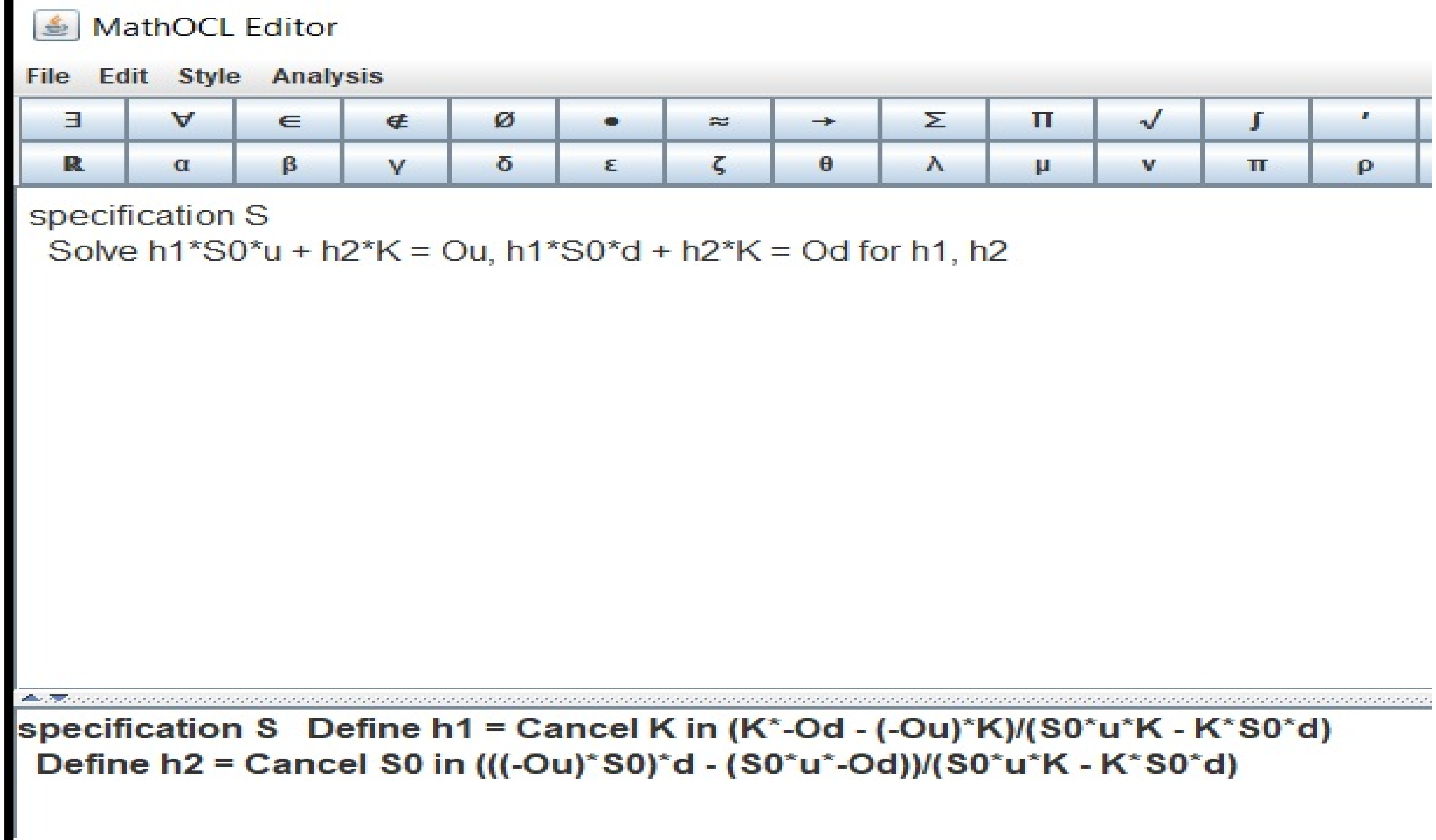
MathOCL model of bond valuation



MathOCL model of CRR validation

Analysis using MathOCL tools

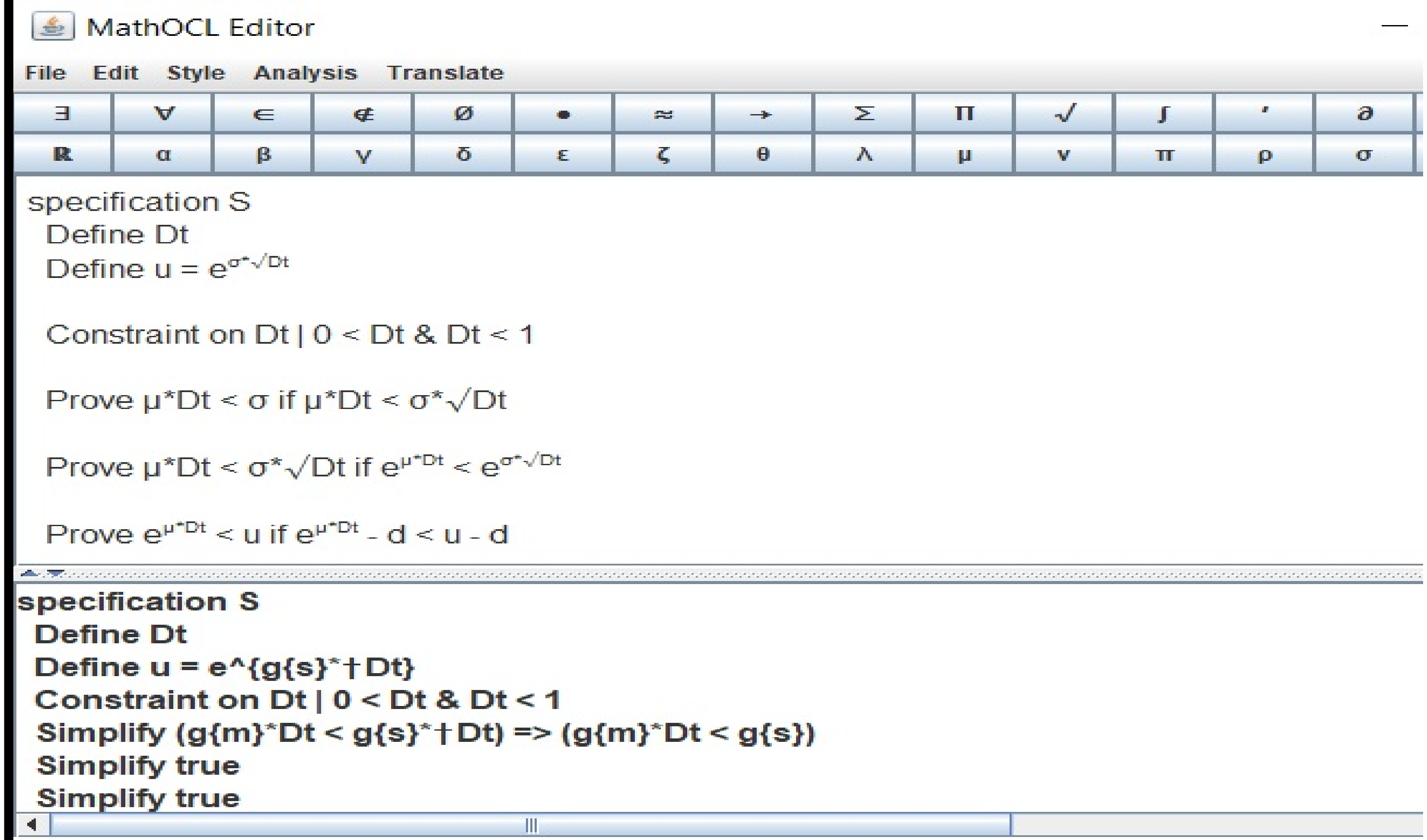
- Top panel gives model in math notation. Tool provides automated keyword guidance + syntax checking.
- The model equation can be automatically solved (symbolically) using MathOCL tools, & interactively simplified to produce more explicit version
- Lower panel provides interactive simplification
- Right-hand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



MathOCL 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.



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

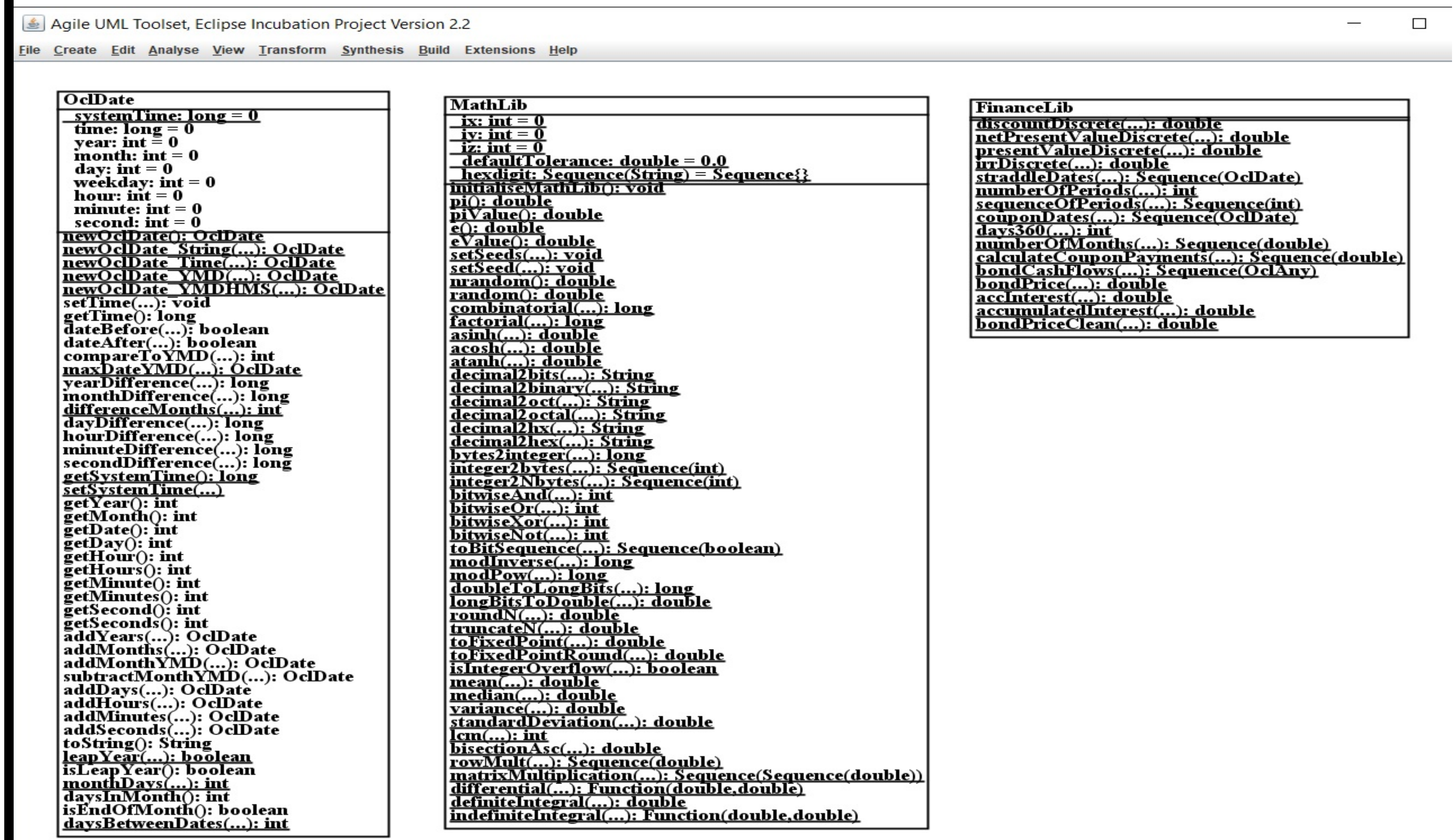
MathOCL Editor

File Edit Style Analysis Translate

```
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



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.