Nelumbo



Logic Meta Language

- Fast logic Reasoning
- Define and parse Syntaxes
- Define and execute Semantics
- Define and run Tests
- IDE integration using LSP
- Lightweight Language Workbench

Requirements for Nelumbo

- Suitable for formalization of, for example, complex tax laws and clinical knowledge
- Semantically rich and proven consistent
- Direct support for predicate logic by reasoning over incomplete facts and falsehoods
- Fully declarative, hence, no over-specification
- Strongly typed for more consistency and extensibility
- Performant by binding variables based on navigating relations only
- Easily extensible and integrable with the use of native classes
- Full editor support in any IDE using the Language Server Protocol

Syntax

Types

```
Smart   :: Object
Living  :: Object
LLM     :: Smart
Person  :: Smart, Living
Male    :: Person
Female  :: Person
```

Patterns

```
Integer ::= <NUMBER>                          // 10
Integer ::= <Integer> - <Integer>  #40,       // 5-7
            <Integer> + <Integer>  #40,       // 5+7
                      - <Integer>  #80,       // -7
            fib(<Integer>)                    // fib(100)

Repetion    ::= { <(> <Integer> <,> , <)*> }  // {3,5,7}
Option      ::= <(> super <)?> fast           // fast, super fast
Alternation ::= <(> A <|> B <|> C <)>         // A, B, C
```

Semantics

Variables

```
Integer a, b, c
Person  x, y, z
```

Rules

```
a<=b     <=>  a<b | a=b

|a|=b    <=>  b=a   if a>=0,
              b=-a  if a<0

descendant(x)=z <=> child(x)=z |
                    E[y](descendant(x)=y & child(y)=z)
```

Running

Queries

```
a+11=21   ?     // [(a=10)][..]
```

Testing

```
a+11=21   ?   [(a=10)][..]
|a|=10    ?   [(a=-10),(a=10)][(a=0),..]
```

Proof

```
a+11=21
  a+11=21 <=> add(a,11,21)
    add(c,11,21) [(c=10)][..]
  c+11=21 [(c=10)][..]
a+11=21 [(a=10)][..]
```

Native Semantics

Declaration

```
private Boolean ::= add(<Integer>,<Integer>,<Integer>)
                    @org.modelingvalue.nelumbo.integers.Integers
```

Java Code

The logic is a `@NelumboMethod` whose name and parameter count match the functor (the preferred style; see `docs/guides/native-cookbook.md`). Each bound argument arrives as its typed `NInteger`, each unbound one as `null`:

```
@NelumboMethod
protected InferResult add(NInteger addend1, NInteger addend2, NInteger sum) {
   if (nrOfUnbound() > 1) {
       return unknown();
   }
   BigInteger a1 = addend1 == null ? null : addend1.value();
   BigInteger a2 = addend2 == null ? null : addend2.value();
   BigInteger s  = sum     == null ? null : sum.value();
```

```
    if (a1 != null && a2 != null) {
        BigInteger r = a1.add(a2);
        if (s != null) {
```

Fibonacci Example

```
Integer ::= fib(<Integer>)

Integer n, f

fib(n)=f <=> f=n                    if n<=1,
             f=fib(n-1)+fib(n-2) if n>1

fib(0)=f       ? [(f=0)][..]
fib(1)=f       ? [(f=1)][..]
fib(5)=f       ? [(f=5)][..]
fib(100)=f     ? [(f=36#22r8fozas3n8w3)][..]
fib(1000)=f    ? [(f=36#18nnysuavughau0hlk8avlvhvagwiagha77pdsgscn5hzwghgaws
```

Literals

```
Male    ::= Hendrik, Bernhard, Claus, Willem
Female  ::= Wilhelmina, Juliana, Beatrix, Maxima, Amalia

Integer ::= <NUMBER>    @org.modelingvalue.nelumbo.integers.Integer
```

Functions

```
Integer ::= <Integer> -  <Integer>  #40,
            <Integer> +  <Integer>  #40

a+b=c  <=>  add(a,b,c)
a-b=c  <=>  add(c,b,a)
```

FactTypes and Facts

```
FactType  ::= pc(<Person>,<Person>)   // parent-child

// Facts
pc(Beatrix, Willem)
pc(Claus, Willem)
pc(Willem, Amalia)
pc(Maxima, Amalia)
```

Predicates

```
Boolean ::= <Integer> "<=" <Integer>  #30

a<=b    <=>  a<b | a=b
```

Operator	Meaning
`::`	Type
`:::=`	Pattern
`<=>`	Rule

Demo

LIVE

Plans

- Namespaces
- Generics (type arguments)
- Lists and Sets
- Language pattern transformations
- LSP (also on WEB)
- Reactive update execution semantics
- Deprecation and migration support
- ....

Contributing

- Open source: create tests and libraries
- Try to falsify the logic
- Help write scientific publications
- GitHub: https://github.com/ModelingValueGroup/nelumbo
- Email: wim.bast@gmail.com

Q&A