数学组建模文档（第二版）

一、语法设计篇（断言逻辑）

1.1 Preliminary

在一个特定的领域内，断言逻辑的语法结构可以被描述为如下的三元组：

$$\langle \mathcal{I}, \mathcal{C}, \mathcal{O} \rangle$$

Individual: 表示该领域的所有objects ---> (可以视作 Set Theory 的 **Elements**)

Concept: 表示具有共同性质的 Individual 组成的集合 ---> (可以视作 Set Theory 的 **Sets**)

Operator: 表示 Individuals 和 Concepts 之间的关系与联系 ---> (可以视作 Set Theory 的 **Functions**)

(注：Concepts 和 Operators 也可以被视为 Individuals)

一个 **Assertion** 由如下形式定义

$$a = b$$

其中 **a** 和 **b** 可以是 **Atomic Individuals** 或者 **Compound Individuals**；上面的表达式表明 **a** 和 **b** 指代相同的元素。

(注：**Compound Individuals** 指那些形如 O(a_{1},..a_{n}) 的 Individuals)

一个 **Knowledge base** 就是一系列 Assertions 的集合，Assertions 同样可以被视为 Individuals

1.2 From NL to AL

我们将形式化断言逻辑的任务定义为：在给定一个问题的 NL 表示下，求解：

- **Declaration List**: 一列用于声明变量的 Declarations
- **Fact List**: 一列用于表示问题的 Assertions
- **Query List**: 一列用于表示查询内容的 Terms
- **Proof List**: 一列用于表示待求证 Assertion的 Lists

(如下例所示)

当求的事实未知的时候

例1.

```
{
    "problem": "If $f(x)$ is a polynomial of degree 3, and $g(x)$ is a polynomial of degree 5, then what is the degree of polynomial $2f(x) + 4g(x)$?",
    "level": "Level 3",
    "type": "Algebra",
    "solution": "Let $f(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$ and $g(x) = b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$.  Then \\begin{align*}\n2f(x) + 4g(x) &= 2
    "Declarations": "P: Polynomial; Q: Polynomial",
    "Facts": "Get_PolyDegree(P) = 3; Get_PolyDegree(Q) = 5",
    "Query": "Get_PolyDegree(2 * P + 4 * Q)",
    "Proof": ""
},
```

例2.

```
{
    "problem": "Let  \\[ f(x) =\n\\begin{cases}\n-\\sqrt[3]x & \\text{if } x \\geq 0,\\\\\nx^2& \\text{if } x <0.\n\\end{cases}\n\\]Compute $f(f(f(f(512)))).$",
    "level": "Level 3",
    "type": "Algebra",
    "solution": "\\begin{align*}\nf(f(f(f(512))))\n&=f(f(f(-8)))\\\\\n&=f(f(64))\\\\\n&=f(-4)\\\\\n&=\\boxed{16}.\n\\end{align*}",
    "Declarations": "f: Function",
    "Facts": "Get_StepFunction_Expression(f, x >= 0) = - x ^ (1 /3); Get_StepFunction_Expression(f, x < 0) = x ^ 2",
    "Query": "Get_Function_Value(f, Get_Function_Composition(f, Get_Function_Composition(f, Get_Function_Composition(f, 512))))",
    "Proof": ""
},
```

当求的东西已知的时候

例3.

"Find $3x^2 y^2$ if x and y are [[integer]]s such that $y^2 + 3x^2 y^2 = 30x^2 + 517$. Show that it is 588."

"Declarations": "x: Integer; y: Integer",

"Facts": "y^2 + 3 * x^2* y^2 = 30 * x^2 + 517",

"Query": "",

"Proof": "3 * x^2 * y^2 = 588"

1.3 AL的基本语法（问题描述）

Basic Syntax

断言逻辑包含的基本语法包括以下内容：

- Sentence -> Assertion
- Assertion -> Term = Term
- Term -> Operator(Terms) | AtomicIndividual | (Assertion) | (Terms) | {Terms}
- Terms -> Term | Terms, Term
- AtomicIndividual -> Constant | Variable
- Constant -> 1 | 2 | True | False | pi | e ...
- Variable -> Parabola_C | Point_A ...
- Operator -> In | PointOnCurve
  | Radius | Length | Sin
  | Focus | Apex | ...

Variable Declaration

This should be clear. Variables declare in this way:

Variable(x) : Concept

For example, we could declare an integer variable like: "x: Integer"

（原则上所有未经声明的变量都是不合法的）

Syntactic Sugar

Symbol	Code	Comments
=	=	
<	<	

>	>	
⩽	<=	
⩾	>=	
+	+	
-	-	
x	*	
÷	/	
a^b	**,^	
∧	&	
....		

总结：（见文档）

1. 基本的算术运算符（加、减、乘、除、乘方）
2. 基本的比较关系 （大于、小于、等于、大于等于、小于等于）
3. 基本的逻辑符号（且、或、非、蕴含、存在、任意）
4. 集合的运算符号（交并补、包含、属于）

Some Tips

1. In the fact list, a sentence is either ~~an assertion (`... = ...`)~~ or a declaration (`... : ...`).
2. The annotation is not sensitive in order. It doesn't matter which translated sentence comes first, so do the declarations.
3. Operators usually start with verbs, for example："Is_OddFunction" ; "Get_Expression_Value". We use underscores "(_)" to separate these components.
4. Concepts are usually very very long words. And there are no underscores. For example, we have the concept "IrreduciblePolynomial"
5. When constructing a concept, we use 'a \\in b' to define that the set of concepts represented by a is a subset of b.

二、K12建模情况

建模原则：

对于概念，仅建模那些**特别有意义的基本数学概念**。自然语言概念如 （真值表） 无需建模

按照知识方程的语法规范，概念只会在声明部分使用；在 Fact List 和 Query List 统一使用具体的 individual

原则是声明此概念是否可以推理出有价值的信息；并且可以直接声明

对于算子，首先有一个统一的算子 Is_XXX (); 用于判断属于 父 Concept 的 individual 是否属于 子 Concept

用于描述概念性质的内容会被建模为算子，如“多项式的根”： Is_Polynomial_Root;

原则是判断该内容是否可以不依托 “某个概念” 而独立存在

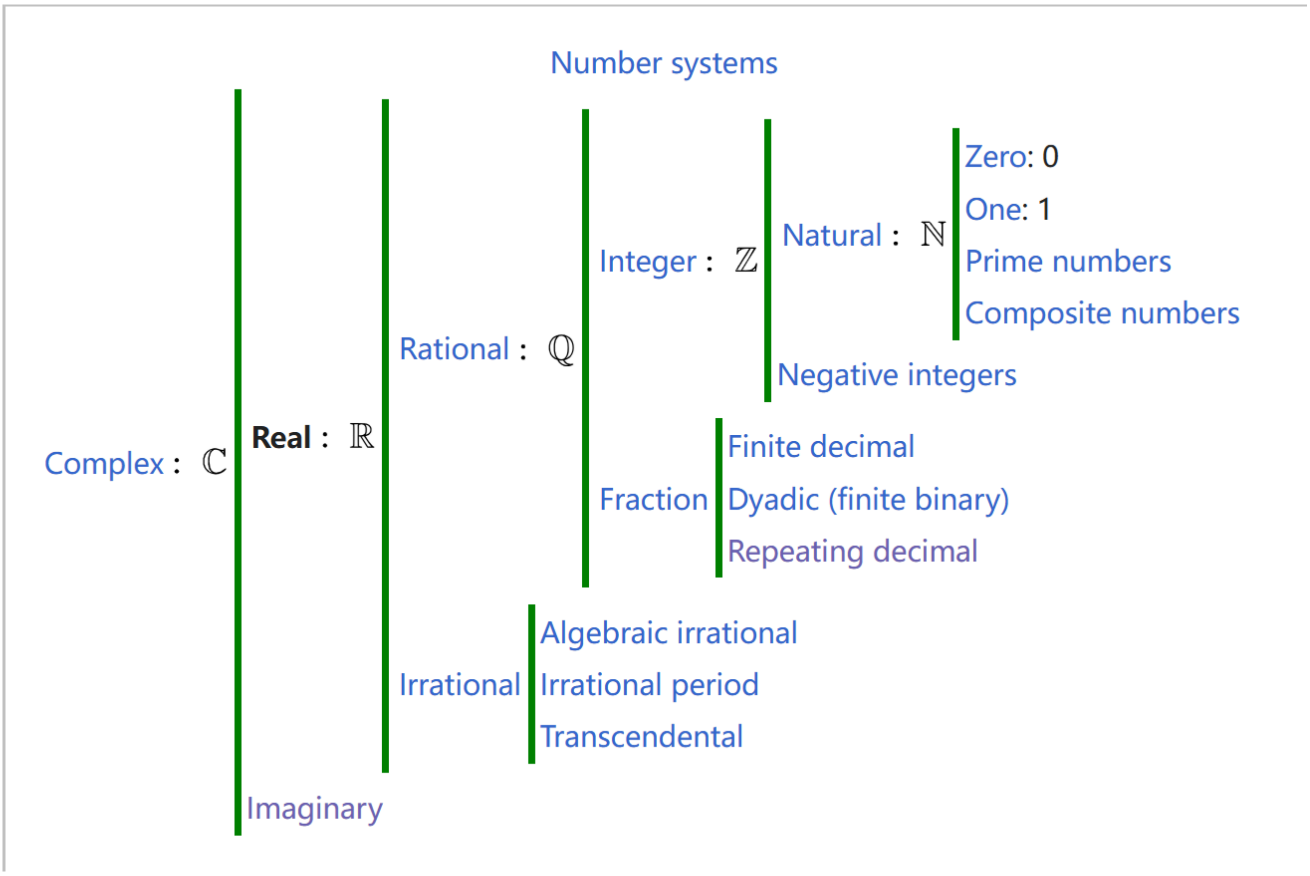
代数部分

概念

数系

包含了K12常见的所有数字类型，如下

实数、整数、自然数、有理数、无理数、复数



多项式

仅包含一个大的概念：Polynomial

其余的概念还有单项式（Monomial）、不可约多项式（IrreduciblePolynomial）

关于根等的性质都建模成了算子

对于两类特殊的多项式

2. QuadraticPolynomial: A polynomial whose expression has the form \(ax^2 + bx + c \), where \(a, b, c \) are constants, and \(a \neq 0 \).

Example: \(2x^2 + 3x - 5 \), \(x^2 - 4x + 4 \)

3. CubicPolynomial: A polynomial whose expression has the form \(ax^3 + bx^2 + cx + d \), where \(a, b, c, d \) are constants, and \(a \neq 0 \).

   Example: \(x^3 - 3x^2 + 2x - 7 \), \(4x^3 + x^2 - 2x + 1 \)

采取了单独的建模

函数

同样是仅包含一个大的概念：函数，此外如奇函数、偶函数等概念也放在了Concept中；

可以在declaration部分直接声明

```
# Function(函数)

1. Function: A mathematical relationship where each input (from the domain) corresponds to exactly one output (in the codomain).
   *Example:* \(f(x) = x^2\), where \(f(2) = 4\).

   2. ConstantFunction ∈ Function: A function that always outputs the same value, regardless of the input, represented as \(f(x) = c\).
   *Example:* \(f(x) = 5\).

   3. IdentityFunction ∈ Function: A function that maps every input to itself, represented as \(f(x) = x\).
   *Example:* \(f(x) = x\).

   4. PeriodicFunction ∈ Function: A function that repeats its values at regular intervals, such as trigonometric functions.
   *Example:* \(f(x) = \sin(x)\), with period \(2\pi\).

   5. IncreasingFunction ∈ Function: A function where the output increases as the input increases.
   *Example:* \(f(x) = 2x\).

   6. DecreasingFunction ∈ Function: A function where the output decreases as the input increases.
   *Example:* \(f(x) = -x\).

   7. StrictlyIncreasingFunction ∈ Function: A function where the output strictly increases as the input increases. That is, if \(x_1 < x_2\), then \(f(x_1) < f(x_2)\).
   *Example:* \(f(x) = x^3\), where for \(x_1 < x_2\), \(f(x_1) < f(x_2)\).

   8. StrictlyDecreasingFunction ∈ Function: A function where the output strictly decreases as the input increases. That is, if \(x_1 < x_2\), then \(f(x_1) > f(x_2)\).
   *Example:* \(f(x) = -x^3\), where for \(x_1 < x_2\), \(f(x_1) > f(x_2)\).

   9. OddFunction ∈ Function: A function that satisfies the condition \(f(-x) = -f(x)\) for all \(x\) in its domain.
   *Example:* \(f(x) = x^3\), where \(f(-2) = -8\) and \(f(2) = 8\).

   10. EvenFunction ∈ Function: A function that satisfies the condition \(f(-x) = f(x)\) for all \(x\) in its domain.
   *Example:* \(f(x) = x^2\), where \(f(-2) = 4\) and \(f(2) = 4\).
```

此外还包含两类特殊函数：

2. QuadraticFunction: A function that its expression has the form of ax^2 + bx + c

   Example: 2x^2 + 1; 3x^2 -4x + 1

3. CubicFunction: A function that its expression has the form of ax^3 + bx^2 + cx + d

   Example: 2x^3 + 1

集合

包含两大类：无序的集合（Set）和有序的集合（List）

```
# Sets and Logic(集合)

1. Set: A collection of distinct objects, considered as an object in its own right. ...

2. List: An ordered collection of elements of type α, implemented as a linked list.(Usually used in NumberTheory)
   *Example:* List(L) = [1, 3, 4, 1]
```

对于集合的大类，在概念上主要对具体的集合（空集、区间）和特殊性质的集合（空集、有限集）做了建模

```
1. Set: A collection of distinct objects, considered as an object in its own right.
   *Example:* \( A = \{1, 2, 3\} \) is a set of numbers.

  2. EmptySet ∈ Set: A set that contains no elements.
     *Example:* \( \emptyset \) or \( \{ \} \).

  3. FiniteSet ∈ Set: A set that contains a finite number of elements.
     *Example:* \( A = \{1, 2, 3\} \) is a finite set.

  4. InfiniteSet ∈ Set: A set that contains an infinite number of elements.
     *Example:* The set of all integers, \( \mathbb{Z} \), is an infinite set.

  5. OrderedSet ∈ Set: A set where the arrangement of elements matters.
     *Example:* \( A = \{1, 2, 3\} \) is ordered if we consider the set as a sequence, like \( A = (1, 2, 3) \).

  6. Interval ∈ Set: A useful set often used when learning function.
     *Example:* A: Interval, \( A = [1, 2] \) ; B: Interval, \( B = (3, 5] \).

  7. OpenInterval ∈ Interval: An open interval is a set of real numbers that includes all numbers between two endpoints, but not the endpoints.
     *Example:* A: OpenInterval, \( A = (1, 2) \) ; B: Interval, \( B = (3, 5) \).

  8. LeftClosedRightOpenInterval ∈ Interval: A left-closed, right-open interval includes the left endpoint but excludes the right endpoint.
     *Example:* A: LeftClosedRightOpenInterval, \( A = [1, 2) \) ; B: Interval, \( B = (3, 5] \).

  9. LeftOpenRightClosedInterval ∈ Interval: A left-open, right-closed interval excludes the left endpoint but includes the right endpoint.
     *Example:* A: LeftOpenRightClosedInterval, \( A = (1, 2] \) ; B: Interval, \( B = (3, 5] \).

  10. ClosedInterval ∈ Interval: A closed interval includes both endpoints.
     *Example:* A: ClosedInterval, \( A = [1, 2] \) ; B: Interval, \( B = [3, 5] \).

  11. RightOpenInterval ∈ Interval: A right-open interval includes all numbers less than the given bound.
     *Example:* A: RightOpenInterval, \( A = (-\infty, 2) \) ; B: Interval, \( B = (-\infty, 5) \).
```

对于 List 的算子建模可能尚不完备，目前仅在数论和数列中有使用到

数列

K12常见的等差等比数列；收敛数列、发散数列

```
# Sequence(数列)

1. Sequence \\in Set: A list of numbers arranged in a specific order, usually following a particular pattern or rule.
   *Example*: \(1, 2, 3, 4, 5, \dots\) is a sequence of natural numbers.

2. ArithmeticSequence \\in Sequence: A sequence where the difference between consecutive terms is constant.
  - General Form: \( a_n = a_1 + (n - 1) \cdot d \), where \(a_1\) is the first term, \(d\) is the common difference, and \(n\) is the term
  - *Example*: \(2, 5, 8, 11, \dots\) with common difference \(d = 3\).

3. GeometricSequence \\in Sequence: A sequence where each term is found by multiplying the previous term by a constant ratio.
  - General Form: \( a_n = a_1 \cdot r^{n-1} \), where \(a_1\) is the first term, \(r\) is the common ratio, and \(n\) is the term number.
  - *Example*: \(3, 6, 12, 24, \dots\) with common ratio \(r = 2\).

4. HarmonicSequence \\in Sequence: A sequence where each term is the reciprocal of a natural number, i.e., \(\frac{1}{2}, \frac{1}{3}, \dots \).
  - General Form: \( a_n = \frac{1}{n} \), where \(a_1\) is the first term, \(n\) is the term number.
  - *Example*: \(1, \frac{1}{2}, \frac{1}{3} \).

5. ConvergentSequence \\in Sequence
   structure ConvergentSequence (s : ℕ → ℝ) where
     (L : ℝ)                             -- 极限 L
     (lim : ∀ ε > 0, ∃ N : ℕ, ∀ n ≥ N, |s n - L| < ε)

6. DivergentSequence \\in Sequence
   structure DivergentSequence (s : ℕ → ℝ) where
     (diverges_to_infinity : ∀ M > 0, ∃ N : ℕ, ∀ n ≥ N, s n > M)
```

表达式（表达式、等式、不等式）

表达式（带变量的）；不等式、等式、匿名表达式

```
# Expression

1. Expression : A mathematical phrase consisting of numbers, variables, and operators (such as +, *) that represents a value or relationship.
   *Example:* \(3x + 5\) is an algebraic expression where \(x\) is a variable.

2. Equation \\in Prop : A statement where two expressions are set equal to each other.
   *Example:* \(2x + 3 = 7\) is an equation where we can solve for \(x\).

3. Inequation \\in Prop : A statement where two expressions are compared using inequality signs (>, <, ≥, ≤, ≠) instead of equality.
   *Example:* \(2x + 3 > 7\) is an inequation where we can solve for \(x\) to find values that satisfy the inequality.

4. LambdaExpression: A function definition expressed in lambda notation, mapping an input to an output using a concise functional syntax.
   Example: λ x : ℕ => x ^ 2 + 1 is a lambda expression that takes a natural number x and returns x ^ 2 + 1
```

逻辑部分

仅包含一个重要的概念：命题 Proposition

算子

见文档

可能遗漏的部分：

三角函数 & 向量（有初步定义，但需要和几何建模兼容）；

导数（? -- 简单）；

统计部分（多模态）-POI

数论部分

算子

数位（进制）

```
### Digits

-- 1. Get_Digit: [(Optional)Base: NaturalNumber](n: NaturalNumber) -> List

-- 2. Covert_Digit_To_Number: [(Optional)Base: NaturalNumber](L: List) -> NaturalNumber

-- 3. Get_DigitCount: [(Optional)Base: NaturalNumber](n: Integer) -> Integer

-- 4. Get_DigitProduct: [(Optional)Base: NaturalNumber](n: Integer) -> Integer

-- 5. Get_DigitSum: [(Optional)Base: NaturalNumber](n: Integer) -> Integer

-- 6. Is_PandigitalNumber: Is_PandigitalNumber(L: List) -> Boolean

-- 7. Is_PalindromeNumber: Is_PalindromeNumber(L: List) -> Boolean

-- 8. Get_Ones_Digit: [(Optional)Base: NaturalNumber]Get_Ones_Digit({n: Integer}) -> Integer
```

分数

```
### Fractions

-- 1. Get_FractionalPart: Get_FractionalPart(x: Real) -> Real

-- 2. Get_IntegerPart: Get_IntegerPart(x: Real) -> Integer

-- 3. Get_LeastCommonDenominator: Get_LeastCommonDenominator(f1: RationalNumbers, f2: RationalNumbers) -> Integer

-- 4. Get_Mediant: Get_Mediant(q1: RationalNumbers, q2: RationalNumbers) -> RationalNumbers

-- 5. UnitFraction: UnitFraction(q: RationalNumbers) -> Prop

-- 6. Is_ProperFraction: Is_ProperFraction(q: RationalNumbers) -> Prop

-- 7. Is_IrreducibleFraction: Is_IrreducibleFraction(q: RationalNumbers) -> Prop
```

素数

```
### Primes

-- 1. Is_Coprime: Is_Coprime({m: NaturalNumber}, {n: NaturalNumber}) -> Prop

-- 2. Is_Factor: Is_Factor({a: NaturalNumber}, {b: NaturalNumber}) -> Prop

-- 3. Get_GCD: Get_GCD({a: NaturalNumber}, {b: NaturalNumber}) -> NaturalNumber

-- 4. Get_LCM: Get_LCM({a: NaturalNumber}, {b: NaturalNumber}) -> NaturalNumber

-- 5. Is_PerfectSquare: Is_PerfectSquare(a: NaturalNumber) -> Prop

-- 6. Get_Remainder: Get_Remainder({a: NaturalNumber}, {b: NaturalNumber}) -> NaturalNumber

-- 7. Is_Prime: Is_Prime(a: NaturalNumber) -> Prop

-- 8. Get_SumOfSquares: Get_SumOfSquares(n: Integer) -> Integer

-- 9. Is_Twin_Prime: Is_Twin_Prime(p: Prime) -> Boolean

-- 10. Is_Factorial_Prime: Is_Factorial_Prime(p: Prime) -> Boolean

-- 11. Is_MersenneNumber: Is_MersenneNumber(N: NaturalNumber) -> Boolean

-- 12. Is_SinglyEvenNumber: Is_SinglyEvenNumber(N: NaturalNumber) -> Boolean

-- 13. Order: Order(a: NaturalNumber, N: NaturalNumber) -> Number
```

概率 & 组合 & 统计

该部分没有特殊建模，原因如下

{ x | P x}

1. 严格的概率定义需要用到实分析，在K12阶段超纲
2. 对于古典概率的情况，只需计算 集合的基数；二者做比就可。无需特殊定义
3. 组合的情况同理，组合部分没有特殊的概念和算子；完全继承于集合论
4. 统计部分，仅需要定义的是平均数、中位数、众数等概念。但该部分算子尚未实现（遗留）

三、待解决的问题

无法表示&解决的问题

因式分解；　（2x^11 + 3x^7 + 3x +1）题目：　求它的因式分解

lean的人：(x^2 - 1) = (x + 1)(x - 1)　(x ^ 3 + x + 1) = (ax^2 + 1 ..)
不可约

化简表达式；：最简

小数部分（无限循环小数）；

函数的定义域；

涉及函数图像的部分概念

歧义问题

多项式 & 函数

数列 & 函数

关于构造一个集合时候，存在全称量词和全局变量的区别（）