A Mathematician's Guide to Kleis

Kleis Language Project

Introduction

Kleis is a formally structured, algebra-oriented language designed to express mathematical objects and their axioms with precision. Unlike programming languages that treat structures as implementation artifacts, Kleis models mathematical practice directly:

- Abstract structures (groups, rings, fields)
- Parameterized structures (vector spaces over a field)
- Substructures (additive and multiplicative components of a ring)
- Explicit implementations of axioms for concrete models (e.g. $\mathbb{R}$ is a field)

This document introduces the central structural features of Kleis: `extends`, `over`, `implements`, `where`, and nested structures.

Each feature corresponds to a familiar concept in algebra or category theory.

1 The `extends` Construct: Inheritance of Structure

When mathematicians define a group, they implicitly note that every group is also a monoid. Formally:

$$(G, \cdot, e, ()^{-1}) \text{ is a group } \implies (G, \cdot, e) \text{ is a monoid.}$$

Kleis expresses this relationship directly:

```
structure Group(G) extends Monoid(G) {
    operation inv : G → G
    axiom left_inverse:
        (x : G). inv(x) • x = e
}
```

Thus:

- all operations of a monoid are *inherited*,
- all axioms of a monoid apply automatically to groups,
- new axioms extend the theory.

This mirrors the extension of algebraic theories in universal algebra.

2 The `over` Construct: Parameterized Structures

Many mathematical structures require a second structure as part of their definition. For example, a vector space is defined:

$$(V, +, \cdot) \quad \text{over a field } F.$$

In Kleis:

```
structure VectorSpace(V) over Field(F) {
    operation (+) : V × V → V
    operation (·) : F × V → V
}
```

This expresses that:

- the structure `VectorSpace(V)` is not meaningful alone,
- it depends on the existence of a field F,
- its axioms relate V and F.

This corresponds exactly to parameterized algebraic structures (modules, vector spaces, algebras) in standard mathematical literature.

3 The `implements` Construct: Concrete Mathematical Models

In mathematics, once an axiomatic structure is defined, one may declare:

$$\mathbb{R} \text{ is a field,} \qquad \mathbb{Z} \text{ is a ring,} \qquad \mathbb{R}^n \text{ is a vector space over } \mathbb{R}.$$

In Kleis:

```
implements Field() {
    element zero = 0
    element one  = 1
    operation (+) = builtin_add
    operation (×) = builtin_mul
}
```

This provides a *model* for the field axioms using the actual real numbers.

Thus `implements` corresponds to constructing an algebra over a given signature, or verifying that a concrete structure satisfies the axioms of an abstract one.

4 The `where` Construct: Logical Constraints

Mathematicians frequently impose side-conditions:

$$\text{Let } G \text{ be a group where } |G| \text{ is prime,}$$

$$\text{Define } \det(A) \text{ where } A \text{ is a square matrix.}$$

Kleis expresses such constraints through `where` clauses:

```
operation det : Matrix(n, n) →
```

implicitly includes the logical condition "where matrix is square."

More generally, Kleis may express:

```
implements SomeStructure(T)
    where Commutative(T)
{
    ...
}
```

This states:

This interpretation of the structure is valid only when T satisfies the given property.

Thus `where` acts as a predicate restricting admissible models, analogous to side-conditions in theorems.

5 Nested Structures: Substructures Within Structures

A ring contains two internal algebraic structures:

$$(R, +, 0) \text{ is an Abelian group,} \qquad (R, \cdot, 1) \text{ is a monoid.}$$

Kleis expresses this literally:

```
structure Ring(R) {
    structure additive : AbelianGroup(R) { ... }
    structure multiplicative : Monoid(R)  { ... }
}
```

Thus nested structures formalize the mathematical idea that an object may possess multiple compatible algebraic structures.

This reflects the treatment of algebraic objects in Bourbaki and universal algebra, where a single underlying set supports multiple signatures.

6 Summary Table

Kleis Construct	**Mathematical Meaning**	**Example**
`extends`	Inheritance of axioms	Group extends Monoid
`over`	Parameterized structure	Vector space over a field
`implements`	Concrete model of axioms	$\mathbb{R}$ is a field
`where`	Logical side-condition	$\det(A)$ where A is square
Nested structures	Internal algebraic components	Ring's additive/multiplicative parts

Conclusion

Kleis mirrors mathematical practice in a direct, structural way. Its constructs are not ad hoc language features but correspond precisely to the methods by which mathematicians build and relate algebraic objects.

Understanding Kleis as a language of abstract structures and their concrete models provides a natural bridge between symbolic algebra, formal logic, and executable mathematics.