Around the *StandardMath* Library.

Manuel Bronstein **Marc Moreno Maza**

January 13, 2003

A bit of history

1977 `scratchpad` / AXIOM [Jenks, Sutor, Davenport, Gianni, Trager, Bronstein, Barton, Williamson, Grabmeier, Gschnitzer, Moeller, ...]

1994 `axllib` [Broadbery, Dooley, Iglio, Morisson, Steinbach, Sutor, Watt]

1996 Σ^{it} [Bronstein + Mannhart, Mulders, Bernardin, Codutti, Prieto]

1997 *BasicMath* [Moreno Maza + Gómez Díaz, Broadbery, Tsikas, Dewar, Dunstan, Bronstein, Rioboo, ...]

1998 *Salli* [Bronstein]

1998 Π^{it} [Mannhart]

2001 *StandardMath* [Bronstein, Moreno Maza + ...]

The *AXIOM* algebraic libraries

- The main concerns: genericity, reusability.
- Lack of emphasis on specialized non-generic algorithms e.g. generic Gaussian elimination is used over small prime fields.
- No access to machine types and data-structures.
- Difficult to interface with external packages.
- But things were tried there for the first time!

The *Salli* library

Salli is a general library for programming with `Aldor` with no emphasis on symbolics or numerical computations.

- Numbers: machine integers and floats, long integers and floats (`Aldor` and GMP).
- Input/Output: strings, files, binary I/O-streams, text I/O-streams
- Data structures: arrays, lists, hash-tables, streams, ...
- Tools: debugging, profiling, exceptions.

The Σ^{it} library

Σ^{it} is a specialized library for Computer Algebra.

- Univariate polynomials and power series with generic domains and efficient domains (dense polynomials).
- Skew polynomials and linear operators.
- Linear Algebra with generic domains and specialized packages for the coefficient rings $\mathbb{Z}$, $\mathbb{F}_p$, $\mathbb{F}_p[x]$, ...
- Solvers for systems of linear ODEs or difference equations.
- Tools: expression trees, parsers, ...

The Π^{it} library

Π^{it} is a library for parallel computation.

- Architecture independent.
- Runs on a network of workstation (single processor or multiprocessor, shared or distributed memory).
- High level in order to hide low level programming (especially synchronization).

Around the *BasicMath* library

BasicMath is a general library for Computer Algebra with emphasis on polynomial system solving.

- *basiclib* with similar aim as *Salli*.
- *BasicMath* providing
  - linear algebra adpated from AXIOM
  - univariate and multivariate polynomials in the spirit of Σ^{it}
  - real algebraic numbers adpated from AXIOM
- *idealsolve* an interface with GB of J. C. Faugère.
- *radicalsolve* for triangular decompositions.

Around *StandardMath* today

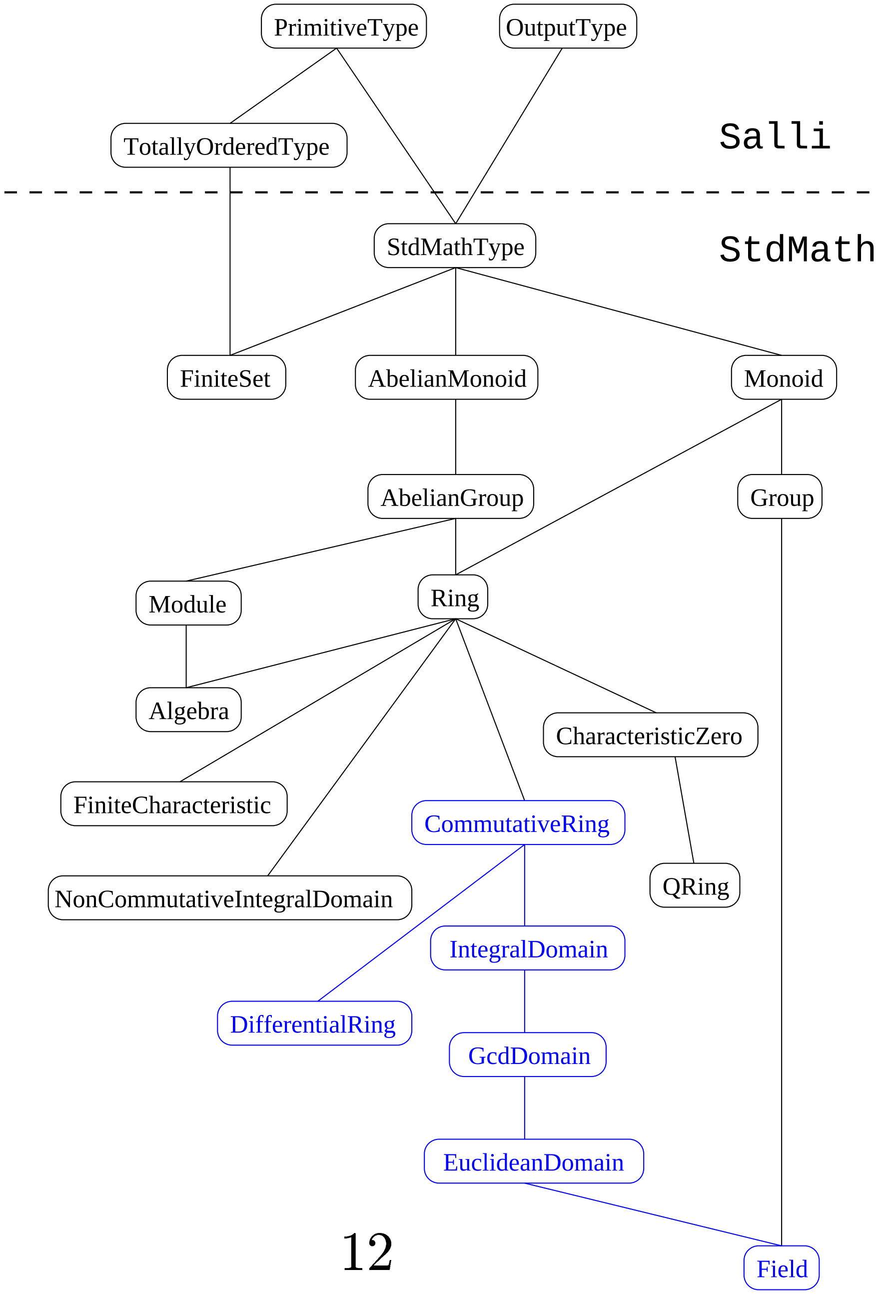
The *Salli* library today

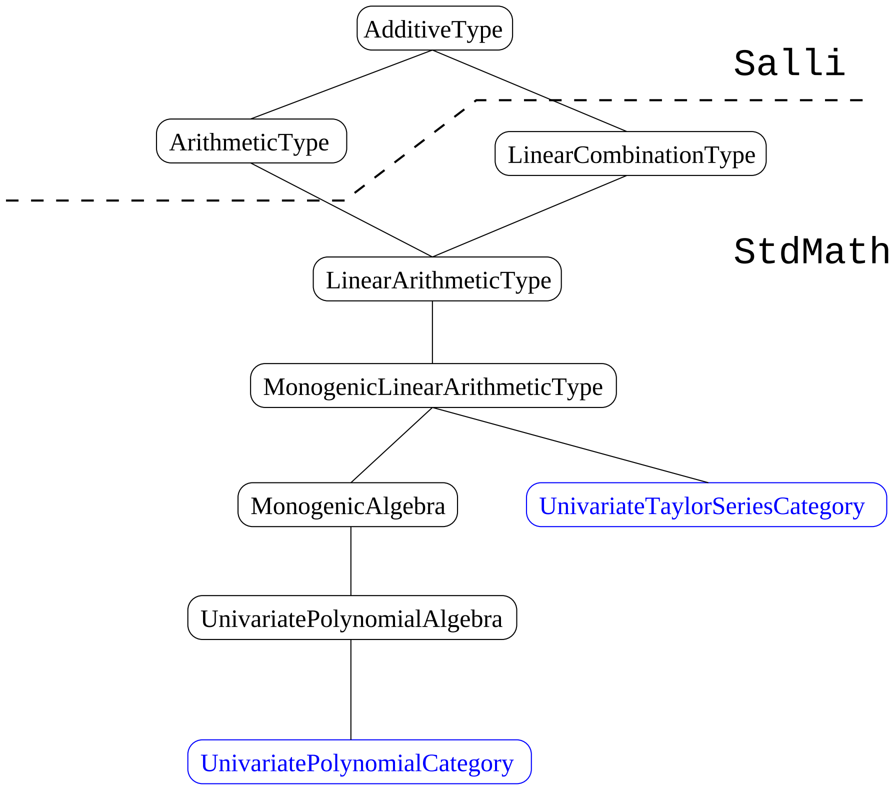
- The category hierarchy.
  - `InputType`, `OutputType`, `SerializableType`.
  - `AdditiveType`, `ArithmeticType`.
  - `AldorInteger`, `GMPInteger`.
- The data structures hierarchy.
  - `List`, `CheckingList`.
  - `PrimitiveArrayType`, `ArrayType`.

The *StandardMath* library

StandardMath is a library for Computer Algebra providing linear algebra and polynomial arithmetic.

- Linear Algebra extracted from Σ^{it}.
- Univariate polynomials and power series extracted from Σ^{it}.
- Multivariate polynomials in the spirit of *BasicMath* but recoded (category reorganisation, simplification, code factorization).
- Tools extracted from Σ^{it}.


PrimitiveType
OutputType
TotallyOrderedType
Salli
StdMathType
StdMath
FiniteSet
AbelianMonoid
Monoid
AbelianGroup
Group
Module
Ring
Algebra
CharacteristicZero
FiniteCharacteristic
CommutativeRing
QRing
NonCommutativeIntegralDomain
IntegralDomain
DifferentialRing
GcdDomain
EuclideanDomain
Field


AdditiveType
Salli
ArithmeticType
LinearCombinationType
StdMath
LinearArithmeticType
MonogenicLinearArithmeticType
MonogenicAlgebra
UnivariateTaylorSeriesCategory
UnivariatePolynomialAlgebra
UnivariatePolynomialCategory

Polynomials in *StandardMath*

- Allow coefficient sets that not rings, e.g. floats.

```
define FreeModule(R:Join(ArithmeticType, StdMathType)): Category ==
        FreeLinearCombinationType R with {
        if R has Ring then Module(R pretend Ring);
...............................................
define FreeLinearCombinationType(R:Join(ArithmeticType, StdMathType)):
  Category  == LinearCombinationType R with {
        map: R -> R -> % -> %;
        map!: R -> R -> % -> %;
................................................
define LinearCombinationType(R:Join(AdditiveType, StdMathType)):
        Category == Join(AdditiveType, StdMathType) with {
                *: (R, %) -> %;
                add!: (%, R, %) -> %;
                times!: (R, %) -> %;
```

....

- Distributed polynomials

```
define FiniteAbelianMonoidRing0(R: Join(ArithmeticType, StdMathType),
  V: VariableType, E: ExponentCategory(V)): Category ==
     Join(PolynomialRing0(R,V), IndexedFreeAlgebra(R,E)) with {
.....................................
}
```

- Recursive polynomials: a category

```
RecursiveMultivariatePolynomialCategory0(
   R:Join(ArithmeticType, StdMathType),
   V: Join(TotallyOrderedType, StdMathType)):
 Category == PolynomialRing(R,V) with {
        rank: % -> %;
        initial: % -> %;
.....................................
}
```

- Recursive polynomials: a domain

```
RecursiveMultivariatePolynomial0(_
  UP: (R: Join(ArithmeticType, StdMathType), avar: Symbol == new()
       ) -> UnivariatePolynomialCategory(R), _
  R :Ring, _%)
  V : VariableType):
 RecursiveMultivariatePolynomialCategory0(R,V) with {
       univariate: % -> UP %;
......................................
```

What remains to be done around *StandardMath* (at least)

- Rebuild *idealsolve* and *radicalsolve* on top of *StandardMath*.

$\Rightarrow$ *radicalsolve* will be renamed Δ.

- Modular methods in $\mathbb{Z}[x_1, \ldots, x_n]$ for $n > 1$.
- However *StandardMath* can easily invoque external packages.