BANs Library Documentation

Lorenzo Fiaschi

`AbstractAlgNum`: abstract type for ANs, it is child of *Number*

`SIZE`: constant which specifies the number of consecutive monosemia in a BAN

`Ban`: encoding for BANs, it is child of `AbstractAlgNum`

- Semantic: $\alpha^p P(\eta)$, $P(0) \neq 0$ but for zero
- Members:
  - `p`: power of the normal form, it is an integer
  - `num`: vector of *Reals* of length `SIZE` and as entries the coefficients of $P(\cdot)$ (in the reversed natural ordering)
- Constructor
  - `Ban(p::Int, num::Array{Real,1})`: instantiates a BAN with power `p` and coefficients `num`; it verifies the consistency of the inputs
  - `Ban(p::Int, num::Array{Real,1}, check::Bool)`: instantiates a BAN with power `p` and coefficients `num` without verifying the consistency of the inputs
  - `Ban(a::Ban)`: returns a copy of `a`
  - `Ban(x::Bool)`: returns 1 of type `Ban`
  - `Ban(x::Real)`: promote `x` to a `Ban`; undefined behavior if `x`= $\pm$`Inf`
- Unique representation of zero: `p`= 0 and `num`= **0**
- No unique representation for `NaN`: it is enough that at least one coefficient of $P(\cdot)$ is `Nan`
- If any entry of $P(\cdot)$ is $\pm$`Inf` that BAN is meaningless

α, η: constants representing the corresponding BANs

`print_ext(a::Ban)`: displays on screen the BAN `a` in the extended form

`println_ext(a::Ban)`: as above with also a new line at the end

`print_latex(a::Ban; precision::Integer=16, digits::Integer=2)`: displays the BAN `a` in a latex-oriented fashion (mathematical environment included)

`print_latex(a::Vector{T}: precision::Integer=16, digits::Integer=2) where T<:AbstractAlgNum`: the same as the previous function but for vectors of BANs

`print_latex(a::Matrix{T}: precision::Integer=16, digits::Integer=2) where T<:AbstractAlgNum`: the same as the previous function but for matrices of BANs

`standard_part(a::Ban)`: if the `a` is infinitesimal it returns 0; if `a` is infinite it returns $\pm$`Inf` depending on the sign of `a`; otherwise returns `num[1]`, i.e., the first coefficient of $P(\cdot)$

`degree(a::Ban)`: returns `p`

`degree(a::Real)`: returns 0 item `min_degree(a::Ban)`: if `a`=0 returns 0; otherwise returns the power of the smallest nonzero monosemium of `a`

`min_degree(a::Real)`: returns 0

`magnitude(a::Ban)`: returns the BAN encoding of α^p

`magnitude(a::Real)`: returns the BAN encoding of α^0

`principal(a::Ban)`: returns the BAN encoding of `num[1]`α^p

`principal(a::Ban)`: returns the BAN encoding of `a`, i.e., is the same as `Ban(a)`

`nextban(a::Ban, n::Int)`: returns a copy of `a` substituting `num[SIZE]` with `nextfloat(num[SIZE], n)`

`prevban(a::Ban, n::Int)`: returns a copy of `a` substituting `num[SIZE]` with `prevfloat(num[SIZE], n)`

`denoise(a::Ban, tol::Real)`: returns a copy of `a` (in normal form) where the entries of `num` whose absolute value is smaller than `tol` are set to 0

`denoise(a::AbstractVector{Ban}), tol::Real`: the same as the previous function but for all the entries of the vector `a`

`denoise(a::AbstractMatrix{Ban}), tol::Real`: the same as the previous function but for all the entries of the matrix `a`

`retrieve_infinitesimals(a::Ban, degree::Int)`: returns a BAN made of only the monosemia of `a` which have power smaller or equal than `degree`

`retrieve_infinitesimals(a::AbstractArray{Ban})`: the same as the previous function but returns an *Array* filled with the output of the previous function applied to all the entries of `a`

`isnan(a::Ban)`: returns `true` if either `p` or any monosemia coefficient in `a` is `NaN`, `false` otherwise

`isinf(a::Ban)` returns `true` if either `p` or any monosemia coefficient in `a` is $\pm$`Inf`, `false` otherwise

`isfinite(a::Ban)` it does the opposite of the previous function

`rand(::Ban)`: returns a positive random finite BAN whose monosemia coefficients are all sampled in [-1, 1] but the first one which is drawn from [0, 1]