Strongly Typed Numerical Computations

MATTHIEU MARTEL

University of Perpignan & Numalis
Laboratory of Mathematics and Physics (LAMPS)

`mxatthieu.martel@univ-perp.fr`

Introduction

Contribution : A type system for numerical accuracy

Type inference, dependent types, ML spirit

Implementation : **`Numl`**

Interest : Bug elimination, early debugging, accuracy information, code generation with optimal precision

Summary

- Introduction
- An overview of **`Numl`**
- The type system
- Type inference / implementation
- Conclusion / perspectives

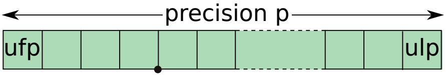
Types Embedding Accuracy

type $\texttt{real}\{s, u, p\}$

s sign of x, $s \in \{+, -, 0, *\}$

u ufp(x), i.e. *u*nit in the *f*irst *p*lace

p precision (number of bits of x)



$err \le 2^{ulp}$

```
> 1.234 ;;
- : real{+,0,53} = 1.2340000000000000 +/- 1.11022302463e-16

> 1.234{4} ;;
- : real{+,0,4} = 1.23 +/- 0.0625
```

Parameterized Types

```
> let f = fun x -> x + 1.0 ;;
val f : real{'a,'b,'c} -> real{<expr>,<expr>,<expr>} = <fun>

> verbose true ;;
- : unit = ()

> f ;;
- : real{'a,'b,'c} -> real(((max 'b 0) +_ (sigma+ 'a 1)),
                           ((max 'b 0) +_ (sigma+ 'a 1)),
                           ((((max 'b 0) +_ 1) -_ (max ('b -_ 'c) - 53))
                          -_ (iota ('b -_ 'c) - 53))) = <fun>
```

+, **−**, **⋆**, **/** real operators

+_, **−_**, **⋆_**, **/_** integer operators

Accuracy of the Results Explicited

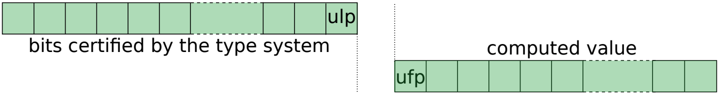
```
> let f = fun x -> x + 1.0 ;;
val f : real{'a,'b,'c} -> real{<expr>,<expr>,<expr>} = <fun>

> f 1.234 ;;
- : real{+,1,53} = 2.2340000000000000 +/- 2.22044604925e-16

> f 1.234{4} ;;
- : real{+,1,5} = 2.23 +/- 0.0625


> (1.0e15 + 1.0) - 1.0e15 ;;
- : real{+,50,54} = 1.0 +/- 0.0625

> (1.0e16 + 1.0) - 1.0e16 ;;
Error: The computed value has no significant digit. Its ufp is 0 but
the ulp of the certified value is 1
```



Recursivity and Subtyping

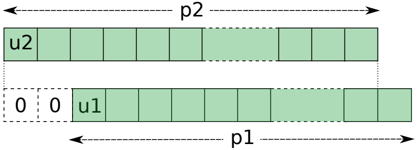
$$< \,:\, \alpha \longrightarrow \alpha \longrightarrow \mathtt{bool}$$

```
> let rec g x = if x < 1.0 then x else g (x * 0.07) ;;
val g : real{+,0,53} -> real{+,0,53} = <fun>

> g 1.0;;
- : real{+,0,53} = 0.070000000000000 +/- 1.11022302463e-16

> g 2.0 ;;
Error: This expression has type real{+,1,53} but an expression was
expected of type real{+,0,53}
```

$$real\{s_1, u_1, p_1\} \sqsubseteq real\{s_2, u_2, p_2\} \iff s_1 \preceq s_2, \quad u_1 \leq u_2, \quad p_1 \geq p_2 + u_1 - u_2$$



Monomorphic Comparisons

$$\texttt{<\{*,u,p\}} : \texttt{real\{*,u,p\}} \longrightarrow \texttt{real\{*,u,p\}} \longrightarrow \texttt{bool}$$

```
let rec g x = if x <{*,10,15} 1.0 then x else g (x * 0.07) ;;
val g : real{*,10,15} -> real{*,10,15} = <fun>

> g 2.0 ;;
- : real{*,10,15} = 0.14 +/- 0.03125

> g 456.7 ;;
- : real{*,10,15} = 0.16 +/- 0.03125

> g 4567.8 ;;
Error: This expression has type real{+,12,53} but an expression was
expected of type real{*,10,15}
```

Another Example

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$$

```
> let rec taylor x{-1,25} xn i n =
          if (i>n) then 0.0{*,10,20}
          else xn + (taylor x (x*xn) (i+_ 1) n) ;;
val taylor : real{*,-1,25} -> real{*,10,20} -> int -> int ->
real{*,10,20} = <fun>

> taylor 0.2 1.0 0 5;;
- : real{*,10,20} = 1.2499 +/- 0.0009765625
```

Yet Another Example

```
> let rec rectangle f a b h =
    if (a >= b) then
       0.0{*,30,40}
    else
       ((f a) * h) + (rectangle f (a + h) b h) ;;
val rectangle : real{<expr>,<expr>,<expr>} -> real{+,29,41} ->
real{<expr>,<expr>,<expr>} -> real{<expr>,<expr>,<expr>}

> let rec g x = x * x ;;
val g : real{'a,'b,'c} -> real{<expr>,<expr>,<expr>} = <fun>

> rectangle g 0.0 2.0 0.01 ;;
- : real{*,30,40} = 2.6867 +/- 0.0009765625
```

Remark : Method error not considered

Summary

- Introduction
- An overview of **`Numl`**
- The type system
- Type inference / implementation
- Conclusion / perspectives

The Type System

Dependent types

Standard typing rules

Accuracy encoded in the types of primitives

Only linear equations among integers in type expressions

Terms and Types

$$
\begin{array}{rcl}
\mathit{Expr} \ni e & ::= & \mathtt{x}\{s,u,p\} \in \mathbb{R}_{s,u,p} \mid \mathtt{i} \in \mathbb{Z} \mid \mathtt{b} \in \mathbb{B} \mid \mathtt{id} \in \mathrm{V} \\
& & \mid \mathtt{if}\ e_0\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \mid \lambda x.e \mid e_0\ e_1 \mid t \\
\mathit{Types} \ni t & ::= & \mid \mathtt{int} \mid \mathtt{bool} \mid \mathit{real}\{e_0, e_1, e_1\} \mid \alpha \mid \pi x : e_0.e_1
\end{array}
$$

Note : $t_0 \rightarrow t_1 \ \equiv\ \pi x : t_0.t_1$ (x not free in t_1)

Higher order constants : `+, -, *, /, +_, -_, *_, /_`

Typing Rules

$$\frac{}{\sigma \vdash \texttt{i} : \texttt{int}} \quad (\textsc{Int}) \qquad \frac{}{\sigma \vdash \texttt{b} : \texttt{bool}} \quad (\textsc{Bool})$$

$$\frac{\mathrm{ufp}(\texttt{x}) \leq u \quad \mathrm{sign}(\texttt{x}) \prec s}{\sigma \vdash \texttt{x\{s,u,p\}} : \mathit{real}\{s, u, p\}} \quad (\textsc{Real}) \qquad \frac{\texttt{id} : t \in \sigma}{\sigma \vdash \texttt{id} : t} \quad (\textsc{Var})$$

$$\frac{\sigma \vdash e_0 : \texttt{bool} \qquad \sigma \vdash e_1 : t_1 \qquad \sigma \vdash e_2 : t_2 \qquad t = t_1 \sqcup t_2}{\sigma \vdash \texttt{if } e_0 \texttt{ then } e_1 \texttt{ else } e_2 : t} \quad (\textsc{Cond})$$

$$\frac{\sigma, x : t_1 \vdash e : t_2}{\sigma \vdash \lambda x.e : \pi x : t_1.t_2} \quad (\textsc{Abs})$$

$$\frac{\sigma \vdash e_1 : \pi x : t_0.t_1 \qquad \sigma \vdash e_2 : t_2 \qquad t_2 \sqsubseteq t_0}{\sigma \vdash e_1\ e_2 : t_2[x \mapsto e_2]} \quad (\textsc{App})$$

Type of Functional Constants

$$
\begin{array}{lll}
+ & : & \pi \mathtt{s}_1 : \mathtt{int}, \mathtt{u}_1 : \mathtt{int}, \mathtt{p}_1 : \mathtt{int}, \mathtt{s}_2 : \mathtt{int}, \mathtt{u}_2 : \mathtt{int}, \mathtt{p}_2 : \mathtt{int}. \\
& & \quad \mathit{real}\{s_1, u_1, p_1\} \to \mathit{real}\{s_2, u_2, p_2\} \\
& & \quad \to \mathit{real}\{\mathcal{S}_+(\mathtt{s}_1, \mathtt{s}_2), \mathcal{U}_+(\mathtt{s}_1, \mathtt{u}_1, \mathtt{s}_2, \mathtt{u}_2), \mathcal{P}_+(\mathtt{u}_1, \mathtt{p}_1, \mathtt{u}_2, \mathtt{p}_2)\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{U}_+(\mathtt{s}_1, \mathtt{u}_1, \mathtt{s}_2, \mathtt{u}_2)) & = & \max(\mathtt{u}_1, \mathtt{u}_2) + \sigma_+(\mathtt{u}_1, \mathtt{u}_2) \\
\mathcal{P}_+(\mathtt{u}_1, \mathtt{p}_1, \mathtt{u}_2, \mathtt{p}_2) & = & \max(\mathtt{u}_1, \mathtt{u}_2) + \sigma_+(\mathtt{u}_1, \mathtt{u}_2) - \max(\mathtt{u}_1 - \mathtt{p}_1, \mathtt{u}_2 - \mathtt{p}_2) - \iota(\mathtt{u}_1 - \mathtt{p}_1, \mathtt{u}_2 - \mathtt{p}_2)
\end{array}
$$

$\mathcal{S}_+$

$\mathtt{s}_1 \backslash \mathtt{s}_2$	0	+	−	$\top$
0	0	+	−	$\top$
+	+	+	+ if $\mathtt{u}_1 < \mathtt{u}_2$ − if $\mathtt{u}_2 < \mathtt{u}_1$ $\top$ otherwise	$\top$
−	−	+ if $\mathtt{u}_2 < \mathtt{u}_1$ − if $\mathtt{u}_1 < \mathtt{u}_2$ $\top$ otherwise	−	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

$$
\iota(x, y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases}
$$

σ_+

	0	+	−	$\top$
0	0	0	0	0
+	0	1	0	1
−	0	0	1	1
$\top$	0	1	1	1

Summary

- Introduction
- An overview of **`Numl`**
- The type system
- Type inference / implementation
- Conclusion / perspectives

Type Inference / Unification

Type variables represented by references

Type **`real`** contains references

Standard type inference and unification

Partial evaluation of types (expression simplification)

Unification of reals relies on a call to **`Z3`**

Type Inference : Case of Application

$\texttt{typeCheck}\ (e_1\ e_2)\ \sigma =$

$t = \texttt{newTypeVar ()}\ ;\ t' = \texttt{newTypeVar ()}\ ;$

$t_1 = \texttt{typeCheck}\ e_1\ \sigma\ ;\ t_2 = \texttt{typeCheck}\ e2\ \sigma\ ;$

$t_1' = \texttt{simplify}\ t_1\ ;\ t_2' = \texttt{simplify}\ t_2\ ;$

$argType = \texttt{getArgType}\ t_1'\ \sigma\ ;$

$\texttt{assert}\ t_2' \sqsubseteq argType\ ;$ (relaxation)

$\texttt{unify}\ t_2'\ t\ ;$

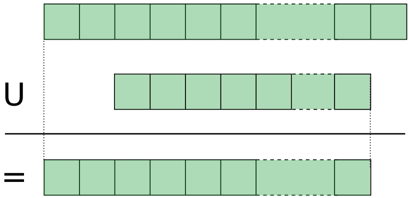
$\alpha = \texttt{newVar()}\ ;\ \texttt{unify}\ t_1'\ (\pi a : t.t')\ ;$

$\texttt{return}\ t'$

Unification of Reals

If no type variable :

Maximize ufp, minimize accuracy



Otherwise **solve** $e_{s_1} = e_{s_2} \wedge e_{u_1} = e_{u_2} \wedge e_{p_1} = e_{p_2}$

Example of Equations Sent to `z3`

```
> let rec taylor x{-1,25} xn i n =
          if (i>n) then 0.0{*,10,20}
          else xn + (taylor x (x*xn) (i+_ 1) n) ;;
val taylor : real{*,-1,25} -> real{*,10,20} -> int -> int ->
real{*,10,20} = <fun>
```

$$\begin{cases} \mathcal{S}_{+}(\alpha,\beta,\gamma,\delta) = * \\ \\ \max(\alpha,\beta) + \sigma_{+}(\gamma,\delta) = 10 \\ \\ \max(\alpha,\beta) + 1 - \mathit{max}(\alpha-\gamma,\beta-\delta) - \iota(\alpha-\gamma,\beta-\delta) = 20 \end{cases}$$

More Examples

```
let deriv f x h = ((f (x + h)) - (f x)) / h ;;
let g x = x*x - 5.0*x + 6.0 ;;
deriv g 1.0 0.1;;

- : real{*,5,52} = -2.900000000000000 +/- 7.1054273576e-15


let rec newton x{10,20} n f fprime =
     if (n=0) then x
     else let xnew = (x-((f x)/(fprime x)))
          in (newton xnew (n-_ 1) f fprime) ;;

real{*,10,20} -> int -> (real*,10,20 -> real0,0,21)
-> (real*,10,20 -> real0,-9,21) -> real*,10,20

let g x = (x*x) - (5.0*x) + 6.0 ;;
let gprime x = 2.0 * x - 5.0 ;;
 newton 9.0 5 g gprime ;;

- : real{*,10,20} = 3.0073 +/- 0.0009765625
```

Summary

- Introduction
- An overview of **`Numl`**
- The type system
- Type inference / implementation
- Conclusion / perspectives

Conclusion / Perspectives

Interpreter uses `GMP` (precision given by types)

Compiler not yet implemented

Improve typable terms by constraint relaxation :

- Improved types of primitives
- Modified type inference with global constraint solving

? ? ? ? ? ? ? ?

? ? ? ? ? ? ? ?

? ? **QUESTIONS** ? ?

? ? ? ? ? ? ? ?

? ? ? ? ? ? ? ?