Units in the OCaml typechecker

Jacques Garrigue and Dara Ly

Nagoya University, ENSTA ParisTech

February 21, 2017

This talk

- A minimalistic extension of the OCaml typechecker
- to allow checking of dimensions / units in calculations
- based on some well-known techniques for unification of dimension types
- but with a twist concerning equality and subsumption of type schemes in OCaml
- Presented as short paper at JFLA 2017

Dimension types

- Annotate numeric types with dimensions / units
- and propagate them through arithmetical operations
- so as to detect programming errors
- and also document the code

Example

```
# let d : <m> dfloat = create 17.
  and t : <s> dfloat = create 2. ;;
val d : <m> dfloat = 17.
val t : <s> dfloat = 2.
# let v = d /: t;;
val v : <m / s> dfloat = 8.5
# d +: t;;
          ^
Error: This expression has type <s> dfloat
       but an expression was expected of type
         <m> dfloat
       Type <s> is not compatible with type <m>
```

Design decisions

- Whether to handle both dimensions and units.
- Whether to allow conversions between units of the same dimension.
  Also allow non-linear conversions (temperatures, decibels) ?
- Whether to allow fractional exponents in dimensions.
  Can we unify `<'a ^ 2> dfloat` with `<m> dfloat` ?
  Seldom used, but possibly simpler.
- Which types should have dimensions ?

Previous implementations

ML-Kit extension for dimension types [Kennedy 1994]

- Only add dimensions to the type system, for the `real` type
- Units are constants (no need for coercions)
- Only allow integer exponents in dimensions

HimML [Goubault 1994]

- Both dimensions and units are allowed, for numeric types
- Automatic linear conversion between units
- Allow rational exponents in dimensions

Caml Light with dimensions [Blanchet 1996]

- Similar to Kennedy's approach, but for any type
- Also allows fractional exponents

Previous implementations

Unit types in F# [Kennedy 2009]

- First wide adoption of dimension types
- Only handle units, all conversions are done by hand
- Apparently restricted to integer exponents, but uses rational exponents internally

Typechecker plugin for GHC [Gundry 2015]

- Similar to F#
- Implemented as a plugin for the GHC constraint solver

Our approach

- A light weight implementation of dimension types
- Follow F# in handling only units, all conversions are by hand
- Only integer exponents
- Dimension types are just normal types (without any constants), which can be used as phantom type parameters to annotate other types
- No syntactic support outside of types

Example

```
module Dim : sig
  type 'a dfloat = private float
  val create : float ->  <'a> dfloat
  val (+:) :
      <'a> dfloat -> <'a> dfloat -> <'a> dfloat
  val ( *: ) :
      <'a> dfloat -> <'b> dfloat -> <'a * 'b> dfloat
  val ( /: ) :
      <'a> dfloat -> <'b> dfloat -> <'a / 'b> dfloat
  val inv :  <'a> dfloat -> <1 / 'a> dfloat
  val dsqrt : <'a ^ 2> dfloat -> <'a> dfloat
end = struct
  type 'a dfloat = float
  let create f = f
  let ( +: ) = ( +. )
  let ( *: ) = ( *. )
  let ( /: ) = ( /. )
  let inv f = 1. /. f
  let dsqrt = sqrt
end
```

Typing of units

Grammar

$$
\begin{array}{rll}
u ::= & 1 \mid u_1 * u_2 \mid u_1/u_2 \mid u_1\hat{}\,int & \\
\mid & \delta & \text{dimension variable} \\
\mid & b & \text{basic unit} \\
\tau ::= & \cdots \mid \langle u \rangle &
\end{array}
$$

Internal representation

```
type unit_desc =
  { ud_vars : (type_expr * int) list ;
    ud_base : (string * int) list }
```

Unit unification [Kennedy 1994]

$$\delta_1^{x_1} \cdot \delta_2^{x_2} \cdots \delta_m^{x_m} \cdot b_1^{y_1} \cdot b_2^{y_2} \cdots b_n^{y_n} = 1$$

If $m = 0$ and $n \neq 0 \Rightarrow$ **failure**

If $m \neq 0$, assume $|x_1|$ is the smallest exponent.

- If $\forall i, x_i \mod x_1 = 0$ and $\forall j, y_j \mod x_1 = 0$, the mgu is

$$\delta_1 \mapsto \delta_2^{-x_2/x_1} \cdots \delta_m^{-x_m/x_1} \cdot b_1^{-y_1/x_1} \cdots b_n^{-y_n/x_1}$$

- Otherwise, if $\exists i, x_i \mod x_1 \neq 0$, substitute

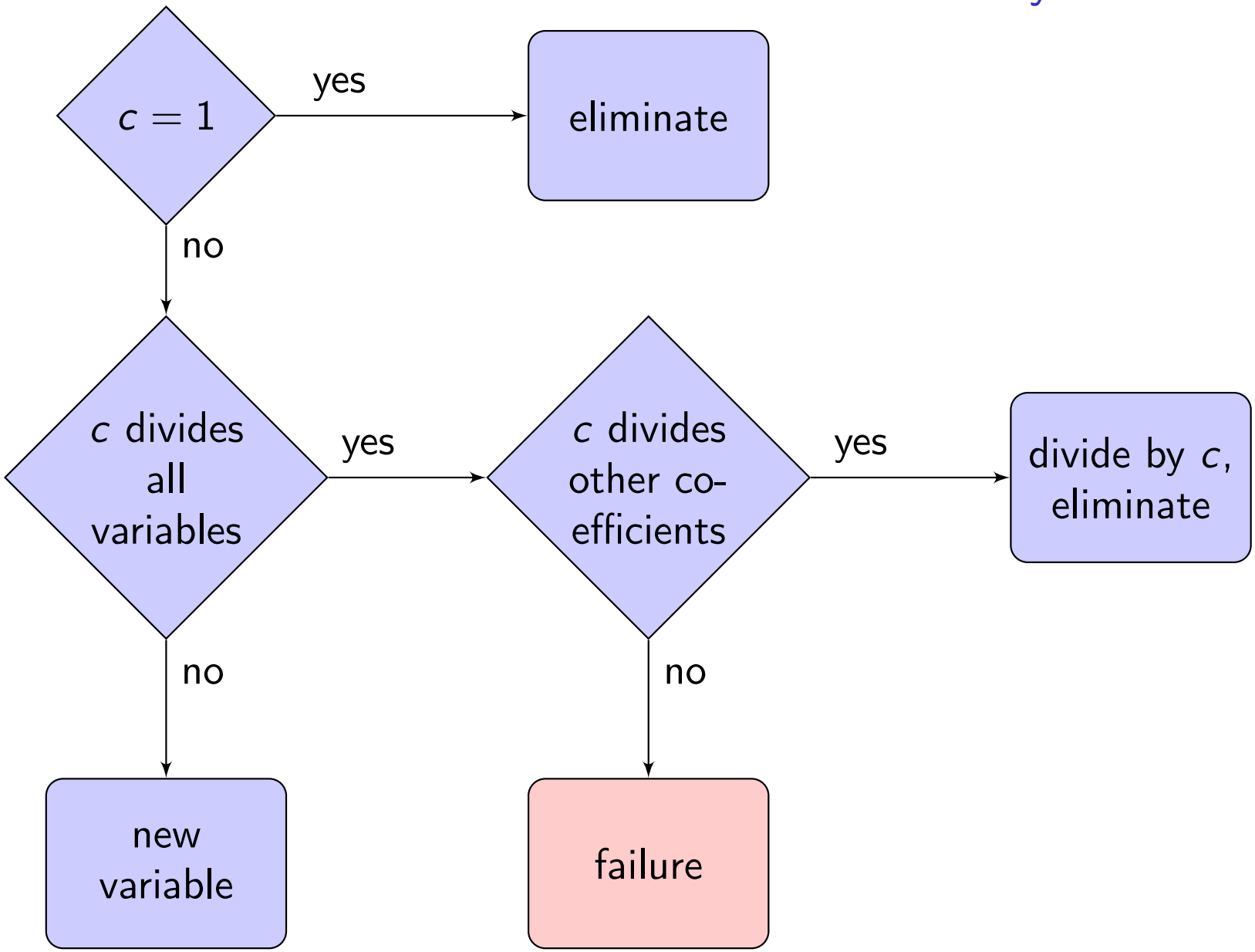
$$\delta_1 \mapsto \delta \cdot \delta_2^{-x_2/x_1} \cdots \delta_m^{-x_m/x_1} \cdot b_1^{-y_1/x_1} \cdots b_n^{-y_n/x_1}$$

  and solve the residual equation

$$\delta^{x_1} \cdot \delta_2^{x_2 \bmod x_1} \cdots \delta_m^{x_m \bmod x_1} \cdot b_1^{y_1 \bmod x_1} \cdots b_n^{y_n \bmod x_1} = 1$$

- Otherwise, if $\exists i, y_i \mod x_1 \neq 0 \Rightarrow$ **failure**

Summary


$c = 1$
yes
eliminate
no
c divides all variables
yes
c divides other co-efficients
yes
divide by c, eliminate
no
no
new variable
failure

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} \quad = \quad 1$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} \quad = \quad 1$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} = 1 \qquad \gamma = \alpha \cdot \beta^{-1} \cdot m^2 \cdot s^{-1}$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\begin{aligned} \alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} &= 1 \\ \gamma^2 \cdot \beta^{-1} \cdot m &= 1 \end{aligned} \qquad \gamma = \alpha \cdot \beta^{-1} \cdot m^2 \cdot s^{-1}$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\begin{aligned} \alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} &= 1 \qquad \gamma = \alpha \cdot \beta^{-1} \cdot m^2 \cdot s^{-1} \\ \gamma^2 \cdot \beta^{-1} \cdot m &= 1 \end{aligned}$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\begin{array}{rcl} \alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} & = & 1 \\ \gamma^2 \cdot \beta^{-1} \cdot m & = & 1 \end{array} \qquad \begin{array}{rcl} \gamma & = & \alpha \cdot \beta^{-1} \cdot m^2 \cdot s^{-1} \\ \beta & = & \gamma^2 \cdot m \end{array}$$

Unit unification

$$\alpha^2 \cdot m^5 = \beta^3 \cdot s^2$$

$$\begin{array}{rcl|rcl} \alpha^2 \cdot \beta^{-3} \cdot m^5 \cdot s^{-2} & = & 1 & \gamma & = & \alpha \cdot \beta^{-1} \cdot m^2 \cdot s^{-1} \\ \gamma^2 \cdot \beta^{-1} \cdot m & = & 1 & \beta & = & \gamma^2 \cdot m \end{array}$$

$$\begin{cases} \alpha = & \gamma^3 \cdot m^{-1} \cdot s \\ \beta = & \gamma^2 \cdot m \end{cases}$$

Type scheme equality and subsumption

Required for module subtyping

Equality $\forall \bar{\alpha}.\tau_1 = \forall \bar{\alpha}'.\tau_2 \Leftrightarrow \exists \rho : \bar{\alpha} \leftrightarrow \bar{\alpha}', \rho(\tau_1) = \tau_2$
used for comparing type definitions

Subsumption $\forall \bar{\alpha}.\tau_1 \leq \forall \bar{\alpha}'.\tau_2 \Leftrightarrow \exists \sigma : \bar{\alpha} \to \mathsf{Types}, \sigma(\tau_1) = \tau_2$
used for comparing value declarations

Implementation

- Subsumption can be checked by rigidification + unification
- Equality by checking subsumption in both directions
- However the OCaml implementation does not rely on unification, and does not allow incremental substitution

OCaml Implementation

- Both equality and subsumption build symbolic substitutions while comparing types structurally.
- However, some equivalent types have different structures

$$\forall\alpha.\langle\alpha \cdot m\rangle \to \langle\alpha\rangle = \forall\beta.\langle\beta\rangle \to \langle\beta \cdot m^{-1}\rangle$$

- We easily instrumented both operations to obtain a list of equations on dimension types, to be solved later.
- We must then solve the equations simultaneously, without substitution.
- This can be done by solving the corresponding system of linear equations.

Matrix representation

Subsumption example

$$\texttt{<'a> dfloat} \to \texttt{<'a * 'b> dfloat}$$
$$\leq$$
$$\texttt{<'c * m> dfloat} \to \texttt{<'d> dfloat}$$

The following equations are extracted, where γ and δ are rigid:

$$\begin{cases} \alpha = & \gamma.m \\ \alpha.\beta = & \delta \end{cases}$$

This is equivalent to the following linear equation system on the exponents, with variables the first 2 columns.

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 0 & -1 & 0 \end{array}$$

Integral Gaussian Elimination

or Gaussian elimination meets Euclid algorithm

- found in Knuth's Art of Computer Programming
- find integer solutions to an integer matrix
- also basis of Kennedy's substitution based approach
- the "pivot" is the non-null coefficient with the smallest absolute value

Elimination example

$$\texttt{<'a> dfloat} \to \texttt{<'a * 'b> dfloat}$$
$$=$$
$$\texttt{<'c * m> dfloat} \to \texttt{<'d> dfloat}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & & \gamma & \delta & & m \\ \left(\begin{array}{cc|cc|c} 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 0 & -1 & 0 \end{array}\right) \end{array}$$

Elimination example

$$\texttt{<'a> dfloat} \to \texttt{<'a * 'b> dfloat}$$
$$=$$
$$\texttt{<'c * m> dfloat} \to \texttt{<'d> dfloat}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 0 & -1 & 0 \end{array}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 1 & -1 & 1 \end{array}$$

Elimination example

$$\texttt{<'a> dfloat} \to \texttt{<'a * 'b> dfloat}$$
$$=$$
$$\texttt{<'c * m> dfloat} \to \texttt{<'d> dfloat}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 0 & -1 & 0 \end{array}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 1 & -1 & 1 \end{array} \quad \begin{cases} \alpha = \gamma \cdot m \\ \beta = \gamma^{-1} \cdot \delta \cdot m^{-1} \end{cases}$$

Elimination example

$$\texttt{<'a> dfloat} \to \texttt{<'a * 'b> dfloat}$$
$$=$$
$$\texttt{<'c * m> dfloat} \to \texttt{<'d> dfloat}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 0 & -1 & 0 \end{array}$$

$$\begin{array}{cc|cc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 1 & -1 & 1 \end{array} \quad \begin{cases} \alpha = \gamma \cdot m \\ \beta = \gamma^{-1} \cdot \delta \cdot m^{-1} \end{cases}$$

For equality, we also need to check again with columns exchanged. Here, trivially:

$$\begin{cases} \gamma = \alpha \cdot m^{-1} \\ \delta = \alpha \cdot \beta \end{cases}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma & \delta & m \\ \left(\begin{array}{cccc|c} 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}\right. \end{array}$$

$$\begin{array}{c} \begin{matrix} \alpha & \beta & \gamma & \delta \end{matrix} \\ \left(\begin{array}{cccc|c} 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}\right) \end{array} \qquad \gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}$$

$$\gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \hline 1 & 0 & 3 & 2 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}$$

$$\begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \quad m \\ \left(\begin{array}{cccc|c} 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}\right) \end{array} \qquad \gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma' & \delta \end{array} \quad m \\ \left(\begin{array}{cccc|c} 1 & 0 & 3 & 2 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}\right) \end{array}$$

$$\begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma' & \delta \end{array} \quad m \\ \left(\begin{array}{cccc|c} 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -18 & -17 & -4 \end{array}\right) \end{array}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array} \qquad \gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \hline 1 & 0 & 3 & 2 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \hline 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -18 & -17 & -4 \end{array} \qquad \beta' = \beta + 2\gamma' + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma & \delta & m \\ \hline 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array} \qquad \gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \hline 1 & 0 & 3 & 2 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \hline 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -18 & -17 & -4 \end{array} \qquad \beta' = \beta + 2\gamma' + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta' & \gamma' & \delta & m \\ \hline 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -4 & -3 & -4 \end{array}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma & \delta & m \\ 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array} \qquad \gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ 1 & 0 & 3 & 2 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -18 & -17 & -4 \end{array} \qquad \beta' = \beta + 2\gamma' + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta' & \gamma' & \delta & m \\ 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -4 & -3 & -4 \end{array} \qquad \delta' = 2\beta' + \gamma' + \delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma & \delta & m \\ \left(\begin{array}{cccc|c} 10 & 3 & 3 & 8 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}\right) & & & & \end{array} \qquad \gamma' = 3\alpha + \beta + \gamma + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \left(\begin{array}{cccc|c} 1 & 0 & 3 & 2 & 1 \\ 6 & -7 & 0 & -5 & 2 \end{array}\right) & & & & \end{array}$$

$$\begin{array}{cccc|c} \alpha & \beta & \gamma' & \delta & m \\ \left(\begin{array}{cccc|c} 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -18 & -17 & -4 \end{array}\right) & & & & \end{array} \qquad \beta' = \beta + 2\gamma' + 2\delta$$

$$\begin{array}{cccc|c} \alpha & \beta' & \gamma' & \delta & m \\ \left(\begin{array}{cccc|c} 1 & 0 & 3 & 2 & 1 \\ 0 & -7 & -4 & -3 & -4 \end{array}\right) & & & & \end{array} \qquad \delta' = 2\beta' + \gamma' + \delta$$

$$\begin{array}{cccc|c} \alpha & \beta' & \gamma' & \delta' & m \\ \left(\begin{array}{cccc|c} 1 & 0 & 3 & 2 & 1 \\ 0 & -1 & -1 & -3 & -4 \end{array}\right) & & & & \end{array}$$

Conclusion

Light weight and modular implementation

- Only 56 lines added or modified in ctype.ml
  (But units.ml is 425 lines long)
- Didn't extend the syntax tree
- Modifications to the parser and front-end are minimal

As plugin?

- Only 3 real entry points in `ctype.ml`
- What else can be done with these entry points?
- Should we just get rid of `moregeneral` and `eqtype` ?

Conclusion

Mergeable?

- Fair amount of interest from industry
- Impact is very small
- Not yet fully compatible with first class polymorphism

Play with it!

Code `https://github.com/tournemire/ocaml/tree/dim`

Paper *Des unités dans le typeur*

`http://www.math.nagoya-u.ac.jp/~garrigue/papers/`