MsML
A Proposal for a successor ML

David MacQueen
Computer Science, University of Chicago, Emeritus
ML Family Workshop 2025

What?

- MsML, my personal version of successor ML

  "***M**acQueen's own version of **s**uccessor **ML***"

  Defined mainly by *differences* from Standard ML

  This talk: description with minimal justification

  Intended to be implemented in the NewFrontEnd project

- An exercise in ***micro*** language design

  Not meant to change the *basic character* of ML

- **Not** concerned about *backward compatibility*

  https://github.com/dmacqueen/NewFrontEnd

  (file proposals/language.txt)

Why?

- New Front End (for SML/NJ) — personal project

  What language? (not necessarily *Standard* ML)

- Nostalgia for the good old days of language design
- Settling old scores (fixing old mistakes)

Character of (S)ML

- Computational Model: strict lambda calculus
- Statically typed

  Classic typed lambda calculus, plus

  Parametric polymorphism, with

  Automatic type inference, and

  Algebraic datatypes (union + recursion + parameterized)
- Functional module language

  structures, signatures, functors (non-recursive)

ML versus Haskell

- Purity

  mutable types (ref, array) vs state monoid

  exceptions vs ?

- Evaluation order

  strict vs lazy

- Interpreted types

  modules (structures and functors) vs type classes

  explicit propagation vs implicit (via type inference)

- Formal Definition

  Yes; No

Other popular(?) non-ML Ideas

- Object-oriented programming

  Objects, classes, inheritance

- Fancy (Fancier) types

  subtypes; GADTs; implicit coercions; overloading

- Fancy (Fancier) control structures

  prompts; continuations; concurrency; …

- Macros
- Programming + Verification languages; Agda, Lean, etc
- Fads or Trends (“aspect-oriented” programming?)

MsML: Some Major Changes - Modules

- **Delete *open* declarations [Core and Modules]**
- **Delete *equality polymorphism* (and thus equality type variables like `''a`) [Core and Modules]**
- Delete *include* specifications in signatures
- Add *higher-order functors* (*a la* SML/NJ 0.93 and onward from 1993)

MsML: Some More Major Changes - Core

- **Require explicit bindings of type variables occurring in declarations and expressions [Core]**
- Delete infix declarations [Core]
- Disallow static declarations (of types, modules) within expressions [Core]
- **Add a record *concatenation* or *overlay* expression construct [Core]**
- Delete "abstype" declarations and "while" expressions [Core]

Delete *open* declarations [Core and Modules]

- Bloated name spaces
- Complexity of static dependence analysis

Delete *open* declarations: Bloated name spaces

```
(* "old style" compiler/Elaborator/modules/moduleutil.sml *)
struct

(* opening 17 structures! *)
open Symbol SymPath InvPath ConvertPaths EntPath PathContext
     Access Types TypesUtil VarCon Bindings EntityEnv Stamp
     Modules ModuleId StaticEnv Primop

     ... name ...
     (* which opened structure did "name" come from? *)

end (* struct *)
```

Delete *open* declarations: Bloated name spaces

```
(* “modern style” *)
struct

local (* imports -- local short names for imported modules *)

   structure S  = Symbol
   structure SP = SymPath
   structure IP = InvPath
   ...
   structure T  = Types
   ...

in  (* body of structure *)

   ... T.name ... (* = Types.name *)

end (* imports local *)
end (* struct *)
```

Delete *equality polymorphism* [Core, Modules]

- Use *generic* (*structural*) equality instead (the old-fashioned way)

  Primitive types with primitive equality

  Propagate equality for products and sums

  Inductively define equality for (structural, recursive) datatypes

  No generic equality for function types and abstract types

- Need to explicitly pass equality for other (i.e. abstract) types

Explicit type variable bindings [Core]

- Principle: variables should be properly (explicitly) bound

(S)ML:

```
fun f x = ... 'a ... 'a ...  OR

fun f(x: 'a list) = ... 'a ... 'a ...
```

MsML:

```
fun f [X] (x: X list) = ... X ... X ...
```

Record overlay construct (fnl record update) [Core]

```
val r = {a = 3, b = true, c = "abc"}

val r' = r with {a = r.a + 1}
```

“**`with`**” is strict, but is not a function because we can’t express its type

Obligations

- New formal definition
  Revise existing Definition? (Archaic formalism)
  New, mechanized Defn (e.g. revise the TWELF Defn)?

- Revise source code of the language system implementation
  compiler, libraries, tools, …

Summary

- MsML: A variant of (Standard) ML, preserving its character

  nothing *essential* thrown out

  very little added

- Module system deserves a deeper rethink

  taking “recent” research into account

- How to formerly define this new variant? (and who will do it?)

- To be implemented by my New Front End project

Resources

- Repository: https://github.com/dmacqueen/NewFrontEnd

  proposals/language.txt — preliminary language changes

- Discussion: standardml.zulipchat.com, Channel # Successor ML

- DBM: dmacqueen@mac.com