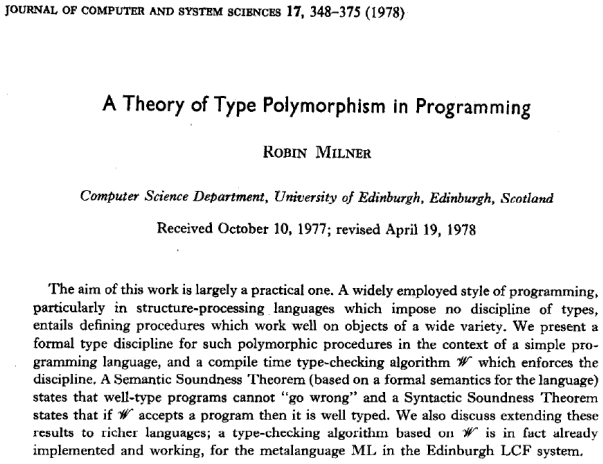
Introduction to Functional Programming in *OCaml*

Roberto Di Cosmo, Yann Régis-Gianas, Ralf Treinen

Week 0 - Sequence 2:

The *OCaml* language : a bit of history

The origins of the ML language family



JOURNAL OF COMPUTER AND SYSTEM SCIENCES 17, 348–375 (1978)

A Theory of Type Polymorphism in Programming

ROBIN MILNER

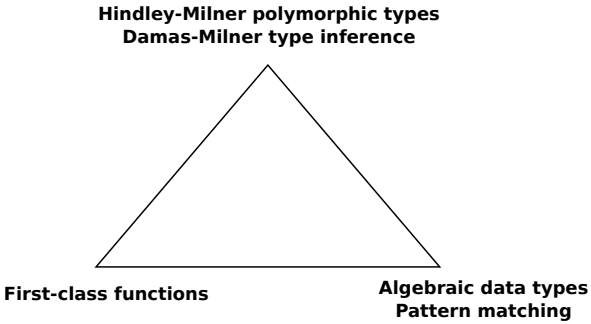
Computer Science Department, University of Edinburgh, Edinburgh, Scotland

Received October 10, 1977; revised April 19, 1978

The aim of this work is largely a practical one. A widely employed style of programming, particularly in structure-processing languages which impose no discipline of types, entails defining procedures which work well on objects of a wide variety. We present a formal type discipline for such polymorphic procedures in the context of a simple programming language, and a compile time type-checking algorithm $\mathscr{W}$ which enforces the discipline. A Semantic Soundness Theorem (based on a formal semantics for the language) states that well-type programs cannot "go wrong" and a Syntactic Soundness Theorem states that if $\mathscr{W}$ accepts a program then it is well typed. We also discuss extending these results to richer languages; a type-checking algorithm based on $\mathscr{W}$ is in fact already implemented and working, for the metalanguage ML in the Edinburgh LCF system.

OCaml belongs to the family of statically strongly typed functional programming languages, started by Sir Robin Milner's ML.

The core features of the ML family



The key additional ingredients

- type inference: *you* don't need to write them!
- pattern-matching: *life saving* data manipulation feature!

History of *OCaml*: the beginnings

1980: the Projet Formel at INRIA

Under the direction of Gérard Huet

- seminal work on mechanising mathematics
- uses Milner's ML language
- contributes to it (notably pattern matching)
- *and then starts developing its own*

History of *OCaml*: the name of the game

Raising the first *Caml*

- 1985 Guy Cousineau, Pierre-Louis Curien and Michel Mauny design the Categorical Abstract Machine
- 1987 Ascander Suarez releases the first *Caml* implementation
- 1988-1992 Michel Mauny and Pierre Weis nurture the *Caml* and make it grow

The system was impressive, but quite complex and needed professional workstations to run.

History of *OCaml*: a new engine

Early nineties: the age of *Caml Light*

1990-1991: Xavier Leroy creates the Zinc abstract machine, Damien Doligez writes a great memory manager, and the result is *Caml Light*

- small footprint: fits in a floppy
- portable: based on a bytecode interpreter
- efficient: runs on a PC

The Zinc machine is very different from the CAM, but the name stayed.

History of *OCaml*: getting up to speed

1995 *Caml Special Light* : compiler to native code, rich module system

1996 *Objective Caml* : efficient, elegant object oriented layer (Jérôme Vouillon and Didier Rémy)

2000 merge of Jacques Garrigue's extended *Objective Label* branch with polymorphic methods, labeled and optional arguments, and polymorphic variants

2011 the name is definitely changed to *OCaml*

Over the years, *OCaml* gained traction, and provides now a rich set of unique features.

Let's see what the *OCaml* users have to say.