TinyHM: Hindley-Milner type inference

How type inference in ML works

Tomas Petricek, Charles University

tomas@tomasp.net
@tomasp.net
https://tomasp.net
https://d3s.mff.cuni.cz/teaching/nprg077

Not a programming system!?

An important part of the ML experience

Makes ML practical and OCaml efficient

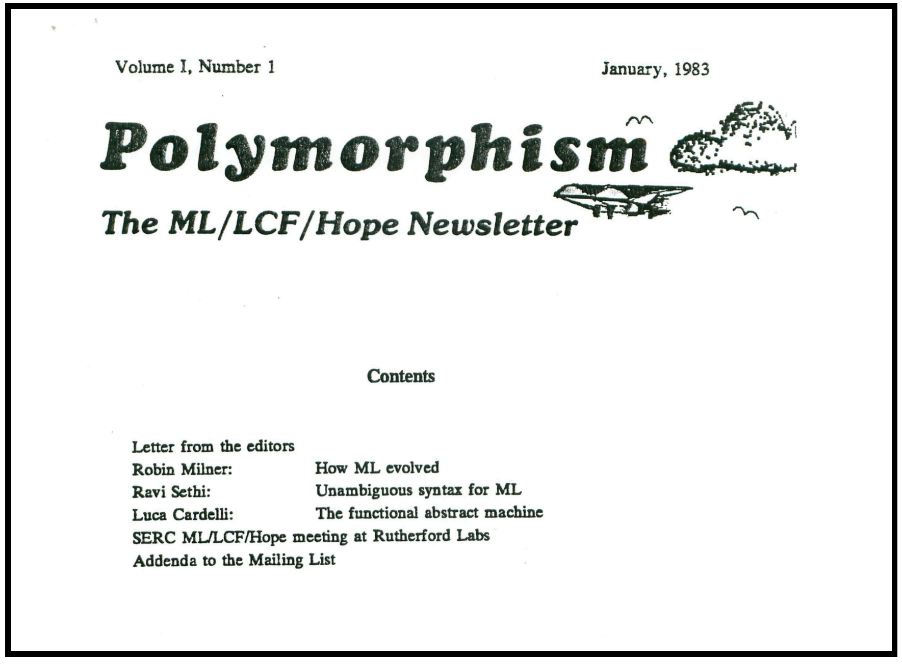
Learn some subtle aspects of F# type inference

Some discovered late through proofs and errors

Good example of constraint solving...

Important technique, used in Prolog & elsewhere

Charles University



Volume I, Number 1 January, 1983

Polymorphism

The ML/LCF/Hope Newsletter

Contents

Origins of ML

LCF theorem prover

ML used for writing meta-programs to generate proofs

Types used to ensure the validity of proofs

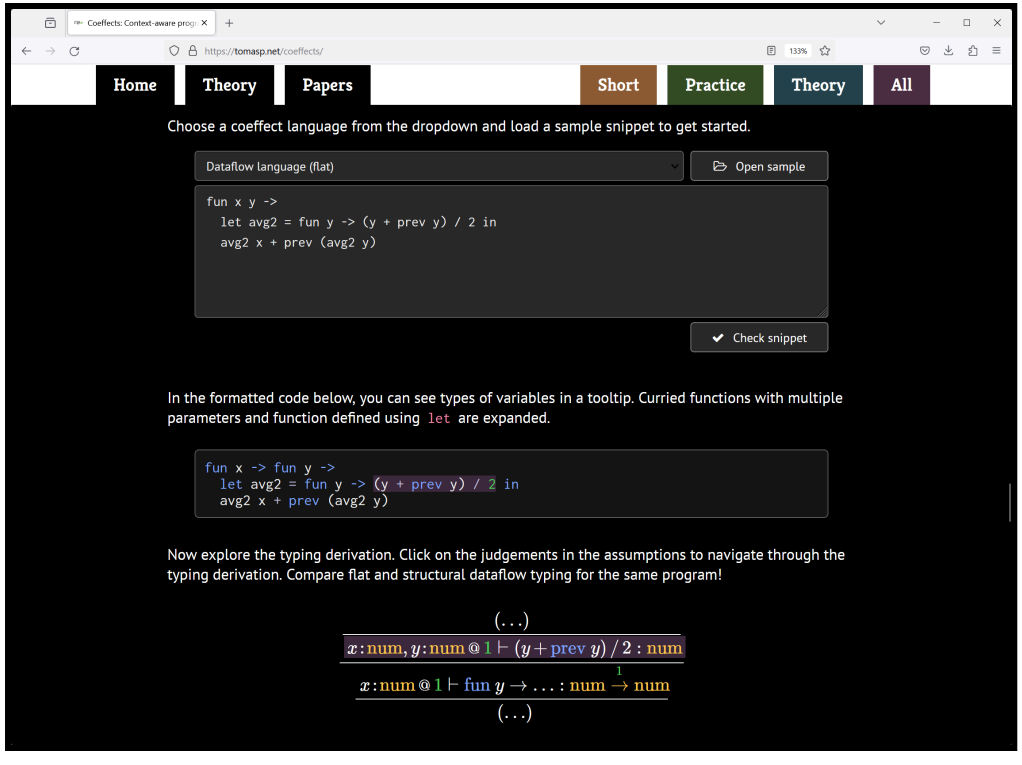
Charles University

Hindley-Milner

A brief history of type inference

- Hindley (1969) for Combinatory Logic
- Milner (1978) for ML with polymorphism
- Damas (1985) with formal analysis and proofs
- Since then - type classes, other extensions

Charles University



Demo Coeffects playground

Constraint solver code on GitHub

Charles University

ML type inference

How does F# figure out the types?

Demo

Basic type inference in F#

How F# type inference works

Constraint-based

- Collect & solve constraints
- No annotations needed for ML!

```
let twice f x = f (f x)
val twice: f: ('a -> 'a) -> x: 'a -> 'a

Full name: Inference.twice
```

Let polymorphism

- Infer generic type of let-bound functions

Limitations in ML and F#

- Value restriction for generic values
- Harder to deal with .NET objects

Charles University

Demo

Type inference limitations in F#