Introduction to Functional Programming in *OCaml*

Roberto Di Cosmo, Yann Régis-Gianas, Ralf Treinen

Week 5 - Sequence 0: Imperative features in *OCaml*

Functional vs. Imperative?

OCaml is a *functional* language...

You have learned to do a lot using *only*

- *immutable* data structures (no in-place modification)
- *identifiers* for values (no variables, no memory cells)
- *pure* functions (no side effects, no alteration of control)

With this fragment of the language,
we can do *pure* functional programming.

And go quite far! (Remember Church thesis...)

Functional and Imperative!

Sometimes, imperative features are useful

OCaml is a *functional* language ... with a range of *imperative features*

- *exceptions* model alteration of the flow of control
- operations that consume *input* and produce *output*
- *mutable* data structures for specific efficient algorithms
- *for* and *while loops* to describe iterations with side effects

It's your choice

In *OCaml*, you can be purely functional,
or program in a fully imperative style.

Your program will stay type safe anyway.

Exploring *OCaml*'s imperative side

This week

We will look at

1. *exceptions*
2. *input/output* and the *unit* type
3. *mutable* data structures and *reference cells*
4. *for* and *while loops*