# A core language for optimised compilation of algebraic effect handlers

Axel Faes
Tom Schrijvers
KULeuven

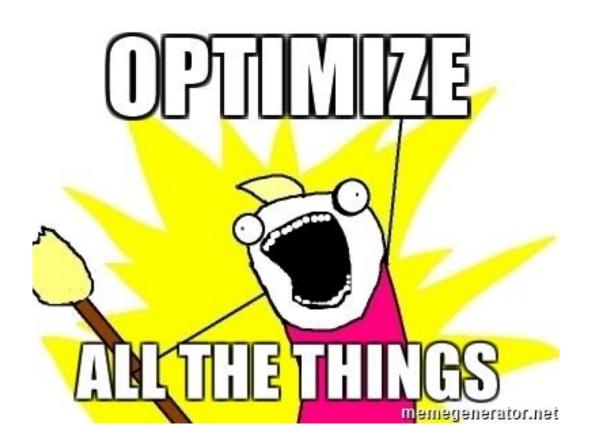
# What did I do?

# Algebraic effect handlers

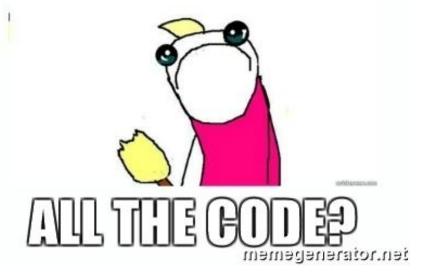
Exception handlers on steroids

**BUT** 

Implementations have runtime penalty



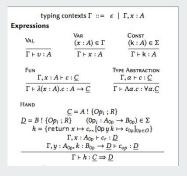
# OPTIMIZE



# New core language

Explicitly typed calculus with row-based effects

Designed a type system



# **Implementation**

Implemented in Eff

Written in OCaml

Rewrote the type inference engine

```
type_expr st {Untyped.term=expr; Untyped.location=loc} = type_plain_e
type_plain_expr st loc = function
 let ty_sch, st = get_var_scheme_env ~loc st x in
Ctor.var ~loc x ty_sch, st
Untyped.Const const ->
Ctor.const ~loc const. st
 let els = List.map (fun (e, ) -> e) (List.map (type_expr st) es)
Ctor.tuple ~loc els, st
Untyped.Record 1st ->
 let lst = List.map (fun (f, (e, _)) \rightarrow (f, e)) (Common.assoc_map (type)
Ctor.record ~loc lst, st
Untyped. Variant (lbl, e) ->
 let exp = Common.option_map (fun (e, _) → e) (Common.option_map (type
Ctor.variant ~loc (lbl, exp), st
Untyped.Lambda (p, c) ->
  et eff = infer effect ~
```

## **ICFP 2017**

**Extended Abstract** 

Poster presentation

Talk at main conference

#### Towards a core language with row-based effects for optimised compilation

Axel Faes student : undergraduate ACM Student Member: 2461936 Department of Computer Science KU Leuven

#### Abstract

Algebraic effects and handlers are a very active area of research. An important aspect is the development of an optimising compiler. Err is an Mt.-style language with support for effects and Genet het testelled for the optimising compiler. However, Err does not offer explicit typing, which makes it say for type checking bugs to be introduced during the construction of optimised compilation. This work presents a new orce language with row-based effects. The core language is explicitly typed in order to reduce bugs in the optimised compilation.

**Keywords** algebraic effect handler, row based effect, optimised compilation

#### 1 Introduction

Agebeaic effect handling is a very active area of research, implementations of algebraic effect annoters are becoming available. Because of this, improving performance is becoming the four of research, factor of research for the research focuses on speeding up the runtime performance. However, a runtime pernally still occur. This happens since handlers or continuations need to be repeatedly copied on the heap. Due to this, we are foliarly towards type-directed optimized compilation of algebraic effect handlers. We want to remove runtime pernally vocus.

In our ongoing, research towards type-directed optimized compilation, term revite rules and purity aware compilation committee the revite rules and purity aware compilation optimize away most handlers. Term rewrite rules use information of the type-di-effict system. Term rewrite rules use perform two types of actions. They remove handlers and apply efficts such that eventually the program does not contain any more handlers. Term rewrite rules can also change the syntactic structure in order to experim erop socializing for optimisations. Purity aware compilation identifies computations that are effectively pure and purifies them.

EFF, an ML-style language, is being used to develop an optimised compiler for algebraic effect handlers. EFF uses a type system based on subtyping [1]. As explained by Bauer Tom Schrijvers advisor Department of Computer Science KU Leuven tom.schrijvers@kuleuven.be

and Pretnar in [2], terms in Err do not contain any information about computational effects. This information has to be inferred using type inference algorithms. The lack of explicit type information makes outer-to-source transformations much more error-prone. Additionally, ensuring that a transformation does not break typesality becomes at transformation great, increase need to reconstruct types after each optimisation pass.

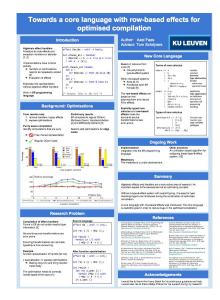
The current type system with subtyping becomes impractical since the typing information is not explicitly contend in each term. There are several solutions to make the type system more practical. It is possible to keep subtyping the use as unification based algorithm [3], implicit effect polymorphism can also be used [7]. The option that is evoded in this work, is to use a simple type-&-effect system based on row-polymorphism [4-6].

In this work, we present a simple explicitly-typed language that can serve as an intermediate language during compilation of Err, and allows for the development of type-preserving core-to-core transformations. Optimisation and term rewriting is done using this core language. This approach will case the development of an optimised compiler since typechecking becomes linear due to the explicit typing,

#### 2 Background The type-&-effect system that is used in Err is based on subtyping and dirty types [1].

Terms Figure 1 shows the two types of terms in EFF. There are values v and computations c. Computations are terms that can contain effects. Effects are denoted as operations Op which can be called.

Types: Figure 2 shows the types of Err. There are two main onto if types. There are (pure) types A. B and dirty types C, D. A dirty type is a pure type A tagged with a finite set of operations A, which we call dirt, that can be called. The type  $C \Rightarrow D$  is used for handlers because a handler takes an input computation C, handles the effects in this computation and outputs computation D as the result.



# Reflection

# **Choosing the honours project**

Do I explore multiple interests?

Focus on a single aspect (area of research)



# **Time Management**

Semester vs Summer

Deadlines

Deal with Summer feeling



# **ICFP 2017**

First big conference

Able to network with a lot of people

Attend talks about different topics

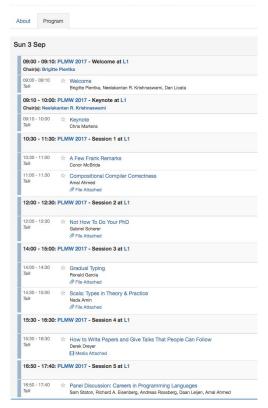


Specifically for beginning researchers

Talks about what (not) to do

Talks about writing papers

#### PLMW 2017 Programming Languages Mentoring Workshop 2017



# **Student Research Competition**

Practice how to explain work to an audience

Highlight: give talk at main conference

### HONOURS STUDENT AXEL FAES WINS BRONZE MODEL IN ACM SIGPLAN

Honours student Axel Faes wins the bronze medal in the undergraduate category of the ACM SIGPLAN Student Research Competition held at the International Conference on Functional Programming in Oxford, UK for his work "Towards a core language with row-based effects for optimised compilation" supervised by prof. Tom Schrijvers.



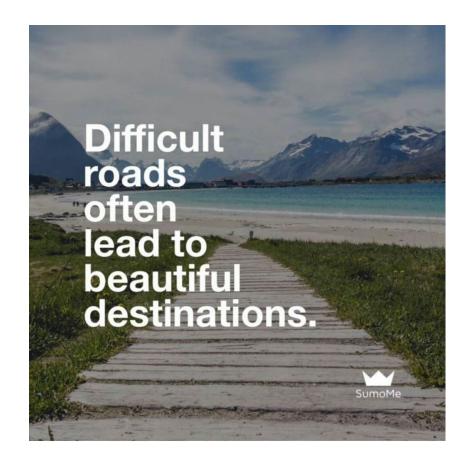
# **Looking back**

Scope of the work

Theoretical

Implementation

Grown as a junior researcher



# Conclusion