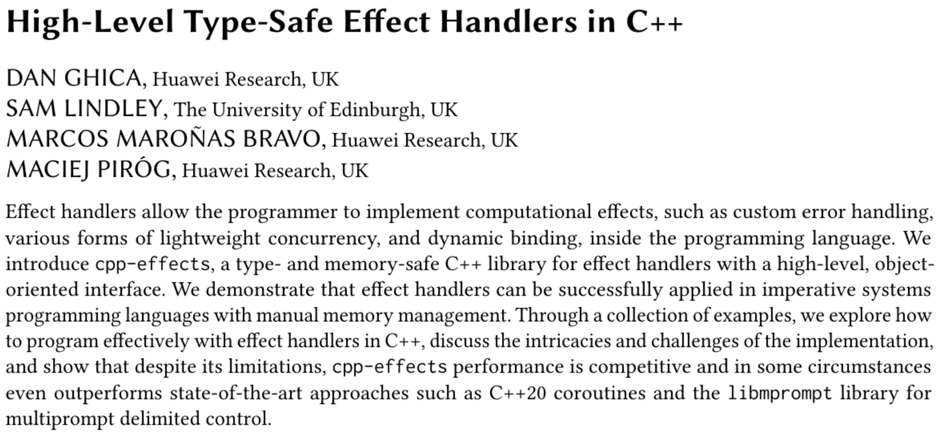
Introduction to Effect Handler Oriented Programming in C++

Daniel Hillerström

Computing Systems Laboratory
Zürich Research Center
Huawei Technologies Switzerland

June 29, 2023

C++Effects: A library for effect handler oriented programming



High-Level Type-Safe Effect Handlers in C++

DAN GHICA, Huawei Research, UK
SAM LINDLEY, The University of Edinburgh, UK
MARCOS MAROÑAS BRAVO, Huawei Research, UK
MACIEJ PIRÓG, Huawei Research, UK

Effect handlers allow the programmer to implement computational effects, such as custom error handling, various forms of lightweight concurrency, and dynamic binding, inside the programming language. We introduce `cpp-effects`, a type- and memory-safe C++ library for effect handlers with a high-level, object-oriented interface. We demonstrate that effect handlers can be successfully applied in imperative systems programming languages with manual memory management. Through a collection of examples, we explore how to program effectively with effect handlers in C++, discuss the intricacies and challenges of the implementation, and show that despite its limitations, `cpp-effects` performance is competitive and in some circumstances even outperforms state-of-the-art approaches such as C++20 coroutines and the `libmprompt` library for multiprompt delimited control.

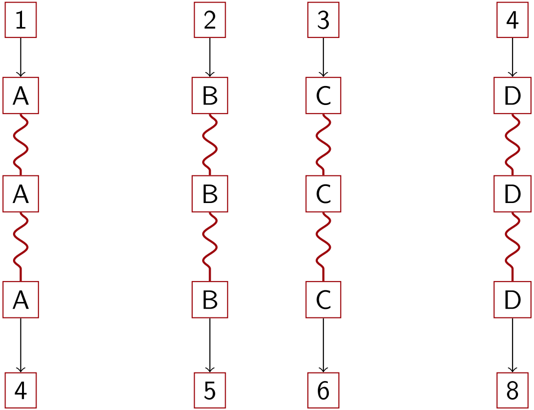
Talk objective

Demonstrate effect handler-oriented programming by example

- Implement a tiny tasking library;
- supporting task-local state;
- and task fault recovery

Example 1: Dynamic binding (1)

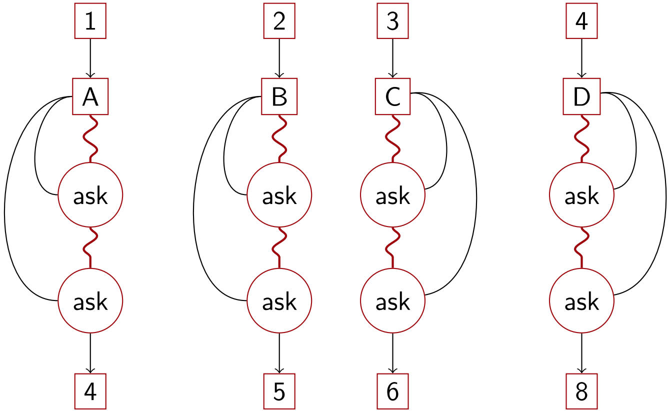
The state passing technique explicitly ‘threads’ the value through the entire computation.



Relevant code: `examples/ex1/state_passing.cpp`

Example 1: Dynamic binding (2)

The environment handler provides a context-dependent variable.



Relevant code: `examples/ex1/env.hpp` and `examples/ex1/env_main.cpp`

The programmer's perspective on effect handlers

Related familiar programming abstractions

- Coroutines
- Generators
- Lightweight threads
- Resumable exceptions
- First-class continuations

The programmer's perspective on effect handlers

Related familiar programming abstractions

- Coroutines
- Generators
- Lightweight threads
- Resumable exceptions
- First-class continuations

Executive summary

*Effect handlers offer a modern interface for highly **composable** and **customisable** programming with non-local control flow. Effect handlers subsume all classic ad-hoc control abstractions.*

A new programming paradigm

Procedure-oriented programming

Imperative steps organised as logical sub-routines

Object-oriented programming

Encapsulation of data and behaviour in objects

Function-oriented programming

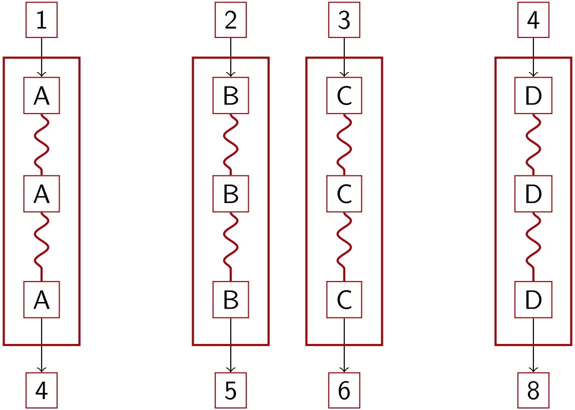
Abstraction through higher-order functions

Effect handler-oriented programming

Interaction with the execution context via continuations

Example 2: Lightweight threads

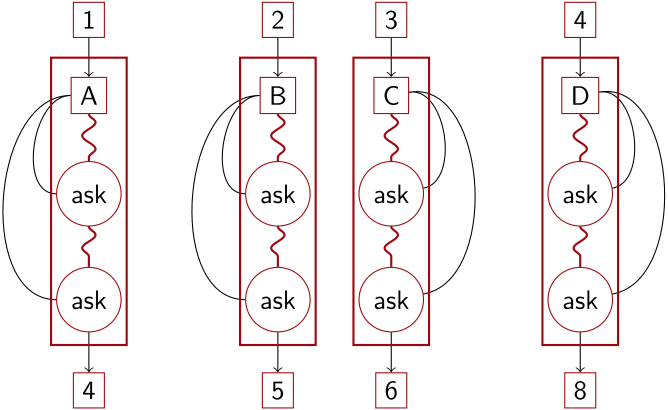
The scheduler handler reifies each task computation as a first-class object.



Relevant code: `examples/ex2/lwt.hpp` and `examples/ex2/lwt_main.cpp`

Example 3: Task-local state

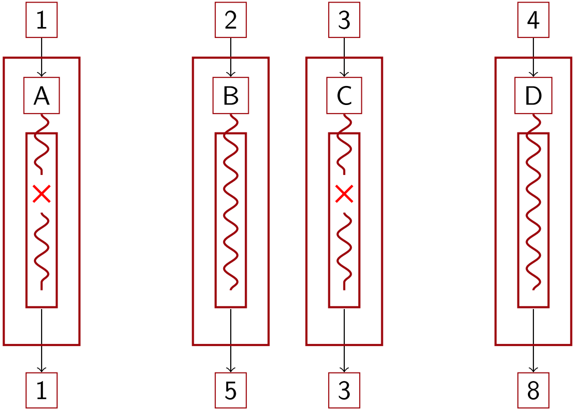
We obtain task-local state by composing the scheduler and environment handlers.



Relevant code: `examples/ex3/lwtenv_main.cpp`

Example 4: Transactional state

We add transactional state by composing yet another handler.



Relevant code: `examples/ex4/rollback.hpp` and `examples/ex4/rollback_main.cpp`

Summary

Summary

- Effect handlers offer modular and composable control
- Common control abstractions are implementable as libraries
- Separation of interface and implementation through continuations
- Extend functionality by composing fine-grained and agnostic handlers

References

Plotkin, Gordon D. and Matija Pretnar (2013). “Handling Algebraic Effects”. In: *Logical Methods in Computer Science* 9.4.

Hillerström, Daniel (2021). “Foundations for Programming and Implementing Effect Handlers”. PhD thesis. The University of Edinburgh, Scotland, UK.

Ghica, Dan et al. (2022). “High-Level Type-Safe Effect Handlers in C++”. In: *Proc. ACM Program. Lang.* 6.OOPSLA, pp. 1–30.