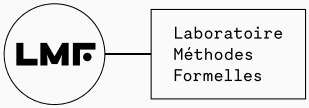
Verified interoperability with exceptions between OCaml and C

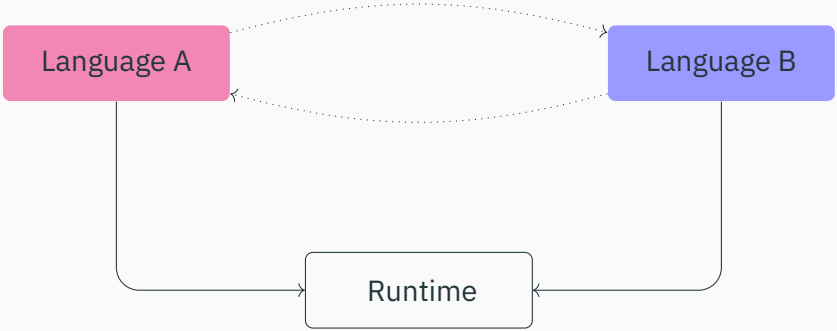
Valeran MAYTIE

Internship supervised by: Armaël Guéneau

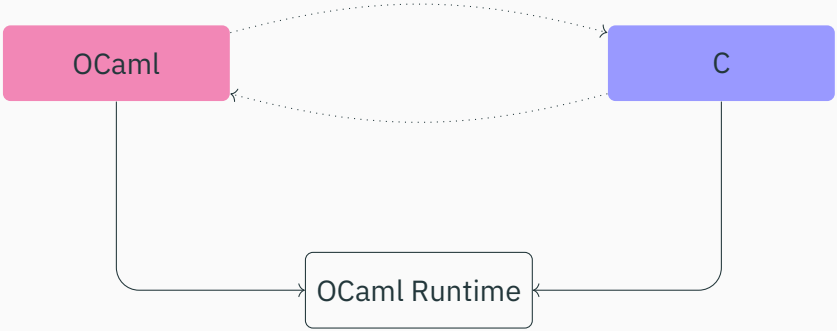
August 29th 2024



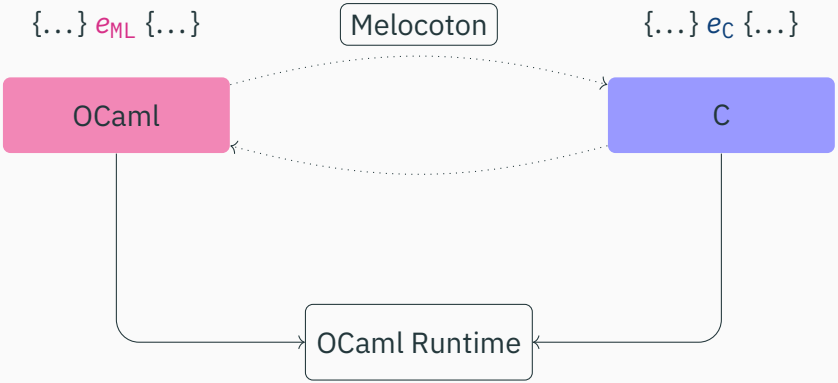
Language interoperability, OCaml Foreign Function Interface



Language interoperability, OCaml Foreign Function Interface



Language interoperability, OCaml Foreign Function Interface



Multi-language program verification with Melocoton

Goal of the internship: Add exceptions to Melocoton

- **Understand** exceptions in OCaml FFI
- **Formalise** step-by-step exceptions in Melocoton
- Make changes **without breaking** Melocoton
- Test changes in little **examples**

OCaml FFI with exception by example

```
external read_file :
  (int -> unit) -> string -> unit =
  "caml_read_file"
```

```
typedef struct {
  bool is_exception;
  value data;
} caml_result;
```

```
value caml_read_file(value fun, value s) {
  CAMLparam2(fun, s);
  char *fname = String_val(s);
  FILE *file = fopen(fname, "r");
  caml_result r;
  char c;
  while ((c = fgetc(file)) != EOF) {
    r = caml_callback_exn(fun,Val_int(c));
    if (r.is_exception) {
      fclose(file);
      caml_raise(r.data);
    }
  }
  fclose(file);
  CAMLreturn(Val_unit);
}
```

Melocoton

October 2023

Melocoton: A Program Logic for Verified Interoperability Between OCaml and C

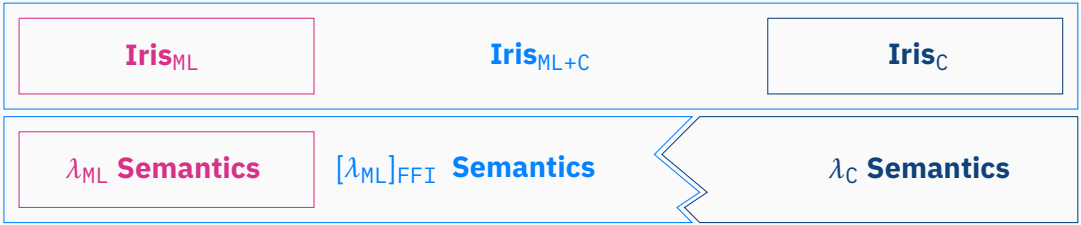
ARMAËL GUÉNEAU*, Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, France
JOHANNES HOSTERT*, Saarland University and MPI-SWS, Germany
SIMON SPIES*, MPI-SWS, Germany
MICHAEL SAMMLER, MPI-SWS, Germany
LARS BIRKEDAL, Aarhus University, Denmark
DEREK DREYER, MPI-SWS, Germany

https://melocoton-project.github.io/

≈ 23 000 lines of code

Iris methodology in Melocoton

Logic is built on top of **semantics**



The main goal is to model the **communication** between OCaml and C $\lambda_C \oplus -$

But, in practice these languages are to different

Solution: Wrap λ_{ML} in $[-]_{FFI}$ $[\lambda_{ML}]_{FFI} \oplus \lambda_C$

Objective: Adding exceptions in Melocoton

Modifications to be made:

- Add exceptions to $\lambda_{\mathtt{ML}}$
- Propagate exception between two languages: extend $- \oplus -$ and $[\![-]\!]_{\mathtt{FFI}}$
- Formalise new primitives: **`caml_callback_exn`** and **`caml_raise`**
- Test the implementation on examples

Extending $- \oplus -$ and $[-]_{\texttt{FFI}}$

1. Adding outcomes in Melocoton abstract languages
2. Modifying the linker $- \oplus -$ to propagate exceptions
3. Adapting the wrapper $[-]_{\texttt{FFI}}$ for outcomes

Outcomes definition:

$$o \in \mathit{Out}(\mathit{Val}) := v_{\texttt{val}} \quad \text{with } v \in \mathit{Val}$$
$$\mid v_{\texttt{exn}} \quad \text{with } v \in \mathit{Val}$$

New results

Challenge: Don’t break Melocoton

- Modification of **all** Melocoton languages
- Understand language combinators
- Repair all proofs at **every** changes
- Methodical **division** of work (the job must be merged into Melocoton)

Reasoning on FFI primitives

Generalization of reasoning rules with outcomes

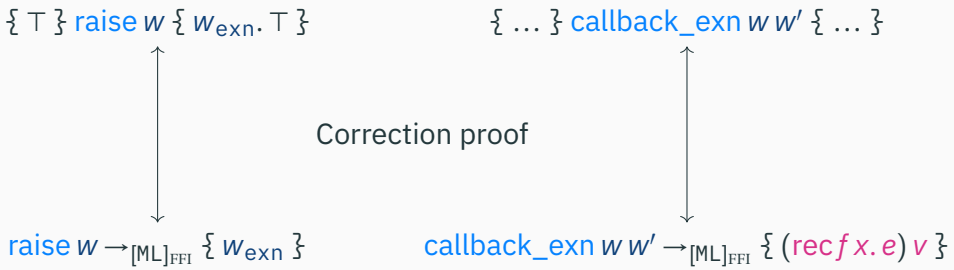
ExecRaise

$$\{\ \top\ \}\ \mathsf{raise}\ w\ \{\ w_{\mathtt{exn}}.\ \top\ \}$$

ExecCallbackExn

$$\frac{\{\ P\ \}\ (\mathsf{rec}\ f\ x.\ e)\ v\ \{\ Q\ \}}{\begin{array}{c}\{\ P * w \approx \mathsf{rec}\ f\ x.\ e * w' \approx v\ \}\\ \mathsf{callback\_exn}\ w\ w'\\ \{\ a_{\mathtt{val}}.\ \exists r\ r\ n.\ a \mapsto_{\mathtt{C}}^{*} [n; r] * r \approx r *\\ (n = 0 \Rightarrow Q(r_{\mathtt{val}})) *\\ (n = 1 \Rightarrow Q(r_{\mathtt{exn}}))\ \}\end{array}}$$

Proving the new rules



Conclusion

Contribution:

- Outcome propagation between incompatible languages (27, 26, 23)
- Intercepting outcomes in the wrapper $[-]_{\texttt{FFI}}$
- Verification of new reasoning rules in $\textbf{Iris}_{\texttt{ML+C}}$: ExecRaise and ExecCallbackExn
- Generalization of $\textbf{Iris}_{\texttt{ML+C}}$ with outcomes (23)

Practically:

- 5 Pull request merged: 18, 23, 25, 26, 27
- There is one branch left to merge “exception”
- Talk in the LMF PhD non-permanent seminar with Gurvan Debaussart