

Exceptions aren't cool

Edvard Aksnes

PwC Norway

2025

Program execution is typically linear...

```
# main.py  
my_hash_map = {"a": 1, "b": 2}  
print(my_hash_map["a"])  
my_hash_map["b"] = 3  
print(my_hash_map["a"] + my_hash_map["b"])
```

Program execution is typically linear...

```
# main.py  
my_hash_map = {"a": 1, "b": 2}  
print(my_hash_map["a"])  
my_hash_map["b"] = 3  
print(my_hash_map["a"] + my_hash_map["b"])
```

Output:

```
$ python main.py  
1  
4
```

Program execution is typically linear...

```
# main.py  
my_hash_map = {"a": 1, "b": 2}  
print(my_hash_map["c"])  
my_hash_map["b"] = 3  
print(my_hash_map["a"] + my_hash_map["b"])
```

Program execution is typically linear... NOT

```
# main.py
my_hash_map = {"a": 1, "b": 2}
print(my_hash_map["c"])
my_hash_map["b"] = 3
print(my_hash_map["a"] + my_hash_map["b"])
```

Output:

```
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    print(my_hash_map["c"])
    ~~~~~^~~~~~
KeyError: 'c'
```

Program execution is typically linear... NOT

```
# main.py
```

```
my_hash
```

```
print(m
```

```
my_hash
```

```
print(m
```

```
h_map["b"])
```

Output:

```
Traceba
```

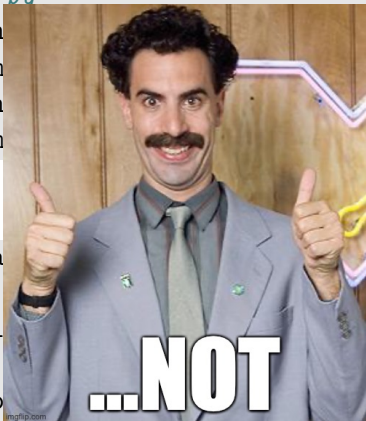
```
File
```

```
pri
```

```
KeyError
```

```
st):
```

```
odule>
```



Becomes worse and worse...

```
# super_important_business_critical_code.py

important = haha_i_only_work_sometimes()

try:
    must_be_done_every_time = lol_dont_care(important)
except:
    # run for your life
    pass
```

Can we do something about this?

```
def only_works_sometimes(a:float, b:float) -> float:  
    return a / b
```

Can we do something about this?

```
def who_knows_what_i_ll_do(a:int, b:int) -> int:  
    return a / b
```

```
def predictable(a:int, b:int) -> <int, ZeroDivError>:  
    return a / b
```

Algebraic effects!

```

value  $v ::= x$ 
      | true | false
      | fun  $x \mapsto c$ 
      |  $h$ 
handler  $h ::= \mathbf{handler} \{ \mathbf{return} \ x \mapsto c_r,$ 
                         $\mathbf{op}_1(x; k) \mapsto c_1, \dots, \mathbf{op}_n(x; k) \mapsto c_n \}$ 
computation  $c ::= \mathbf{return} \ v$ 
      |  $\mathbf{op}(v; y. c)$ 
      | do  $x \leftarrow c_1$  in  $c_2$ 
      | if  $v$  then  $c_1$  else  $c_2$ 
      |  $v_1 \ v_2$ 
      | with  $v$  handle  $c$ 

```

Overall idea

Leading question

What if the type system was expanded with **effects** and **handlers**?

Overall idea

Leading question

What if the type system was expanded with **effects** and **handlers**?

In addition to functions and values, we add *effects* $\text{op}_1, \dots, \text{op}_n$:

$$\text{op}_i(v, y.c) \iff \text{let } y = \text{perform } (\text{op}_i \ v) \text{ in } c$$

and *handlers*:

$$\begin{aligned} h &:= \{\text{return } x \mapsto c_r, \text{op}_\alpha(v) \mapsto c_\alpha\} \\ \iff h &= \text{handler } \{ \\ &\quad \text{return } x \mapsto c_{\text{out}}, \\ &\quad \text{op}_i(v, k) \mapsto c_i \\ &\} \end{aligned}$$

Overall idea

Leading question

What if the type system was expanded with **effects** and **handlers**?

In addition to functions and values, we add *effects* $\text{op}_1, \dots, \text{op}_n$:

$$\text{op}_i(v, y.c) \iff \text{let } y = \text{perform } (\text{op}_i \ v) \text{ in } c$$

and *handlers*:

$$\begin{aligned} h &:= \{\text{return } x \mapsto c_r, \text{op}_\alpha(v) \mapsto c_\alpha\} \\ \iff h &= \text{handler } \{ \\ &\quad \text{return } x \mapsto c_{\text{out}}, \\ &\quad \text{op}_i(v, k) \mapsto c_i \\ &\} \end{aligned}$$

The **Eff** programming language

Handlers let us say *"when this happens, do that."*

But what if we could decide what happens next?

Introduction to `call/cc`

In **Scheme**, the `call/cc` function allows you to capture the context of a computation, i.e. the rest of the program.

Introduction to call/cc

In **Scheme**, the `call/cc` function allows you to capture the context of a computation, i.e. the rest of the program.



Figure: You should have seen this coming



Figure: What we're about to do

A story for another time...

*programming
language theory talk*

monads:



FIN