Algebraic Subtyping for Algebraic Effects and Handlers

Axel Faes
KULeuven

What are algebraic effects and handlers?

Exception handlers on steroids

Defining a new effect

```
class DivisionByZero extends Exception {
    public DivisionByZero() {
        super("Division by zero");
    }
}
```

```
effect DivisionByZero : unit -> empty
```

Using an effect

```
{
        throw new DivisionByZero();
}
```

#DivisionByZero ()

Using an effect

```
public static int divide(int a, int b) {
    if (b == 0) {
        throw new DivisionByZero();
    } else {
        return a / b;
    }
}
```

```
let divide a b =
    if (b == 0) then
        #DivisionByZero ()
    else
        a / b
```

Effect inference

Inference of effects

```
public static int divide(int a, int b) throws DivisionByZero {
    if (b == 0) {
        throw new DivisionByZero();
    } else {
        return a / b;
    }
}
```

Effect handlers

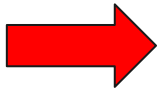
```
public static int safeDivide(int a, int b) {
    try {
        return divide(a, b);
    } catch(DivisionByZero ex) {
        return 0;
    }
}
```

```
let safeDivide a b =
    handle (divide a b) with
        | #DivisionByZero () k -> 0
```

Continuation

Effect inference

```
public static int safeDivide(int a, int b) {
    try {
        return divide(a, b);
    } catch(DivisionByZero ex) {
        return 0;
    }
}
```



Doesn’t throw exception

What is algebraic subtyping?

The select function

```
public static ?1 select(?2 p, ?3 v, ?4 d) {
    if (p(v)) {
        return v;
    } else {
        return d;
    }
}
```

```
let select p v d =
    if (p v) then
        v
    else
        d
```

The select function in ML

```
public static ?1 select(?2 p, ?3 v, ?4 d) {
    if (p(v)) {
        return v;
    } else {
        return d;
    }
}
```

```
let select p v d =
    if (p v) then
        v
    else
        d
```

$?_1 = ?_3 = ?_4$

$?_2 = ?_1 \rightarrow \text{bool}$

$\forall \alpha \,.\, (\alpha \rightarrow \text{bool}) \rightarrow \alpha \rightarrow \alpha \rightarrow \alpha$

The select function with subtyping

```
public static ?1 select(?2 p, ?3 v, ?4 d) {
    if (p(v)) {
        return v;
    } else {
        return d;
    }
}
```

```
let select p v d =
    if (p v) then
        v
    else
        d
```

$(\alpha \to \text{bool}) \to \alpha \to \beta \to \gamma \mid \alpha \leq \gamma, \beta \leq \gamma$

The select function with subtyping

```
public static ?1 select(?2 p, Dog v, Cat d) {
    if (p(v)) {
        return v;
    } else {
        return d;
    }
}
```

```
let select p v d =
    if (p v) then
        v
    else
        d
```

$\mathbf{Dog} \leq ?_1$, $\mathbf{Cat} \leq ?_1$

$\alpha \leq \gamma$, $\beta \leq \gamma$

The select function with subtyping

```
public static Animal select(?2 p, Dog v, Cat d) {
    if (p(v)) {
        return v;
    } else {
        return d;
    }
}
```

```
let select p v d =
    if (p v) then
        v
    else
        d
```

Dog ≤ **Animal**, **Cat** ≤ **Animal**

α ≤ γ, β ≤ γ

The select function with (algebraic) subtyping

Dog $\leq ?_1$, **Cat** $\leq ?_1$ → **Animal** → **Dog** or **Cat**

The select function with algebraic subtyping

```
public static (Dog or Cat)
  select(? p, Dog v, Cat d) {
        if (p(v)) {
              return v;
        } else {
              return d;
        }
}
```

```
let select p v d =
      if (p v) then
            v
      else
            d
```

$(\alpha \to \text{bool}) \to \alpha \to \beta \to \alpha \sqcup \beta$

What is the goal?

Algebraic subtyping with effects

```
let select p v d =
    if (p v) then
        v
    else
        d
```

$\forall \alpha, \beta . (\alpha \to \text{bool}) \to \alpha \to \beta \to \alpha \sqcup \beta$

Add effect information

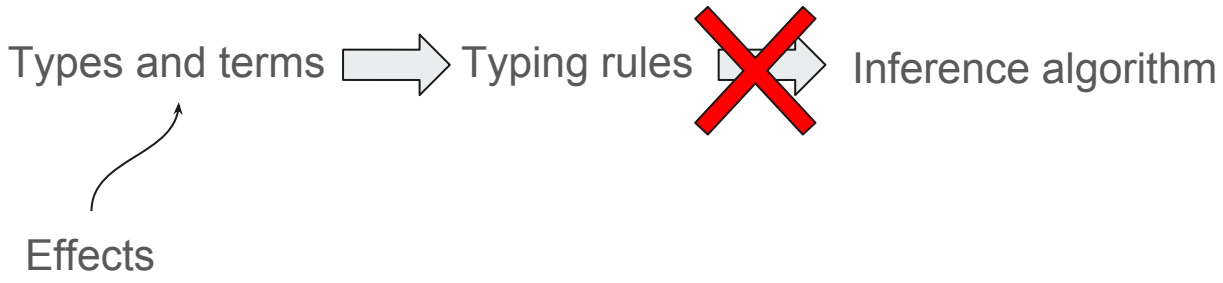
How do you build a type system?

Algebraic subtyping with effect

Types and terms → Typing rules → Inference algorithm

Effects

Algebraic subtyping with effect



Algebraic subtyping with effect

Types and terms → Typing rules → Transformations → Inference algorithm

Effects

Transformations

Typing rules → Reformulate Typing rules → Add polarity → Inference algorithm

Transformations

What inference algorithm?

Transformations

What inference algorithm?

Hindley-Milner type inference (with minor changes)

What have I done?

Done

Implementation

Eff programming language
written in OCaml

Fully featured

Todo: simplification using finite automata

```
and type_expr st {Untyped.term=expr; Untyped.location=loc} = type_plain_e

(* Type a plain expression *)
and type_plain_expr st loc = function
  | Untyped.Var x ->
    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
  | Untyped.Tuple es ->
    let els = List.map (fun (e, _) -> e) (List.map (type_expr st) es) in
    Ctor.tuple ~loc els, st
  | Untyped.Record lst ->
    let lst = List.map (fun (f, (e, _)) -> (f, e)) (Common.assoc_map (typ
    Ctor.record ~loc lst, st
  | Untyped.Variant (lbl, e) ->
    let exp = Common.option_map (fun (e, _) -> e) (Common.option_map (typ
    Ctor.variant ~loc (lbl, exp), st
  | Untyped.Lambda (p, c) ->
    let pat = type_pattern st p in
    let comp, st = type_comp st c in
    Ctor.lambda ~loc pat comp, st
  | Untyped.Effect eff ->
    let eff = infer_effect ~loc st eff in
```

TODO

Theory

Proofs

- Instantiation
- Weakening
- Substitution
- Soundness
- Type preservation
- Reformulated typing rules

TODO

Validation

Testing against other systems

- Coercion subtyping
- Subtyping
- Row polymorphism

Usecase

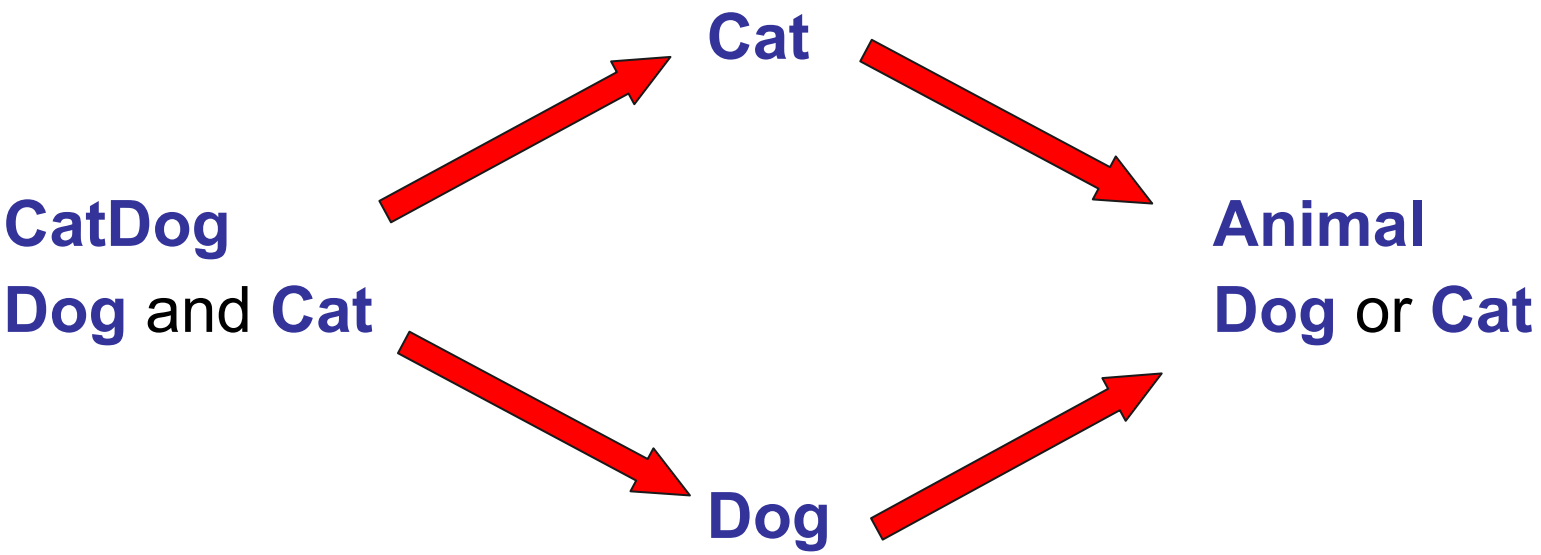
- Optimized compilation

Summarize

Algebraic effects and handlers

Algebraic subtyping

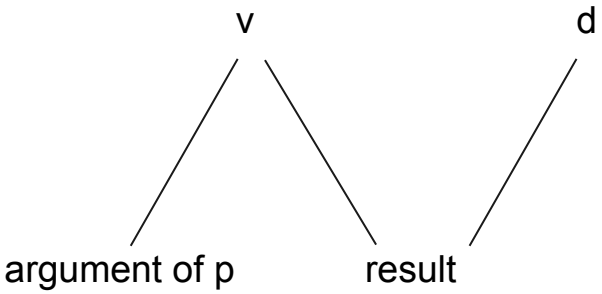
The select function with (algebraic) subtyping



The select function in ML

```
let select p v d =
    if (p v) then
        v
    else
        d
```

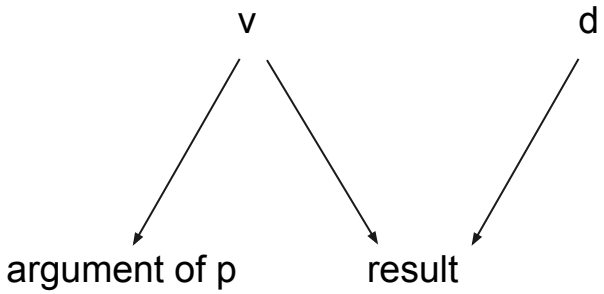
$\forall \alpha \,.\, (\alpha \to \text{bool}) \to \alpha \to \alpha \to \alpha$



The select function with Subtyping

```
let select p v d =
    if (p v) then
        v
    else
        d
```

$(\alpha \to \text{bool}) \to \alpha \to \beta \to \gamma \mid \alpha \leq \gamma, \beta \leq \gamma$



The select function with Subtyping

$\alpha \leq \gamma$, $\beta \leq \gamma$ → Dog ≤ Animal,
Cat ≤ Animal

The select function with Subtyping

Dog ≤ Animal,
Cat ≤ Animal

Animal is a Dog or a Cat

The select function with algebraic subtyping

```
let select p v d =
    if (p v) then
        v
    else
        d
```

$\forall \alpha, \beta . (\alpha \to \text{bool}) \to \alpha \to \beta \to \alpha \sqcup \beta$