Nullable Type Inference

Michel Mauny, Benoît Vaugon
Ensta-ParisTech
Nullable Types

A **nullable type** $t?$ includes `NULL` and (unboxed) values of type $t$

$$
t ::= t_b \mid \alpha \mid \tau_1 \rightarrow \tau_2 \\
\tau ::= t \mid t？
$$

We provide a type inference algorithm:

- featuring Hindley-Milner polymorphism
- that statically guarantees that `NULL` cannot be used as a regular value
- whose soundness has been proved

Nullable types in practice

- Hack (Facebook)
- Swift (Apple)
- ...

```swift
func f (b : Bool, s : String) -> String? {
    if b { return s } else { return nil }
}
```
Swift:

```swift
var f = { (b : Bool, s : String) -> String? in
    if b { s } else { nil }
}
→ Ok

var f = { (b : Bool, s) -> String? in
    if b { s } else { nil }
}
→ Error: Cannot convert type '(Bool, $T0) -> String?' to type '$T1'
```

OCaml:

```ocaml
let f b (s : ['String]) =
    if b then s else 'NULL

→ Error: This expression has type ['NULL]
    but an expression was expected of type ['String]
    The second variant type does not allow tag(s) 'NULL
Replace Unification by Subtyping

The algorithm interleaves inference of subtyping constraints...