Point sur FunLoft

1. Accès aux champs des structures
2. Compilation séparée
3. Integers

Structures

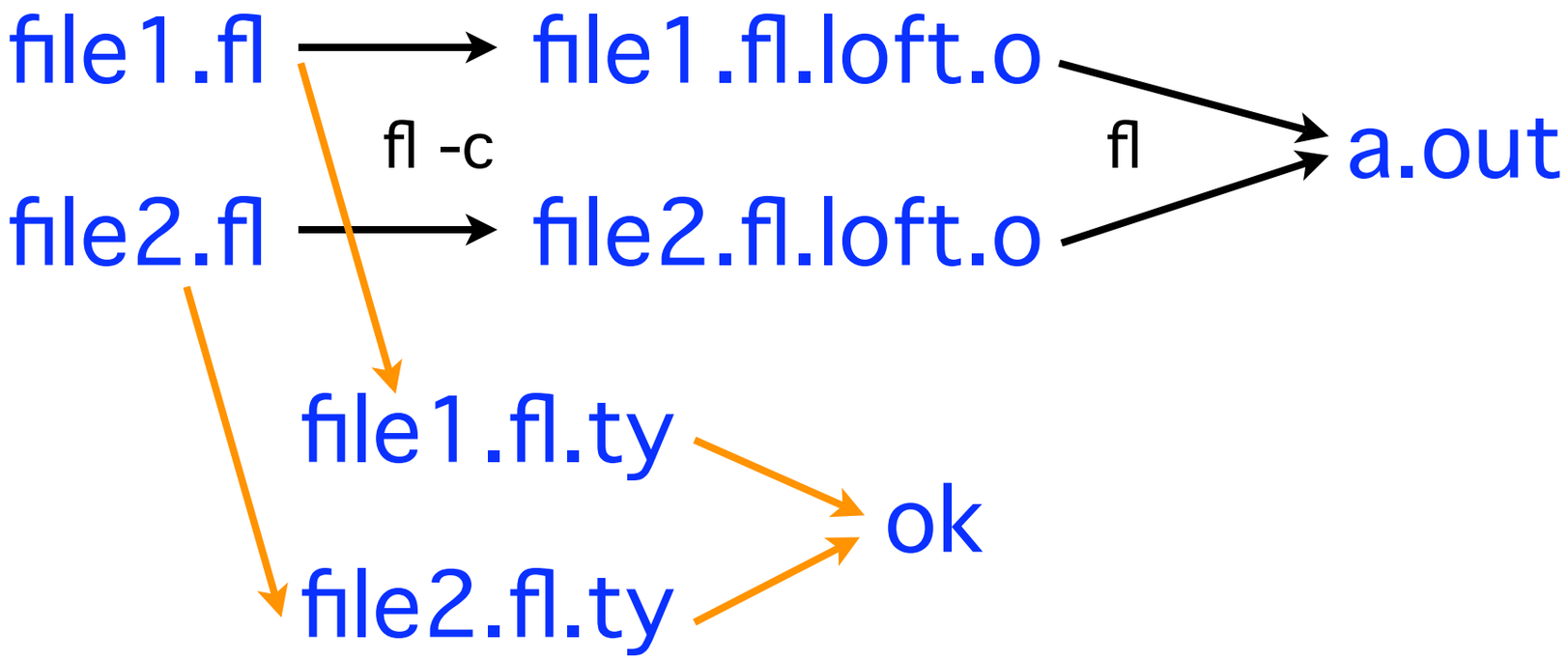
En présence d'un seul constructeur on peut nommer les champs

```
type point_t = Point of int * int
let f (p) = match p with Point (x,y) -> x + y end
```

Alternative :

```
type point_t = Point of x: int * y: int
let f (p) = p.x + p.y
```

Compilation séparée




```
extern g: int->int
let r = ref 0
let f (x) = g(x) + !r
let module main () = f(4)
```

file1.fl

```
extern g: int -> int
r: int ref
f: int -> int +access
main:  unit => thread_t
```

file1.fl.ty

Compilation séparée

Pas complètement implémentée

file1.fl

```
extern g: int->int
let f(x) = g(x)
let module main () = f(0)
```

file2.fl

```
extern f: int->int
let g(x) = f(x)
```

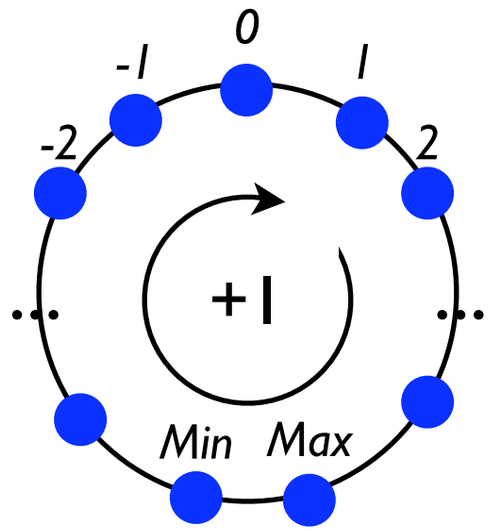
```
fl -c file1.fl
fl -c file2.fl
fl file1.fl.loft.o file2.fl.loft.o
a.out
```

Bus error

Cycles de dépendance non détectés

Standard approach to Integers

[Min..Max]



Max+1 = Min

1/0 = Error

⌈ n ⌉ ∈ **Z**, ⌈ < ⌉ ≠ **<**

⌈ Max-1 < Min-1 ⌉ = *true*

∃n ⌈ n∗n < 0 ⌉ = *true*, ...

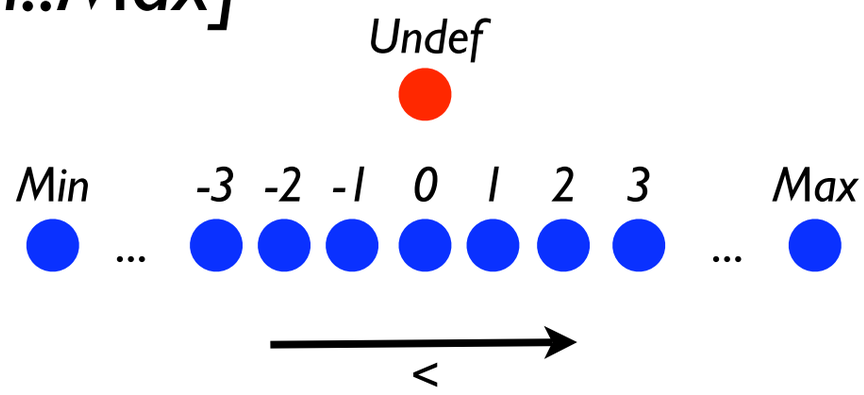
type int in C

max == 2147483647 min == -2147483648
max+1 == -2147483648 min+1 == -2147483647
max-1 == 2147483646 min-1 == 2147483647
max-1 < min-1 == 1

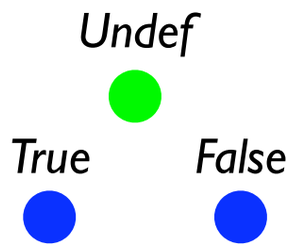
y == 1000000
y*y == -727379968

Integers & Booleans

[Min..Max]



Booleans



Max+1 = Min-1 = Undef = 1/0

n < Undef = Undef = Undef < n

Integers & Booleans - 2

n,m of type int

- ⌈n+m⌉ = *if* ⌈n⌉ , ⌈m⌉ , ⌈n+m⌉ ∈*[Min..Max] then n+m*

else *Undef*

- ⌈n<m⌉ = *if* ⌈n⌉ , ⌈m⌉ ∈*[Min..Max] then n<m else* *Undef*

Soundness:

- ⌈n<m⌉ = *true* ⇒ ⌈n⌉ < ⌈m⌉
- ⌈n<m⌉ = *false* ⇒ ⌈n⌉ ≥ ⌈m⌉
- ⌈n<m⌉ = *Undef* ⇒ ⌈n⌉ *or* ⌈m⌉ *not in [Min..Max]*