TD 3 : Inférence de types

1 Unification de types

Donnez les unificateurs des types suivants (`'a, 'b, 'c` sont des variables de type) :
— `int -> int` et `'a -> 'b`
— `(int -> int) -> 'a` et `int -> (int -> 'b)`
— `(int -> bool) -> 'a` et `'a -> 'b`
— `'a -> 'a` et `'a`

2 Inférence de types

Inférez le type des expressions suivantes :
— `fun x -> f (f x)`, pourvu que `f : int -> int`
— `fun f -> f (f 42)`
— `fun f -> f (f true)`
— `fun f -> fun x -> f (f x)`
— `fun (f: 'a -> int) -> ((f 3) + (f 4))`
— `fun (f: 'a -> int) -> ((f 3) + (f true))`
— `fun g -> List.map g [[1];[2];[3]]`, où
`List.map : ('a -> 'b) -> 'a list -> 'b list`
— `fun f -> (f (fun x -> 3))`
— `fun x -> (x x)`

Dans le cas d'un échec, expliquez les origines par la cause de l'échec de l'algorithme d'unification sous-jacent (*Clash*/*Check*).

3 Inférence de types avec `let`

Expliquez le processus d'inférence de types des expressions suivantes :

1. `fun x -> let f = (fun g -> g x) in (f (fun z -> 3))`
2. `fun x -> let f = (fun g -> g x) in (f (fun z -> z + 3))`
3. `fun x -> let f = (fun g -> g x) in (f (fun z -> z + 3)) + (f (fun z -> if x then 1 else 2))`
4. `fun x -> let f = (fun g -> g x) in (f (fun z -> if x then 1 else 2))`
5. `fun x -> let f = (fun g -> g x) in f`