

GOSPEL

un langage de spécification pour OCaml

Jean-Christophe Filliâtre Clément Pascutto Mário Pereira

journée LVP
12 mars 2021



NOVALINCS

projet ANR **VOCaL** (**V**erified **OC**aml **L**ibrary), 2015–2021

objectif : construire une bibliothèque vérifiée de structures de données et algorithmes

participants : Paris-Saclay, Inria, Verimag, TrustInSoft, OCamlPro

une contribution du projet VOCaL est un langage de spécification pour OCaml, appelé GOSPEL

(pour Generic OCaml SPECification Language)

1. une brève présentation de GOSPEL
2. écosystème

principaux choix :

- spécification écrite dans le `.mli`, dans des commentaires
- une syntaxe proche de celle d'OCaml
- de la logique du premier ordre
- pas nécessairement exécutable

une sémantique en termes de logique de séparation [FM'19]

```
val mjrty: string array -> string
```

```
val mjrty: string array -> string
```

```
(** [mjrty a] returns the element of [a] with absolute  
    majority, if any, or raises [Not_found] otherwise *)
```

```
val mjrty: string array -> string
```

```
(** [mjrty a] returns the element of [a] with absolute  
majority, if any, or raises [Not_found] otherwise *)
```

```
(*@ r = mjrty a  
  requires 1 <= length a  
  ensures numof a r > length a / 2  
  raises   Not_found ->  
           forall c. numof a c <= length a / 2    *)
```


- champs modèles

```
type 'a t  
(*@ mutable model view: 'a seq *)
```

- champs modèles
- effets de bord

```
val push: 'a t -> 'a -> unit
(*@ push a x
    modifies a
    ...
```

- champs modèles
- effets de bord
- entiers mathématiques dans les spécifications

```
val f: int -> int
(*@ y = f x
   ensures 3 * y > 2 * x *)
```

- champs modèles
- effets de bord
- entiers mathématiques dans les spécifications
- code fantôme

```
(*@ type witness *)
```

```
(*@ val create: int -> witness *)
```

```
val f: int -> int  
(*@ y = f [w:witness] x ... *)
```

```
val binary_search:  
  ('a -> 'a -> int) -> 'a array -> int -> int -> 'a -> int  
(*@ r = binary_search cmp a fromi toi v  
  requires 0 <= fromi <= toi <= length a  
  requires is_pre_order cmp  
  requires forall i j.  
    fromi <= i <= j < toi -> cmp a[i] a[j] <= 0  
  ...
```

les préconditions sont supposées vérifiées, statiquement,
au point d'appel

```
val binary_search:  
  ('a -> 'a -> int) -> 'a array -> int -> int -> 'a -> int  
(*@ r = binary_search cmp a fromi toi v  
  checks    0 <= fromi <= toi <= length a  
  requires  is_pre_order cmp  
  requires  forall i j.  
             fromi <= i <= j < toi -> cmp a[i] a[j] <= 0  
  ...
```

checks signifie une vérification dynamique
(et la levée de `Invalid_argument` en cas d'échec)

```
val binary_search:  
  ('a -> 'a -> int) -> 'a array -> int -> int -> 'a -> int  
(*@ r = binary_search cmp a fromi toi v  
  checks    0 <= fromi <= toi <= length a  
  requires  is_pre_order cmp  
  requires  forall i j.  
             fromi <= i <= j < toi -> cmp a[i] a[j] <= 0  
  ...
```

un tel contrat suppose que la fonction cmp est **pure**

il n'est pas toujours possible de faire une telle hypothèse

```
val iter: ('a -> unit) -> 'a t -> unit
```


il n'est pas toujours possible de faire une telle hypothèse

```
val iter: ('a -> unit) -> 'a t -> unit
(*@ iter f c
  equivalent "List.iter f (elements c)" *)
```

c'est un morceau de code, pas une formule logique !

2

écosystème

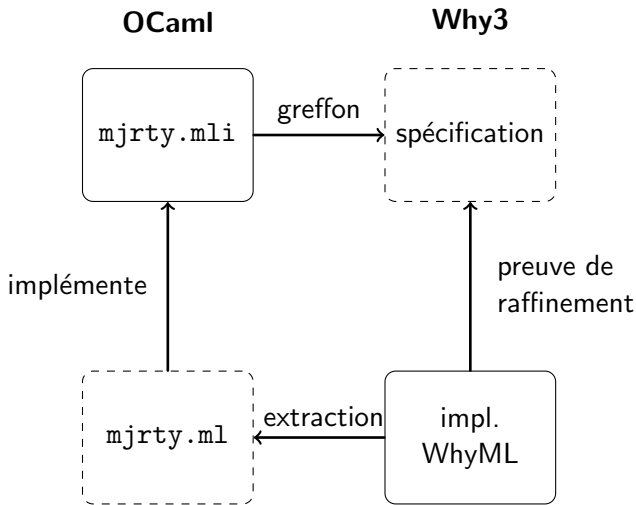
- implémentation de GOSPEL
 - ▶ analyse syntaxique, typage
 - ▶ bibliothèque standard
 - ▶ documentation
- deux outils de vérification déductive
 - ▶ un greffon Why3
 - ▶ cameleer (Mário Pereira)
- vérification dynamique
 - ▶ gospel-rtac (Clément Pascutto)
- bibliothèque VOCaL

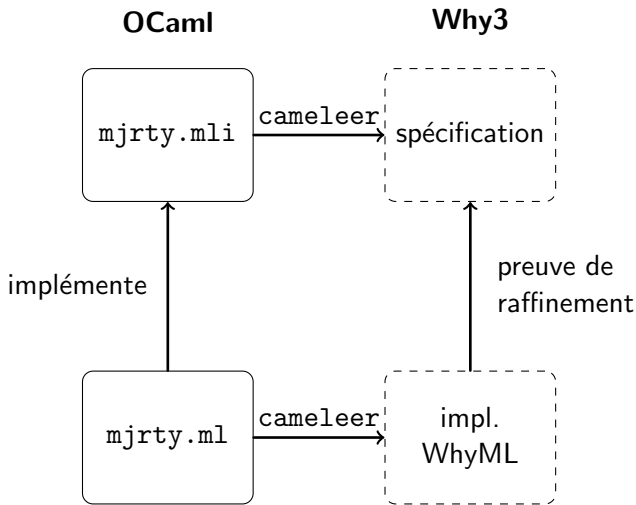
implémentation de GOSPEL

github.com/ocaml-gospel/gospel

- contribution initiale de Cláudio Lourenço (post-doc)
- utilise les attributs d'OCaml et ppxlib

```
val create: int -> 'a -> 'a array
[@@gospel {| arr = create n v
  requires n >= 0
  ensures length arr = n
  ensures forall i. 0 <= i < n -> arr[i] = v |}]
```





extension de GOSPEL pour les fichiers `.ml`

- invariants et variants de boucle
- assertions intermédiaires

limitations imposées par WhyML

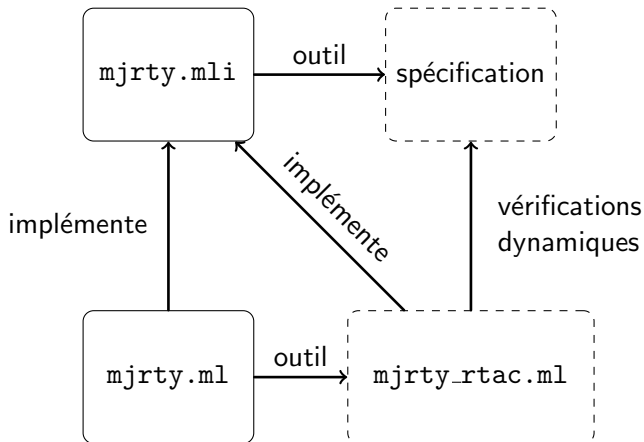
- mutabilité bornée
- ordre supérieur sans effet

plus de 20 modules complètement vérifiés :

- structures de données (e.g., leftist heap, union find)
- ordre supérieur (e.g., `List.fold_left`, code en CPS)
- foncteurs (e.g., AVL)

```
let binary_search cmp a fromi toi v =  
  let l = ref fromi in  
  let u = ref toi in  
  let exception Found of int in  
  try  
    while !l < !u do  
      (*@ invariant fromi <= !l && !u <= toi *)  
      (*@ invariant forall i. fromi <= i < toi ->  
        cmp a.(i) v = 0 -> !l <= i < !u *)  
      (*@ variant !u - !l *)  
      ...  
    done;  
    raise Not_found  
  with Found r ->  
    r
```


OCaml



- identifie le fragment exécutable des contrats de fonctions
- produit un code instrumenté autour de l'implémentation d'origine
 - ▶ préconditions, postconditions (exceptionnelles)
 - ▶ levée d'exceptions illégales
- sémantique identique en cas de succès
- ignore les clauses non exécutables

permet de vérifier autant la correction

- des appels à l'API
- de l'implémentation de l'API

1. on produit le module instrumenté

```
$ gospel-rtac lib.mli > lib_rtac.ml  
$ cp lib.mli lib_rtac.mli
```

2. on lie ensemble

- ▶ la bibliothèque originale lib.ml
- ▶ la bibliothèque support gospel_runtime
- ▶ le code instrumenté lib_rtac.ml
- ▶ le code client

3. on lance

```
$ ./test
```

File "lib.mli", line 7, characters 0-125:

```
Runtime error: specification unmet in function 'bad_create':  
post-condition 'forall i. 0 <= i < n -> arr[i] = v' is violated.
```

module	loc	tool	
HashTable	150	CFML	dont iter, fold
UnionFind	60	Why3,CFML	complexité amortie
Vector	150	Why3	
PriorityQueue	81	Why3	utilisé dans le code de Why3
PairingHeap	42	Why3	
ZipperList	58	Why3	
Lists	50	Coq	dont le tri fusion d'OCaml
Arrays	63	Why3	
Mjrtty	26	Why3	
RingBuffer	34	Why3	
CountingSort	24	Why3	

<https://vocal.lri.fr/>

- publications

<https://github.com/ocaml-gospel/>

- implémentation de GOSPEL
- greffon Why3
- cameleer
- gospel-rtac
- la bibliothèque VOCaL