

# Soutenance de projet

---

Anthony Araye et Camille Schnell

# Sommaire

- I. Introduction du sujet
- II. La machine abstraite
- III. Scénarios de test
- IV. Raffinements
- V. Preuves
- VI. Conclusion

# I. Introduction

Objectif : modéliser un système d'emprunt en libre service de vélo.

Contraintes :

- 4 gares contenant 6 bornes
- 12 vélos à l'état initial
- restrictions lorsqu'un utilisateur est bloqué (emprunt)

## II. La machine abstraite

### Ensembles et constantes

#### **MACHINE**

Velhop

#### **CONSTANTS**

velos, gare

#### **PROPERTIES**

velos = 1 .. 12 & gare = 1 .. 4

#### **SETS**

PERSONNES; ETATSVELO = {bon\_etat,use,abime}; STATUT = {actif, bloque}

#### **VARIABLES**

usagers, etatVelos, usagerVelo, gares

## II. La machine abstraite

### Variables et invariant

#### **VARIABLES**

usagers, etatVelos, usagerVelo, gares

#### **INVARIANT**

```
etatVelos : velos --> ETATSVELO &  
usagers : PERSONNES +-> STATUT &  
usagerVelo : dom(usagers) >+> velos &  
gares : gare --> (1 .. 6 +-> velos) &  
!g1.(g1 : dom(gares) => (!v1.(v1 : ran(gares(g1)) => (!g2.(g2 : dom(gares) & g2 /= g1 =>(v1  
/: ran(gares(g2))))))))))
```

## II. La machine abstraite

### Initialisation

#### INITIALISATION

```
usagers, etatVelos, usagerVelo := {}, velos * {bon etat}, {} ||  
gares := {1 |-> {1 |-> 1, 2 |-> 2, 3 |-> 3}, 2 |-> {1 |-> 4, 2 |-> 5, 3 |-> 6}, 3 |-> {1 |->  
7, 2 |-> 8, 3 |-> 9}, 4 |-> {1 |-> 10, 2 |-> 11, 3 |-> 12}}
```

## II. La machine abstraite

### Opérations implémentées (1)

#### OPERATIONS

```
    creer_usager(user) =
```

```
    PRE
```

```
        user : PERSONNES & user /: dom(usagers)
```

```
    THEN
```

```
        usagers := usagers \ / {user |-> actif}
```

```
    END;
```

```
    supprimer_usager(user) =
```

```
    PRE
```

```
        user : PERSONNES & user : dom(usagers) & usagers(user) /= bloque & user /:
```

```
dom(usagerVelo)
```

```
    THEN
```

```
        usagers := {user} <<| usagers
```

```
    END;
```

## II. La machine abstraite

### Opérations implémentées (2)

```
debloquer_usager(user) =  
PRE  
    user : PERSONNES & user : dom(usagers) & usagers(user) = bloque & user /: dom(usagerVelo)  
THEN  
    usagers := usagers <+ {user |-> actif}  
END;
```

```
emprunter(user, vv, gg) =  
PRE  
    user : PERSONNES & user : dom(usagers) & usagers(user) /= bloque & user /:  
dom(usagerVelo) & vv : velos & gg : gare & vv /: ran(usagerVelo) & vv : ran(gares(gg)) &  
etatVelos(vv) = bon_etat  
THEN  
    usagerVelo := usagerVelo \ / {user |-> vv} ||  
    gares(gg) := gares(gg) |>> {vv}  
END;
```



## II. La machine abstraite

### Opérations implémentées (3)

```
rendre(velo,gg,temps,etat) =  
PRE  
    velo : velos & gg : gare & velo : ran(usagerVelo) & temps : NAT1 & etat : ETATSVELO  
THEN  
    ANY xx WHERE xx : 1 .. 6 & xx /: dom(gares(gg)) THEN  
        usagerVelo := usagerVelo |>> {velo} ||  
        etatVelos := etatVelos <+ { velo |-> etat } ||  
        gares(gg) := gares(gg) \ / {xx |-> velo} ||  
        IF temps > 60 or etat = abime THEN  
            usagers := usagers <+ usagerVelo~[{velo}] * {bloque}  
        END  
    END  
END;  

```

## II. La machine abstraite

### Opérations implémentées (4)

```
deplacer_velo(vv,gg)=
PRE
  vv : velos & gg : gare & etatVelos(vv) = bon_etat & card(gares(gg))<6 & vv /:
ran(gares(gg)) & vv /: ran(usagerVelo)
THEN
  ANY ggg WHERE ggg : gare & vv : ran(gares(ggg)) THEN
    ANY xx WHERE xx : 1 .. 6 & xx /: dom(gares(gg)) THEN
      gares := gares <+ {ggg |-> (gares(ggg) |>> {vv})}, gg |-> (gares(gg) \ / {xx |->
vv}}})
    END
  END
END;

reparer_velo(velo) =
PRE
  velo : velos & etatVelos(velo) /= bon_etat
THEN
  etatVelos(velo) := bon_etat
END
```

# III. Scénarios de test

## 1er scénario :

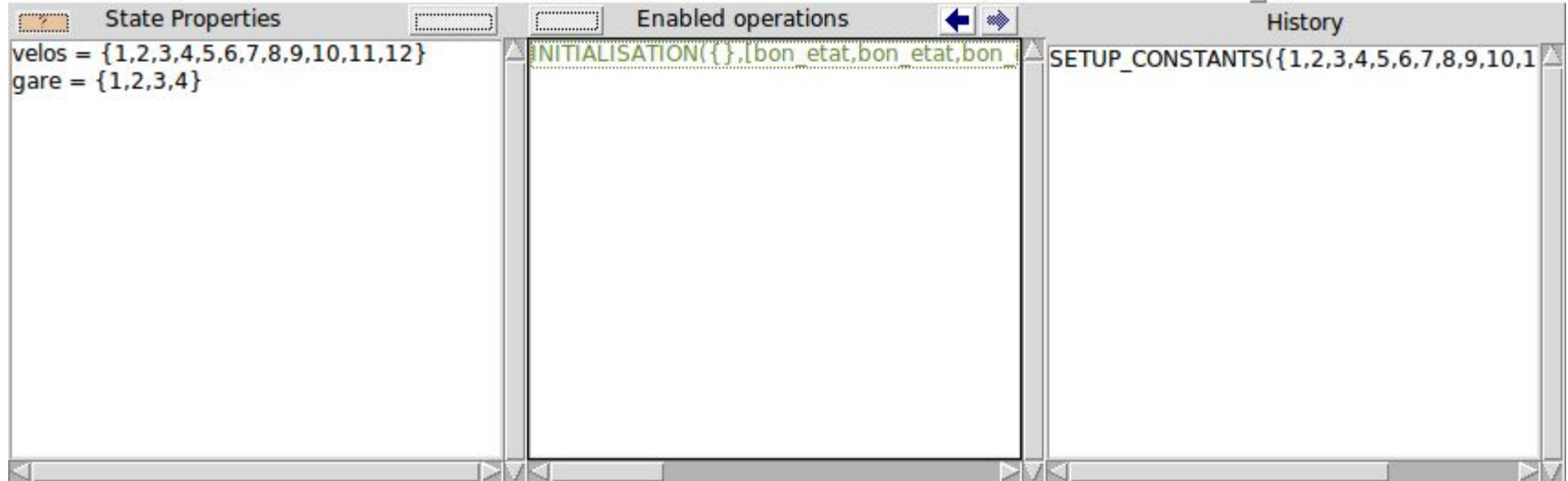
- Initialisation
- Création d'un utilisateur u1
- u1 emprunte le vélo 1 à la gare 1
- u1 rend le vélo 1 à la gare 1 en bon état en 30 minutes
- u1 est supprimé.

### III. Scénarios de test

```
MAXINT = 100  
MININT = -1  
card(PERSONNES) = 2 (assumed for deferre  
card(ETATSVELO) = 3  
card(STATUT) = 2
```

```
SETUP_CONSTANTS({1,2,3,4,5,6,7,8,9,10,1
```

### III. Scénarios de test



### III. Scénarios de test

The screenshot displays a software interface with three main panels. The left panel, titled 'State Properties', contains a list of variables and their values: 'invariant\_ok', 'velos = {1,2,3,4,5,6,7,8,9,10,11,12}', 'gare = {1,2,3,4}', 'usagers = {}', and a series of 'etatVelos' assignments for indices 1 through 11, all set to 'bon\_etat'. The middle panel, titled 'Enabled operations', shows a sequence of operations: 'creer\_usager(PERSONNES1)', 'creer\_usager(PERSONNES2)', and ten 'deplacer\_velo' calls with varying parameters. The right panel, titled 'History', shows the initial state: 'INITIALISATION({},{bon\_etat,bon\_etat,bon\_etat})' and 'SETUP\_CONSTANTS({1,2,3,4,5,6,7,8,9,10,11})'. The interface includes an 'OK' button in the top left and a 'max' button in the top middle.

State Properties	Enabled operations	History
invariant_ok	creer_usager(PERSONNES1)	INITIALISATION({},{bon_etat,bon_etat,bon_etat})
velos = {1,2,3,4,5,6,7,8,9,10,11,12}	creer_usager(PERSONNES2)	SETUP_CONSTANTS({1,2,3,4,5,6,7,8,9,10,11})
gare = {1,2,3,4}	deplacer_velo(4,1)	
usagers = {}	deplacer_velo(4,1)	
etatVelos(1) = bon_etat	deplacer_velo(4,1)	
etatVelos(2) = bon_etat	deplacer_velo(5,1)	
etatVelos(3) = bon_etat	deplacer_velo(5,1)	
etatVelos(4) = bon_etat	deplacer_velo(5,1)	
etatVelos(5) = bon_etat	deplacer_velo(6,1)	
etatVelos(6) = bon_etat	deplacer_velo(6,1)	
etatVelos(7) = bon_etat	deplacer_velo(6,1)	
etatVelos(8) = bon_etat	deplacer_velo(6,1)	
etatVelos(9) = bon_etat	deplacer_velo(7,1)	
etatVelos(10) = bon_etat		
etatVelos(11) = bon_etat		

### III. Scénarios de test

The screenshot displays a software interface with three main panes:

- State Properties:** Contains the following text:

```
invariant_ok
velos = {1,2,3,4,5,6,7,8,9,10,11,12}
gare = {1,2,3,4}
usagers(PERSONNES1) = actif
etatVelos(1) = bon_etat
etatVelos(2) = bon_etat
etatVelos(3) = bon_etat
etatVelos(4) = bon_etat
etatVelos(5) = bon_etat
etatVelos(6) = bon_etat
etatVelos(7) = bon_etat
etatVelos(8) = bon_etat
etatVelos(9) = bon_etat
etatVelos(10) = bon_etat
etatVelos(11) = bon_etat
```
- Enabled operations:** Contains a list of operations, with the first one highlighted:

```
creer_usager(PERSONNES2)
supprimer_usager(PERSONNES1)
emprunter(PERSONNES1,1,1)
emprunter(PERSONNES1,2,1)
emprunter(PERSONNES1,3,1)
emprunter(PERSONNES1,4,2)
emprunter(PERSONNES1,5,2)
emprunter(PERSONNES1,6,2)
emprunter(PERSONNES1,7,3)
emprunter(PERSONNES1,8,3)
emprunter(PERSONNES1,9,3)
emprunter(PERSONNES1,10,4)
deplacer_velo(4,1)
deplacer_velo(4,1)
deplacer_velo(4,1)
```
- History:** Contains a list of operations:

```
creer_usager(PERSONNES1)
INITIALISATION({},[bon_etat,bon_etat,bon_
SETUP_CONSTANTS({1,2,3,4,5,6,7,8,9,10,1
```

### III. Scénarios de test

OK State Properties

```
invariant_ok  
velos = {1,2,3,4,5,6,7,8,9,10,11,12}  
gare = {1,2,3,4}  
usagers(PERSONNES1) = actif  
etatVelos(1) = bon_etat  
etatVelos(2) = bon_etat  
etatVelos(3) = bon_etat  
etatVelos(4) = bon_etat  
etatVelos(5) = bon_etat  
etatVelos(6) = bon_etat  
etatVelos(7) = bon_etat  
etatVelos(8) = bon_etat  
etatVelos(9) = bon_etat  
etatVelos(10) = bon_etat  
etatVelos(11) = bon_etat  
etatVelos(12) = bon_etat  
usagerVelo(PERSONNES1) = 1  
gares(1) = {(2|->2),(3|->3)}  
gares(2) = [4,5,6]  
gares(3) = [7,8,9]  
gares(4) = [10,11,12]
```

Enter Parameters for rendre

Parameters for rendre

velo:

1

gg:

1

temps:

30

etat:

bon\_etat

Additional PRE:

Cancel Execute



### III. Scénarios de test

The screenshot displays a software interface with three main panels. The left panel, titled 'State Properties', contains a list of variables and their values. The middle panel, titled 'Enabled operations', shows a list of operations with 'supprimer usager(PERSONNES1)' highlighted. The right panel, titled 'History', shows a sequence of operations performed.

**State Properties**

```
invariant_ok
velos = {1,2,3,4,5,6,7,8,9,10,11,12}
gare = {1,2,3,4}
usagers(PERSONNES1) = actif
etatVelos(1) = bon_etat
etatVelos(2) = bon_etat
etatVelos(3) = bon_etat
etatVelos(4) = bon_etat
etatVelos(5) = bon_etat
etatVelos(6) = bon_etat
etatVelos(7) = bon_etat
etatVelos(8) = bon_etat
etatVelos(9) = bon_etat
etatVelos(10) = bon_etat
etatVelos(11) = bon_etat
etatVelos(12) = bon_etat
usagerVelo = {}
gares(1) = [1,2,3]
gares(2) = [4,5,6]
gares(3) = [7,8,9]
gares(4) = [10,11,12]
```

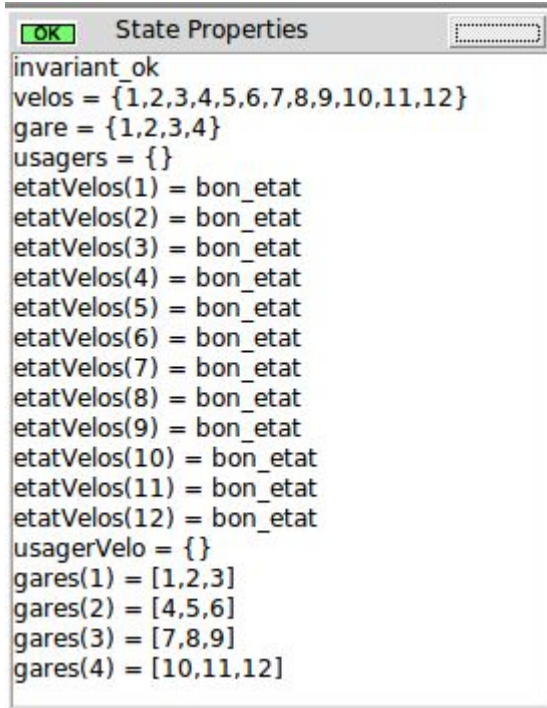
**Enabled operations**

```
creer_usager(PERSONNES2)
supprimer_usager(PERSONNES1)
emprunter(PERSONNES1,1,1)
emprunter(PERSONNES1,2,1)
emprunter(PERSONNES1,3,1)
emprunter(PERSONNES1,4,2)
emprunter(PERSONNES1,5,2)
emprunter(PERSONNES1,6,2)
emprunter(PERSONNES1,7,3)
emprunter(PERSONNES1,8,3)
emprunter(PERSONNES1,9,3)
emprunter(PERSONNES1,10,4)
deplacer_velo(4,1)
deplacer_velo(4,1)
deplacer_velo(4,1)
deplacer_velo(5,1)
deplacer_velo(5,1)
deplacer_velo(5,1)
deplacer_velo(6,1)
deplacer_velo(6,1)
deplacer_velo(6,1)
deplacer_velo(7,1)
```

**History**

```
rendre(1,1,30,bon_etat)
emprunter(PERSONNES1,1,1)
creer_usager(PERSONNES1)
INITIALISATION({},[bon_etat,bon_etat,bon_etat,bon_etat])
SETUP_CONSTANTS({1,2,3,4,5,6,7,8,9,10,11,12},{1,2,3,4})
```

### III. Scénarios de test



```
invariant_ok
velos = {1,2,3,4,5,6,7,8,9,10,11,12}
gare = {1,2,3,4}
usagers = {}
etatVelos(1) = bon_etat
etatVelos(2) = bon_etat
etatVelos(3) = bon_etat
etatVelos(4) = bon_etat
etatVelos(5) = bon_etat
etatVelos(6) = bon_etat
etatVelos(7) = bon_etat
etatVelos(8) = bon_etat
etatVelos(9) = bon_etat
etatVelos(10) = bon_etat
etatVelos(11) = bon_etat
etatVelos(12) = bon_etat
usagerVelo = {}
gares(1) = [1,2,3]
gares(2) = [4,5,6]
gares(3) = [7,8,9]
gares(4) = [10,11,12]
```

### III. Scénarios de test

#### 2ème scénario :

Test du blocage d'un usager (vélo rendu abîmé), son déblocage et la réparation du vélo (vérification qu'on peut encore l'utiliser après).

#### 3ème scénario :

Test du blocage d'un usager (pour avoir rendu du vélo trop tard) et test du déplacement d'un vélo d'une gare à une autre.

## IV. Raffinements

### Raffinement de la variable etatVelos en etats

etats : ETATSVELO --> FIN(velos)

→ Ensemble des vélos en bon état : etats(bon\_etat)

L'invariant de collage est :

#### **INVARIANT**

```
etats : ETATSVELO --> FIN(velos) & etatVelos~[{bon_etat}] = etats(bon_etat) &  
etatVelos~[{use}] = etats(use) & etatVelos~[{abime}] = etats(abime)
```

Principales modifications dans les opérations rendre et reparer\_velo

## IV. Raffinements

### Raffinement de la variable etatVelos en etats : opération rendre

```
46     rendre(velo,gg,temps,etat) =
47-     PRE
48         velo : velos & gg : gare & velo : ran(usagerVelo_r) & temps : NAT1 & etat :
ETATSVELO
49-     THEN
50-         ANY xx WHERE xx : 1 .. 6 & xx /: dom(gares_r(gg)) THEN
51             usagerVelo_r := usagerVelo_r |>> {velo} ||
52-             SELECT etat = use THEN
53                 etats := {bon_etat |-> (etats(bon_etat) - {velo}), use |-> (etats(use)
\ / {velo}), abime |-> etats(abime))}
54-             WHEN etat = abime THEN
55                 etats := {bon_etat |-> (etats(bon_etat) - {velo}), use |-> etats(use),
abime |-> (etats(abime) \ / {velo})}
56             END ||
57             gares_r(gg) := gares_r(gg) \ / {xx |-> velo} ||
58-             IF temps > 60 or etat = abime THEN
59                 usagers_r := usagers_r <+ usagerVelo_r~[{velo}] * {bloque}
60             END
61         END
62     END;
```

## IV. Raffinements

Raffinement de la variable etatVelos en etats : opération reparer\_velo

```
77      reparer_velo(velo) =  
78 -    PRE  
79      velo : velos & velo /\ etats(bon_etat)  
80 -    THEN  
81      etats := {bon_etat |-> (etats(bon_etat) \/{velo}), use |-> (etats(use) -  
82 {velo}), abime |-> (etats(abime) - {velo})}  
      END
```

## IV. Raffinements

### Raffinement de la variable usagers en statuts

statuts : STATUT  $\rightarrow$  FIN(PERSONNES)

→ Ensemble des usagers bloqués : statuts(bloque)

L'invariant de collage est :

#### **INVARIANT**

statuts : STATUT  $\rightarrow$  FIN(PERSONNES) & statuts(aktif) = usagers~[{aktif}] &  
statuts(bloque) = usagers~[{bloque}] &

Principales modifications dans les opérations de création/suppression/déblocage d'un utilisateur et dans l'opération rendre

## IV. Raffinements

Raffinement de la variable etatVelos en etats : opération creer\_usager

```
19   creer_usager(user) =  
20 -   PRE  
21       user : PERSONNES & user /\ statuts(aktif) & user /\ statuts(bloque)  
22 -   THEN  
23       statuts(aktif) := statuts(aktif) \/{user}  
24   END;
```






## IV. Raffinements

Raffinement de la variable etatVelos en etats : opération rendre

```
48     rendre(velo,gg,temps,etat) =
49 -   PRE
50       velo : velos & gg : gare & velo : ran(usagerVelo_r) & temps : NAT1 & etat :
ETATSVELO
51 -   THEN
52 -     ANY xx WHERE xx : 1 .. 6 & xx /: dom(gares_r(gg)) THEN
53       usagerVelo_r := usagerVelo_r |>> {velo} ||
54       etatVelos_r := etatVelos_r <+ { velo |-> etat } ||
55       gares_r(gg) := gares_r(gg) \/{xx |-> velo} ||
56 -     IF temps > 60 or etat = abime THEN
57       statuts := {actif |-> (statuts(actif) - usagerVelo_r~[{velo}]),bloque |-
> (statuts(bloque) \/{usagerVelo_r~[{velo}])}
58     END
59   END
60 END;
```

# V. Preuves

## Résumé

Composant ▼	Typage vérifié	OPs générées	Obligations de Preuve	Prouvé	Non-prouvé	B0 Vérifié
 Velhop	OK	OK	50	44	6	-
 Velhop_r_users	OK	OK	76	64	12	-
 Velhop_r_velo	OK	OK	137	124	13	-

Velhop : 88% prouvé

Velhop\_r\_users : 84% prouvé

Velhop\_r\_velo : 90% prouvé

# V. Preuves

## Méthode :

- Vérification de type, génération des POs, Preuve automatique
- Preuve interactive : dd, mp puis pp principalement

Les preuves non vérifiées sont dues à un manque de temps.

# V. Preuves

## Exemple de preuve vérifiée :

The screenshot displays a software verification tool interface with two main panels: 'Preuve' (Proof) and 'Situation' (Situation).

**Preuve Panel:**

- Tree structure:
  - Force(0)
    - dd
      - mp
      - pr

- Text area content:

```
-----  
      ""Check that the invariant (usagers: PERSONNES +-> STATUT) is  
preserved by the operation - ref 3.4'" => usagers\/{user|->actif}:  
PERSONNES +-> STATUT  
-----  
      usagers\/{user|->actif}: PERSONNES +-> STATUT  
-----  
      ""Check that the invariant (usagers: PERSONNES +-> STATUT) is  
preserved by the operation - ref 3.4'" => usagers\/{user|->actif}:  
PERSONNES +-> STATUT
```

**Situation Panel:**

- Checkbox: ☐ Afficher seulement les OPs non prouvées
- Text field: OPs récemment prouvées
- Tree structure:
  - Initialisation (marked with a red X)
  - creer\_usager (marked with a green checkmark)
    - PO1 (highlighted in blue)
    - PO2
    - PO3

# V. Preuves

## Exemple de preuve non vérifiée :

The screenshot displays a proof assistant interface with two main panels on the left and a large text area on the right.

**Preuve Panel:** Contains a tree view with a single node labeled "Force(0)" and a sub-label "Next".

**Situation Panel:** Includes a checkbox labeled "Afficher seulement les OPs non prouvées" which is currently unchecked. Below it is a list box titled "OPs récemment prouvées" containing four items: "rendre" (with a red 'x' icon), "PO1" (with a green checkmark), "PO2" (with a green checkmark), "PO3" (with a green checkmark), and "PO4" (with a red 'x' icon and highlighted in blue).

**Main Text Area:** Contains the following text:

```
not(xx: dom(gares_r$l(gg))) &  
61<=temps &  
  ``Check that the invariant (gares_r = gares) is preserved by  
the operation - ref 4.4, 5.5'' & ``Check operation refinement - ref  
4.4, 5.5''  
=>  
  #(xx$0).(xx$0: 1..6 & not(xx$0: dom(gares(gg))) &  
gares_r$l<+{gg|->(gares_r$l(gg)\/{xx|->velo}}) = gares<+{gg|-  
>(gares(gg)\/{xx$0|->velo}}))
```

## VI. Conclusion

- La machine Velhop est fonctionnelle (testée avec ProB)
- Quelques OPs restent non prouvées pour Velhop et les 2 raffinements
- Principales difficultés : manque de temps, preuves compliquées