COMPTE-RENDU

TP numéro 6 [LO22]

31 MAI 2016

Marion Chan & Laviolette Etienne

Chan Marion
Laviolette Etienne

Le 28 Mars 2016

Rappel des objectifs du TP :

L'objectif de ce TP est de nous familiariser avec l'outil « Atelier B ». L'atelier B est un atelier de génie logiciel disponible gratuitement qui permet de développer des logiciels prouvés sans défaut. L'objectif, grâce à ce logiciel, est de nous faire appréhender la puissance de la méthode formelle B à l'aide d'un logiciel facile d'accès, très puissant et réutilisable aisément.

Dans ce TP numéro 6 nous avons tout d'abord pris en main le logiciel proposé grâce à un tutoriel détaillé pour comprendre le fonctionnement de l'Atelier B. Une fois les bases acquises nous les avons mises à contribution dans le cadre d'un exemple.

I. Création d'un nouveau projet et réalisation du tutoriel

Le début de notre tutoriel introductif commençait à l'étape 1.4 du PDF : « TP_Methode_B_pour_tous », ainsi nous ne détaillerons pas les manipulations que nous avons effectué avant cela pour découvrir le logiciel et apprendre à manier son interface.

4 . Création d'un nouveau projet

1. Tout d'abord, le dossier « trad » va contenir la «transcription » de nos implémentations réalisées en langage B vers le code en C.
   Le dossier « bdp », quant à lui, va contenir, au fur et à mesure du projet, tous les fichiers créées lors de l'analyse de nos différentes machines et implémentations. Ainsi nous retrouverons ici les obligations de preuve, les preuves de ces obligations et d'autres éléments comme les librairies.
   Le dossier « scpe », pour sa part, contiendra les informations spéciales qui concernent les spécifications des différentes machines de notre projet.

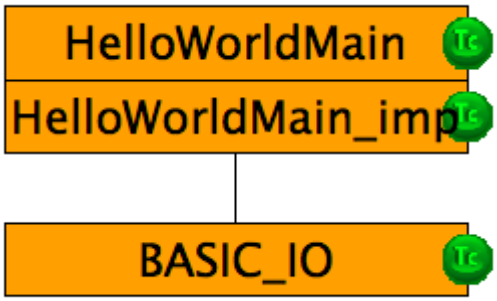
2. Voir projet

3. Les fichiers .mch sont des fichiers dont l'extension correspond aux fichiers machines. De l'autre côté, les fichiers .imp sont des fichiers contenant le code d'implémentation des machines.

5. Rappel

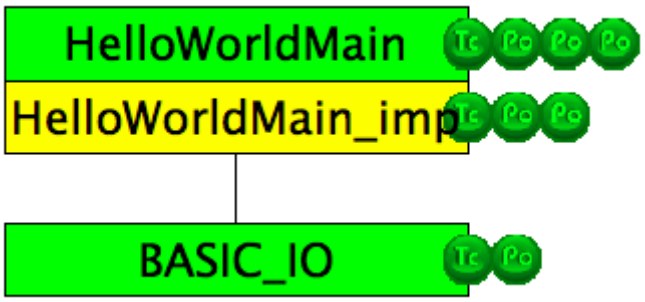
Nous avons ajouté notre machine et son implémentation puis la librairie BASIC_IO nécessaires pour ce tutoriel

4. Grâce à l'outil de vérification de type de l'Atelier B nous avons pu corriger des erreurs dans la syntaxe et ainsi vérifier le type de nos composants

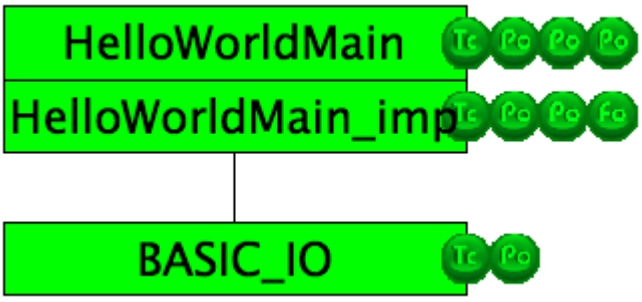


Machines et implémentations nécessaires à la réalisation de la partie tutoriel

5. L'instruction « skip » est une instruction très pratique qui est toujours VRAIE et surtout ne modifie pas l'état de la machine.
   BASIC_IO est une librairie libre et qui sert notamment à l'affichage, elle est nécessaire dans le cadre de la réalisation de ce tutoriel.

6. 

Génération des obligations de preuve

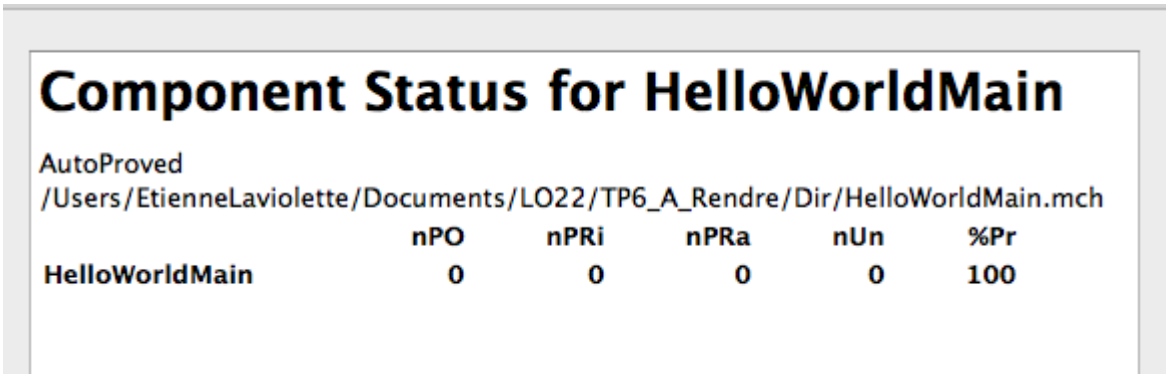
7. 

8. Le code en langage C se trouve dans le dossier trad de notre répertoire « Dir ». On peut étudier le code du fichier HelloWorldMain_imp.c. Celui-ci compile correctement et affiche ce que l'on souhaite : HelloWorld

```c
#ifndef _HelloWorldMain_h
#define _HelloWorldMain_h

#include <stdint.h>
#include <stdbool.h>
#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */


/* Clause SETS */

/* Clause CONCRETE_VARIABLES */


/* Clause CONCRETE_CONSTANTS */
/* Basic constants */
/* Array and record constants */
extern void HelloWorldMain__INITIALISATION(void);

/* Clause OPERATIONS */

extern void HelloWorldMain__afficher(void);

#ifdef __cplusplus
}
#endif /* __cplusplus */


#endif /* _HelloWorldMain_h */
```

Code .c issu de la génération de code de l'Atelier B

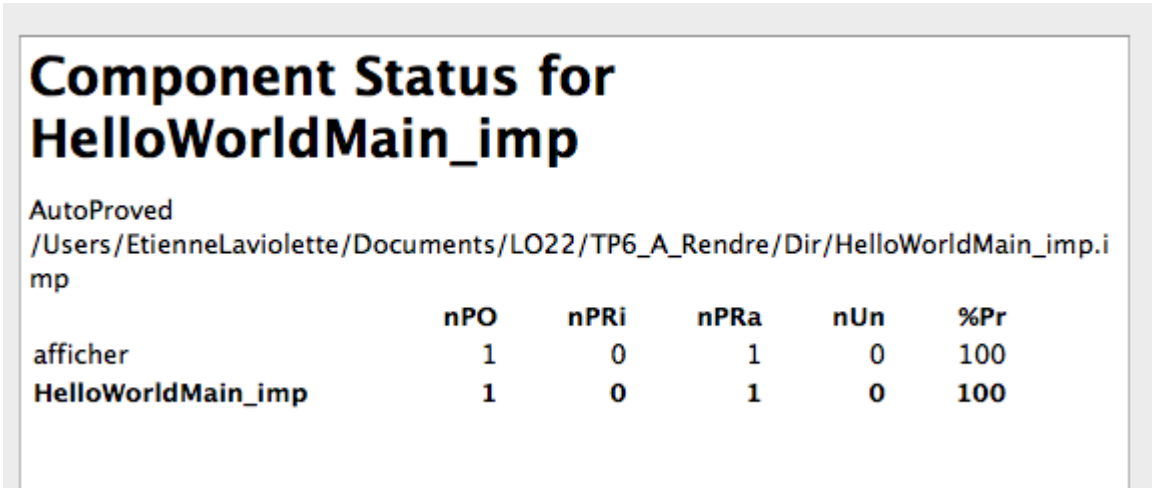
9.



Component Status for HelloWorldMain

AutoProved
/Users/EtienneLaviolette/Documents/LO22/TP6_A_Rendre/Dir/HelloWorldMain.mch

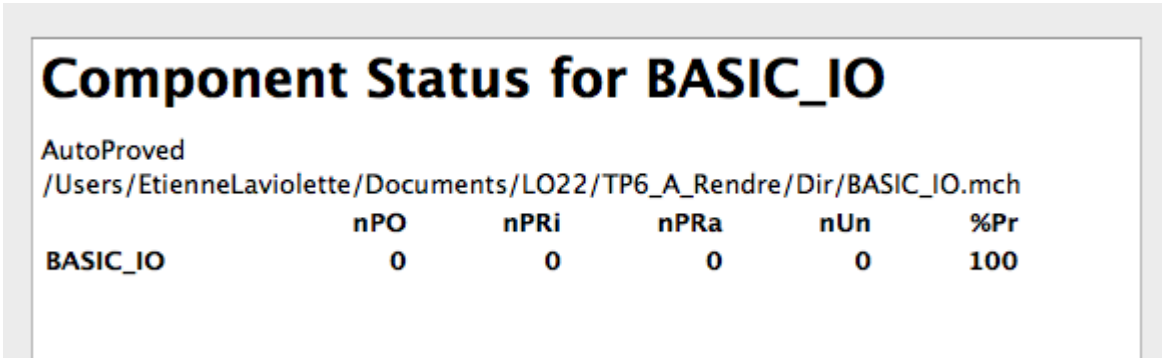
	nPO	nPRi	nPRa	nUn	%Pr
HelloWorldMain	**0**	**0**	**0**	**0**	**100**



Component Status for HelloWorldMain_imp

AutoProved
/Users/EtienneLaviolette/Documents/LO22/TP6_A_Rendre/Dir/HelloWorldMain_imp.imp

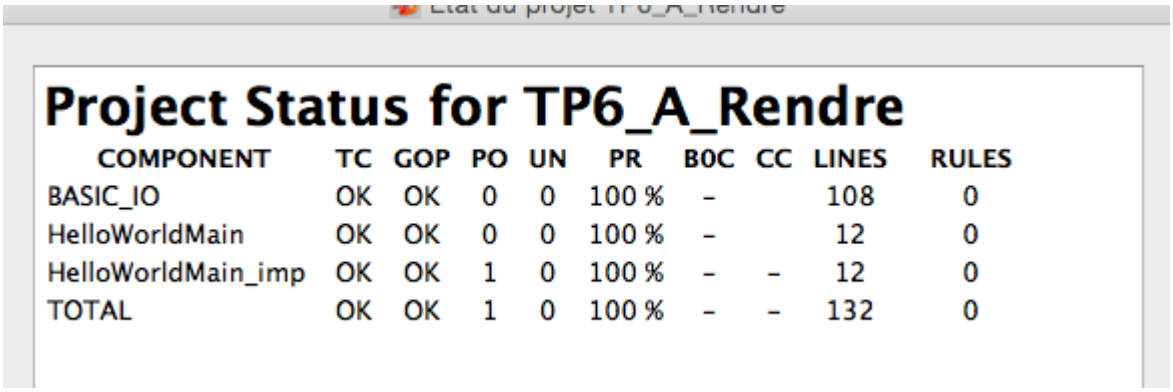
	nPO	nPRi	nPRa	nUn	%Pr
afficher	1	0	1	0	100
HelloWorldMain_imp	**1**	**0**	**1**	**0**	**100**



Component Status for BASIC_IO

AutoProved
/Users/EtienneLaviolette/Documents/LO22/TP6_A_Rendre/Dir/BASIC_IO.mch

	nPO	nPRi	nPRa	nUn	%Pr
BASIC_IO	**0**	**0**	**0**	**0**	**100**

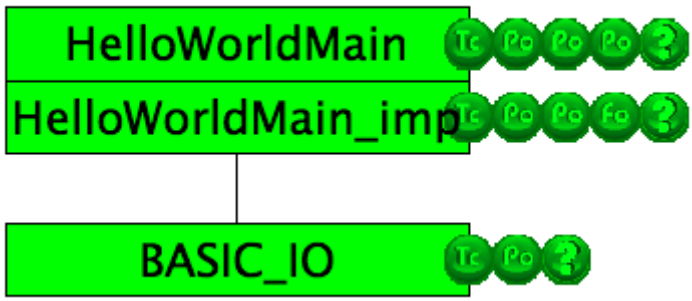


Project Status for TP6_A_Rendre

COMPONENT	TC	GOP	PO	UN	PR	B0C	CC	LINES	RULES
BASIC_IO	OK	OK	0	0	100 %	-		108	0
HelloWorldMain	OK	OK	0	0	100 %	-		12	0
HelloWorldMain_imp	OK	OK	1	0	100 %	-	-	12	0
TOTAL	OK	OK	1	0	100 %	-	-	132	0

Les différentes abréviations utilisées par l'Atelier B ont la signification suivante :

- TC : Etat de l'analyse syntaxique et du contrôle de type (OK = succès)
- GOP : Etat de l'obligation des preuves (OK génération avec succès)
- PO : Nombre de preuves
- UN : Nombre de preuves non prouvées
- %PR : Pourcentage des obligations de preuve qui ont été prouvée
- BOC : Etat du contrôle du langage d'implémentation (OK : succès)
- Lines : Nombre de lignes
- nPO : Nombre d'obligations de preuves non triviales
- nUn : Nombre d'obligations de preuve non encore prouvées
- nPRi : Nombre de preuve prouvées avec prouveur interactif
- nPRa : Nombre de preuve prouvées automatiquement



Etat de notre projet après avoir réalisation les preuves

Projet	Composant	Action	Statut	Messages	Serveur
TP6_A_R...	BASIC_IO		Fini	Vérification de type terminée	localhost
TP6_A_R...	HelloWorld...		Fini	Vérification de type terminée	localhost
TP6_A_R...	HelloWorld...		Fini	Vérification de type terminée	localhost
TP6_A_R...	HelloWorld...		Fini	0 obligations de preuve générées.	localhost
TP6_A_R...	HelloWorld...		Fini	1 obligations de preuve générées.	localhost
TP6_A_R...	HelloWorld...		Fini	Proof obligations for HelloWorldMain have already been generated	localhost
TP6_A_R...	BASIC_IO		Fini	0 obligations de preuve générées.	localhost
TP6_A_R...	HelloWorld...		Fini	Proof obligations for HelloWorldMain_imp have already been generated	localhost
TP6_A_R...	HelloWorld...		Fini	Proof obligations for HelloWorldMain have already been generated	localhost
TP6_A_R...	HelloWorld...		Fini	End of Proof	localhost
TP6_A_R...	BASIC_IO	Générati...	Fini	(ComenC2) Translation into C successful	localhost
TP6_A_R...	HelloWorld...	Générati...	Fini	(ComenC2) Translation into C successful	localhost
TP6_A_R...	HelloWorld...	Générati...	Fini	(ComenC2) Translation into C successful	localhost

Historique des actions effectuées

10. Après toute la démarche suivie à la lettre on peut en conclure que notre projet (ici très simple) fonctionne bien. En effet le code C est sans erreur et compile correctement directement après sa génération. D'autre part le statut global du projet semble satisfaisant puisque tous les contrôles et autres obligations ont été réalisées avec succès.

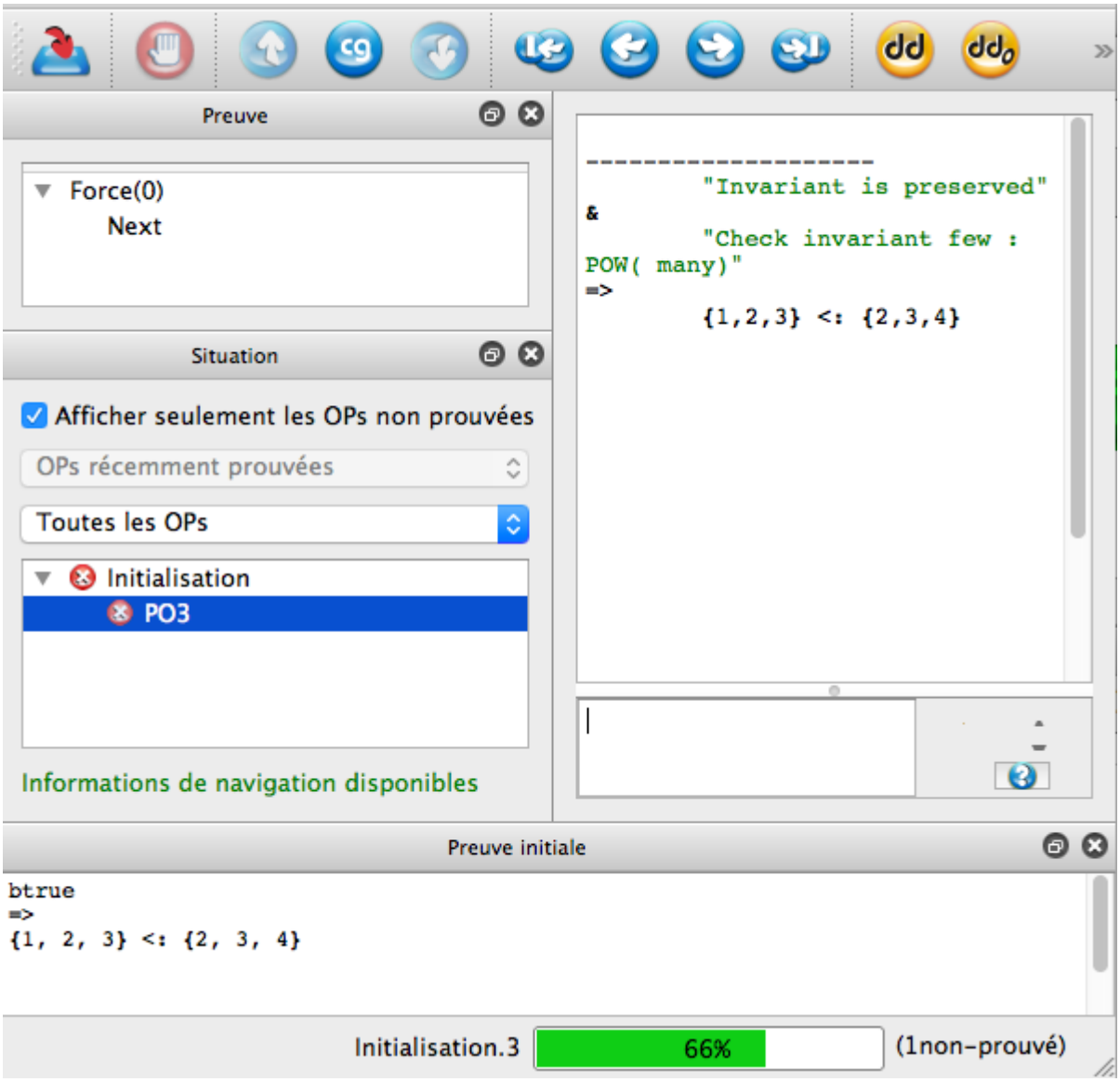
II. Exemple avec prouveur intéractif

Nous démarrons un nouveau projet avec le code fourni dans l'énonce du TP.

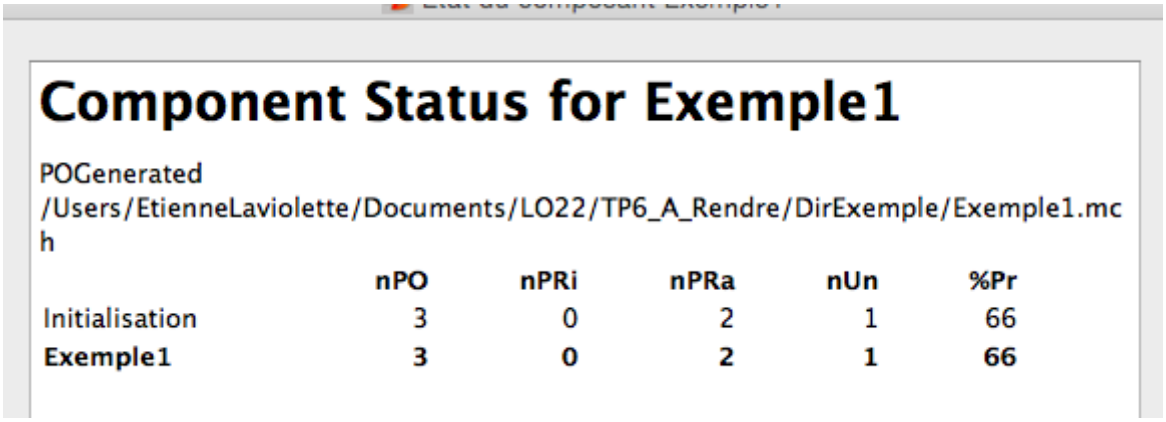
1. 2. 3.

Etat de notre machine après vérification de type, génération des obligations de preuve et prouveur automatique

On remarque ici que le prouveur automatique de force 0 (et meme force 1, 2 ou 3) fait passer notre machine dans un état « jaune » au lieu du vert habituel rencontré dans le tutoriel. Nous en déduisons que l'un des obligations de preuve n'arrive pas à être prouvée. Nous allons utiliser le prouveur interactif puis faire un bilan des statuts pour évaluer si le problème de la force automatique 0 ou d'une erreur dans le code.



Erreur dans le prouveur interactif.



Component Status for Exemple1

POGenerated
/Users/EtienneLaviolette/Documents/LO22/TP6_A_Rendre/DirExemple/Exemple1.mch

	nPO	nPRi	nPRa	nUn	%Pr
Initialisation	3	0	2	1	66
Exemple1	**3**	**0**	**2**	**1**	**66**

Statut de notre machine

4. Comme nous le pressentions, une obligation de preuve n'arrive pas à être prouvée. Seulement 2/3 des obligations de preuves soit 66% sont prouvées. En effet si on étudie le code fourni dans le PDF du TP, on se rend compte que la troisième obligation ne peut pas être prouvée car elle est tout simplement fausse. Une erreur dans le code est présente : le « few » ne peut jamais être inclus dans le « many » et donc l'initialisation de notre machine ne respecte pas une propriété fondamentale : l'invariant.

5. On peut appliquer des modifications et réitérer le processus, voici la démarche et le résultat :

```
/* Exemple1
 * Author: lo22pXXX
 * Creation date: 10/06/2016
 */
MACHINE
    Exemple1
VARIABLES
    few, many
INVARIANT
    few <: NATURAL &
    many <: NATURAL &
    few <: many
INITIALISATION
    few,many := {1,2,3},{1,
2,3,4}
END
```

Modifications réalisées

Etat de notre système après modifications et preuve automatique et interactive (l'interactive n'était pas nécessaire)

Project Status for Exemple

COMPONENT	TC	GOP	PO	UN	PR	B0C	CC	LINES	RULES
Exemple1	OK	OK	3	0	100 %	-		15	0
TOTAL	OK	OK	3	0	100 %	-	OK	15	0

Statut de notre projet après modifications et preuves

Projet	Composant	Action	Statut	Messages	Serveur
Exemple1	Exemple1		Fini	Vérification de type terminée	localhost
Exemple1	Exemple1		Fini	3 obligations de preuve générées.	localhost
Exemple1	Exemple1		Fini	End of Proof	localhost
Exemple1	Exemple1		Fini	Nothing to prove in Exemple1	localhost
Exemple1	Exemple1		Fini	Vérification de type terminée	localhost
Exemple1	Exemple1		Fini	3 obligations de preuve générées.	localhost
Exemple1	Exemple1		Fini	End of Proof	localhost
Exemple	Exemple1		Fini	Vérification de type terminée	localhost
Exemple	Exemple1		Fini	3 obligations de preuve générées.	localhost
Exemple	Exemple1		Fini	End of Proof	localhost
Exemple	Exemple1		Fini	End of Proof	localhost
Exemple	Exemple1		Fini	End of Proof	localhost
Exemple	Exemple1		Fini	End of Proof	localhost
Exemple	Exemple1		Fini		localhost
Exemple	Exemple1		Fini	Vérification de type terminée	localhost
Exemple	Exemple1		Fini	3 obligations de preuve générées.	localhost
Exemple	Exemple1		Fini	End of Proof	localhost

Historique de nos actions

Conclusion :

Même si la méthode formelle semble difficile d'accès a priori nous avons été agréablement surpris de découvrir que le logiciel Atelier B présentait l'avantage d'être facile d'utilisation. Grâce au tutoriel introductif et à notre curiosité nous avons pu comprendre globalement la machinerie mathématique se cachant derrière les obligations de preuves, les preuves de ces obligations et l'utilisation des librairies. Même si l'exemple pour lequel nous avons mis en application nous acquis semblait basic, ce fut une bonne opportunité de nous confronté à la correction d'erreur et à sa correction grâce aux outils de l'Atelier.