

UNIVERSITÉ DE LA RÉUNION

TRAVAIL D'ÉTUDE ET DE RECHERCHE

M1 INFORMATIQUE

DÉPARTEMENT D'INFORMATIQUE

Portage de Mace4/Prover9 vers des technologies web

Auteur :
Kamarouzamane COMBO
N° étudant : 34002711

Encadant :
Pr. Fred MESNARD

Responsable du TER UFR Sciences et Technologies :
Dr. Anilhoussen CASSAM-CHENAI

10 juin 2019

TER de Master 1 Informatique

Remerciements

Tout d'abord, je tiens à remercier mon encadrant de TER Pr Fred MESNARD *pour ses bons conseils qui m'ont orienté dans la bonne direction, pour son incroyable patience, sa disponibilité et son aide sur ce projet. Je remercie également Dr. Anilhoussen* CASSAM-CHENAI *d'avoir rendu ce TER possible.*

K.Combo

Résumé

Pour clôturer la première année d'étude du Master Informatique à l'Université de la Réunion, nous sommes amenés à effectuer un TER durant le deuxième semestre, afin d'apprendre à connaître les conditions de travail en entreprise ou en laboratoire et de mettre en pratique les connaissances acquises au cours du cursus universitaire.
Mon TER consistait à d'utiliser un logiciel de démonstrateur automatique, qui a pour nom *Prover9* développer par William McCune. En utilisant le code source du projet de McCune, il fallait l'intégrer sur une plateforme web.

Mots-clés : Emscripten, Javascript, Mace4, Prover9

Abstract

To complete the first year of study of the Master in Computer Science at the University of Reunion Island, students must complete a TER during the second semester, in order to get to know the working conditions in the company or laboratory and to put into practice the knowledge acquired during the university course.
My TER was to use an automatic demonstrator software, which has the name *Prover9* developed by William McCune. Using McCune's project source code, it had to be integrated on a web platform.

Keywords : Emscripten, Javascript, Mace4, Prover9

Table des matières

1 Introdution

Mon TER s'est déroulé du 1 janvier au 14 juin 2019 au sein du laboratoire du LIM (Laboratoire d'Informatique et de Mathématiques), il est situé au Parc Technologique Universitaire de La Réunion, à Saint Denis.
Durant celui-ci, j'ai pu mettre en œuvre mes compétences techniques acquises au cours de mon cursus universitaire dans plusieurs domaines.
Avec l'aide de l'encadreur (Pr Fred MESNARD), on a compilé le projet de MCCUNE. Le projet se composait de deux démonstrateurs automatique (Prover9 et Mace4).
Par la suite, on a transformé les scriptes Prover9 et Mace4 (qui étaient écrit en C) en Javascript.
Ensuite, on s'également occupé de l'élaboration d'application web. Ainsi, on a utilisé du HTML et du JavaScript. J'appris à travailler avec différents outils que je n'utilisais pas auparavant et j'ai aussi appris à m'autoformer sur des technologies que je ne maîtrisais pas et que je ne connaissais pas pour mener à bien les différentes missions qui m'ont été confiées.

Je vais donc développer, tout au long de ce rapport, les différentes étapes de mon travail. Tout d'abord, j'effectuerai une présentation de l'entreprise, puis dans un deuxième temps, je décrirai les différentes missions qui m'ont été confiées. Pour finir, la dernière partie contiendra le bilan du travail effectué durant le TER.

2 Présentation de l'entreprise d'accueil

J'effectue mon TER de Master 1 au sein du Laboratoire d'Informatique et de Mathématiques (LIM) de l'Université de la Réunion. Le LIM est dirigé par Pr. Jean Diatta et est structuré en trois axes :

- Epistémologie et Didactique de l'Informatique et des Mathématiques (EDIM)
- InformaTique et Applications (ITA)
- Mathématiques (MATHS)

Le LIM développe, à travers ses membres, des partenariats académiques aux niveaux local, régional (zone Océan Indien), national et international, ainsi que des partenariats industriels locaux. Il participe à la formation par la recherche, adosse le Master Informatique et Mathématiques de l'UFR Sciences et Technologies, le département Informatique et Télécommunica-

tions de l'ESIROI (Ecole Supérieure d'Ingénieurs Réunion-Océan Indien), et il co-adosse le master MEEF (Métiers de l'Enseignement, de l'Éducation et de la Formation) de l'ESPE (École Supérieure du Professorat et de l'Éducation). Il est membre de l'Ecole doctorale STS (Sciences Technologies Santé) et de la fédération de recherche OMNCG (Observatoire des milieux naturels et des changements globaux), de l'UR.

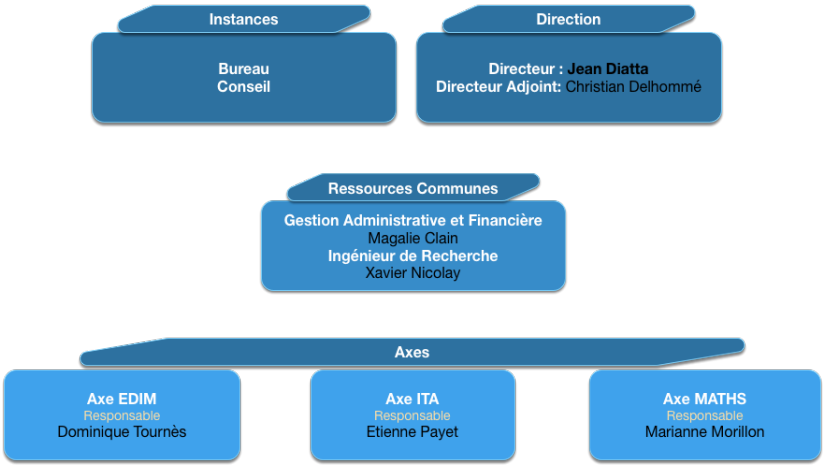


FIGURE 1 – Organigramme du LIM.

3 Présentation du TER

3.1 Les Technologies / Outils utilisées

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. Le langage supporte le paradigme objet, impératif et fonctionnel. JavaScript est le langage possédant le plus large écosystème grâce à son gestionnaire de dépendances npm, avec plus de 350 000 paquets.

HyperText Markup Language (HTML) est le langage de balisage standard pour la création de pages Web et d'applications Web. Avec le CSS et JavaScript, il forme une triade de technologies angulaires pour le World Wide Web. Les navigateurs Web reçoivent des documents HTML d'un serveur web ou d'un stockage local et les rendent en pages Web multimédias.

Bootstrap est un framework CSS, mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript. qui a travers un ensemble d'outils, aide à mettre en forme une page web : organisation, aspect, animation,..

Mozilla Firefox est un navigateur web (webGL) libre et gratuit, développé et distribué par la Mozilla Foundation.

Sublime Text est un éditeur de texte générique codé en C++ et Python, disponible sur Windows, Mac et Linux. Le logiciel a été conçu tout d'abord comme une extension pour Vim, riche en fonctionnalités.

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

Bash est l'acronyme de Bourne-Again shell. C'est un interpréteur en ligne de commande de type script. C'est le shell Unix du projet GNU.

Prover9 et Mace4 : Prover9 est un prouveur de théorème automatisé pour le premier ordre et en logique propositionnelle, et Mace4 recherche des modèles finis et des contre-exemples. Prover9 est le successeur du prouveur Otter.

Emscripten est un compilateur source à source Open Source permettant de compiler du bitcode LLVM en asm.js, qui peut être exécuté par les navigateurs web.

3.2 Travaux réaliés

3.2.1 Compilation native

La première activité qui m'a été confié a été de compiler le projet de William MCCUNE : *Prover9 & Mace4* [3].
Le but de cette compilation est de vérifié le bon fonctionnement de l'application en local, et cela permet aussi de comprendre comment fonction *Prover9* et *Mace4*.
Prover9 a pour objectif de trouver une preuve de n'importe quel énoncé prouvable du logique du premier ordre ou en logique propositionnelle.
Mace4 permet de trouver un contre-exemple en logique du premier ordre ou en logique propositionnelle.

Prover9 & Mace4 prennent un fichier en entrée, dans ce dernier on peut trouver *des axiomes, des lemmes, des théorèmes et des formules*.
Le type de fichier d'entrée le plus basique consiste en une liste de clauses

appelées « sos » représentant la négation de la conjecture, comme dans les exemples ci-dessous.

Exemples de fichier attendu en entrée :

```
formulas(assumptions).
% Mace4 should produce a counterexample in a few seconds.
% Lattice Axioms

x ^ y = y ^ x.
(x ^ y) ^ z = x ^ (y ^ z).
x v y = y v x.
(x v y) v z = x v (y v z).
x v (x ^ y) = x.
x ^ (x v y) = x.

% The following gives us ortholattices.
x1 ^ x = 0.
x1 v x = 1.
x2 = x.
x ^ y = (x1 v y1).

% Ortholattice lemmas.
% 1 v x = 1.
% 1 ^ x = x.

% 0 ^ x = 0.
% 0 v x = x.

% Weak orthomodular law (*1 of mail 68).
(x\prime ^ (x v y)) v (y\prime v (x ^ y)) = 1  # label(mail_68_1).

end_of_list.

formulas(goals).

% Equation in question (*3 of mail 68).
x ^ (y v (x ^ (x\prime v (x ^ y)))) = x ^ (x\prime v (x ^ y)) #
    label(mail_68_3).

end_of_list.
```

FIGURE 2 – Fichier d'entrer pour Mace4

```
formulas(sos).

  e * x = x.
  x\prime * x = e.
  (x * y) * z = x * (y * z).

  x * x = e.

end_of_list.

formulas(goals).

  x * y = y * x.

end_of_list.
```

FIGURE 3 – Fichier d'entrer pour Prover9

Mace4/Prover9 transformera les formules de cette entrée en clauses identiques à celles de l'entrée, avant de commencer la recherche d'une contre-exemple ou d'une preuve.

Exemples de fichier attendu en sortie :

```
======= CLAUSES FOR SEARCH =========

formulas(mace4_clauses).
p | r.
q.
 p |  r.
end_of_list.

======= end of clauses for search =========

% There are no natural numbers in the input.

========== DOMAIN SIZE 2 ============

============ MODEL ===============

interpretation( 2, [number=1, seconds=1], [

```

```
        relation(p, [ 0 ]),

        relation(q, [ 1 ]),

        relation(r, [ 1 ])
]).

========== end of model ==============
```

FIGURE 4 – Fichier de sortie pour Mace4

```
========== SEARCH ==========

% Starting search at 0.00 seconds.

given #1 (wt=5): 6 e * x = x. [input].

given #2 (wt=6): 7 x\prime * x = e. [input].

given #3 (wt=11): 8 (x * y) * z = x * (y * z). [input].

given #4 (wt=5): 9 x * x = e. [input].

given #5 (wt=7): 10 c2 * c1 != c1 * c2. [clausify].

given #6 (wt=8): 11 x\prime * (x * y) = y. [para(7(a,1),8(a,1,1))
    ,demod(6(2)),flip(a)].

given #7 (wt=4): 19 x\prime = x. [back_demod(15),demod(17(4))].

given #8 (wt=5): 20 x * e = x. [back_demod(17),demod(19(1))].

given #9 (wt=7): 12 x * (x * y) = y. [para(9(a,1),8(a,1,1)),demod
    (6(2)),flip(a)].

given #10 (wt=9): 13 x * (y * (x * y)) = e. [para(9(a,1),8(a,1)),
    flip(a)].

given #11 (wt=11): 21 x * (y * (x * (y * z))) = z. [back_demod
    (16),demod(19(2),8(4))].

given #12 (wt=7): 23 x * (y * x) = y. [para(13(a,1),12(a,1,2)),
    demod(20(2)),flip(a)].
```

```
28
29 % Operation * is associative commutative; redundancy checks
       enabled.
30
31 ========== PROOF ============
32
33 % Proof 1 at 0.00 (+ 0.00) seconds.
34 % Length of proof is 15.
35 % Level of proof is 7.
36 % Maximum clause weight is 11.
37 % Given clauses 12.
38
39 6 e * x = x.  [input].
40 7 x \prime * x = e.  [input].
41 8 (x * y) * z = x * (y * z).  [input].
42 9 x * x = e.  [input].
43 10 c2 * c1 != c1 * c2.  [clausify].
44 11 x\prime * (x * y) = y.  [para(7(a,1),8(a,1,1)),demod(6(2)),flip
       (a)].
45 12 x * (x * y) = y.  [para(9(a,1),8(a,1,1)),demod(6(2)),flip(a)].
46 13 x * (y * (x * y)) = e.  [para(9(a,1),8(a,1)),flip(a)].
47 15 x\prime\prime * e = x.  [para(7(a,1),11(a,1,2))].
48 17 x \prime * e = x.  [para(9(a,1),11(a,1,2))].
49 19 x\prime = x.  [back_demod(15),demod(17(4))].
50 20 x * e = x.  [back_demod(17),demod(19(1))].
51 23 x * (y * x) = y.  [para(13(a,1),12(a,1,2)),demod(20(2)),flip(a)
       ].
52 28 x * y = y * x.  [para(23(a,1),12(a,1,2))].
53 29 $F.  [resolve(28,a,10,a)].
54
55 ====== end of proof ==========
```

FIGURE 5 – Fichier de sortie pour Prover9

Voici les points importants concernant les clauses et les formules :

— Les clauses sont un sous-ensemble de formules. Toutes les formules saisies, y compris les clauses, apparaissent dans une liste intitulée *formules(list_name)*.
— Il existe une règle pour distinguer les variables des constantes, car les clauses et autres formules peuvent avoir des variables libres (variables non liées par des quantificateurs). La règle par défaut est que les variables commencent par (minuscules) u à z. Par exemple, dans

la formule P(a,x), le terme a est une constante, et x est une variable.
— Les variables libres dans les clauses et les formules sont supposées être universellement quantifiées au niveau le plus éloigné.
— Les règles d'inférence de Prover9 s'appliquent aux clauses. Si des formules non clausales sont saisies, Prover9 les traduit immédiatement par des conversions NNF et CNF.

3.2.2 Emscripten

Nous avions utilisé l'outils *Emscripten* [2] pour pouvoir transformer le projet *Prover9 & Mace4* [3] qui était écrit en C en JavaScript, pour pouvoir l'intégrer directement sur une interface web.
Emscripten permet aux applications qui sont codées en C/C++ de s'exécutés sur un navigateur.

FIGURE 6 – *Le but d'Emscripten.*

Le compilateur Emscripten prend le code C ou C+++ et le convertit en bitcode LLVM, puis le noyau du compilateur est utilisé pour compiler le bitcode en WASM (web assembly), qui peut être exécuté dans le navigateur.
Pour utiliser Emscripten, on doit installer le compilateur [1] lui-même, donc voici les prérequis :
— Installer Python
— Clone Emscripten de GitHub
— Installer Emscripten
— Créer un programme C/C+++ et le compiler dans le module WASM

Tout d'abord, on doit installer Python, au moment d'écrire ce billet, la dernière version était Python 3.7.2. Il suffit d'aller sur le site web et de trouver l'installateur web, l'installation est simple. On a besoin parce que le noyau Emscripten SDK est en Python.

Git doit également être installé, pour cela on peut le fait se le terminal.

```
git   version
```

maintenant on a besoin d'un dossier pour cloner le compilateur Emscripten. On a créé le dossier «c : \temp\EmscriptenSDK » sur ma machine, puis cd vers ce dossier, puis on lance la commande suivante :

```
git clone https://github.com/juj/emsdk.git
```

Cela téléchargerait le bootstrap, maintenant on doit l'installer. On exécute ensuite les commandes suivantes :

```
cd emsdk
git pull
emsdk install latest
emsdk activate latest
```

donc maintenant on a un compilateur Emscript installé, il faut maintenant exécuter la commande suivante :

```
emsdk_env.bat
```

Ceci devrait ajouter la variable d'environnement au PATH, de sorte qu' on puisse appeler le compilateur même après avoir fermé l'invite de commande.

```
emcc   version
```

emcc –version, nous montre la version du compilateur.

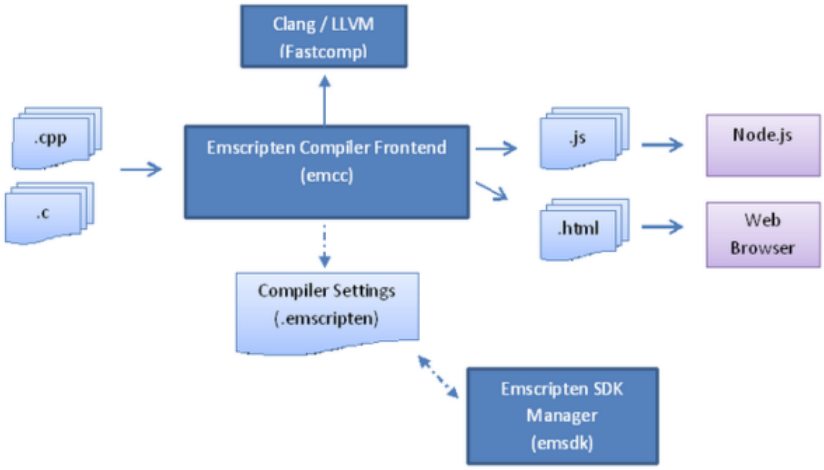


FIGURE 7 – Le fonctionnement d'Emscripten.

Exemple de compilation :

```
1 emcc hello.c -s WASM=1 -o hello.html
```

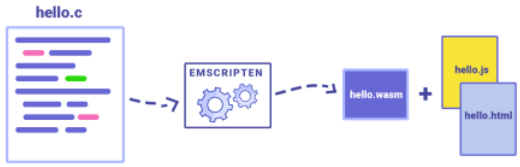


FIGURE 8 – Compiler du code C en WebAssembly avec *Emscripten*.

3.2.3 Migration vers le web

Cette mission consiste à créer une application web en intégrant le script *Mace4 & Prover9*, qui est générée par Emscripten.
L'objectif est d'avoir une plateforme universelle, qui pourrait être uilisé par n'importe quel système d'exploitation.
Dès l'arrivé sur la page d'accueil, on a une courte présentation de *Mace4* et

de *Prover9*.

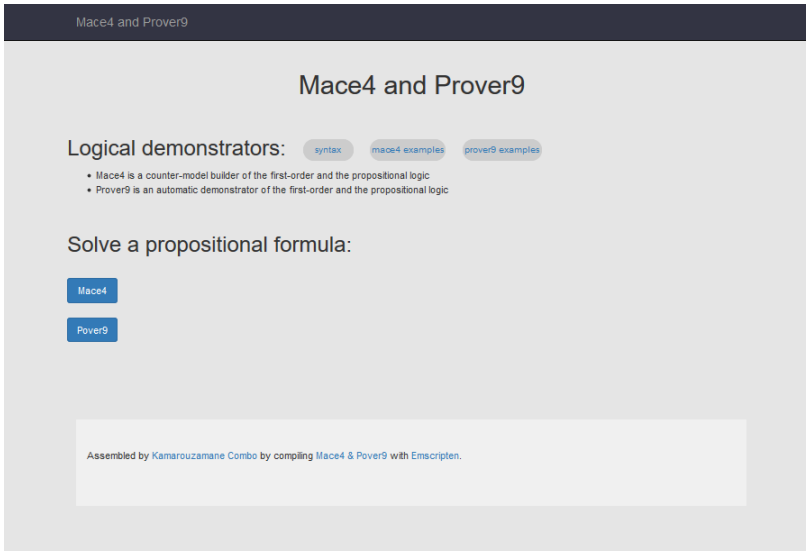
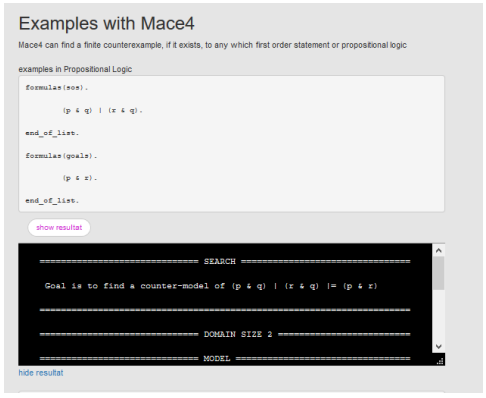


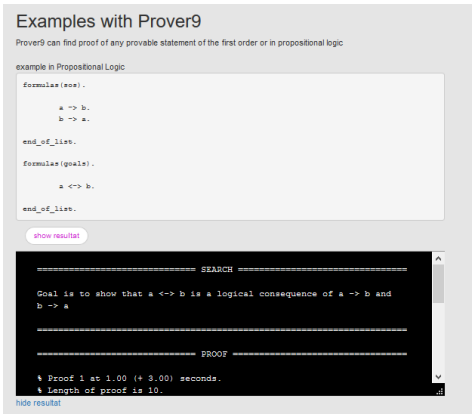
FIGURE 9 – Page d'accueil.

Depuis cette page, utilisateur peut se diriger soit vers une description générale des syntaxes à utiliser ou soit vers des exemple de *Mace4* ou de *Prover9*.

FIGURE 10 – Description de la syntaxe attendue.



(a) Mace4.



(b) Prover9.

FIGURE 11 – Des exemples pour *Mace4* et *Prover9*.

Pour les deux démonstrateurs logique (*Mace4/Prover9*), ils possèdent une zone de saisie (*Inpute*) et une zone d'affichage (*Output*)

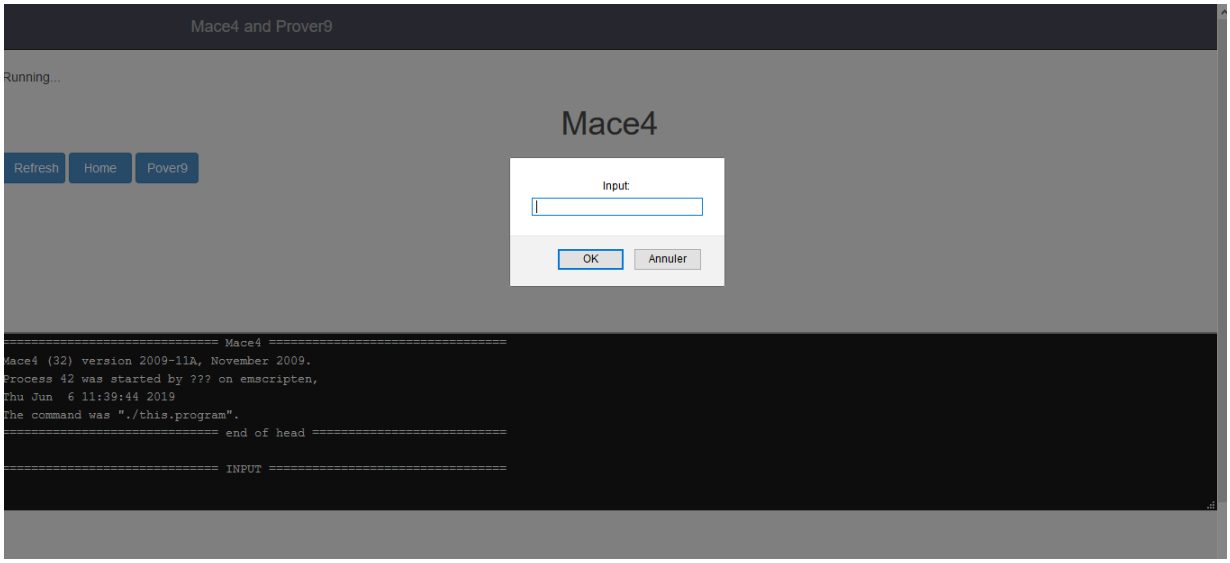


FIGURE 12 – La zone d'*intput*.

FIGURE 13 – La zone d'*output*.

4 Retours critiques

4.1 Expérience du TER

4.1.1 Préambule

Cette partie n'a aucune ambition de dénigrer qui que ce soit. Elle doit simplement me permettre de partager mon expérience du TER.

4.1.2 Difficultés

Le TER en lui même n'a pas présenté de grosses difficultés. Le sujet étant assez modulaire, cela m'a permis de contourner certaines difficultés. Il a fallut faire face aux nombreux échecs dus à la prise en main de l'environnement de développement. Une difficulté qui n'était pas bloquante mais morale fût la frustration face à ces nombreux échecs, aux blocages pendant plusieurs jours et à la pression d'exigence de résultats.

5 Conclusion

J'ai effectué mon TER de Master1 au sein du LIM. Ma mission était de développer une applications web, afin d'obtenir une plateforme universelle pour les démonstrateurs *Mace4* et *Prover9*.
Ce TER a été très enrichissant, il m'a permis d'acquérir une méthode de travail, qui est nécessaire à l'élaboration et la conception d'une application web.

6 Bibliographie

Références

[1] Compilation avec emcripten. https://developer.mozilla.org/fr/docs/WebAssembly/C_to_wasm.

[2] Empscripten. https://emscripten.org.

[3] Mace4 & prover9. https://www.cs.unm.edu/~mccune/mace4/.