

Rendu final de projet génie logiciel : Proost

Arthur Adjedj Augustin Albert
Vincent Lafeychine Lucas Tabary-Maujean

Jean Abou-Samra Tanguy Bozec Antonin Bretagne Vivien Ducros
Antoine Guilmin-Crépon Balthazar Patiachvili

ENS Paris-Saclay

12 janvier 2023



Sommaire

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

➊ Présentation générale

➋ Description de la théorie implémentée

➌ Caractéristiques de Proost

➍ Pour la suite



Présentation du projet

Qu'est-ce que Proost ?

Proost est un assistant de preuve écrit en Rust basé sur le calcul des constructions. À terme, il est prévu que cela soit remplacé par le calcul observationnel des constructions (CC^{obs+}).

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Présentation du projet

Rendu final :
Proost

Projet génie
logiciel

Qu'est-ce que Proost ?

Proost est un assistant de preuve écrit en Rust basé sur le calcul des constructions. À terme, il est prévu que cela soit remplacé par le calcul observationnel des constructions (CC^{obs+}).

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Quel est le langage de preuve ?

Il s'agit de Madeleine, un nouveau langage inspiré de Coq, Lean, ...



Présentation du projet

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Qu'est-ce que Proost ?

Proost est un assistant de preuve écrit en Rust basé sur le calcul des constructions. À terme, il est prévu que cela soit remplacé par le calcul observationnel des constructions ($CC^{\text{obs+}}$).

Quel est le langage de preuve ?

Il s'agit de Madeleine, un nouveau langage inspiré de Coq, Lean, ...

Exemple de code

```
def And: Prop -> Prop -> Prop :=  
  fun A B: Prop => (C: Prop) -> (A -> B -> C) -> C
```



Philosophie du projet

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Points importants

- Isolation du kernel pour assurer la correction de l'implémentation



Philosophie du projet

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Points importants

- Isolation du kernel pour assurer la correction de l'implémentation
- API d'interface avec le noyau



Philosophie du projet

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Points importants

- Isolation du kernel pour assurer la correction de l'implémentation
- API d'interface avec le noyau
- Expérience utilisateur agréable (LSP, coloration syntaxique, toplevel)



Philosophie du projet

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Points importants

- Isolation du kernel pour assurer la correction de l'implémentation
- API d'interface avec le noyau
- Expérience utilisateur agréable (LSP, coloration syntaxique, toplevel)
- Optimisation en temps et en mémoire du type-checking



Sommaire

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

① Présentation générale

② Description de la théorie implémentée

③ Caractéristiques de Proost

④ Pour la suite



Hiérarchie prédictive

Type 0 : Type 1 : ...

$$\frac{\Gamma \vdash A : \text{Type } u \quad \Gamma, x : A \vdash B : \text{Type } v}{\Gamma \vdash \Pi(x : A). B : \text{Type } \max u v}$$

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Univers

Rendu final :
Proost

Projet génie
logiciel

Hiérarchie prédicative

Type 0 : Type 1 : ...

$$\frac{\Gamma \vdash A : \text{Type } u \quad \Gamma, x : A \vdash B : \text{Type } v}{\Gamma \vdash \Pi(x : A). B : \text{Type } \max u v}$$

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Univers imprédictatif proof-irrelevant : Prop

Prop : Type 0

$$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : \text{Prop}}{\Gamma \vdash \Pi(x : A). B : \text{Prop}}$$



Polymorphisme d'univers

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Hiérarchie contenant Prop

Prop : Type 0 : Type 1 : ...

Sort 0 : Sort 1 : Sort 2 : ...

Exemple

```
def id.{u} := fun A: Sort u, x: A => x
```



Types inductifs variés

Types inductifs implémentés

- False et True
- Nat, Eq, ...

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Types inductifs variés

Types inductifs implémentés

- False et True
- Nat, Eq, ...

Exemple d'utilisation de Eq et de Nat :

```
def add := fun x: Nat => Nat_rec.{1}
  (fun _: Nat => Nat) x (fun _ n: Nat => Succ n)

def two := Succ (Succ Zero)
def four := Succ (Succ two)

check Refl.{1} Nat four:
  Eq.{1} Nat four (add two two)
```

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Égalité

Actuellement

Égalité intensionnelle à la MLTT

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Égalité

Actuellement

Égalité intensionnelle à la MLTT

Dans le futur

Égalité observationnelle à la CC^{obs+}

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Sommaire

- 1 Présentation générale
- 2 Description de la théorie implémentée
- 3 Caractéristiques de Proost
- 4 Pour la suite

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Caractéristiques

- Mémoïsation quasi-systématique des calculs sur les termes (WHNF, substitution, *var-shifting*, ...) : soit dans des *hashmaps*, soit dans les termes eux-mêmes ;

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Caractéristiques

- Mémoïsation quasi-systématique des calculs sur les termes (WHNF, substitution, *var-shifting*, ...) : soit dans des *hashmaps*, soit dans les termes eux-mêmes ;
- rendu plausible par la mise en place d'une *arène* (cf. implémentation de DDB) telle que :

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Caractéristiques

- Mémoïsation quasi-systématique des calculs sur les termes (WHNF, substitution, *var-shifting*, ...) : soit dans des *hashmaps*, soit dans les termes eux-mêmes ;
- rendu plausible par la mise en place d'une *arène* (cf. implémentation de DDB) telle que :
 - chaque terme est unique dans l'arène ;

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Caractéristiques

- Mémoïsation quasi-systématique des calculs sur les termes (WHNF, substitution, *var-shifting*, ...) : soit dans des *hashmaps*, soit dans les termes eux-mêmes ;
- rendu plausible par la mise en place d'une *arène* (cf. implémentation de DDB) telle que :
 - chaque terme est unique dans l'arène ;
 - chaque arène est associée de manière unique à une *lifetime* (garanties statiques, ...)

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Caractéristiques

- Mémoïsation quasi-systématique des calculs sur les termes (WHNF, substitution, *var-shifting*, ...) : soit dans des *hashmaps*, soit dans les termes eux-mêmes ;
- rendu plausible par la mise en place d'une *arène* (cf. implémentation de DDB) telle que :
 - chaque terme est unique dans l'arène ;
 - chaque arène est associée de manière unique à une *lifetime* (garanties statiques, ...)
- API *safe* utilisable seule et avec des syntaxes simples (en interne, pour la **construction** de termes : type clôture).

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Toplevel

Caractéristiques

- Accessible via l'invite de commandes

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Toplevel

Caractéristiques

- Accessible via l'invite de commandes
- Coloration syntaxique

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Toplevel

Caractéristiques

- Accessible via l'invite de commandes
- Coloration syntaxique
- Localisation précise et visuelle des erreurs

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Toplevel

Caractéristiques

- Accessible via l'invite de commandes
- Coloration syntaxique
- Localisation précise et visuelle des erreurs

Exemple

```
> check add Zero (fun id x: Prop => x)
X      ^-----^
X function (λNat => (((NatRec.{1}) (λ Nat => Nat)) 1)
X      (λNat => λNat => Succ 1)) Zero
X      expects a term of type Nat,
X      received λ Prop => 1: Prop -> Prop
```

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



LSP

Description

Le LSP s'appelle Tilleul et se compose :

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Description

Le LSP s'appelle Tilleul et se compose :

- d'une partie I/O répartissant les requêtes ;



Description

Le LSP s'appelle Tilleul et se compose :

- d'une partie I/O répartissant les requêtes ;
- d'une partie logique traitant les requêtes.



Description

Le LSP s'appelle Tilleul et se compose :

- d'une partie I/O répartissant les requêtes ;
- d'une partie logique traitant les requêtes.

Avancement

Une preuve de concept est actuellement disponible sur la partie I/O.
Cependant la partie logique n'a pas encore été commencée.



Fonctionnement du projet

Utilisation de GitLab

Le projet est disponible sur le GitLab du CRANS.

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Fonctionnement du projet

Utilisation de GitLab

Le projet est disponible sur le GitLab du CRANS.

Nous effectuons du *peer-reviewing* à chaque *merge request*.

Infrastructure

À chaque modification du projet, une *pipeline* s'exécute et :

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Fonctionnement du projet

Utilisation de GitLab

Le projet est disponible sur le GitLab du CRANS.

Nous effectuons du *peer-reviewing* à chaque *merge request*.

Infrastructure

À chaque modification du projet, une *pipeline* s'exécute et :

- vérifie le formatage et effectue des analyses statiques ;

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Fonctionnement du projet

Rendu final :
Proost

Projet génie
logiciel

Utilisation de GitLab

Le projet est disponible sur le GitLab du CRANS.

Nous effectuons du *peer-reviewing* à chaque *merge request*.

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Infrastructure

À chaque modification du projet, une *pipeline* s'exécute et :

- vérifie le formatage et effectue des analyses statiques ;
- génère des couvertures de tests ;



Fonctionnement du projet

Rendu final :
Proost

Projet génie
logiciel

Utilisation de GitLab

Le projet est disponible sur le GitLab du CRANS.

Nous effectuons du *peer-reviewing* à chaque *merge request*.

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Infrastructure

À chaque modification du projet, une *pipeline* s'exécute et :

- vérifie le formatage et effectue des analyses statiques ;
- génère des couvertures de tests ;
- génère une documentation.



Sommaire

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

① Présentation générale

② Description de la théorie implémentée

③ Caractéristiques de Proost

④ Pour la suite



Projets pour la suite

Dans un futur proche

- Égalité observationnelle

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Projets pour la suite

Dans un futur proche

- Égalité observationnelle
- LSP entièrement fonctionnel

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Projets pour la suite

Dans un futur proche

- Égalité observationnelle
- LSP entièrement fonctionnel
- Espaces de noms importés

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Projets pour la suite

Dans un futur proche

- Égalité observationnelle
- LSP entièrement fonctionnel
- Espaces de noms importés

Dans un second temps

- Types inductifs

Rendu final :
Proost

Projet génie
logiciel

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite



Projets pour la suite

Rendu final :
Proost

Projet génie
logiciel

Dans un futur proche

- Égalité observationnelle
- LSP entièrement fonctionnel
- Espaces de noms importés

Présentation
générale

Description de la
théorie
implémentée

Caractéristiques de
Proost

Pour la suite

Dans un second temps

- Types inductifs
- Unification
- Vérification du kernel avec Creusot.

