Application spontanée de transformations logiques en assistance au raisonnement automatique

Stage de L3

Alexis CARRÉ [1]

Sous la direction de :

Chantal KELLER [2] Louise DUBOIS DE PRISQUE [2]

[1]École Normale Supérieure de Lyon

[2]Laboratoire Méthodes Formelles
Université Paris Saclay

4 septembre 2023 : Soutenance

La preuve formelle

- Une démonstration rigoureuse et systématique
- Des axiomes et une suite d'étapes logiques

- Grande confiance dans la validité de la preuve
- Peut être vérifiée par un ordinateur

- Processus souvent long et sujet à erreurs

Un assistant de preuve

- Un logiciel pour manipuler les preuves formelles
- Coq, Isabelle, Agda, Lean, ...
- Interaction Homme-Machine

- Facilite la création d'une preuve
- Permet de vérifier automatiquement la validité d'une preuve

- Pas de construction automatique de preuve

Exemple : $x + 0 = x$

```
Lemma add_zero : forall x : nat, x + 0 = x.
Proof.
  intros x.
  induction x as [| x' IHx'].
    simpl. reflexivity.
    simpl. rewrite IHx'. reflexivity.
Qed.
```

Un peu d'architecture

Assistant de preuve

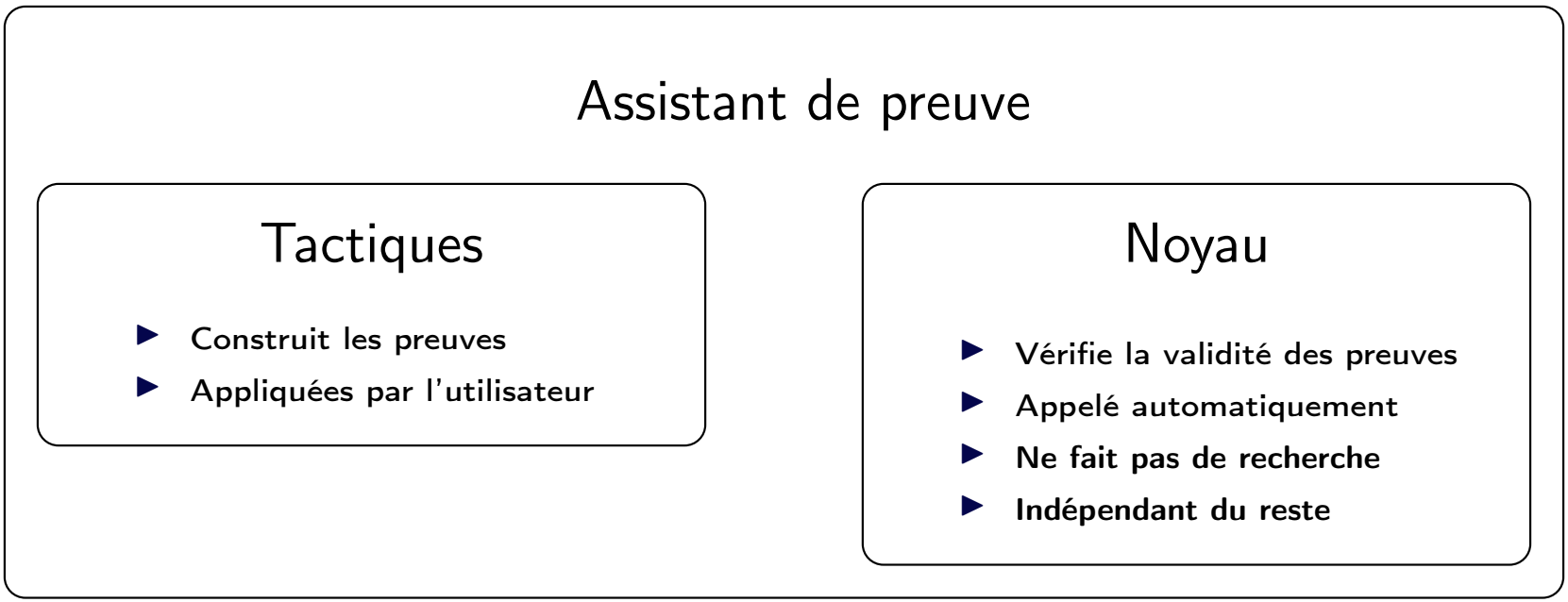
Tactiques

- Construit les preuves
- Appliquées par l'utilisateur

Noyau

- Vérifie la validité des preuves
- Appelé automatiquement
- **Ne fait pas de recherche**
- **Indépendant du reste**

Un peu d'architecture



Architecture simplifiée par rapport à Coq, mais retrouvée dans notre prototype

Automatisation

- Gagner du temps
- Rendre les assistants de preuve plus accessibles

Automatisation

- Gagner du temps
- Rendre les assistants de preuve plus accessibles

```
Require Import Lia.

Lemma add_zero : forall x : nat, x + 0 = x.
Proof.
  intros x.
  lia.
Qed.
```

Toujours plus d'automatisation

Le cas du solveur SMT

- Logiciel à part entière
- Permet de résoudre des problèmes de décision
- Reçoit une formule logique et des contraintes
- Renvoie `SAT`[1], `UNSAT`[2] ou `UNKNOWN`

Exemple de formule :

$$sorted(t, i, j) = \forall i', j'.\, i \leq i' \wedge i' \leq j' \wedge j' \leq j \Rightarrow t[i'] \leq t[j']$$

1. Il existe des valeurs telles que la formule est vraie
2. La formule est toujours fausse

Encore un peu d'architecture

Du point du vue du solveur SMT

$$\forall i', j'.\ \ i \leq i' \ \wedge\ i' \leq j' \ \wedge\ j' \leq j \ \Rightarrow\ t[i'] \leq t[j']$$

Encore un peu d'architecture

Du point du vue du solveur SMT

$$\forall i', j'.\ i \leq i' \ \wedge\ i' \leq j' \ \wedge\ j' \leq j \ \Rightarrow\ t[i'] \leq t[j']$$

- Instantiation

Encore un peu d'architecture

Du point du vue du solveur SMT

$$\forall i', j'.\ i \leq i' \ \wedge\ i' \leq j' \ \wedge\ j' \leq j \ \Rightarrow\ t[i'] \leq t[j']$$

- Instantiation
- Logique

Encore un peu d'architecture

Du point du vue du solveur SMT

$$\forall i', j'.\ i \leq i' \ \wedge\ i' \leq j' \ \wedge\ j' \leq j \ \Rightarrow\ t[i'] \leq t[j']$$

- Instantiation
- Logique
- Théories

Exemple : [1,2,3] est trié

$\neg sorted([1,2,3], 0, 2)$

- $\neg \forall i', j'. i \leq i' \wedge i' \leq j' \wedge j' \leq j \Rightarrow t[i'] \leq t[j']$

Exemple : [1,2,3] est trié

$\neg sorted([1,2,3],0,2)$

- $\neg\forall i', j'.0 \leq i' \wedge i' \leq j' \wedge j' \leq 2 \Rightarrow [1,2,3][i'] \leq [1,2,3][j']$

Exemple : [1,2,3] est trié

$\neg sorted([1,2,3],0,2)$

- $\neg\forall i', j'.0 \leq i' \wedge i' \leq j' \wedge j' \leq 2 \Rightarrow [1,2,3][i'] \leq [1,2,3][j']$
- `UNSAT`

Un problème de communication

Des logiques différentes :

- Assistant Coq : Calcul des constructions inductives
- Solveurs SMT : Logique du 1er ordre

Immédiat :

$\exists x, x + 0 = x$

À traduire :

$\exists R, \forall x, R(x, x)$

Un problème de communication

Des logiques différentes :

- Assistant Coq : Calcul des constructions inductives
- Solveurs SMT : Logique du 1er ordre

Du progrès, des tactiques de traduction :

L. D. de Prisque, C. Keller, V. Blot, ...
Compositional Pre-processing for Automated Reasoning in Dependent Type Theory
CPP 2023.

Objectifs du stage

Un ordre de traduction fixe

- Résultat d'expérimentation
- Changer d'ordre en fonction du contexte ?

Vers un ordre déterminé à la volée

- Détecter les formules traduisibles
- Appliquer automatiquement les tactiques de traduction

Limite d'un ordre fixe

1. Traduction équations

```
H  : forall A, length A [] = 0
H1 : forall A x xs,
       length A (x::xs) = length A xs + 1
----------------------------
length nat [1; 2; 3] = 3
```

2. Traduction polymorphisme

```
H  : length nat [] = 0
H1 : forall x xs,
       length nat (x::xs) = length nat xs + 1
----------------------------
length nat [1; 2; 3] = 3
```

1. Traduction polymorphisme

```
----------------------------
length nat [1; 2; 3] = 3
```

2. Traduction équations

```
H  : forall A, length A [] = 0
H1 : forall A x xs,
       length A (x::xs) = length A xs + 1
----------------------------
length nat [1; 2; 3] = 3
```

Aperçu

`TEq (TGoal, TSomeHyp)`

Le but est-il égal à une hypothèse ?

`TIs (TSomeHyp, TAnd (TVar "A", TVar "B"))`

Existe-t-il une hypothèse de la forme $A \wedge B$?

Syntaxe

```
type trigger_var = TGoal | TSomeHyp

type trigger_form =
  | TVar of string
  | TArr of trigger_form * trigger_form
  | TAnd of trigger_form * trigger_form
  | TOr of trigger_form * trigger_form
  | TTop
  | TBottom
  | TDiscard
  | TMetaVar

type trigger =
  | TEq of trigger_var * trigger_var
  | TIs of trigger_var * trigger_form
  | TContains of trigger_var * trigger_form
```

Sémantique et interprétation

Variables

- `TGoal`
- `TSomeHyp`

Formes

- `TVar`, `TArr`, `TAnd`, `TOr`, `TTop`, `TBottom`
- `TDiscard`
- `TMetaVar`

Autres

- `TEq`
- `TIs`
- `TContains`

Conclusion

- Un langage de spécification
- Une implémentation dans un prototype

Perspectives

- Travaux futurs de Louise
- Implémentation dans Coq avec Ltac2
- Étudier l'ordre et son impact