Un Algorithme de Compilation du Filtrage par Motifs en Coq

Soutenance de Stage de L3

Meven Bertrand

26 juillet 2016

ENS DE LYON

Introduction

Qu'est-ce que c'est que Coq ?

- langage de preuve formelle
- cadre théorique (CIC = Calculus of Inductive Constructions) : lambda-calcul

Qu'est-ce que c'est que le pattern-matching ?

```
match n with
  | O => O
  | S n' => n'
end
```

Introduction

Qu'est-ce que c'est que Coq ?

- langage de preuve formelle
- cadre théorique (CIC = Calculus of Inductive Constructions) : lambda-calcul

Qu'est-ce que c'est que le pattern-matching ?

```
match n with
  | O => O
  | S n' => n'
end
```

Plan

Les Constructeurs

- Variables : Briques de base
- Sortes `Type`, `Prop` et `Set` : Classification des types
- Abstraction $\lambda x : T \cdot M$: Fonctions
- Application $M\ N$: Application de la fonction M au terme N
- Produit dépendant $\forall x : T, M$: Type des fonctions dépendantes

Les Constructeurs

- Variables : Briques de base
- Sortes `Type`, `Prop` et `Set` : Classification des types
- Abstraction $\lambda x : T \cdot M$: Fonctions
- Application $M \ N$: Application de la fonction M au terme N
- Produit dépendant $\forall x : T, M$: Type des fonctions dépendantes

Les Types

Tous le termes de Coq ont un type !
Pour définir le type, on définit une relation par induction.

Contexte

$x_1 : T_1, \ldots, x_n : T_n$: liste de variables, chacune avec un type

Assertion de Typage

$\Gamma \vdash M : T =$ dans le contexte Γ le terme M est bien formé et de type T

Types Inductifs

- Ajout au CoC pour former le Coq actuel
- Construction d'un type à partir de constructeurs définis par l'utilisateur
- Prototype :

$$
\begin{gathered}
I \overrightarrow{a} : \forall \overrightarrow{b} : \overrightarrow{B}, S := \\
|C_1 : \forall \overrightarrow{\phi_1} : \overrightarrow{\Phi_1}, I \overrightarrow{a} \overrightarrow{b_1}(\overrightarrow{\phi_1}) \\
\vdots \\
|C_n : \forall \overrightarrow{\phi_n} : \overrightarrow{\Phi_n}, I \overrightarrow{a} \overrightarrow{b_n}(\overrightarrow{\phi_n})
\end{gathered}
$$

ENS DE LYON

Exemples de Types Inductifs

- ```
  Inductive nat : Set :=
  | O : nat
  | S : nat -> nat.
  ```

- ```
  Inductive vect (A : Type) : nat -> Type :=
  | nil : vect A O
  | cons : forall (a : A) (n : nat), vect A n -> vect A (S n).
  ```

Plan

Motifs

$$p ::= v | C_1 \underbrace{p \dots p}_{r_1} | \dots | C_n \underbrace{p \dots p}_{r_n}$$

terme x unifiable avec motif p si :

- p est une variable
- p est de la forme $C_i p_1 \dots p_{r_i}$, x est de la forme $C_i x_1 \dots x_{r_i}$ et chaque x_j est unifiables avec p_j

Filtrage

Filtrage général

- termes $x_1 \dots x_n$
- matrice de motifs $(p_{i,j})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}}$
- termes $t_1 \dots t_m$
- **objectif** : si j est minimal t.q. pour tout i, x_i est unifiable avec $p_{i,j}$, alors retourner t_j avec remplacements

Filtrage simple

- un seul terme x en entrée
- motifs $(C_i v \dots v)_{1 \leq i \leq k}$

ENS DE LYON

Compilation du Filtrage

Objectif : transformer un filtrage complexe en filtrages simples.
Utilisation de clauses `in` et `return` pour faciliter cette compilation.
Exemples de choses qu'on aimerait savoir compiler :

```
Variables (n : nat) (v : vect nat n).
Definition w :=
match n, v in vect n' return vect (S n') with
| O, nil nat => cons nat 1 O (nil nat)
| S n', cons nat k n' v' => cons nat 2 (S n') (cons nat 1 n')
end.

Variables (n : nat) (v : vect nat (S n)).
Definition w :=
match v in vect (S n') return vect n' with
| cons nat k n' v' => v'
end.
```

Plan

Problème Général

Syntaxe Coq

$$\Gamma \vdash \begin{array}{c} \texttt{match } x_1 \texttt{ as } x'_1 \texttt{ in } I_1\_\overrightarrow{\pi_1}, \ldots, x_n \texttt{ as } x'_n \texttt{ in } I_n\_\overrightarrow{\pi_n} \texttt{ return } P \texttt{ with} \\ \begin{array}{lccc} | & p_{1,1} \ldots p_{1,n} & \Rightarrow & t_1 \\ | & \vdots & \vdots & \vdots \\ | & p_{m,1} \ldots p_{m,n} & \Rightarrow & t_m \end{array} \\ \texttt{end.} \end{array}$$

transcrit sous la forme

$$\Gamma \vdash \begin{array}{cc} (x_1 \ldots x_n) & [\overrightarrow{w_1}, x'_1(\overrightarrow{\pi_1}), \ldots, \overrightarrow{w_n}, x'_n(\overrightarrow{\pi_n}) \vdash P] \\ \begin{pmatrix} p_{1,1} \ldots p_{1,n} \\ \vdots \\ p_{m,1} \ldots p_{m,n} \end{pmatrix} & \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \end{array}$$

Cas Terminal

$$\begin{array}{ccc} & () & [\vdash P] \\ \Gamma \vdash & () & \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \end{array}$$

est compilé en t_1
$+$ vérification de type

Suppression d'une Variable

$$\Gamma \vdash \begin{array}{c} (x_1 \dots x_n) \\ \begin{pmatrix} v_1 \ p_{1,2} \dots p_{1,n} \\ \vdots \\ v_m \ p_{m,2} \dots p_{m,n} \end{pmatrix} \end{array} \quad \begin{array}{c} [\overrightarrow{w_1}, x'_1(\overrightarrow{\pi_1}), \dots, \overrightarrow{w_n}, x'_n(\overrightarrow{\pi_n}) \vdash P] \\ \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \end{array}$$

On aimerait le compiler en

$$\Gamma \vdash \begin{array}{c} (x_2 \dots x_n) \\ \begin{pmatrix} p_{1,2} \dots p_{1,n} \\ \vdots \\ p_{m,2} \dots p_{m,n} \end{pmatrix} \end{array} \quad \begin{array}{c} [\overrightarrow{w_2}, x'_2(\overrightarrow{\pi_2}), \dots, \overrightarrow{w_n}, x'_n(\overrightarrow{\pi_n}) \vdash \ \textbf{?}] \\ \begin{pmatrix} t_1[x_1/v_1] \\ \vdots \\ t_m[x_1/v_m] \end{pmatrix} \end{array}$$

Mais que mettre pour **?** ? La solution à un autre problème de filtrage !

$$\Gamma, \overrightarrow{w_2}, x'_2(\overrightarrow{\pi_2}), \dots, \overrightarrow{w_n}, x'_n(\overrightarrow{\pi_n}) \vdash \begin{array}{c} (\overrightarrow{b_1}) \\ \begin{pmatrix} \overrightarrow{\pi_1} \\ v \end{pmatrix} \end{array} \quad \begin{array}{c} [\vdash S] \\ \begin{pmatrix} P[x_1/x'_1] \\ \text{True} \end{pmatrix} \end{array}$$

ENS DE LYON

Présentation

$$\Gamma \vdash \begin{array}{c} (x_1 \dots x_n) \\ \left(\begin{array}{c} p_{1,1} \dots p_{1,n} \\ \vdots \\ p_{m,1} \dots p_{m,n} \end{array} \right) \end{array} \qquad \begin{array}{c} [\overrightarrow{w_1}, x'_1(\overrightarrow{\pi_1}), \dots, \overrightarrow{w_n}, x'_n(\overrightarrow{\pi_n}) \vdash P] \\ \left(\begin{array}{c} t_1 \\ \vdots \\ t_m \end{array} \right) \end{array}$$

Idée : filtrage simple sur x_1 + classement des patterns par constructeur de tête + un nouveau filtrage à compiler dans chaque branche

Le Filtrage Maître

$$\Gamma \vdash \quad \begin{pmatrix} (x_1) & [\overrightarrow{y_1}, x'_1 \vdash \textbf{?}] \\ \begin{pmatrix} \mathcal{C}_1(z_1, \dots z_{r_1}) \\ \vdots \\ \mathcal{C}_{k'}(z_1 \dots z_{r_{k'}}) \end{pmatrix} & \begin{pmatrix} \textbf{?} \\ \vdots \\ \textbf{?} \end{pmatrix} \end{pmatrix}$$

Le Filtrage Maître

$$\Gamma', \Delta, \Gamma'' \vdash \begin{pmatrix} (x_1) & [\overrightarrow{y_1}, x_1' \vdash \forall \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'', \textbf{?}] \\ \begin{pmatrix} \mathcal{C}_1(z_1, \ldots z_{r_1}) \\ \vdots \\ \mathcal{C}_{k'}(z_1 \ldots z_{r_{k'}}) \end{pmatrix} & \begin{pmatrix} \lambda \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'' \cdot \textbf{?} \\ \vdots \\ \lambda \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'' \cdot \textbf{?} \end{pmatrix} \end{pmatrix} \overrightarrow{\gamma}''$$

Le Type

$$\Gamma, \overrightarrow{y_1} : \overrightarrow{B_1}, x'_1 : I_1 \overrightarrow{a_1} \overrightarrow{y_1}, \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'' \vdash \begin{pmatrix} \left(\overrightarrow{y_1}\ \overrightarrow{b_2} \dots \overrightarrow{b_n}\right) \\ \overrightarrow{\pi_1}\ \overrightarrow{\pi_2} \dots\ \overrightarrow{\pi_n} \\ v \end{pmatrix} \begin{pmatrix} [\vdash S] \\ P \\ \text{True} \end{pmatrix} : S$$

Solution à ce problème : s_T

Le Filtrage Maître

$$\Gamma', \Delta, \Gamma'' \vdash \begin{pmatrix} (x_1) & [\overrightarrow{y_1}, x'_1 \vdash \forall \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'', s_T] \\ \begin{pmatrix} \mathcal{C}_1(z_1, \dots z_{r_1}) \\ \vdots \\ \mathcal{C}_{k'}(z_1 \dots z_{r_{k'}}) \end{pmatrix} & \begin{pmatrix} \lambda \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'' . \; ? \\ \vdots \\ \lambda \overrightarrow{\gamma}'' : \overrightarrow{\Gamma}'' . \; ? \end{pmatrix} \end{pmatrix} \overrightarrow{\gamma}''$$

Les Sous-Problèmes : Problème Intermédiaire

Contexte Γ_i de la branche :

$$\Gamma, \overrightarrow{z} : \overrightarrow{\Phi_i}, \overrightarrow{b_1}' := \overrightarrow{b_{1,i}}(\overrightarrow{z}), x_1' := C_i(z_1 \dots z_{r_i}) : I_1 \overrightarrow{a_1} \overrightarrow{b_1}', \Gamma''[\overrightarrow{b_1}'/\overrightarrow{b_1}][x_1'/x_1]$$

Introduction d'un problème intermédiaire :

$$\Gamma_i \vdash \begin{pmatrix} \overrightarrow{b_1}' \end{pmatrix} \quad [\overrightarrow{b_1}' \vdash s_T]$$
$$\begin{pmatrix} \overrightarrow{\pi_1} \\ v \end{pmatrix} \quad \begin{pmatrix} ? \\ | \end{pmatrix}$$

Les Sous-Problèmes : Problème Intermédiaire

Contexte Γ_i de la branche :

$$\Gamma, \overrightarrow{z} : \overrightarrow{\Phi_i}, \overrightarrow{b_1}' := \overrightarrow{b_{1,i}}(\overrightarrow{z}), x'_1 := C_i(z_1 \dots z_{r_i}) : I_1 \overrightarrow{a_1}\overrightarrow{b_1}', \Gamma''[\overrightarrow{b_1}'/\overrightarrow{b_1}][x'_1/x_1]$$

Introduction d'un problème intermédiaire :

$$\begin{array}{ccc} & \left(\overrightarrow{b_1}'\right) & [\overrightarrow{b_1}' \vdash s_T] \\ \Gamma_i \vdash & \begin{pmatrix} \overrightarrow{\pi_1} \\ v \end{pmatrix} & \begin{pmatrix} ? \\ | \end{pmatrix} \end{array}$$

Les Sous-Problèmes : Compilation Récursive

Contexte après le problème intermédiaire $\Gamma_i' : \Gamma_i, \overrightarrow{w_1} : \overrightarrow{W_1}$

$$\Gamma_i' \vdash \quad \begin{array}{c} (z_1 \dots z_{r_i} \ x_2 \dots x_n) \\ \left(\begin{array}{c} \overrightarrow{q_{i,1}} \ \overrightarrow{p_{i,1}} \\ \vdots \\ \overrightarrow{q_{i,m_i}} \ \overrightarrow{p_{i,m_i}} \\ \overrightarrow{q} \ \overrightarrow{p_{\omega,1}} \\ \vdots \\ \overrightarrow{q} \ \overrightarrow{p_{\omega,m_\omega}} \end{array} \right) \end{array} \quad \begin{array}{c} [\overrightarrow{w_2}, x_2'(\overrightarrow{\pi_2}), \dots, \overrightarrow{w_n}, x_n'(\overrightarrow{\pi_n}) \vdash P] \\ \left(\begin{array}{c} t_{i,1}[x_1'/x_1][\overrightarrow{b_1}'/\overrightarrow{b_1}][x_1'/\theta_{i,1}] \\ \vdots \\ t_{i,m_i}[x_1'/x_1][\overrightarrow{b_1}'/\overrightarrow{b_1}][x_1'/\theta_{i,m_i}] \\ t_{\omega,1}[x_1'/x_1][\overrightarrow{b_1}'/\overrightarrow{b_1}][x_1'/\theta_{\omega,1}] \\ \vdots \\ u_{\omega,m_\omega}[x_1'/x_1][\overrightarrow{b_1}'/\overrightarrow{b_1}][x_1'/\theta_{\omega,m_\omega}] \end{array} \right) \end{array}$$

trouver une solution à ce problème, puis remonter

Apports et Limites

Apports

- généralisation systématique $\rightarrow$ conservation de l'information
- filtrages intermédiaire/motifs dans le `in` $\rightarrow$ simplifie les motifs entrés

Limites

gestion des affectations (généralisation)
à comparer avec Goguen, McBride McKinna

Merci pour votre attention !