Cours 4: réécriture et types inductifs

David Delahaye

Polytech' Montpellier
David.Delahaye@lirmm.fr

AIGLE M1

Réécriture

Un exemple simple

```
Coq < Parameter E : Set.
E is assumed

Coq < Parameters a b : E.
a is assumed
b is assumed

Coq < Parameter P : E -> Prop.
P is assumed

Coq < Axiom eq : a = b.
eq is assumed
```

Réécriture

Un exemple simple

```
Coq < Goal P(b) -> P(a).
1 subgoal

  ============================
   P b -> P a

Coq < intro.
1 subgoal

  H : P b
  ============================
   P a
```

Réécriture

Un exemple simple

```
Coq < rewrite eq.
1 subgoal

  H : P b
  ============================
   P b

Coq < assumption.
No more subgoals.
```

Réécriture

Sens de la réécriture

```
Coq < Goal P(a) -> P(b).
1 subgoal

  ============================
   P a -> P b

Coq < intro.
1 subgoal

  H : P a
  ============================
   P b
```

Réécriture

Sens de la réécriture

```
Coq < rewrite <- eq.
1 subgoal

  H : P a
  ============================
   P a

Coq < assumption.
No more subgoals.
```

Isomorphismes de types dans les CCC

```
Open Scope type_scope.

Section Iso_axioms.

Variables A B C : Set.

Axiom Com : A * B = B * A.
Axiom Ass : A * (B * C) = A * B * C.
Axiom Cur : (A * B -> C) = (A -> B -> C).
Axiom Dis : (A -> B * C) = (A -> B) * (A -> C).
Axiom P_unit : A * unit = A.
Axiom AR_unit : (A -> unit) = unit.
Axiom AL_unit : (unit -> A) = A.

End Iso_axioms.
```

Isomorphismes de types dans les CCC

Un exemple de preuve

```
Coq < Goal forall A B : Set, A * (B -> unit) = A.
1 subgoal

  ============================
   forall A B : Set, A * (B -> unit) = A

Coq < intros.
1 subgoal

  A : Set
  B : Set
  ============================
   A * (B -> unit) = A
```

Isomorphismes de types dans les CCC

Un exemple de preuve

```
Coq < rewrite AR_unit.
1 subgoal

  A : Set
  B : Set
  ============================
   A * unit = A
```

Isomorphismes de types dans les CCC

Un exemple de preuve

```
Coq < rewrite P_unit.
1 subgoal

  A : Set
  B : Set
  ============================
   A = A

Coq < reflexivity.
No more subgoals.
```

Isomorphismes de types dans les CCC

Tactiques

```
Coq < Ltac remove_unit :=
Coq <   repeat
Coq <     rewrite P_unit || rewrite AR_unit ||
Coq <     rewrite AL_unit.
remove_unit is defined
```

Isomorphismes de types dans les CCC

Tactiques

```
Coq < Goal forall A B : Set, A * (B -> unit) = A.
1 subgoal

  ============================
   forall A B : Set, A * (B -> unit) = A

Coq < intros.
1 subgoal

  A : Set
  B : Set
  ============================
   A * (B -> unit) = A
```

Isomorphismes de types dans les CCC

Tactiques

```
Coq < remove_unit.
1 subgoal

  A : Set
  B : Set
  ============================
   A = A

Coq < reflexivity.
No more subgoals.
```

Isomorphismes de types dans les CCC

Tactiques

```
Coq < Ltac remove_unit :=
Coq <   repeat
Coq <     match goal with
Coq <     | |- context[?A * unit] => rewrite P_unit
Coq <     | |- context[?A -> unit] => rewrite AR_unit
Coq <     | |- context[unit ->?A] => rewrite AL_unit
Coq <     end.
remove_unit is defined
```

Exercices

Propositions à démontrer

```
Lemma isos_ex1 : forall A B:Set,
  A * unit * B = B * (unit * A).

Lemma isos_ex2 : forall A B C:Set,
  (A * unit -> B * (C * unit)) =
  (A * unit -> (C -> unit) * C) * (unit -> A -> B).
```

Tactiques à écrire

- Écrire une tactique qui normalise les expressions ;
- Démontrer les propositions précédentes à l'aide de cette tactique.

Arithmétique de Peano

```
Section Peano.

Parameter N : Set.
Parameter o : N.
Parameter s : N -> N.
Parameters plus mult : N -> N -> N.
Variables x y : N.
Axiom ax1 : ~((s x) = o).
Axiom ax2 : exists z, ~(x = o) -> (s z) = x.
Axiom ax3 : (s x) = (s y) -> x = y.
Axiom ax4 : (plus x o) = x.
Axiom ax5 : (plus x (s y)) = s (plus x y).
Axiom ax6 : (mult x o) = o.
Axiom ax7 : (mult x (s y)) = (plus (mult x y) x).

End Peano.
```

Arithmétique de Peano

Un exemple de preuve

```
Coq < Goal (plus (s o) (s (s o))) = (s (s (s o))).
1 subgoal

  ============================
   plus (s o) (s (s o)) = s (s (s o))

Coq < rewrite ax5.
1 subgoal

  ============================
   s (plus (s o) (s o)) = s (s (s o))
```

Arithmétique de Peano

Un exemple de preuve

```
Coq < rewrite ax5.
1 subgoal

  ============================
   s (s (plus (s o) o)) = s (s (s o))
```

Arithmétique de Peano

Un exemple de preuve

```
Coq < rewrite ax4.
1 subgoal

  ============================
   s (s (s o)) = s (s (s o))

Coq < reflexivity.
No more subgoals.
```

Exercices

Propositions à démontrer

- $2 + 2 = 4$;
- $2 \times 2 = 4$.

Tactiques à écrire

- Écrire une tactique qui calcule automatiquement ;
- Démontrer les propositions précédentes à l'aide de cette tactique ;
- Même question en utilisant la tactique `autorewrite`.

Anneaux (travail à la maison)

Anneaux

- Construire une structure d'anneau sur un ensemble donné.

Propositions à démontrer

- Démontrer les identités remarquables ;
- Même question en utilisant la tactique `ring`.

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < Print nat.
Inductive nat : Set :=  O : nat | S : nat -> nat
For S: Argument scope is [nat_scope]

Coq < Check (S O).
1
     : nat

Coq < Check 1.
1
     : nat
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < Print plus.
plus =
fix plus (n m : nat) {struct n} : nat :=
  match n with
  | 0 => m
  | S p => S (plus p m)
  end
     : nat -> nat -> nat
Argument scopes are [nat_scope nat_scope]
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < Goal forall x : nat, 0 + x = x.
1 subgoal

  ============================
   forall x : nat, 0 + x = x

Coq < intro.
1 subgoal

  x : nat
  ============================
   0 + x = x
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < simpl.
1 subgoal

  x : nat
  ============================
   x = x

Coq < reflexivity.
No more subgoals.
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < Goal forall x : nat, x + 0 = x.
1 subgoal

  ============================
   forall x : nat, x + 0 = x

Coq < intro.
1 subgoal

  x : nat
  ============================
   x + 0 = x
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < elim x.
2 subgoals

  x : nat
  ============================
   0 + 0 = 0
subgoal 2 is:
 forall n : nat, n + 0 = n -> S n + 0 = S n
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < simpl.
2 subgoals

  x : nat
  ============================
   0 = 0
subgoal 2 is:
 forall n : nat, n + 0 = n -> S n + 0 = S n
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < reflexivity.
1 subgoal

  x : nat
  ============================
   forall n : nat, n + 0 = n -> S n + 0 = S n
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < intros.
1 subgoal

  x : nat
  n : nat
  H : n + 0 = n
  ============================
   S n + 0 = S n
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < simpl.
1 subgoal

  x : nat
  n : nat
  H : n + 0 = n
  ============================
   S (n + 0) = S n
```

Types inductifs

Exemples de preuves

- Entiers naturels :

```
Coq < rewrite H.
1 subgoal

  x : nat
  n : nat
  H : n + 0 = n
  ============================
   S n = S n

Coq < reflexivity.
No more subgoals.
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < Print list.
Inductive list (A : Type) : Type :=
    nil : list A | cons : A -> list A -> list A
For nil: Argument A is implicit and maximally inserted
For cons: Argument A is implicit
For list: Argument scope is [type_scope]
For nil: Argument scope is [type_scope]
For cons: Argument scopes are [type_scope _ _]

Coq < Open Scope list.

Coq < Check (0 :: 1 :: nil).
0 :: 1 :: nil
     : list nat
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < Open Scope list.

Coq < Print app.
app =
fun A : Type =>
fix app (l m : list A) {struct l} : list A :=
  match l with
  | nil => m
  | a :: l1 => a :: app l1 m
  end
     : forall A : Type, list A -> list A -> list A
Argument A is implicit
Argument scopes are [type_scope list_scope list_scope]
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < Goal forall (E : Type) (l : list E), l ++ nil = l.
1 subgoal

  ============================
   forall (E : Type) (l : list E), l ++ nil = l

Coq < intros.
1 subgoal

  E : Type
  l : list E
  ============================
   l ++ nil = l
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < elim l.
2 subgoals

  E : Type
  l : list E
  ============================
   nil ++ nil = nil
subgoal 2 is:
 forall (a : E) (l0 : list E),
 l0 ++ nil = l0 -> (a :: l0) ++ nil = a :: l0
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < simpl.
2 subgoals

  E : Type
  l : list E
  ============================
   nil = nil
subgoal 2 is:
 forall (a : E) (l0 : list E),
 l0 ++ nil = l0 -> (a :: l0) ++ nil = a :: l0
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < reflexivity.
1 subgoal

  E : Type
  l : list E
  ============================
   forall (a : E) (l0 : list E),
   l0 ++ nil = l0 -> (a :: l0) ++ nil = a :: l0
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < intros.
1 subgoal

  E : Type
  l : list E
  a : E
  l0 : list E
  H : l0 ++ nil = l0
  ============================
   (a :: l0) ++ nil = a :: l0
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < simpl.
1 subgoal

  E : Type
  l : list E
  a : E
  l0 : list E
  H : l0 ++ nil = l0
  ============================
   a :: l0 ++ nil = a :: l0
```

Types inductifs

Exemples de preuves

- Listes :

```
Coq < rewrite H.
1 subgoal

  E : Type
  l : list E
  a : E
  l0 : list E
  H : l0 ++ nil = l0
  ============================
   a :: l0 = a :: l0

Coq < reflexivity.
No more subgoals.
```

Exercices avec les types inductifs

Propositions à démontrer

1. $\forall n : \mathbb{N}. n \times 1 = n$;
2. $f(10) = 1024$, où :
   $$f(n) = \begin{cases} 1 & \text{si } n = 0 \\ 2 \times f(n-1) & \text{si } n > 0 \end{cases}$$
3. $\forall E : \mathtt{Type}. \forall l : (\mathit{list}\ E). \forall e : E. \mathit{rev}\ (l \mathbin{++} [a]) = a\ ::\ \mathit{rev}\ l$;
4. $\forall E : \mathtt{Type}. \forall l : (\mathit{list}\ E). \mathit{rev}\ (\mathit{rev}\ l) = l$.

Exercices avec les types inductifs (travail à la maison)

Entiers naturels

- Démontrer la décidabilité de l'égalité :

```
Lemma my_eq_nat_dec :
  forall n m : nat, {n = m} + {n <> m}.
```

  - Manuellement ;
  - En utilisant la tactique `decide equality`.

Arbres

- Définir le type des arbres binaires d'entiers naturels ;
- Démontrer la décidabilité de l'égalité :
  - Manuellement ;
  - En utilisant la tactique `decide equality` ;
  - En utilisant la commande `Scheme Equality` ;
  - En écrivant la fonction qui teste l'égalité :

```
Fixpoint eq_tree (t1 t2 : tree) : bool := ...
```

Relations inductives

Exemples de preuves

- Entiers pairs :

```
Coq < Inductive is_even : nat -> Prop :=
Coq < | is_even_O : is_even O
Coq < | is_even_S : forall n : nat,
Coq <   is_even n -> is_even (S (S n)).
is_even is defined
is_even_ind is defined
```

Relations inductives

Exemples de preuves

- Entiers pairs :

```
Coq < Goal (is_even 2).
1 subgoal

  ============================
   is_even 2

Coq < apply is_even_S.
1 subgoal

  ============================
   is_even 0

Coq < apply is_even_O.
No more subgoals.
```

Relations inductives

Exemples de preuves

- Entiers pairs :

```
Coq < Goal ~(is_even 3).
1 subgoal

  ============================
   ~ is_even 3

Coq < intro.
1 subgoal

  H : is_even 3
  ============================
   False
```

Relations inductives

Exemples de preuves

- Entiers pairs :

```
Coq < inversion H.
1 subgoal

  H : is_even 3
  n : nat
  H1 : is_even 1
  H0 : n = 1
  ============================
   False

Coq < inversion H1.
No more subgoals.
```

Exercices avec les relations inductives

Tactiques à écrire

- Écrire une tactique qui démontre des buts de la forme `(is_even n)` ;
- Écrire une tactique qui démontre des buts de la forme `~(is_even n)` ;
- Écrire une tactique qui démontre les buts précédents indifféremment.

Propositions à démontrer

1. $\forall n : \mathbb{N}. f_{is\_even}(n) \rightarrow is\_even(n)$ (correction) ;
2. $\forall n : \mathbb{N}. is\_even(n) \rightarrow f_{is\_even}(n)$ (complétude) ;

où f_{is_even} est la fonction qui teste la parité d'un entier naturel.

Exercices avec les relations inductives (travail à la maison)

Listes d'entiers naturels triées

- Définir la relation « être trié » pour une liste d'entiers naturels ;
- Démontrer que la liste `1::2::3::4::5::nil` est triée ;
- Écrire une fonction de tri sur les listes d'entiers naturels ;
- Démontrer que cette fonction renvoie bien des listes triées.