Coq入門

D2 宇田 拓馬

自己紹介

• 名前: 宇田 拓馬

• 所属: JAIST 青木研 D2

• 研究キーワード: 形式仕様記述, 定理証明, 自動運転システム

• 趣味: 読書, ゲーム, プログラミング

• Twitter: @hennin ltn

カリー=ハワード同型対応を利用した証明支援システム

- カリー=ハワード同型対応 ... 形式体系と計算モデルの間の対応
 - 。 論理式と型の対応
 - 証明とプログラムの対応

形式体系 ... 直観主義論理 or 古典論理 (の形式体系)

計算モデル ... Calculus of Inductive Constructions

すごく型のつよい関数型言語

型がつよすぎて証明もできるようになっちゃった

カリー=ハワード同型対応を利用した証明支援システム

- カリー=ハワード同型対応 ... 形式体系と計算モデルの間の対応
 - 。 論理式と型の対応
 - 証明とプログラムの対応

形式体系 ... 直観主義論理 or 古典論理 (の形式体系)

計算モデル ... Calculus of Inductive Constructions

例: Coqによる証明

```
Inductive N : Set :=
  Zero
  | S (n: N).
Fixpoint add (m n : N) : N :=
  match n with
  | Zero => m
 \mid S n' => S (add m n')
  end.
Infix "+" := add.
Lemma Commutativity: forall (m n : N),
 m + n = n + m.
```

なにがうれしいか

- ソフトウェアテスト
 - メリット: 導入が容易
 - デメリット: 場当たり的
- Coqによる証明 (形式検証)
 - メリット: すべての状態について
 - デメリット:
 - 強い事実は言えなかったり難しかったりする
 - 専門知識が必要

例: Coqによる証明

```
Inductive N : Set :=
  Zero
  | S (n: N).
Fixpoint add (m n : N) : N :=
  match n with
  | Zero => m
 \mid S n' => S (add m n')
  end.
Infix "+" := add.
Lemma Commutativity: forall (m n : N),
 m + n = n + m.
```

- Gallina ... 仕様記述言語, 証明を記述する
- Vernacular ... 処理系に命令を与える言語, Tacticを定義できる
- Tactic ... ある推論規則をもって, その goal を subgoals で置き換える機能
 - 。 subgoals … 推論規則の仮定
 - 。 goal … 推論規則の結論
- Ltac ... 既存のTacticを組み合わせて複雑なTacticを定義するための言語
- Ltac2 ... Ltacのいくつかの欠点を補うため新しく導入された言語

例: Coqによる証明

```
Inductive N : Set :=
  Zero
  | S (n: N).
Fixpoint add (m n : N) : N :=
  match n with
  | Zero => m
 \mid S n' => S (add m n')
  end.
Infix "+" := add.
Lemma Commutativity: forall (m n : N),
 m + n = n + m.
```

```
Lemma Commutativity : forall (m n : N),
14
       m + n = n + m.
     Proof.
15
       intros m n.
       elim m.
       - simpl.
         elim n.
         simpl.
         trivial.
21
         intro n0.
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
28
         simpl.
29
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
         rewrite H0.
         trivial.
```

```
(1/1)
forall m n : N, m + n = n + m
```

```
Lemma Commutativity: forall (m n : N),
14
       m + n = n + m.
15
       intros m n.
16
       elim m.
       - simpl.
         elim n.
         simpl.
         trivial.
         intro n0.
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
         simpl.
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
         rewrite H0.
         trivial.
```

```
(1/1)
m + n = n + m
```

```
Lemma Commutativity : forall (m n : N),
14
       m + n = n + m.
       intros m n.
16
       elim m.
17
       - simpl.
         elim n.
         simpl.
         trivial.
         intro n0.
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
         simpl.
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
         rewrite H0.
         trivial.
```

```
m, n: N

(1/2)
Zero + n = n + Zero

(2/2)
forall n0 : N, n0 + n = n + n0 -> S n0 + n = n + S n0
```

```
Lemma Commutativity : forall (m n : N),
14
      m + n = n + m.
       intros m n.
       elim m.
       - simpl.
18
        elim n.
19
        simpl.
20
21
        trivial.
        intro n0.
22
        intro H.
23
        simpl.
24
        rewrite H.
25
        trivial.
26
       - intro m'.
        intro H.
28
        simpl.
29
        rewrite <- H.
        elim n.
        simpl.
        trivial.
        intro n'.
        intro H0.
        simpl.
        rewrite H0.
        trivial.
```

```
m, n: N

(1/1)
Zero + n = n + Zero
```

```
Lemma Commutativity: forall (m n: N),
      m + n = n + m.
      intros m n.
      elim m.
      - simpl.
18
       elim n.
        simpl.
       trivial.
        intro n0.
        intro H.
        simpl.
        rewrite H.
        trivial.
      - intro m'.
        intro H.
        simpl.
        rewrite <- H.
        elim n.
        simpl.
        trivial.
        intro n'.
        intro H0.
        simpl.
        rewrite H0.
        trivial.
    Qed.
```

```
(1/1)
Zero + n = n
```

```
Lemma Commutativity : forall (m n : N),
       intros m n.
       elim m.
       - simpl.
        elim n.
19
         simpl.
         trivial.
21
         intro n0.
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
         simpl.
29
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro HO.
         simpl.
         rewrite HO.
         trivial.
```

```
m, n: N

(1/2)
Zero + Zero = Zero

(2/2)
forall n0: N, Zero + n0 = n0 -> Zero + S n0 = S n0
```

```
Lemma Commutativity: forall (m n : N),
       m + n = n + m.
       intros m n.
       elim m.
       - simpl.
        elim n.
         simpl.
20
         trivial.
         intro no.
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
         simpl.
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
         rewrite H0.
37
         trivial.
     Qed.
```

```
m, n: N

(1/2)
Zero = Zero

(2/2)
forall n0: N, Zero + n0 = n0 -> Zero + S n0 = S n0
```

```
Lemma Commutativity : forall (m n : N),
       intros m n.
       elim m.
       - simpl.
         elim n.
         simpl.
         trivial.
21
         intro n0.
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
         simpl.
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
         rewrite H0.
         trivial.
```

```
m, n: N

(1/1)
forall n0: N, Zero + n0 = n0 -> Zero + S n0 = S n0
```

```
Lemma Commutativity : forall (m n : N),
       intros m n.
       elim m.
       - simpl.
         elim n.
         simpl.
         trivial.
21
         intro n0.
22
         intro H.
         simpl.
         rewrite H.
         trivial.
       - intro m'.
         intro H.
         simpl.
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro HO.
         simpl.
         rewrite H0.
         trivial.
     Qed.
```

```
m, n, n0: N

(1/1)
Zero + n0 = n0 -> Zero + S n0 = S n0
```

```
Lemma Commutativity: forall (m n : N),
       m + n = n + m.
     Proof.
       intros m n.
       elim m.
       - simpl.
        elim n.
         simpl.
20
         trivial.
21
         intro n0.
22
        intro H.
23
        simpl.
24
        rewrite H.
25
        trivial.
26
       - intro m'.
27
         intro H.
28
         simpl.
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
        rewrite H0.
         trivial.
```

```
m, n, n0: N
H: Zero + n0 = n0
(1/1)
Zero + S n0 = S n0
```

```
Lemma Commutativity: forall (m n : N),
      m + n = n + m.
     Proof.
      intros m n.
      elim m.
      - simpl.
        elim n.
        simpl.
        trivial.
        intro no.
        intro H.
        simpl.
24
        rewrite H.
        trivial.
      - intro m'.
        intro H.
        simpl.
        rewrite <- H.
        elim n.
        simpl.
        trivial.
        intro n'.
        intro H0.
        simpl.
        rewrite H0.
        trivial.
```

```
m, n, n0: N
H: Zero + n0 = n0
(1/1)
S (Zero + n0) = S n0
```

```
Lemma Commutativity: forall (m n: N),
       m + n = n + m.
       intros m n.
       elim m.
       - simpl.
       elim n.
        simpl.
        trivial.
21
        intro n0.
        intro H.
        simpl.
25
        rewrite H.
        trivial.
       - intro m'.
        intro H.
         simpl.
        rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro H0.
         simpl.
        rewrite H0.
         trivial.
     Qed.
```

```
m, n, n0: N
H: Zero + n0 = n0
(1/1)
S n0 = S n0
```

```
Lemma Commutativity : forall (m n : N),
14
       m + n = n + m.
       intros m n.
       elim m.
       - simpl.
        elim n.
         simpl.
20
         trivial.
21
         intro n0.
         intro H.
         simpl.
24
         rewrite H.
25
         trivial.
26
       - intro m'.
         intro H.
28
         simpl.
29
         rewrite <- H.
         elim n.
         simpl.
         trivial.
         intro n'.
         intro HO.
         simpl.
         rewrite H0.
         trivial.
```

There are unfocused goals.

```
Lemma Commutativity: forall (m n : N),
14
15
       intros m n.
16
       elim m.
17
       - simpl.
18
        elim n.
19
        simpl.
20
        trivial.
21
        intro n0.
22
        intro H.
23
        simpl.
24
        rewrite H.
25
        trivial.
26
       - intro m'.
27
        intro H.
28
        simpl.
        rewrite <- H.
        elim n.
        simpl.
        trivial.
        intro n'.
        intro HO.
        simpl.
        rewrite H0.
        trivial.
```

```
(1/1)
forall n0 : N, n0 + n = n + n0 -> S n0 + n = n + S n0
```

```
Lemma Commutativity: forall (m n : N),
 m + n = n + m.
Proof.
 intros m n.
 elim m.
 - simpl.
   elim n.
   simpl.
   trivial.
   intro n0.
   intro H.
   simpl.
   rewrite H.
   trivial.
 - intro m'.
   intro H.
   simpl.
   rewrite <- H.
   elim n.
   simpl.
   trivial.
   intro n'.
   intro H0.
   simpl.
   rewrite H0.
   trivial.
```

No more subgoals.