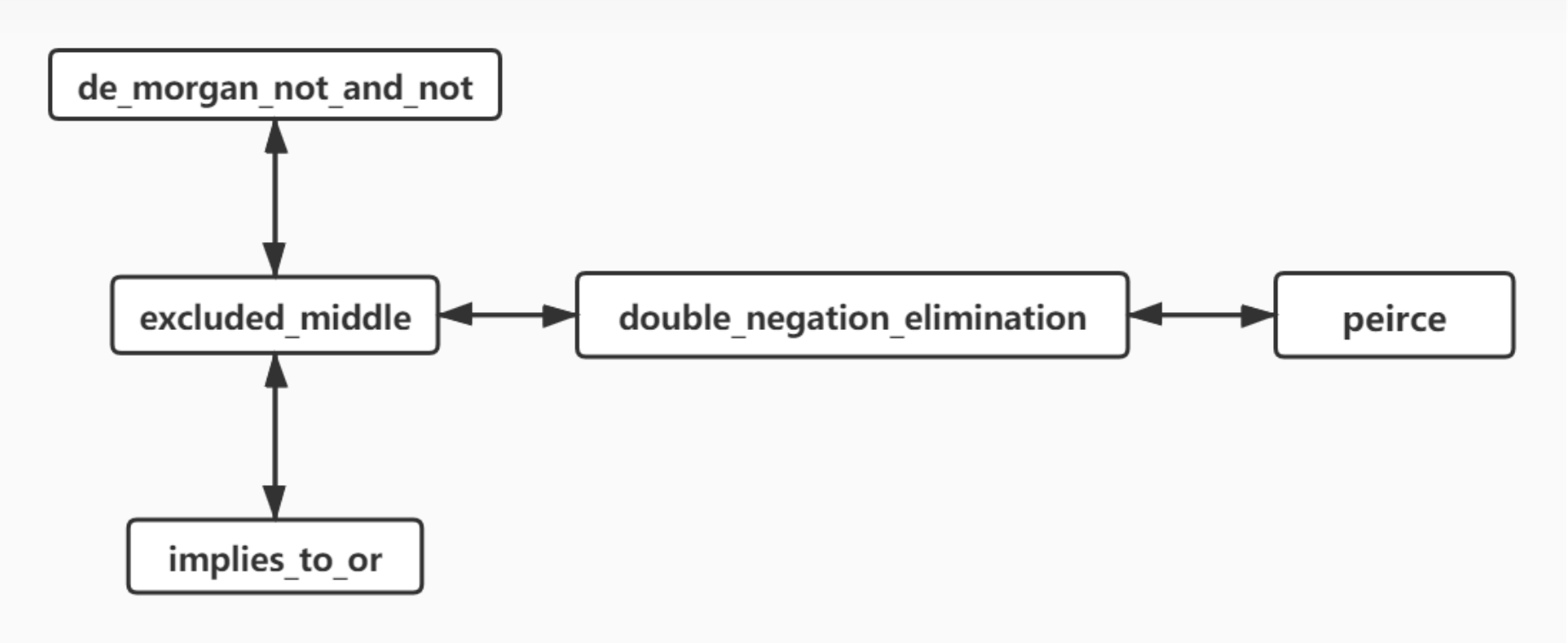


在Coq中证明经典逻辑命题的等价性

MisakaCenter
学习中 |-- 啥都不会

作者： @MisakaCenter

本文为学习笔记，解释权归 诚实鸽友研究社 所有，转载请标明出处。

一、概述

最近在学《Software Foundations》的时候，遇到了一道很有意思的题目（也是这本书中出现的第二道五星题）：

> Exercise: 5 stars, standard, optional (classical_axioms)
> For those who like a challenge, here is an exercise taken from the Coq'Art book by Bertot and Casteran (p. 123). Each of the following four statements, together with excluded_middle, can be considered as characterizing classical logic. We can't prove any of them in Coq, but we can consistently add any one of them as an axiom if we wish to work in classical logic.

简单来说，虽然我们没法在Coq中直接证明这五个经典逻辑命题，但我们可以证明它们的等价性。

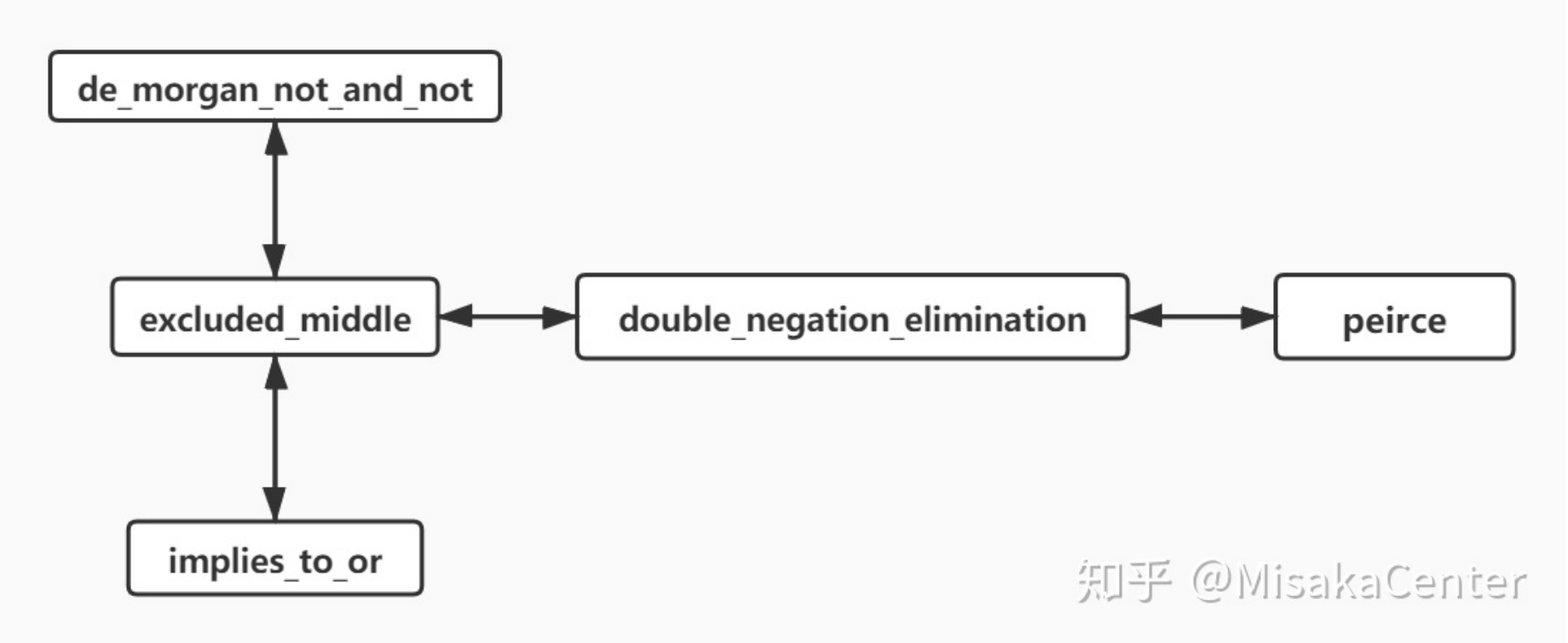
其中五条公理在Coq中的定义如下

```
Definition excluded_middle := forall P : Prop,
  P \/ ~ P.

Definition peirce := forall P Q: Prop,
  ((P->Q)->P)->P.

Definition double_negation_elimination := forall P:Prop,
  ~~P -> P.

Definition de_morgan_not_and_not := forall P Q:Prop,
  ~(~P /\ ~Q) -> P\/Q.

Definition implies_to_or := forall P Q:Prop,
  (P->Q) -> (~P\/Q).
```

为了证明它们是等价的，比较好的方法是证明一条将它们连接起来的单向蕴含环链。经过尝试，我在Coq中证明出了如下的推出关系，从而证明了这五条公理的等价性。

1. classical_axioms_1: excluded_middle -> double_negation_elimination
2. classical_axioms_2: double_negation_elimination -> excluded_middle
3. classical_axioms_3: excluded_middle -> de_morgan_not_and_not
4. classical_axioms_4: de_morgan_not_and_not -> excluded_middle
5. classical_axioms_5: excluded_middle -> implies_to_or
6. classical_axioms_6: implies_to_or -> excluded_middle
7. classical_axioms_7: double_negation_elimination -> peirce
8. classical_axioms_8: peirce -> double_negation_elimination

更加直观一点的话，如下图所示：



得到的推出关系

二、证明

excluded_middle <-> double_negation_elimination

```
Theorem classical_axioms_1 (* 很简单 *)
: excluded_middle -> double_negation_elimination.
Proof.
  unfold double_negation_elimination.
  unfold excluded_middle. unfold not.
  intros H P H0. destruct (H P) as [H1|H2].
  - apply H1.
  - apply H0 in H2. destruct H2.
Qed.

Theorem classical_axioms_2
: double_negation_elimination -> excluded_middle.
Proof.
  unfold double_negation_elimination.
  unfold excluded_middle. unfold not.
  intros H P. apply H.
  intro H0. (* H0 : P \/ (P -> False) -> False *)
  assert ((P -> False) -> False).
    { intros. apply or_intror with (A:=P) in H1.
      apply H0 in H1. (* H1 : False *)
      apply H1. }
   (* H1 : (P -> False) -> False *)
  apply H in H1. (* H1 : P *)
  apply or_introl with (B:=P->False) in H1.
  apply H0 in H1. (* H1 : False *)
  apply H1.
Qed.
```

excluded_middle <-> de_morgan_not_and_not

```
Theorem classical_axioms_3
: excluded_middle -> de_morgan_not_and_not.
Proof.
  unfold de_morgan_not_and_not.
  unfold excluded_middle.
  unfold not. intros.
  destruct (H Q).
  - right. apply H1.
  - destruct (H P).
    + left. apply H2.
    + destruct H0. split. ++ apply H2. ++ apply H1.
Qed.

Theorem classical_axioms_4
: de_morgan_not_and_not -> excluded_middle.
Proof.
  unfold de_morgan_not_and_not.
  unfold excluded_middle.
  unfold not. intros.
  apply H. intros [H1 H2]. apply H2. apply H1.
Qed.
```

excluded_middle<->implies_to_or

```
Theorem classical_axioms_5
: excluded_middle -> implies_to_or.
Proof.
  unfold excluded_middle.
  unfold implies_to_or.
  unfold not. intros.
  destruct (H P).
  - right. apply H0 in H1. apply H1.
  - left. apply H1.
Qed.

Theorem classical_axioms_6
: implies_to_or -> excluded_middle.
Proof.
  unfold excluded_middle.
  unfold implies_to_or.
  unfold not. intros.
  assert((P -> False)\/P ->  P \/ (P -> False)).
  { intros. destruct H0. right. apply H0. left. apply H0. }
  apply H0.
  apply H.
  intros. apply H1.
Qed.
```

double_negation_elimination<->peirce

```
Theorem classical_axioms_7
: double_negation_elimination -> peirce.
Proof.
  unfold double_negation_elimination.
  unfold peirce.
  unfold not. intros.
  apply H.
  intros. apply H1. apply H0. intros.
  apply H1 in H2. destruct H2.
Qed.

Theorem classical_axioms_8
: peirce -> double_negation_elimination.
Proof.
  unfold peirce.
  unfold double_negation_elimination. unfold not.
  intros.
  apply H with (Q:=False).
  intros. apply H0 in H1. destruct H1.
Qed.
```

三、？

Coq好有意思！！！

编辑于 2020-09-01 23:41

编程 编程学习 编程语言