CS6380 – Artificial Intelligence

Interactive Theorem Proving.

Nov 6–7, 2025

Interactive Theorem Proving.

Proof Assistant: software tool which aids in the development of a formal proof for a theorem using human-machine collaboration.

Interactive Theorem Proving.

Proof Assistant: software tool which aids in the development of a formal proof for a theorem using human-machine collaboration.

Why do we need a proof assistant?

- Proofs written on paper may be buggy. – Famous example : Four color theorem

Interactive Theorem Proving.

Proof Assistant: software tool which aids in the development of a formal proof for a theorem using human-machine collaboration.

Why do we need a proof assistant?

- Proofs written on paper may be buggy. – Famous example : Four color theorem Kempe's proof of the 4 color theorem had an incorrect assumption. It took 11 years for this to be identified.

Interactive Theorem Proving.

Proof Assistant: software tool which aids in the development of a formal proof for a theorem using human-machine collaboration.

Why do we need a proof assistant?

- Proofs written on paper may be buggy. – Famous example : Four color theorem Kempe's proof of the 4 color theorem had an incorrect assumption. It took 11 years for this to be identified.
  – The current proof of Four color theorem is verified via a proof assistant.

Interactive Theorem Proving.

Proof Assistant: software tool which aids in the development of a formal proof for a theorem using human-machine collaboration.

Why do we need a proof assistant?

- Proofs written on paper may be buggy. – Famous example : Four color theorem Kempe's proof of the 4 color theorem had an incorrect assumption. It took 11 years for this to be identified.
  – The current proof of Four color theorem is verified via a proof assistant.
- Fully automated proofs may be hard to find or may take too long.
- Proof assistants serve as middle ground.

Interactive Theorem Proving.

Proof Assistant: software tool which aids in the development of a formal proof for a theorem using human-machine collaboration.

Why do we need a proof assistant?

- Proofs written on paper may be buggy. – Famous example : Four color theorem Kempe's proof of the 4 color theorem had an incorrect assumption. It took 11 years for this to be identified.
  – The current proof of Four color theorem is verified via a proof assistant.
- Fully automated proofs may be hard to find or may take too long.
- Proof assistants serve as middle ground.

Proof assistant is like a pedantic student who always raises his hand and asks "why is this true?" –Kevin Buzzard.

Basics of CoQ.

- CoQ is an Interactive Theorem Proving Environment.
- CoQ compiler checks for correctness of definitions and proofs.
- The name comes from Calculus of Constructions and its first implementation was in 1984 due to Coquand and Huet.
- Constant enhancements to the language and environment. (Now renamed as RocQ).

Basics of CoQ.

- CoQ is an Interactive Theorem Proving Environment.
- CoQ compiler checks for correctness of definitions and proofs.
- The name comes from Calculus of Constructions and its first implementation was in 1984 due to Coquand and Huet.
- Constant enhancements to the language and environment. (Now renamed as RocQ).

Syntax and Coqide

- All commands in coq end with a full-stop.
- (* This is a comment in coq *)
- coqide is interactive tool which provides three windows:
  - Main window containing coq commands (tactics) which are sequentially interpreted.
  - In Proof mode, a window displays the current status of the proof including unfinished goals.
  - A window to display messages (if any).

Warm-up with CoQ: Propositions.

1. If A is a proposition then, then we have $A \to A$.

Warm-up with CoQ: Propositions.

1. If A is a proposition then, then we have $A \to A$.
2. For any proposition P, we have $P \to P$.

Warm-up with CoQ: Propositions.

1. If A is a proposition then, then we have $A \to A$.
2. For any proposition P, we have $P \to P$.
3. If Q and R are propositions, then $(Q \to R) \to (Q \to R)$.

Warm-up with CoQ: Propositions.

1. If A is a proposition then, then we have $A \to A$.
2. For any proposition P, we have $P \to P$.
3. If Q and R are propositions, then $(Q \to R) \to (Q \to R)$.

Tactics in CoQ

- intro
- assumption

Warm-up with CoQ: Propositions.

```
Section FirstSet.
Variables A: Prop.
Theorem Thm0: A -> A.
Proof.
   intro.
   assumption.
Qed.

Theorem Thm1 : forall P: Prop, P -> P.
Proof.
    intro.
    intro.
    assumption.
Qed.
End FirstSet.
```

Warm-up with CoQ: Propositions.

```
Section FirstSet.
Variables A: Prop.
Theorem Thm0: A -> A.
Proof.
   intro.
   assumption.
Qed.

Theorem Thm1 : forall P: Prop, P -> P.
Proof.
    intro.
    intro.
    assumption.
Qed.
End FirstSet.
```

How do we prove the third statement using Thm1?

Warm-up with CoQ: Propositions.

```
Section FirstSet.
Theorem Thm1 : forall P: Prop, P -> P.
Proof.
    intro.
    intro.
    assumption.
Qed.

Variables Q R: Prop.
Theorem Thm2: (Q->R) -> (Q->R).
Proof.
    pose proof Thm1.
    specialize H with (P:=(Q->R)).
    assumption.
Qed.
End FirstSet.
```

Warm-up with CoQ: Propositions.

```
Section FirstSet.
Theorem Thm1 : forall P: Prop, P -> P.
Proof.
    intro.
    intro.
    assumption.
Qed.

Variables Q R: Prop.
Theorem Thm2: (Q->R) -> (Q->R).
Proof.
    pose proof Thm1.
    specialize H with (P:=(Q->R)).
    assumption.
Qed.
End FirstSet.
```

Similarly, prove that for all propositions, A if and only if A.

Warm-up with CoQ: Propositions.

```
Section FirstSet.
Theorem Thm1 : forall P: Prop, P -> P.
Proof.
    intro.
    intro.
    assumption.
Qed.
Theorem Thm3 : forall A: Prop, A <-> A.
Proof.
    split. (* note the new tactic *)
    pose proof Thm1.
    specialize H with (P:=A).
    assumption.
    pose proof Thm1.
    specialize H with (P:=A).
    assumption.
Qed.
End FirstSet.
```

What tactics have we learnt?

Tactics allow us to transform the goal or the hypothesis to complete the proof.

What tactics have we learnt?

Tactics allow us to transform the goal or the hypothesis to complete the proof.

Tactics that modify the goal:

- intros : useful when there is an implication or forall quantified variables. The tactic brings the quantified variables and / or implications from the goal to the proof context.

What tactics have we learnt?

Tactics allow us to transform the goal or the hypothesis to complete the proof.

Tactics that modify the goal:

- intros : useful when there is an implication or forall quantified variables. The tactic brings the quantified variables and / or implications from the goal to the proof context.
- split : useful when the goal has a bi-implication. Also useful when the goal has an AND. Splits the goal into two sub-goals.

What tactics have we learnt?

Tactics allow us to transform the goal or the hypothesis to complete the proof.

Tactics that modify the goal:

- intros : useful when there is an implication or forall quantified variables. The tactic brings the quantified variables and / or implications from the goal to the proof context.
- split : useful when the goal has a bi-implication. Also useful when the goal has an AND. Splits the goal into two sub-goals.

Tactics that modify the hypothesis:

- specialize : useful when we have a universally quantified hypothesis and we want to create a specific instance of that. It removes the original hypothesis from the proof context. There is a way to retain it as well.

Let's deal with NOT, AND, OR

Try these yourself. You will need destruct and left / right tactics.

- For any propositions P, Q, we have $(P \text{ and } Q) \to (Q \text{ and } P)$.
- For any propositions P, Q, we have $(P \text{ or } Q) \to (Q \text{ or } P)$.
- For any propositions P, Q, we have $(P \text{ and } \bar{P}) \to Q$.

Let's deal with NOT, AND, OR

Try these yourself. You will need destruct and left / right tactics.

- For any propositions P, Q, we have $(P \text{ and } Q) \to (Q \text{ and } P)$.
- For any propositions P, Q, we have $(P \text{ or } Q) \to (Q \text{ or } P)$.
- For any propositions P, Q, we have $(P \text{ and } \bar{P}) \to Q$.
- For any propositions P, Q, R, we have $(P \text{ or } Q) \text{ and } (\bar{Q} \text{ or } R) \implies (P \text{ or } R)$.

Contradiction Implies anything.

```
Theorem contrad_implies_anything:
        forall P Q: Prop, (P /\ ~P) -> Q.
Proof.
    intros.
    destruct H as [H1 H2].
    unfold not in H2.
    apply H2 in H1.
    contradiction.
Qed.
```

Resolution for 3 variables.

```
Theorem resolution:
        forall P Q R: Prop, (P \/ Q) /\ (~Q \/ R) -> (P \/ R).
Proof.
   intros.
   destruct H as [H1 H2].
   destruct H1 as [HP | HQ].
   left.
   assumption.
   destruct H2.
   contradiction.
   right.
   exact H.
Qed.
```

World of Knights and Knaves.

World of Knights and Knaves.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

World of Knights and Knaves.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

A stranger visits the island and meets two people A and B.

1. A says " B is a Knight".
2. B says " A is a Knight".
3. It is a fact that at least one of them is truthful.

World of Knights and Knaves.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

A stranger visits the island and meets two people A and B.

1. A says " B is a Knight".
2. B says " A is a Knight".
3. It is a fact that at least one of them is truthful.

Can you find out who are A and B?

Knight and Knaves: Setup

```
Require Import Coq.Logic.Classical_Prop.

Variables A_is_Knight B_is_Knight : Prop.

Definition A_is_Knave := ~ A_is_Knight.
Definition B_is_Knave := ~ B_is_Knight.

(* This definition says thet A says something P *)
(* P is true iff A is Knight*)
Definition A_says (P : Prop) := A_is_Knight <-> P.
Definition B_says (P : Prop) := B_is_Knight <-> P.
```

Knight and Knaves: Theorem statement

```
Theorem Both_Are_Knights :
     (* Fact 1: A says "B is a Knight" *)
     A_says (B_is_Knight) /\

     (* Fact 2: B says "A is a Knight" *)
     B_says (A_is_Knight) /\

     (* Fact 3: At least one of them us truthful *)
     (A_is_Knight \/ B_is_Knight) ->

     (* Goal: Both are Knights *)
     A_is_Knight /\ B_is_Knight.
```

Knight and Knaves: Proof

```
Proof.
     intro.
     destruct H as [H1 H2].
     destruct H2 as [H3 H4].
     unfold A_says in H1.
     unfold B_says in H3.
     destruct H4.
     split.
     assumption.destruct H1.
     apply H0 in H. assumption.
     split.
     destruct H1.
     apply H1 in H.
     assumption.
     assumption.
Qed.
```

World of Knights and Knaves: Puzzle 2.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

World of Knights and Knaves: Puzzle 2.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

A stranger visits the island and meets two people A and B.

- A says " A is a Knave and B is a Knave".

World of Knights and Knaves: Puzzle 2.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

A stranger visits the island and meets two people A and B.

- A says " A is a Knave and B is a Knave".

Can you find out who are A and B?

World of Knights and Knaves: Puzzle 2.

On an remote island, there are two types of people

- Knight: a person who always speaks the truth.
- Knave: a person who always lies.

A stranger visits the island and meets two people A and B.

- A says " A is a Knave and B is a Knave".

Can you find out who are A and B?

Formalize the theorem statement and try the proof.

Acknowledgements.

- For CoQ: Prof. K.C. Sivaramkrishnan, Sanket Tarafder.

Acknowledgements.

- For CoQ: Prof. K.C. Sivaramkrishnan, Sanket Tarafder.
- For discussions and advice: Prof. Deepak Khemani, Prof. Sutanu, Prof. Ravindran B., Prof. Chandra, Prof. Mani, Prof. Narayanswamy.

Acknowledgements.

- For CoQ: Prof. K.C. Sivaramkrishnan, Sanket Tarafder.
- For discussions and advice: Prof. Deepak Khemani, Prof. Sutanu, Prof. Ravindran B., Prof. Chandra, Prof. Mani, Prof. Narayanswamy.
- To our TA Team : Ritwiz, Dinesh and Shristhi.

Acknowledgements.

- For CoQ: Prof. K.C. Sivaramkrishnan, Sanket Tarafder.
- For discussions and advice: Prof. Deepak Khemani, Prof. Sutanu, Prof. Ravindran B., Prof. Chandra, Prof. Mani, Prof. Narayanswamy.
- To our TA Team : Ritwiz, Dinesh and Shristhi.
- To each and every participant of our class.

Acknowledgements.

- For CoQ: Prof. K.C. Sivaramkrishnan, Sanket Tarafder.
- For discussions and advice: Prof. Deepak Khemani, Prof. Sutanu, Prof. Ravindran B., Prof. Chandra, Prof. Mani, Prof. Narayanswamy.
- To our TA Team : Ritwiz, Dinesh and Shristhi.
- To each and every participant of our class.

Acknowledgements.

- For CoQ: Prof. K.C. Sivaramkrishnan, Sanket Tarafder.
- For discussions and advice: Prof. Deepak Khemani, Prof. Sutanu, Prof. Ravindran B., Prof. Chandra, Prof. Mani, Prof. Narayanswamy.
- To our TA Team : Ritwiz, Dinesh and Shristhi.
- To each and every participant of our class.

Thank You!