CS420

Logical Foundations of Computer Science

Lecture 7: Formal languages

Tiago Cogumbreiro

Today we will learn...

- Existential operator
- Mock Mini-Test 1
- Formal language
- Language operators
- Language equivalence

Existential quantification

$$\exists x . \boldsymbol{P}$$

Existential quantification

```
Inductive ex (A : Type) (P : A → Prop) : Prop :=
  | ex_intro : forall (x : A) (_ : P x), ex P.
```

Notation:

```
exists x:A, P x
```

- To conclude a goal `exists x:A, P x` we can use tactics `exist x.` which yields `P x`. Alternatively, we can use `apply ex_intro`.

```
forall n, exists z, z + n = n
```

- To use a hypothesis of type `H:exists x:A, P x`, you can use `destruct H as (x,H)`, or `inversion H`

```
forall n, (exists m, m < n) → n <> 0.
```

Mock Mini-Test 1

Q1.1

- Is there any type such that any two values of that type are distinct?

```
exists X:Type, forall x y:X, x <> y
```

Q1.1

- Is there any type such that any two values of that type are distinct?

```
exists X:Type, forall x y:X, x <> y
```

Solution: Yes

```
Goal
  exists X:Type, forall x y:X, x <> y.
Proof.
  exists False.
  intros.
  unfold not.
  destruct x.
Qed.
```

Q1.2

Is the following statement is provable in Coq?

```
true = false
```

Q1.2

Is the following statement is provable in Coq?

```
true = false
```

Solution: No

```
Goal
  true = false → False.
Proof.
  intros.
  inversion H.
Qed.
```

Q1.3

All functions defined in Coq via `Fixpoint` must terminate on all inputs.

Q1.3

All functions defined in Coq via `Fixpoint` must terminate on all inputs.

Solution: Yes!

Q1.4

Is the type foo infinite?

```
Inductive foo : Type :=
  | mk_foo: foo → foo.
```

Q1.4

> Is the type foo infinite?

```
Inductive foo : Type :=
  | mk_foo: foo → foo.
```

Solution: No!

Type foo is **empty**, as can be proved below.

```
Goal
  forall (f:foo), False.
Proof.
  intros.
  induction f.
  assumption.
Qed.
```

Q2.1

What is the type of the following expression?

```
forall (X : Type), forall (Y : Type), forall (x : X), x = x
```

Q2.1

What is the type of the following expression?

```
forall (X : Type), forall (Y : Type), forall (x : X), x = x
```

Answer: `Prop`

Q2.2

What is the type of the following expression?

```
fun (x:nat) ⇒ match [56;24] with [] ⇒ 32 | x::l ⇒ x+1 end
```

Q2.2

What is the type of the following expression?

```
fun (x:nat) => match [56;24] with [] => 32 | x::l => x+1 end
```

Answer: `nat -> nat`

Q2.3

What is the type of the following expression?

```
Nat.eqb 56
```

Implementation of `Nat.eqb`:

```
Fixpoint eqb n m : bool :=
  match n, m with
    | 0, 0 ⇒ true
    | 0, S _ ⇒ false
    | S _, 0 ⇒ false
    | S n', S m' ⇒ eqb n' m'
  end.
```

Q2.3

What is the type of the following expression?

```
Nat.eqb 56
```

Answer: `nat → bool`

Implementation of `Nat.eqb`:

```
Fixpoint eqb n m : bool :=
  match n, m with
    | 0, 0 ⇒ true
    | 0, S _ ⇒ false
    | S _, 0 ⇒ false
    | S n', S m' ⇒ eqb n' m'
  end.
```

Q2.4

```
forall (X : Type) (x y : X), x * y = 32
```

```
Fixpoint mul n m :=
  match n with
  | 0 => 0
  | S p => m + p * m
  end

where "n * m" := (mul n m) : nat_scope.
```

Q2.4

```
forall (X : Type) (x y : X), x * y = 32
```

```
Fixpoint mul n m :=
  match n with
  | 0 => 0
  | S p => m + p * m
  end

where "n * m" := (mul n m) : nat_scope.
```

Answer: `ill formed`, because `X` cannot be any type. Only works with nat.

Q3.1

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

```
forall n, n = S n
```

Q3.1

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

```
forall n, n = S n
```

Answer: NOT PROVABLE

Q3.2

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

```
forall x y, x * y = y * x
```

```
Fixpoint mul n m :=
  match n with
  | 0 => 0
  | S p => m + p * m
  end

where "n * m" := (mul n m) : nat_scope.
```

Q3.2

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

```
forall x y, x * y = y * x
```

```
Fixpoint mul n m :=
  match n with
  | 0 ⇒ 0
  | S p ⇒ m + p * m
  end

where "n * m" := (mul n m) : nat_scope.
```

Answer: BY INDUCTION

Q3.3

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

```
forall n, n <> S n
```

Q3.3

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

```
forall n, n <> S n
```

Answer: BY INDUCTION

Q4.1

Prove this goal:

```
H : P -> Q
H0 : P \/ ~ P
______________________________________(1/1)
~ P \/ Q
```

Q4.2

Prove this goal:

```
P : Prop
H0 : P
______________________________________(1/1)
~ ~ P
```