The contents of Coq what I understood

First of all

Trips:

- Every Coq command should be end with `.`
- Using annotation by `(* COMMENTS HERE *)`

Pred

We can declare set variables like:

```
Variables A B : Set.
```

And also assume some predicate variables like:

```
Variables P Q : A -> Prop.
```

∀ & ∃

- Using `forall` to declare some universal quantification. Just like ∀.
- Using `exists` to declare some existential quantification. Just like ∃.

E.g.

```
forall x:A
exists x:A
```

Hello World

Start the first proof
E.g.

```
Theorem my_first_proof : (forall A : Prop, A -> A).
Proof.
  intros A.
  intros proof_of_A.
  exact proof_of_A.
Qed.
```

- Using `Theorem` to declare a theorem.
- `forall` means ∀.
- `Prop` means proposition.
- Using `Proof` to start to prove something.
- Using `intros` to introduce some assumptions.
- Using `exact` if the subgoal matches an hypothesis what we can finish the proof by using this assumption.

- Using `Qed` to end the demonstration.

True or False

It's like a kind of `structure` in C++, and you can create a new type by using command `Inductive`.

```
Inductive False : Prop := .

Inductive True : Prop :=
  | I : True.

Inductive bool : Set :=
  | true : bool
  | false : bool.
```

Prop

`Prop` means proposition.

Connectives and Logical Constants

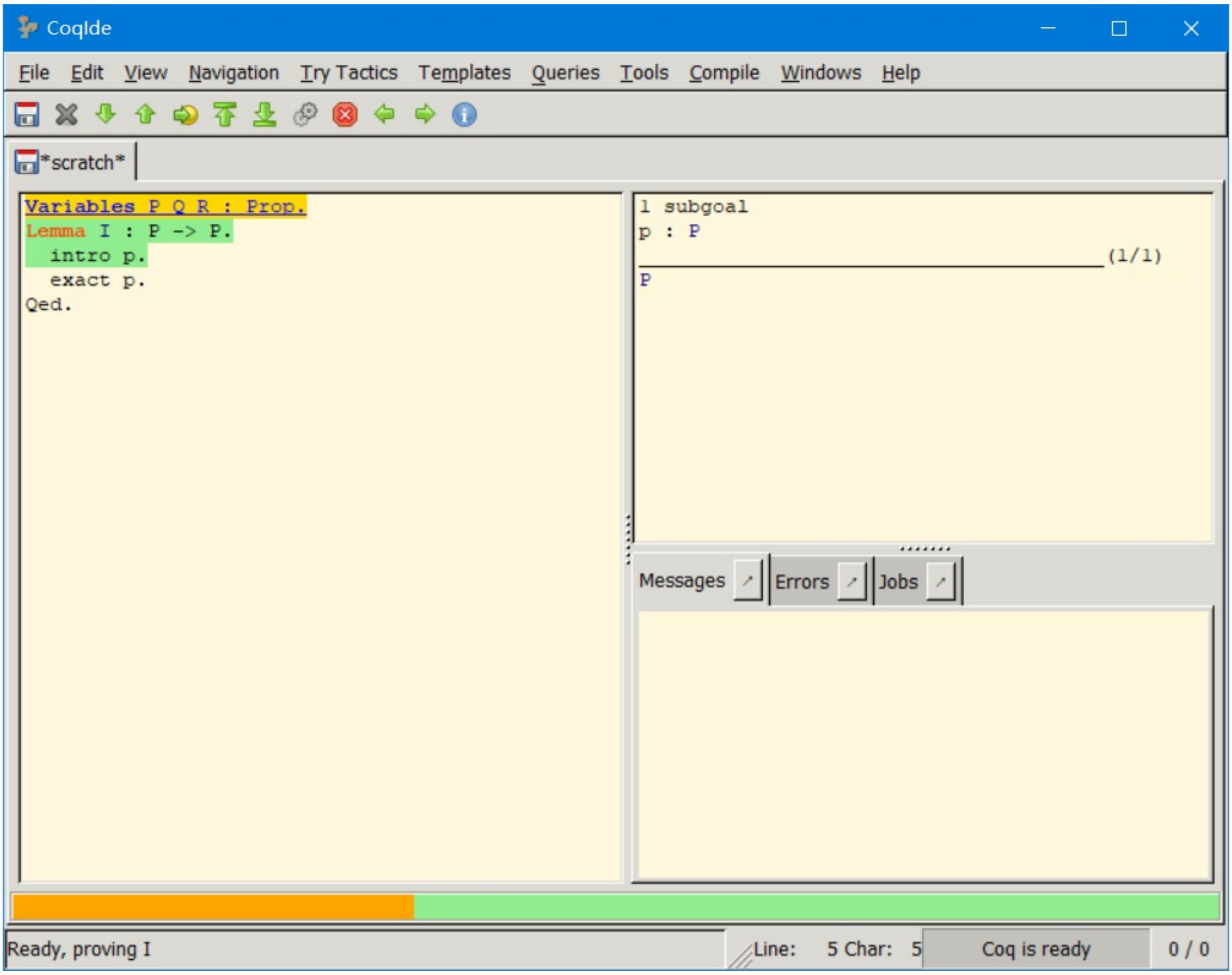
- `->`, P -> Q means if P then Q. It's just `→` in Discrete mathematics.
- `/\`, P /\ Q means P and Q. It's just `∧` in Discrete mathematics.
- `\/`, P \/ Q means P or Q. It's just `∨` in Discrete mathematics.
- `~`, ~ P means not P. It's just `┐` in Discrete mathematics.
- `<->`, P <-> Q means P is equivalent to Q. P <-> Q ≡ (P -> Q) /\ (Q -> P).

First proof of this case

E.g.

```
Variables P Q R : Prop.
Lemma I : P -> P.
  intro p.
  exact p.
Qed.
```

- Using `Lemma` to declare a lemma.
- Using `Proof` to start to prove something.
- Using `intro` to introduce the assumptions.
- Using `exact` if the subgoal matches an hypothesis what we can finish the proof by using this assumption.
- Using `Qed` to end the demonstration.



- `1 subgoal` means you have one goal need to prove.
- `============================` is the dividing line. The above is the assumptions and conditions. The following is the part that needs proof.

Assumptions

E.g.

```
Variables P Q R : Prop.
Lemma C : (P -> Q) -> (Q -> R) -> P -> R.
intro pq.
intro qr.
intro p.
apply qr.
apply pq.
exact p.
Qed.
```

- `intro pq` means P -> Q.
- Using `apply qr` to apply the assumption which we want.

Disjunction

E.g.

```
Variables P Q R : Prop.
Lemma inl : P -> P \/ Q.
intros p.
left.
exact p.
Qed.
```

Bool

Defining & Operating

First of all

Trips:

- `bool = { true , false }` it's a definition.
- `negb` is a function and it can be defined by pattern.

E.g.

```
Definition negb (b:bool) : bool :=
  match b with
  | true => pattern1
  | false => pattern2
  end.
```

Some boolean functions can be defined easily.

```
Definition andb(b c:bool) : bool :=
  if b then c else false.
```

Reasoning about Bool

E.g.

```
Lemma negb_idem' : forall b :bool, negb (negb b) = b.
intro b.
destruct b;
  reflexivity.
Qed.
```

E.g.

```
Lemma andb_comm : forall x y : bool, andb x y = andb y x.
intros x y.
destruct x;
  (destruct y;
     reflexivity).
Qed.
```

- Using `destruct` b creates a case for b = true and one for b = false.
- Using `reflexivity` to make some simplified goals.