CSE-433 Assignment: Proof Terms

gla@postech

In this assignment, you will practice proof terms in propositional logic (Section `ProofTerm`).

Proof terms

In the previous assignments, we exploited tactics and tacticals of Coq to prove theorems in propositional logic. Since proof terms are compact representations of proofs, we can translate all these proofs into corresponding proof terms. In fact, we can just use the Coq command `Print` to displays all such proof terms. For example, we can print the proof term for `A -> A` once we complete its proof using tactics as follows:

```
Coq < Theorem id : A -> A.
1 subgoal

  ============================
   A -> A

...

id < Qed.
intro x.
assumption.
id is defined

Coq < Print id.
id = fun x : A => x
     : A -> A
```

In order to use a proof term in proving a theorem, we use the command `Definition`. For example, we can define `id` by directly providing a proof term for it as follows:

```
Coq < Definition id : A -> A := fun x : A => x.
id is defined

Coq < Print id.
id = fun x : A => x
     : A -> A
```

`Definition` uses the following syntax:

`Definition` ⟨*identifier*⟩ `:` ⟨*proposition*⟩ `:=` ⟨*proof term*⟩`.`

Proof terms for propositional logic in Coq use slightly different syntax from the simply typed λ-calculus. The following table shows how to convert proof terms in the simply typed λ-calculus into Coq:

Simply-typed λ-calculus	Coq
$\lambda x\!:\!A.\, M$	`fun` x `:` A `=>` M
$\lambda x\!:\!A.\, \lambda y\!:\!B.\, M$	`fun (`x `:` A`) (`y `:` B`) =>` M
$\lambda x\!:\!A.\, \lambda y\!:\!B. \cdots \lambda z\!:\!C.\, M$	`fun (`x `:` A`) (`y `:` B`)` $\cdots$ `(`z `:` C`) =>` M
$M\ N$	$M\ N$
(M, N)	`conj` $M\ N$
$\mathsf{fst}\ M$ where $M : A \wedge B$	`and_ind (fun (p :` A`) (q :` B`) => p)` M
$\mathsf{snd}\ M$ where $M : A \wedge B$	`and_ind (fun (p :` A`) (q :` B`) => q)` M
$\mathsf{inl}_A\ M$	`or_introl` $A\ M$
$\mathsf{inr}_A\ M$	`or_intror` $A\ M$
$\mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N_1 \mid \mathsf{inr}\ y.\, N_2$ where $M : A \vee B$	`or_ind (fun` x `:` A `=>` N_1`) (fun` y `:` B `=>` N_2`)` M
$()$	`I`
$\mathsf{abort}_C\ M$	`False_ind` $C\ M$

Note that Coq provides just a single term `and_ind` for eliminating conjunction, which can be thought of as combining the two elimination rules for conjunction. To see how `and_ind` works, `Check` it out!

```
Coq < Check and_ind.
and_ind
     : forall A B P : Prop, (A -> B -> P) -> A /\ B -> P
```

Also `Check` out other terms such as `conj`, `or_introl`, `or_intror`, `or_ind`, and `False_ind`.

Proof terms in propositional logic

Complete all definitions in Section `ProofTerm` using proof terms.