Constructive Logic (15-317), Fall 2015
Recitation 4: Classical Logic and Exam Review (9/30/15)
Vincent Huang (vincenth@andrew), Anna Gommerstadt (hgommers@cs), Michael Coblenz (mcoblenz@cs)

Part I
Classical Logic

Early in class, someone proposed defining $A \supset B$ as $\neg A \vee B$. We've found that this is not provable constructively, but it's provable if we assume classical rules PBC and contra. Use these rules to prove the proposition.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \supset B\ true}\,u \quad \dfrac{}{A\ true}\,w}{B\ true}\supset E}{\neg A \vee B\ true}\vee I_R \quad \dfrac{}{\neg A \vee B\ false}\,k}{\dfrac{\bot}{\neg A\ true}\supset I^w}\,contra \qquad \dfrac{}{\neg A \vee B\ false}\,k}{\dfrac{\dfrac{\#}{\neg A \vee B\ true}\,PBC^k}{(A \supset B) \supset \neg A \vee B\ true}\supset I^u}\,contra \qquad \dfrac{}{\neg A \vee B\ false}\,k}{\dfrac{\dfrac{\dfrac{\dfrac{}{u : A \supset B\ true}\,u \quad \dfrac{}{w : A\ true}\,w}{u\ w : B\ true}\supset E}{\mathbf{inr}\ u\ w : \neg A \vee B\ true}\vee I_R \quad \dfrac{}{k : \neg A \vee B\ false}\,k}{\mathbf{throw}\ k\ (\mathbf{inr}\ u\ w) : \bot}\,contra}{\lambda w : A.\mathbf{throw}\ k\ (\mathbf{inr}\ u\ w) : \neg A\ true}\supset I^w}{\mathbf{inl}\ (\lambda w : A.\mathbf{throw}\ k\ (\mathbf{inr}\ u\ w)) : (\neg A \vee B\ true)}\vee I_L}\,contra}{\mathbf{throw}\ k\ (\mathbf{inl}\ (\lambda w : A.\mathbf{throw}\ k\ (\mathbf{inr}\ u\ w))) : \#}}{\mathcal{C}k.\mathbf{throw}\ k\ (\mathbf{inl}\ (\lambda w : A.\mathbf{throw}\ k\ (\mathbf{inr}\ u\ w))) : \neg A \vee B\ true}\,PBC^k}{\lambda u : (A \supset B).\mathcal{C}k.\mathbf{throw}\ k\ (\mathbf{inl}\ (\lambda w : A.\mathbf{throw}\ k\ (\mathbf{inr}\ u\ w))) : (A \supset B) \supset \neg A \vee B\ true}\supset I^u$$

Part II
Exam Review

Task 1. Prove $A \wedge (A \supset B) \wedge (B \supset C) \supset C\ true$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge (A \supset B) \wedge (B \supset C)\ true}\,u}{(A \supset B) \wedge (B \supset C)\ true}\wedge E_R}{B \supset C\ true}\wedge E_R \quad \dfrac{\dfrac{\dfrac{}{A \wedge (A \supset B) \wedge (B \supset C)\ true}\,u}{(A \supset B) \wedge (B \supset C)\ true}\wedge E_2}{A \supset B\ true}\wedge E_L \quad \dfrac{\dfrac{}{A \wedge (A \supset B) \wedge (B \supset C)\ true}\,u}{A\ true}\wedge E_L}{B\ true}\supset E}{C\ true}\supset E}{A \wedge (A \supset B) \wedge (B \supset C) \supset C\ true}\supset I^u$$

Harmony

Task 2. Define a connective that is either not locally sound or locally complete, but not both. Write down its introduction and elimination rules:

Check: your rules above should not use any connectives other than the one you're defining. If they do, revise your rules.

Task 3. *Semantics*: Write down an English description of what your connective means. Assume that you are a verificationist. Now, write down a description again assuming you are not a verificationist (i.e you are a pragmatist).

Task 4. *Harmony*: Trade papers with a classmate. Attempt to write soundness and completeness proofs for your classmate's connective.

Task 5. *Proof terms* Define proof terms for your new connective. Check your classmate's definitions and see if you think they make sense.

Quantifiers Prove or explain why these are not provable. Solutions are given in Tutch code because I happen to have it on hand:

Task 6.

$$(\exists x : \tau.P \wedge Q(x)) \supset (P \wedge \exists x : \tau.Q(x))$$

```
proof ExLem : (?x:t. P & Q x) => P & ?x:t. Q x =
begin
[?x:t. P & Q x;
        [x:t, P & Q x;
                P;
                Q x;
                ? x:t . Q x;
                P & ?x:t . Q x];
        P & ?x:t. Q x];
 (?x:t. P & Q x) => P & ?x:t. Q x;
end;
```

Task 7.

$$(P \wedge \exists x : \tau.Q(x)) \supset (\exists x : \tau.P \wedge Q(x))$$

```
proof AndEx: (P & ?x:t. Q(x)) => (?x:t. P & Q(x)) =
begin
```

```
[P & ?x:t. Q x;
        P;
        ?x:t. Q x;
        [x:t, Q x;
                P;
                P & Q(x);
                ?x:t. P & Q(x)];
        ?x:t. P & Q(x)];
 (P & ?x:t. Q(x)) => (?x:t. P & Q(x));
end;
```

Task 8.

$$(P \vee \forall x : \tau.Q(x)) \supset (\forall x : \tau.P \vee Q(x))$$

```
proof OrAll: (P | !x:t. Q(x)) => (!x:t. P | Q(x)) =
begin
[(P | !x:t. Q(x));
        [P;
                [x:t;
                        P;
                        P | Q (x)];
                !x:t.P | Q(x)];
        [!x:t. Q(x);
                [x:t;
                        Q(x);
                        P | Q(x)];
                !x:t. P | Q(x)];
        (!x:t. P | Q(x))];
 (P | !x:t. Q(x)) => (!x:t. P | Q(x));
end;
```

Primitive recursion

Task 9. Assume R as in class. Use R to implement `test-primality`, which takes a natural number n and returns true or false according to whether the number is prime. You may assume you have a function *mod* that tells you the remainder upon dividing one natural number by another.