Classical logic is cool! Standard ML is cool! 
... oh, no...  
The Tutch portion of your work should be submitted electronically using the command

```
$ /afs/andrew/course/15/317/bin/submit -r hw4 hw4.tut
```

from any Andrew server. You may check the status of your submission by running the command

```
$ /afs/andrew/course/15/317/bin/status hw4
```

If you have trouble running either of these commands, email Anna, Michael, or Vincent.

The SML portion of your work should be submitted electronically by placing `hw04.sml` in your handin directory (`/afs/andrew/course/15/317/submit/andrewid/hw4`).

The written portion of your work should be submitted at the beginning of class. If you are familiar with \LaTeX, you are encouraged to use this document as a template for typesetting your solutions, but you may alternatively write your solutions neatly by hand.

\section{An instant classic (6 points)}

Tutch allows you to do classical proofs by employing the following rule:

\[
\begin{array}{c}
\neg A \\
\vdots \\
\bot \\
\hline
\end{array}
\quad
\begin{array}{c}
u \\
\vdots \\
PBC^u_{\text{Tutch}}
\end{array}
\]

\[
A \quad \text{true}
\]

1
Task 1. Derive this rule using only the PBC\(^k\) and contra rules for classical logic, plus the usual natural deduction rules.

Solution

\[
\frac{\lnot \text{true} \quad \text{true}}{
\lnot \text{true} \quad \lnot \text{true} \quad \text{true} \quad \text{false}}
\]

2 Can Tutch this (6 points)

Prove the following theorems in Tutch, using classical reasoning if necessary (i.e. don’t sprinkle contradictions all over the place just because you can).

classical proof greaterThanDNE : \(~\lnot A \rightarrow \lnot A;\)
classical proof notAllProps : \(\lnot (!x:t. \lnot A(x)) \rightarrow ?x:t. A(x);\)

Solution

Solutions not provided for Tutch problems

3 The best programming language in the world (6 points)

In the code handout for this assignment, you will find...gasp...some actual code this week! It may be helpful for you to look at http://www.smlnj.org/doc/SMLofNJ/pages/cont.html. Then write the following:

Task 2. val magic : unit \(\rightarrow\) ('a \(\rightarrow\) 'b, 'b \(\rightarrow\) 'a) either

Task 3. val more_magic : ('a \(\rightarrow\) 'b) \(\rightarrow\) ('a \(\rightarrow\) 'c, 'b) either

In case the SMLofNJ in the code handout wasn’t enough of a hint, please use SML/NJ for this assignment. We have provided a signature file and a sources.cm file for you to load your code into the REPL.

\(^1\)You might also want to note that the person who set this assignment has no idea what isolate, 'a control, cont, capture, and escape are, so you probably don’t need to know either.
Also, please don’t use exceptions, things from the Unsafe structure, or whatever other twisted ways you can think of to do this. You get that we want you to practise using continuations. Be good. Do that.

Solution

Solutions not provided for SML problems

4 Konsidering komputation (2 points)

Task 4. Explain the computational content of the SML code you wrote. That is, what do the results of magic and more_magic do if you run them?

Solution

magic () produces a function of type 'a -> 'b that, when called, steals the 'a you gave it, time-travels, and gives you a function that, given a 'b, produces the aforementioned 'a and gives it back to you.

more_magic, when you supply it a function that takes 'a’s to 'b’s, gives you what’s essentially an 'a cont, or a ¬A, but if you try to use that, it’ll steal your 'a, time travel, apply your function, and give you back the resulting 'b.