Constructive Logic (15-317), Fall2021
Assignment 3: Proofs as Programs + Verifications and Uses

Instructor: Karl Crary
TAs: Avery Cowan, Katherine Cordwell, Matias Scharager, Antian Wang

Due: Wednesday, September 22, 2021, 11:59 pm

The assignments in this course must be submitted electronically through Gradescope. Written homework PDFs and coding SML files will both go to Gradescope. For this homework, submit two files:

- `hw3.pdf` (your written solutions)
- `hw3.sml` (your coding solutions)

Trees are Programs

Task 1 (12 points)**.** Prove the following theorems using the proof-as-program logic in SML. You can look at `support/pap_examples.sml` for reference proofs as programs proof trees.

a. `prove deMorgagain`: $\neg A \wedge \neg B \supset \neg(A \vee B)$

b. `prove toptobottom`: $(A \supset \top) \wedge (\bot \supset A)$

c. `prove reuse`: $((A \supset B) \wedge (A \supset C)) \supset (A \supset B \wedge C)$

d. `prove ormap`: $((A \vee B) \supset C) \supset (A \supset C) \wedge (B \supset C)$

You can compile your code the same way as for natural deduction. You can pretty print your proof-as-program proof trees by running the following command in your repl:

```
>> Out.print_pap Homework3.{proof_name_here}
```

A wild FUNCTION has appeared!

Task 2 (8 points)**.** For this task, you will be directly writing the code that inhabits the corresponding type for a proposition. For each proposition, either submit `SOME(v)` where `v` is a value[1] of that type or leave it as `NONE` if the proposition is unprovable[2]. Rather than using the course infrastructure for proof-as-program trees, this question will now study SML programs in their natural habitat.

We provide you with the `void` type[3] and `abort` function[4] to deal with falsehood. Similarly, you have access to the built-in structure `Either` [5] in order to deal with $\vee$.

a. `prove curry`: $(A \wedge B \supset C) \supset A \supset B \supset C$

b. `prove abba`: $((A \supset B) \supset B) \supset A$

c. `prove contrapositive`: $(A \supset B) \supset (\neg B \supset \neg A)$

d. `prove exclusion`: $((A \vee B) \wedge \neg A) \supset B$

I thunk therefore I am

Task 3 (8 points)**.** Consider a unary connective $\circ$ defined by the following rules:

$$\dfrac{\begin{array}{c}\overline{\top\ \textit{true}}\ u \\ \vdots \\ A\ \textit{true}\end{array}}{\circ A\ \textit{true}}\ \circ I^u \qquad \dfrac{\circ A\ \textit{true} \quad \top\ \textit{true}}{A\ \textit{true}}\ \circ E$$

1. Can you prove a simple relationship between A *true* is $\circ A$ *true*?
2. Using **thunk**$(u.M)$ as the proof term for the intro rule (aka introduction form), give the appropriate intro rule. for **thunk**$(u.M) : \circ A$.
3. Using $M << N$ as the proof term for the elim rule (aka elimination form), give the appropriate elim rule. for $M << N : A$.
4. Does $\circ$ have a contraction rule[6]? Write out a contraction rule for $\circ$ if one exists. Otherwise, show that no reduction rule is possible.
5. Why might a programming language or programmer want to use thunks in code?[7]

[1] A value is an expression that has finished executing. For this problem, we will also accept your answer if v is an expression that reduces to a value.

[2] Proving the totality of functions using exceptions or recursion is nontrivial so do not use exceptions or recursion for this task

[3] `datatype void = (* no constructors *)`

[4] `abort:  void -> 'a`

[5] `dataype ('a, 'b) either = INL of 'a | INR of 'b`

[6] Remember that a contraction rule shows how to reduce the elimination form of a connective to a simpler term

[7] Any reasonable guess is fine

Verifications

Consider the $\clubsuit$ connective.

$$\dfrac{\begin{array}{cc}\overline{A\ \mathit{true}}^{\,u} & \overline{A\ \mathit{true}}^{\,v}\\ \vdots & \vdots \\ B\ \mathit{true} & C\ \mathit{true}\end{array}}{\clubsuit(A,B,C)\ \mathit{true}}\ \clubsuit I^{u,v} \qquad \dfrac{\begin{array}{ccc} & & \overline{B\ \mathit{true}}^{\,u} \\ & & \vdots \\ \clubsuit(A,B,C)\ \mathit{true} & A\ \mathit{true} & D\ \mathit{true}\end{array}}{D\ \mathit{true}}\ \clubsuit E_1^u \qquad \dfrac{\begin{array}{ccc} & & \overline{C\ \mathit{true}}^{\,u} \\ & & \vdots \\ \clubsuit(A,B,C)\ \mathit{true} & A\ \mathit{true} & D\ \mathit{true}\end{array}}{D\ \mathit{true}}\ \clubsuit E_2^u$$

Task 4 (5 points)**.** Give rules for forming the judgments that $\clubsuit(A,B,C)$ has a verification and that $\clubsuit(A,B,C)$ can be used.

Task 5 (4 points)**.** Give a verification for this proposition

$$(\neg A \wedge B) \supset ((A \supset B) \supset (\neg A \supset \neg B)) \supset \bot$$

For clarification on how to write a verifications-and-uses proof, please look at `examples/vau_examples.sml`

Task 6 (10 points)**.** For each of the following propositions, give a verification-and-uses proof and its **corresponding** proofs-as-programs term.

1. $\bot \supset \top$
2. $\bot \supset \top$ **(Do not use the same verification/proof term as part a. Use a new one.)**
3. $(A \supset B) \supset (\neg B \supset \neg A)$
4. $(A \supset B) \supset (B \supset C) \supset (A \supset C)$