Constructive Logic (15-317), Spring 2023 Assignment 3: Sequent Calculus and Rules as Algorithms (60 points)

Instructor: Frank Pfenning

Due: February 16, 2023, 11:59 pm

This assignment has a written portion and two coding portions. You will submit all portions through Gradescope.

We recommend that you typeset your written solutions. Most students use LaTeX, but other software is acceptable. If you choose not to typeset your solutions, be aware that you are answerable for your handwriting. Any that the grader has difficulty reading (in the sole judgement of the grader), will be marked wrong.

For part the coding portion you will use Dcheck, for the other SML. You can find documentation on Dcheck at `cs.cmu.edu/~crary/dcheck/dcheck.pdf` and a sample file at `cs.cmu.edu/~crary/dcheck/example.deriv`.

1 Sequent Calculus (12 points)

Provide Dcheck derivations of the following Sequent Calculus judgements using the `SC` system. Your solutions should go in `hw3.deriv`.

Task 1 (4 points)**.** Define a derivation named `task1` that derives:

$$\Longrightarrow ((P \supset R) \wedge (Q \supset R)) \supset ((P \vee Q) \supset R)$$

Task 2 (4 points)**.** Define a derivation named `task2` that derives:

$$\Longrightarrow (P \supset (Q \supset R)) \supset ((P \supset Q) \supset (P \supset R))$$

Task 3 (4 points)**.** Define a derivation named `task3` that derives:

$$R \supset P \vee Q, (P \wedge Q) \supset R \Longrightarrow P \supset (Q \supset R)$$

Note that Dcheck takes P, Q, and R to be atomic propositions.

2 Unprovability (10 points)

Task 4 (5 points)**.** Prove that double negation elimination does not hold in the sequent calculus. In other words, show that the following is not provable for a propositional variable P.

$$\Longrightarrow (\neg\neg P) \supset P$$

Task 5 (5 points)**.** In lecture, we discussed that the elimination rules for conjunction are not locally complete if we had "forgotten" the $\wedge E_2$ rule, but we did not have the means to prove it rigorously.

Which theorem *about* the sequent calculus fails if we "forget" the corresponding rule $\wedge L_2$? *Prove* that this important property indeed fails.

3 Applications of the Admissibility of Cut (20 points)

Task 6 (10 points)**.** Using the admissibility of weakening, contraction, cut, and identity from Lecture 8 as needed, prove:

If $\Delta, A \wedge B \Longrightarrow C$ *then* $\Delta, A, B \Longrightarrow C$.

In particular, you should not use any induction in your argument.

Task 7 (10 points)**.** Using the admissibility of weakening, contraction, cut, and identity from Lecture 8 as needed, prove:

If $\Delta \Longrightarrow A \supset B$ *and* $\Delta \Longrightarrow A$ *then* $\Delta \Longrightarrow B$.

In particular, you should not use any induction in your argument.

4 Rules as Algorithms (18 points)

In recitation, we implemented addition and multiplication for natural numbers as follows:

```
datatype nat = zero | succ of nat
fun plus (zero) y = y
  | plus (succ x) y = succ (plus s y)
fun times zero y = zero
  | times (succ x) y = plus (y) (times (x) (y))
```

We turn now to binary numbers. Let `e` be the empty number (representing 0), $\texttt{b0}(n)$ be n followed by the bit 0. Similarly, let $\texttt{b1}(n)$ denote n followed the bit 1. For example, $\texttt{b0}(\texttt{b1}(\texttt{b1}(\texttt{e})))$ represents the number 6, $\texttt{b1}(\texttt{b0}(\texttt{b1}(\texttt{b1}(\texttt{e}))))$ represents the number 13, and $\texttt{b1}(\texttt{b0}(\texttt{b1}(\texttt{b1}(\texttt{b0}(\texttt{e})))))$ also represents the number 13.

In this system, the successor of `e` is $\texttt{b1}(\texttt{e})$. Furthermore, the successor of $\texttt{b0}(m)$ is $\texttt{b1}(m)$ for all m, and for arbitrary numbers m and n, the successor of $\texttt{b1}(m)$ is $\texttt{b0}(n)$ if the successor of m is n.

Task 8 (5 points)**.** Write a collection of inference rules such that $\mathsf{succ}\ m\ n$ can be derived if and only if n is a successor m.

Task 9 (2 points)**.** State the theorem that every binary number has a successor. You do not need to prove it.

Task 10 (3 points)**.** Extract a *function* $\mathsf{inc} : \mathsf{num} \to \mathsf{num}$ from your rules. It should be the computational content of a proof of the theorem in Task 9.

Task 11 (3 points)**.** State a theorem that binary numbers may or may not have a predecessor. You do not need to prove it.

Task 12 (5 points)**.** Give the type corresponding to your theorem in Task 11 and extract a function dec of this type from your rules. It should be the computational content of a proof of the theorem in Task 11.

The functions from Tasks 10 and 12 should be in a file `hw3.sml`, the template of which is provided in the starter code.