Constructive Logic (15-317), Fall 2024
Assignment 7: Theorem Proving

Constructive Logic Staff
(Instructor: Karl Crary)

Due: Wednesday, October 23, 2024, 11:59 pm

This is an SML coding assignment. Please submit a file named "`hw.sml`" to "Homework 7."

1 Theorem Proving

In this assignment you will implement a theorem prover for propositional logic. You have considerable flexibility in how to accomplish this task. The specification only requires you to implement a module `Prover` that matches the signature:

```
signature PROVER =
   sig
      val prove : Logic.prop -> bool
   end
```

The `Prover.prove` function called with P should return true if and only if $\longrightarrow P$ is derivable (equivalently, if and only if P `true` is derivable). Propositions are given (in the `Logic` module) by the datatype:

```
datatype prop =
   Atomic of string
 | And of prop * prop
 | Implies of prop * prop
 | Or of prop * prop
 | True
 | False
```

Although you have considerable latitude in how to implement your prover, we have some advice:

- Unless you are already knowledgeable about propositional theorem proving, **you should definitely use the G4ip system presented in class.** We have not taught any other technique that will work. For your convenience, the G4ip rules are given in the appendix.
- Theorem proving is the sort of search-and-backtrack problem that continuations excel at implementing. We recommend you use continuation-passing style with a success continuation and a failure continuation. If you do, you will probably find the problem much easier than if you do not.

- Certain rules are *invertible* (a.k.a. *asynchronous*), which means that it is never a mistake to apply them. It is a good idea to use all applicable asynchronous rules first, before you start to consider synchronous rules that you might need to backtrack from. Asynchronous G4ip rules are listed in the appendix.
- The rules governing atomic propositions (*init* and $P{\supset}L$) are special cases that may require special handling. It is never a mistake to use them when applicable, so it makes sense to use them eagerly when you can. However, when they are not applicable, one must account for the possibility that they might become applicable later on.
- One way to realize the above advice is distilled in the set of rules given in the second appendix.
- The rules of G4ip (and even G4ip in the Inversion Calculus) are non-deterministic. The most challenging part of this assignment may be to convert the into a deterministic algorithm. Think carefully about your algorithm before you start code.
- Many students find this assignment challenging. We recommend that you start early.
- To assist in testing and debugging, the `Logic` module provides some utilities for displaying and parsing propositions.

Grading The typechecker requires that you provide a well-typed module `Prover :> PROVER`. Submissions that fail to type check will receive a zero.

Submissions that cause the grader to timeout will also receive a zero. The easiest way to timeout is if your prover loops, but it can also happen if your prover suffers from particularly bad exponential behavior.[1] Think carefully about your algorithm!

Any submission that passes all of our tests will be deemed apparently correct, and will receive 100%. Submissions that fail any tests are incorrect, and will receive a maximum of 75%. The actual score will vary depending on how many tests are failed.

We will **not** be disclosing the tests. To do so would enable submissions that are based on recognizing specific instances, rather than on general theorem proving, which would defeat the purpose of the exercise.

Collaboration All the code you submit must be your own. You are not permitted to share your code with anyone, or to use any code that you did not write yourself. **Except:** you are permitted and indeed encouraged to share test cases with other students. You may also discuss your algorithm with other students abstractly, so long as you do not share SML code.

[1]Alas, some exponential behavior is almost certainly inevitable since propositional constructive logic is PSPACE-complete.

A G4ip rules

The rules of G4ip (a.k.a. Dyckhoff's contraction-free sequent calculus) are as follows. In these rules, P stands for atomic propositions.

$$\frac{P \in \Delta}{\Delta \longrightarrow P}\; \mathit{init}$$

$$\frac{\Delta, A, B \longrightarrow C}{\Delta, A \wedge B \longrightarrow C}\; {\wedge}L \qquad \frac{\Delta \longrightarrow A \quad \Delta \longrightarrow B}{\Delta \longrightarrow A \wedge B}\; {\wedge}R$$

$$\frac{\Delta, A \longrightarrow C \quad \Delta, B \longrightarrow C}{\Delta, A \vee B \longrightarrow C}\; {\vee}L \qquad \frac{\Delta \longrightarrow A}{\Delta \longrightarrow A \vee B}\; {\vee}R1 \qquad \frac{\Delta \longrightarrow B}{\Delta \longrightarrow A \vee B}\; {\vee}R2$$

$$\frac{\Delta \longrightarrow C}{\Delta, T \longrightarrow C}\; \mathsf{T}L \qquad \frac{}{\Delta \longrightarrow T}\; \mathsf{T}R \qquad \frac{}{\Delta, F \longrightarrow C}\; \mathsf{F}L$$

$$\frac{\Delta, A \longrightarrow B}{\Delta \longrightarrow A \supset B}\; {\supset}R$$

The rules above are the same as in the reduced sequent calculus. The following are the strange rules of G4ip that replace $\supset L$:

$$\frac{\Delta, A \longrightarrow D}{\Delta, \mathsf{T} \supset A \longrightarrow D}\; \mathsf{T}{\supset}L \qquad \frac{\Delta \longrightarrow D}{\Delta, \mathsf{F} \supset A \longrightarrow D}\; \mathsf{F}{\supset}L$$

$$\frac{\Delta, A \supset B \supset C \longrightarrow D}{\Delta, (A \wedge B) \supset C \longrightarrow D}\; {\wedge}{\supset}L \qquad \frac{\Delta, A \supset C, B \supset C \longrightarrow D}{\Delta, (A \vee B) \supset C \longrightarrow D}\; {\vee}{\supset}L$$

$$\frac{P \in \Delta \quad \Delta, A \longrightarrow D}{\Delta, P \supset A \longrightarrow D}\; P{\supset}L$$

$$\frac{\Delta, B \supset C, A \longrightarrow B \quad \Delta, C \longrightarrow D}{\Delta, (A \supset B) \supset C \longrightarrow D}\; {\supset}{\supset}L$$

The rules $\wedge L$, $\wedge R$, $\vee L$, $\mathsf{T}L$, $\mathsf{T}R$, $\mathsf{F}L$, $\supset R$, $\mathsf{T}{\supset}L$, $\mathsf{F}{\supset}L$, $\wedge{\supset}L$, and $\vee{\supset}L$ are asynchronous. The rules *init*, $\vee R1$, $\vee R2$, $P{\supset}L$, and ${\supset}{\supset}L$ are synchronous.

B G4ip in the Inversion Calculus

Three judgements indicate different states in the proof-search process. First decompose A on the right, then decompose members of Ω on the left, then search through synchronous rules:

right inversion	$\Delta;\Omega \xrightarrow{R} A$
left inversion	$\Delta;\Omega \xrightarrow{L} C^+$
search	$\Delta \xrightarrow{S} C^+$

Here, P refers only to atomic propositions, A^+ (etc.) refers to right synchronous propositions, A^- (etc.) refers to left synchronous propositions, Δ contains only left synchronous propositions, and Ω contains arbitrary propositions. Left and right synchronous propositions are given by the grammar:

$$\begin{array}{lrcl} \text{left synchronous} & A^- & ::= & P \mid P \supset B \mid (B \supset C) \supset D \\ \text{right synchronous} & A^+ & ::= & P \mid B \vee C \mid F \end{array}$$

Right Inversion

$$\frac{\Delta;\Omega \xrightarrow{R} A \quad \Delta;\Omega \xrightarrow{R} B}{\Delta;\Omega \xrightarrow{R} A \wedge B}\,\wedge R \qquad \frac{}{\Delta;\Omega \xrightarrow{R} T}\,\mathsf{T}R \qquad \frac{\Delta;\Omega, A \xrightarrow{R} B}{\Delta;\Omega \xrightarrow{R} A \supset B}\,\supset\! R$$

Switch

$$\frac{\Delta;\Omega \xrightarrow{L} P}{\Delta;\Omega \xrightarrow{R} P}\,LR_P \qquad \frac{\Delta;\Omega \xrightarrow{L} A \vee B}{\Delta;\Omega \xrightarrow{R} A \vee B}\,LR_\vee \qquad \frac{\Delta;\Omega \xrightarrow{L} F}{\Delta;\Omega \xrightarrow{R} F}\,LR_F$$

Left Inversion

$$\frac{\Delta;\Omega, A, B \xrightarrow{L} C^+}{\Delta;\Omega, A \wedge B \xrightarrow{L} C^+}\,\wedge L \qquad \frac{\Delta;\Omega \xrightarrow{L} C^+}{\Delta;\Omega, T \xrightarrow{L} C^+}\,\mathsf{T}L \qquad \frac{\Delta;\Omega, A \xrightarrow{L} C^+ \quad \Delta;\Omega, B \xrightarrow{L} C^+}{\Delta;\Omega, A \vee B \xrightarrow{L} C^+}\,\vee L$$

$$\frac{}{\Delta;\Omega, F \xrightarrow{L} C^+}\,\mathsf{F}L \qquad \frac{\Delta;\Omega, A \xrightarrow{L} D}{\Delta;\Omega, \mathsf{T} \supset A \xrightarrow{L} D}\,\mathsf{T}\!\supset\! L \qquad \frac{\Delta;\Omega \xrightarrow{L} D}{\Delta;\Omega, \mathsf{F} \supset A \xrightarrow{L} D}\,\mathsf{F}\!\supset\! L$$

$$\frac{\Delta;\Omega, A \supset B \supset C \xrightarrow{L} D}{\Delta;\Omega, (A \wedge B) \supset C \xrightarrow{L} D}\,\wedge\!\supset\! L \qquad \frac{\Delta;\Omega, A \supset C, B \supset C \xrightarrow{L} D}{\Delta;\Omega, (A \vee B) \supset C \xrightarrow{L} D}\,\vee\!\supset\! L$$

Shift

$$\frac{\Delta, P;\Omega \xrightarrow{L} C^+}{\Delta;\Omega, P \xrightarrow{L} C^+}\,\mathit{shift}_P \qquad \frac{\Delta, P \supset B;\Omega \xrightarrow{L} C^+}{\Delta;\Omega, P \supset B \xrightarrow{L} C^+}\,\mathit{shift}_{P\supset} \qquad \frac{\Delta, (A \supset B) \supset C;\Omega \xrightarrow{L} D^+}{\Delta;\Omega, (A \supset B) \supset C \xrightarrow{L} D^+}\,\mathit{shift}_{\supset\supset}$$

Search

$$\frac{\Delta \xrightarrow{S} C^+}{\Delta;\cdot \xrightarrow{L} C^+}\,\mathit{search} \qquad \frac{P \in \Delta}{\Delta \xrightarrow{S} P}\,\mathit{init} \qquad \frac{\Delta;\cdot \xrightarrow{R} A}{\Delta \xrightarrow{S} A \vee B}\,\vee R1 \qquad \frac{\Delta;\cdot \xrightarrow{R} B}{\Delta \xrightarrow{S} A \vee B}\,\vee R2$$

$$\frac{P \in \Delta \quad \Delta; A \xrightarrow{L} D^+}{\Delta, P \supset A \xrightarrow{S} D^+}\,P\!\supset\! L \qquad \frac{\Delta; B \supset C, A \xrightarrow{R} B \quad \Delta; C \xrightarrow{L} D^+}{\Delta, (A \supset B) \supset C \xrightarrow{S} D^+}\,\supset\supset\! L$$