Constructive Logic (15-317), Fall 2016
Assignment 8

Jon Sterling

Due November 9

This assignment is due at the beginning of class on the above date and must be submitted electronically via autolab. Submit your homework via autolab as a file named **hw8.pdf**.

Focusing and Chaining

A major theme of this course has been the discovery of theory through practice: strategies for efficient proof search in the concrete conditions of real-world implementations are transformed into razor-edged intellectual weapons, entirely new logics which sharpen the principal contradiction of proof theory: the dialectic of the *positive* and *negative* (polarity).

The decomposition of *truth* into *verification* and *use* was our first encounter with the scientific law, "One Divides Into Two". By studying invertibility in the context of the sequent calculus (when does a conclusion imply its premises?), we were able to achieve a firmer grasp of the fault-lines at play, summarized in a dangerously over-simplified form below:

	LEFT RULE	RIGHT RULE
POSITIVE	invertible	**non-invertible**
NEGATIVE	**non-invertible**	invertible

Inversion Invertible rules can always be applied without any need for backtracking: since the conclusion of an invertible rule implies its premises, the "future truth" of the goal is preserved under free application of such rules. This practical insight, which is crucial for implementing a performant proof search engine, can be codified by sharpening the logic to include deterministic inversion phases $\Gamma; \Omega \longrightarrow_{\mathsf{L}} C$ and $\Gamma; \Omega \longrightarrow_{\mathsf{R}} C$ (where Ω is an ordered context of propositions).

Chaining While the above gives a clear and deterministic account of invertible rules, the non-invertible ones beg for something similar. In this week's lecture, we began to study *chaining*, which fixes a dynamics for the non-invertible rules based on two forms of judgment, $\Gamma \longrightarrow [A^+]$ and $\Gamma; [A^-] \longrightarrow C$. Chaining is a technique to minimize backtracking by applying a sequence of non-invertible rules in one go.

1 Practicing focusing

Task 1 (10 pts)**.** Consider following polarized sequent:

$$\cdot \longrightarrow \big((a^+ \supset b^-) \wedge (a^+ \supset c^-)\big) \supset \big(a^+ \supset (b^- \wedge c^-)\big)$$

First, apply inversion to this sequent until it is stable (i.e. there are no more invertible rules to apply). Then, prove the resulting stable sequent(s) in the chaining fragment introduced in the lecture (where shifts are only applied to atoms).

2 Converting between rules and propositions

In class, we learned how to convert between polarized formulas and rules of inference. First, a polarized formula is decomposed into stable sequents using invertible rules; in this exercise, we perform this step for you. Such a stable sequent will present any number of possible opportunities for entering focus, such as left-focusing on a negative hypothesis or right-focusing a positive succedent.

To find all the possible derived rules for such a sequent, try out all possible focusing proofs by fixing an arbitrary context Γ (and in the case of left-focus, an arbitrary conclusion C). For instance, to test left-focus for an assumption A^-, begin proving $\Gamma; [A^-] \longrightarrow C$, and then consider the leaves of your proof tree to be the premises of a derived rule with conclusion $\Gamma \longrightarrow C$. To test right-focus for a succedent A^+, begin proving $\Gamma \longrightarrow [A^+]$, and then consider the leaves of your proof tree to the premises of a derived rule with conclusion $\Gamma \longrightarrow A^+$.

During focusing, if there are multiple rules that you can apply (such as when focusing on a conjunction on the left), try them exhaustively, generating as many derived rules as you can.

Task 2 (5 pts)**.** Consider the following sequent:

$$a^-, b^-, {\downarrow}a^- \wedge {\downarrow}b^- \supset c^- \longrightarrow c^-$$

Construct the derived rules that correspond to the above sequent; then, derive the sequent using only these derived rules.

Task 3 (5 pts)**.** Construct the derived rules that correspond to the following sequent, and then use these rules to derive it:

$${\downarrow}a^- \supset ({\downarrow}b^- \supset c^-), a^- \wedge b^- \longrightarrow c^-$$

Task 4 (5 pts)**.** Construct the derived rules that correspond to the following sequent, and then use these rules to derive it:

$$a^+, b^+, a^+ \wedge b^+ \supset {\uparrow}c^+ \longrightarrow c^+$$

Task 5 (5 pts)**.** Construct the derived rules that correspond to the following sequent, and then use these rules to derive it:

$$a^+ \supset (b^+ \supset {\uparrow}c^+), {\uparrow}a^+ \wedge {\uparrow}b^+ \longrightarrow c^+$$

3 Saturation

Consider the following grammar of ground terms representing binary numbers:

$$n \quad ::= \quad \epsilon \mid \mathtt{b0}(n) \mid \mathtt{b1}(n)$$

In class, we learned to write forward logic programs using inference rules; a forward logic programming engine will apply these inference rules until saturation is reached, and then the result of our program can be read from the saturated proof state. In the tasks that follow, you are free to introduce any auxiliary predicates that you require. You need to ensure that your rules *saturate* when new facts of the indicated form are added to the database.

In the problems that follow, you are required to implement forward logic programs by writing down systems of inference rules. You may find it useful to experiment with **DLV**, an implementation of forward logic programming which can be downloaded here: `http://www.dlvsystem.com/dlv/`. **DLV** can be used to test your ideas on specific cases and quickly determine if they are likely to work; but it is not required.

Task 6 (5 pts)**.** Implement a forward logic program $\mathsf{std}(n)$ which derives the atom no iff it is not the case that n is in standard form. You may assume that n is ground (i.e. not subject to unification).

Task 7 (5 pts)**.** Next, implement a forward logic program $\mathsf{succ}(m, n)$ which derives no when it is not the case that $m + 1 = n$. For the purpose of this exercise, you may assume that m and n are ground. You may also assume that m and n are in standard form.