Constructive Logic (15-317), Spring 2021
Assignment 9: Linear Logic, Focusing, and Chaining

Instructor: André Platzer
TAs: Zhibo Chen, Avery Cowan, Julia Gu, Akshina Gupta, Ethan Rosenthal

Due: Friday, May 7, 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, you will be submitting both written pdf files and SML coding files:

- `hw9.pdf` (your written solutions)
- `hw9.sml` (your coding solutions)

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[1] 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 (5 pts)**.** Consider the following depolarized formula:

$$\cdot; \cdot \longrightarrow_{\mathsf{R}} ((A \supset B) \wedge (A \supset C)) \supset (A \supset (B \wedge C))$$

Polarize the formula above and construct a derivation in focused logic for the resulting sequent. (Hint: Make sure to use a single polarization for each atom). You can look at examples of focusing proofs in `examples/f_examples.sml`. There are focusing rules provided in the appendix for reference. [2]

[1] In *structural* or *persistent* logic, some rules which ought to be non-invertible turn out to be invertible; polarity arises properly from the proof search dynamics of *linear logic*, and casts an imperfect shadow in persistent logic.

[2] Make sure that your proofs use the provided rules in the Appendix rather than variations from the notes.

2 Mean, Median...

For each of the following Prolog predicates, give all possible modalities relative to the described intended behavior. For the sake of this problem, negatively-moded parameters *must* eventually terminate when backtracked, but do *not* need to generate all correct possibilities. You may assume that predicates are passed "sane" values (i.e., all arguments are of the correct form).

Task 2 (6 points)**.**

1. `lookup(Map,Key,Value)` is satisfiable when `(Key, Value)` occurs in `Map`.

```
lookup([],_,_) :- false.
lookup([(Key, Value) | _], Key, Value).
lookup([_ | Ms], Key, Value) :- lookup(Ms, Key, Value).
```

2. `sublist_sum(L,N,S)` is satisfiable when S is a sublist of L and sums to N.

```
sublist_sum(_,0,[]).
sublist_sum([],_,_) :- false.
sublist_sum([_ | Xs], N, Ss) :- sublist_sum(Xs, N, Ss).
sublist_sum([X | Xs], N, [X | Ss]) :- N2 is N-X, sublist_sum(Xs, N2, Ss).
```

3. `subset_sum(L,N,S)` is satisfiable when S is a permutation of some sublist of L and sums to N.

```
subset_sum(_,0,[]).
subset_sum([],_,_) :- false.
subset_sum([_ | Xs], N, Ss) :- subset_sum(Xs, N, Ss).
subset_sum([X | Xs], N, [X | Ss]) :- N2 is N-X, subset_sum(Xs, N2, Ss).
```

3 Saturation

Consider the following grammar of ground terms representing binary numbers:

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

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 3 (1 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 4 (3 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.

4 Practicing Linear Logic

Task 5 (9 pts)**.** Prove the following judgments in linear sequent calculus, or state that they do not hold. You can look at examples of linear sequent calculus in `examples/ll_examples.sml`. For your convenience, we supply a self-contained presentation of the rules of linear natural deduction in the Appendix.

1. $A \multimap B \multimap C\ \mathit{true} \Vdash A \otimes B \multimap C\ \mathit{true}$
2. $(A \mathbin{\&} B) \multimap C\ \mathit{true} \Vdash A \multimap (B \multimap C)\ \mathit{true}$
3. $((A \otimes \top) \mathbin{\&} (B \otimes \top)) \multimap C\ \mathit{true} \Vdash A \multimap (B \multimap C)\ \mathit{true}$

5 Proof-Theoretic Harmony

Just like we did in the beginning of the course, we can check a local correctness condition for the rules of linear natural deduction: proof-theoretic harmony.[3] Hint: exhibiting local reductions and expansions in linear logic is subtle: you must be sure to not constrain the contexts Δ in your local reductions and expansions any more than is warranted by the rules of linear logic.

Remark 1 (Linear substitution principle)**.** When exhibiting local reductions and expansions, you will need to use substitutions $[\mathcal{D}/u]\mathcal{E}$. These are governed by the *linear substitution principle*, which states:

$$\text{If } \Delta \Vdash \overset{\mathcal{D}}{A\ \mathit{true}} \text{ and } \Delta', u : A\ \mathit{true} \Vdash \overset{\mathcal{E}}{B\ \mathit{true}}, \text{ then } \Delta, \Delta' \Vdash \overset{[\mathcal{D}/u]\mathcal{E}}{B\ \mathit{true}}.$$

You *must* ensure that your resulting derivations have the correct contexts.

Task 6 (6 pts)**.** Verify that the rules for the tensor $\otimes$ are harmonious in linear natural deduction.

6 Proof Expansions in Linear Logic

Dual to cut and proof reductions are identity and proof expansions. The admissibility of cut in linear sequent calculus tells us that if we can derive $A\ \mathit{true}$ from some assumptions, then those same assumptions entitle us to use the $A\ \mathit{true}$ resource — that is, true propositions can be used as resources. By contrast, the admissibility of a general identity rule in linear logic

$$\frac{}{A\ \mathit{res} \Vdash A\ \mathit{true}}\ \mathsf{id}(A)$$

[3] Harmony is a necessary condition for the correctness of rules, but not a sufficient condition.

tells us the opposite — a resource A *true* is sufficient to conclude that A *true* is derivable. Just as we proved cut admissible by an induction whose individual cases were proof reductions, we prove identity admissible by an induction (this time just on the structure of A) whose individual cases are proof expansions. An example expansion is shown below:

$$\frac{}{A \,\&\, B \text{ true} \Vdash A \,\&\, B \text{ true}}\ \mathsf{id}(A \,\&\, B)$$

(1)	A *true* $\Vdash A$ *true*	By i.h. on A, as A is smaller than $A \,\&\, B$
(2)	$A \,\&\, B$ *true* $\Vdash A$ *true*	By rule $\&L_1$ on (1)
(3)	B *true* $\Vdash B$ *true*	By i.h. on B, as B is smaller than $A \,\&\, B$
(4)	$A \,\&\, B$ *true* $\Vdash B$ *true*	By rule $\&L_2$ on (3)
(5)	$A \,\&\, B$ *true* $\Vdash A \,\&\, B$ *true*	By rule $\&R$ on (2) and (4)

Make sure to clearly state what parameter is smaller than in the current case when using your inductive hypothesis.

Task 7 (3 pts)**.** Complete the following expansion:

$$\frac{}{A \multimap B \text{ true} \Vdash A \multimap B \text{ true}}\ \mathsf{id}(A \multimap B)$$

Task 8 (3 pts)**.** Complete the following expansion:

$$\frac{}{A \otimes B \text{ true} \Vdash A \otimes B \text{ true}}\ \mathsf{id}(A \otimes B)$$

7 Applications

Blocks World is a class of scenarios in which there is a table, some number of blocks which can be stacked on top of each other, and a robotic arm which can pick up and move blocks. We will briefly look at how to model this situation using linear logic. The following atomic predicates are used:

- empty means that the robotic arm's hand is empty.
- $\mathsf{holds}(x)$ means that the hand is holding block x.
- $\mathsf{clear}(x)$ means that the block x does not have anything on top of it.
- $\mathsf{on}(x, y)$ means that the block x is directly on top of the block y.
- $\mathsf{on\_table}(x)$ means that the block x is sitting directly on the table.
- space means that there is an empty space on the table that can fit a block.

There are four types of possible state transitions in Blocks World:

1. The hand, if not holding any block, can pick up a block that is on the table and has nothing on top of it, leaving a block-sized space on the table.

2. The hand, if not holding any block, can pick up the top block of a stack of blocks, exposing the next block down.
3. The hand, if holding a block, can place it in an empty space on the table.
4. The hand, if holding a block, can place it on top of an existing stack of blocks.

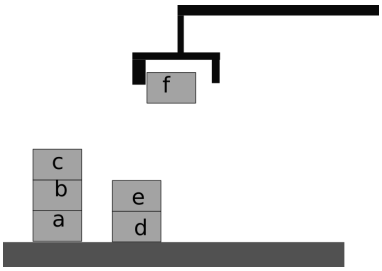
We can formalize transition (a) as the following axiom in linear logic: [4]

$$!\,(\mathsf{empty} \otimes \mathsf{clear}(x) \otimes \mathsf{on\_table}(x) \multimap \mathsf{holds}(x) \otimes \mathsf{space})$$

Note that we use the exponential ! here to indicate that this is unrestricted, as we may take this action any number of times.

Task 9 (3 pts)**.** Write linear logic axioms describing transitions (2)-(4) from above. Please provide proposition in linear logic that describe these axioms. [5]

Task 10 (2 pts)**.** Consider the following Blocks World scenario:



Write a proposition in linear logic which expresses this configuration, assuming that the table can fit **three** blocks total directly on it.

For comparison, the configuration with no blocks would be represented as $\mathsf{empty} \otimes \mathsf{space} \otimes \mathsf{space} \otimes \mathsf{space}$.

Hint: It may help to think about what invariants hold for all configurations, and then to think about what invariants hold for all configurations with the same set of blocks.

Task 11 (2 pts)**.** Write a proposition in linear logic expressing that the blocks are sorted alphabetically in a single stack, with a at the bottom of the stack.

Task 12 (2 pts)**.** Do you think the proposition from the previous task is provable from the axioms and the initial state given in task 5 for the scenario pictured above? If so, briefly justify why. If not, briefly justify why not. **You do not need to write a proof.**

Task 13 (3 pts)**.** Suppose we are in a state with two towers of blocks and at least one empty space — something of the form

$$\mathsf{space} \otimes \mathsf{clear}(a) \otimes \mathsf{on}(a, b) \otimes \mathsf{clear}(c) \otimes \mathsf{on}(c, d) \otimes \ldots$$

Write a procedure to swap the top blocks a and b of each tower by listing the sequence of axioms that need to be applied and which block is being moved at each step. You will be specifying this as list of axioms and blocks. For example if we wanted to indicate that the hand picks up block c and then puts it on an empty space we would provide the list $[(1, c), (3, c)]$.

[4] Technically, this is an axiom *schema*, and to get an axiom, you need to instantiate all free variables with blocks.

[5] Make sure that your axioms apply generically to all blocks by using a variable rather than using a specific block

Task 14 (2 pts)**.** How general is your procedure? Describe two different changes to the initial state that would require you to change your procedure, and briefly explain what goes wrong.

Appendix

Focusing

Positive and Negative

$$C^- \coloneqq A^- \wedge^- B^- | A^+ \supset B^- | \top^- | P^- | \uparrow C^+$$

$$C^+ \coloneqq A^+ \wedge^+ B^+ | A^+ \vee B^+ | \bot | \top^+ | P^+ | \downarrow C^-$$

Inversion

$$\frac{\Gamma^-; \Omega^+ \to_R A^- \quad \Gamma^-; \Omega^+ \to_R B^-}{\Gamma^-; \Omega^+ \to_R A^- \wedge^- B^-} \wedge R \qquad \frac{}{\Gamma^-; \Omega^+ \to_R \top^-} \top R \qquad \frac{\Gamma^-; \Omega^+, A^+ \to_R B^-}{\Gamma^-; \Omega^+ \to_R A^+ \supset B^-} \supset R$$

$$\frac{\Gamma^-; \Omega^+ \to_L \downarrow P^-}{\Gamma^-; \Omega^+ \to_R P^-} PR \qquad \frac{\Gamma^-; \Omega^+ \to_L A^+}{\Gamma^-; \Omega^+ \to_R \uparrow A^+} \uparrow R$$

$$\frac{\Gamma^-; \Omega^+, A^+, B^+ \to_L C^+}{\Gamma^-; \Omega^+, A^+ \wedge^+ B^+ \to_L C^+} \wedge L \qquad \frac{\Gamma^-; \Omega^+ \to_L C^+}{\Gamma^-; \Omega^+, \top^+ \to_L C^+} \top L$$

$$\frac{\Gamma^-; \Omega^+, A^+ \to_L C^+ \quad \Gamma^-; \Omega^+, B^+ \to_L C^+}{\Gamma^-; \Omega^+, A^+ \vee B^+ \to_L C^+} \vee L \qquad \frac{}{\Gamma^-; \Omega^+, \bot \to_L C^+} \bot L$$

$$\frac{\Gamma^-, \uparrow P^+; \Omega \to_L C^+}{\Gamma^-; \Omega^+, P^+ \to_L C^+} PL \qquad \frac{\Gamma^-, A^-; \Omega^+ \to_L C^+}{\Gamma^-; \Omega^+, \downarrow A^- \to_L C} \downarrow L$$

Stablization and Focus

$$\frac{\Gamma^- \to C^+}{\Gamma^-; \cdot \to_L C^+} \text{ stable} \qquad \frac{\Gamma^- \to [A^+]}{\Gamma^- \to A^+} \text{ focus}_R \qquad \frac{\Gamma^-, A^-; [A^-] \to C^+}{\Gamma^-, A^- \to C^+} \text{ focus}_L$$

Chaining

$$\frac{\Gamma^- \to [A^+] \quad \Gamma \to [B^+]}{\Gamma^- \to [A^+ \wedge^+ B^+]} \wedge R \qquad \frac{\Gamma \to [A^+]}{\Gamma \to [A^+ \vee B^+]} \vee R_1 \qquad \frac{\Gamma \to [B^+]}{\Gamma \to [A^+ \vee B^+]} \vee R_2 \qquad \frac{}{\Gamma \to [\top^+]} \top R$$

$$\frac{}{\Gamma, \uparrow P^+ \to [P^+]} PR \qquad \frac{\Gamma; \cdot \to_R A^-}{\Gamma \to [\downarrow A^-]} \downarrow R \qquad \frac{\Gamma \to [A^+] \quad \Gamma; [B^-] \to C^+}{\Gamma; [A^+ \supset B^-] \to C^+} \supset L$$

$$\frac{\Gamma; [A^-] \to C^+}{\Gamma; [A^- \wedge^- B^-] \to C^+} \wedge L_1 \qquad \frac{\Gamma; [B^-] \to C^+}{\Gamma; [A^- \wedge^- B^-] \to C^+} \wedge L_2 \qquad \frac{\Gamma; A^+ \to_L C^+}{\Gamma; [\uparrow A^+] \to C^+} \uparrow L$$

$$\frac{}{\Gamma; [P^-] \to \downarrow P^-} PL$$

Linear Sequent Calculus

In both presentations of linear logic that we use in this course, contexts Δ should be taken as unordered lists; therefore, the principle of *exchange* is automatic. Weakening and contraction are *not* included in linear logic.

$$\frac{}{P\ true \Vdash P\ true}\ init$$

Multiplicative Conjunction

$$\frac{\Delta \Vdash A\ true \quad \Delta' \Vdash B\ true}{\Delta, \Delta' \Vdash A \otimes B\ true}\ \otimes R \qquad \frac{\Delta', A\ res, B\ res \Vdash C\ true}{\Delta, A \otimes B\ res \Vdash C\ true}\ \otimes L$$

$$\frac{}{\cdot \Vdash \mathbf{1}\ true}\ \mathbf{1}R \qquad \frac{\Delta \Vdash C\ true}{\Delta, \mathbf{1}\ res \Vdash C\ true}\ \mathbf{1}L$$

Additive Conjunction

$$\frac{\Delta \Vdash A\ true \quad \Delta \Vdash B\ true}{\Delta \Vdash A \mathbin{\&} B\ true}\ \&R \qquad \frac{\Delta, A\ res \Vdash C\ true}{\Delta, A \mathbin{\&} B\ res \Vdash C\ true}\ \&L_1 \qquad \frac{\Delta, B\ res \Vdash C\ true}{\Delta, A \mathbin{\&} B\ res \Vdash C\ true}\ \&L_2$$

$$\frac{}{\Delta \Vdash \top\ true}\ \top R$$

There is no elimination rule for the unit $\top$.

Additive Disjunction

$$\frac{\Delta \Vdash A\ true}{\Delta \Vdash A \oplus B\ true}\ \oplus R_1 \qquad \frac{\Delta \Vdash B\ true}{\Delta \Vdash A \oplus B\ true}\ \oplus R_2 \qquad \frac{\Delta, A\ res \Vdash C\ true \quad \Delta, B\ res \Vdash C\ true}{\Delta, A \oplus B\ res \Vdash C\ true}\ \oplus L$$

$$\frac{}{\Delta, \mathbf{0}\ res \Vdash C\ true}\ \mathbf{0}L$$

There is no introduction rule for the unit $\mathbf{0}$.

Implication

$$\frac{\Delta, A\ res \Vdash B\ true}{\Delta \Vdash A \multimap B\ true}\ \multimap R \qquad \frac{\Delta \Vdash A\ true \quad \Delta', B\ res \Vdash C\ true}{\Delta, \Delta', A \multimap B\ res \Vdash C\ true}\ \multimap \mathsf{E}$$

Linear Natural Deduction

$$\frac{}{u : A \; true \Vdash A \; true} \; hyp$$

Multiplicative Conjunction

$$\frac{\Delta \Vdash A \; true \quad \Delta' \Vdash B \; true}{\Delta, \Delta' \Vdash A \otimes B \; true} \otimes\mathsf{I} \qquad \frac{\Delta \Vdash A \otimes B \; true \quad \Delta', u : A \; true, v : B \; true \Vdash C \; true}{\Delta, \Delta' \Vdash C \; true} \otimes\mathsf{E}^{u,v}$$

$$\frac{}{\cdot \Vdash \mathbf{1} \; true} \; \mathbf{1}\mathsf{I} \qquad \frac{\Delta \Vdash \mathbf{1} \; true \quad \Delta' \Vdash C \; true}{\Delta, \Delta' \Vdash C \; true} \; \mathbf{1}\mathsf{E}$$

Additive Conjunction

$$\frac{\Delta \Vdash A \; true \quad \Delta \Vdash B \; true}{\Delta \Vdash A \mathbin{\&} B \; true} \mathbin{\&}\mathsf{I} \qquad \frac{\Delta \Vdash A \mathbin{\&} B \; true}{\Delta \Vdash A \; true} \mathbin{\&}\mathsf{E}_1 \qquad \frac{\Delta \Vdash A \mathbin{\&} B \; true}{\Delta \Vdash B \; true} \mathbin{\&}\mathsf{E}_2$$

$$\frac{}{\Delta \Vdash \top \; true} \; \top\mathsf{I}$$

There is no elimination rule for the unit $\top$.

Additive Disjunction

$$\frac{\Delta \Vdash A \; true}{\Delta \Vdash A \oplus B \; true} \oplus\mathsf{I}_1 \qquad \frac{\Delta \Vdash B \; true}{\Delta \Vdash A \oplus B \; true} \oplus\mathsf{I}_2$$

$$\frac{\Delta \Vdash A \oplus B \; true \quad \Delta', u : A \; true \Vdash C \; true \quad \Delta', v : B \; true \Vdash C \; true}{\Delta, \Delta' \Vdash C \; true} \oplus\mathsf{E}^{u,v}$$

$$\frac{\Delta \Vdash \mathbf{0} \; true}{\Delta, \Delta' \Vdash C \; true} \; \mathbf{0}\mathsf{E}$$

There is no introduction rule for the unit $\mathbf{0}$.

Implication

$$\frac{\Delta, u : A \; true \Vdash B \; true}{\Delta \Vdash A \multimap B \; true} \multimap\mathsf{I}^u \qquad \frac{\Delta \Vdash A \multimap B \; true \quad \Delta' \Vdash A \; true}{\Delta, \Delta' \Vdash B \; true} \multimap\mathsf{E}$$