Constructive Logic (15-317), Fall 2016
Assignment 10: Learning Linear Logic

Contact: Giselle Reis (`giselle@cmu.edu`)

Due Thursday, December 8, 2016, 1:30pm

This assignment is due at the beginning of class on the above date and must be submitted electronically via autolab. Submit your homework as a **tar** archive containing the files `hw10.pdf` and `poly.pl`.

After submitting via autolab, please check the submission's contents to ensure it contains what you expect. No points can be given to a submission that isn't there.

1 A logic of conscious resources

Classical logic was about truth, intuitionistic logic was about proofs, now linear logic is about *resources*!

Until now, structural rules could be applied to any formula on the left side of a sequent. By structural rule I mean contraction and weakening, respectively:

$$\frac{\Gamma, A, A \longrightarrow C}{\Gamma, A \longrightarrow C}\ C \qquad \frac{\Gamma \longrightarrow C}{\Gamma, A \longrightarrow C}\ W$$

Because of that, the two following rules for conjunction on the right are basically the same[1]:

$$\frac{\Gamma \longrightarrow A \quad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B}\ \wedge_a \qquad \frac{\Gamma_1 \longrightarrow A \quad \Gamma_2 \longrightarrow B}{\Gamma_1, \Gamma_2 \longrightarrow A \wedge B}\ \wedge_m$$

But if we disallow structural rules on the formulas in the context, then these rules can no longer be used interchangeably. The way to distinguish them in linear logic is to have two connectives for conjunction, namely, $\&$ (called *with*) and $\otimes$ (called *tensor*). Since we now have two conjunctions, we will also have two "trues" that will be their neutral elements: $\mathbf{1}$ is the neutral element for $\otimes$ and $\top$ for $\&$.

[1]Think about how you can get one from the other by using structural rules.

Disjunction can also be split into two connectives, but for our formulation of intuitionistic linear logic we will need only one, namely, $\oplus$ (called *plus*)[2]. Its neutral element is $\mathbf{0}$.

The intuitionistic implication $A \supset B$ represented the fact that we can transform a proof of A into a proof of B. The linear implication $A \multimap B$ denotes that we can transform a resource A into a resource B. And once this is done, A is no longer available.

Finally, we might want to still have resources that can be used an unbounded number of times. These are indicated by the unary operator ! (called *bang*).

The calculus in Figure 1 is using a dyadic notation, meaning that it splits the context into two parts: Δ containing linear resources (also the working context, where the rules are applied) and Γ containing unbound resources. It admits cut, but we will not go into the details of the cut admissibility proof for now.

Task 1 (10)**.** For each of the following sequents, prove that they hold by constructing a cut-free sequent calculus proof (using the calculus above) or explain why they do not hold.

- $\cdot; A \multimap B \multimap C \Vdash (A \otimes B) \multimap C$
- $\cdot; A \& B \Vdash A \oplus B$
- $\cdot; A \otimes B \Vdash A \oplus B$
- $\cdot; A \& B \Vdash A \otimes B$
- $\cdot; !(A \& B) \Vdash A \otimes B$

Task 2 (5)**.** Take the sequents above which you were not able to prove and transform their conclusion C into $C \otimes \top$. Can you prove them now? What can you conclude about the role of $\otimes\top$? Can you think of an interesting way for using this? (Be creative :)

Task 3 (5)**.** It is possible to have a linear logic calculus where the left side of the sequent is not split into two (multi-)sets. Explain how the calculus would change and show the rules that would be different.

2 Practice some more Prolog

In this question we will study ways of computing the derivative of polynomials in one variable with Prolog. Assume that polynomial expressions are represented as data structures of type `poly` built in an arbitrary shape from these constructors:

[2] If you are curious, the other one is denoted by $\parr$ and called *par*.

```
plus(S,T)  represents the sum of S and T
times(S,T) represents the product of S and T
var        indicates the variable (only one variable occurs so no need for a name)
num(N)     represents the number literal N (say as an integer)
```

In this problem you will define a predicate `diff/2` to compute the derivative of a polynomial expression represented in this way. For example, the following query is expected to succeed:

```
?- diff(plus(var,num(5)), plus(num(1), num(0))).
```

Modes in Prolog describe the intended ways of using a predicate. Mode `+poly` refers to an input argument of type `poly` that needs to be provided. Mode `-poly` refers to an output argument of type `poly` that will be computed by the predicate when all inputs are provided.

Task 4 (6)**.** Write a Prolog program `diff(+poly,-poly)` that takes the polynomial as an input in the first argument and produces its derivative as an output in the second argument.

Task 5 (2)**.** With mode `diff(+poly,-poly)`, the predicate from Task 4 computes a derivative. Is there a mode with which the predicate from Task 4 computes antiderivatives (also known as indefinite integrals)? Justify.

Task 6 (2)**.** Is there a mode with which the predicate from Task 4 can be used to check whether a given polynomial expression is the integral of another given polynomial expression? Justify.

3 This is (almost) the end

Congratulations on reaching this far on the Constructive Logic course! Hopefully you are now more insightful of what a proof really means and how we can use logic to reason about the world. You might also have noticed that the notion of "reasoning" is quite subjective.

Now it is time to prepare for your final sprint. Here's checklist of topics you might want to review. Make sure you understand correctly how all those things are related, and take a step back to appreciate the big picture of the course.

- Natural deduction: first representation of a logic, proofs can be viewed as proof terms, nice correspondence to functional programming. Rules must be harmonious, meaning that they are locally sound and complete.

- Verification: non-redundant version of natural deduction – disallows silly steps that eliminate and introduce the same connective over and over again. Sound and complete w.r.t. natural deduction.

- Sequent calculus (unrestricted): Can be obtained by a direct translation from the verification calculus. Sound and complete w.r.t. verification. Cut and init are admissible.

- Connections between the concepts of harmony, local soundness and completeness, and cut and init admissibility.

- Sequent calculus (restricted): "Slim" version of the previous sequent calculus.

- G4ip: decision procedure for intuitionistic logic. Cases on the left side of implications on the left side of the sequent.

- Quantifiers: all the calculi above can be formulated for first-order logic as well, simply by adding the quantifier rules. The term being replaced for the quantified variable has different restrictions depending on the quantifier and the rule (left or right, introduction or elimination).

- Backward logic programming (Prolog): a fragment of intuitionistic logic is used as a programming language.

- Operational semantics: understanding how backward logic programming evaluates the clauses to try to prove a goal.

- Unification: a key concept in logic programming. In backward logic programming, the system tries to unify the query with the head of clauses. The unification calculus can be used to do that.

- Forward logic programming: "executing" a logic program in a pro-active way and generating all possible facts instead of querying it for one thing might be a good choice in terms of complexity. A forward logic program saturates the database which you can then check for facts.

- Linear logic: formulas are interpreted as resources – they are consumed when used and, unless stated otherwise, cannot be used more than once. New connectives show up here.

Good luck on finals!

$$\frac{}{\Gamma; P \Vdash P}\ \text{id} \qquad \frac{\Gamma, A; \Delta, A \Vdash C}{\Gamma, A; \Delta \Vdash C}\ \text{copy}$$

$$\frac{\Gamma; \Delta_1 \Vdash A \quad \Gamma; \Delta_2 \Vdash B}{\Gamma; \Delta_1, \Delta_2 \Vdash A \otimes B}\ {\otimes}R \qquad \frac{\Gamma; \Delta, A, B \Vdash C}{\Gamma; \Delta, A \otimes B \Vdash C}\ {\otimes}L$$

$$\frac{}{\Gamma; \cdot \Vdash \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma; \Delta \Vdash C}{\Gamma; \Delta, \mathbf{1} \Vdash C}\ \mathbf{1}L$$

$$\frac{\Gamma; \Delta, A \Vdash B}{\Gamma; \Delta \Vdash A \multimap B}\ {\multimap}\, R \qquad \frac{\Gamma; \Delta_1 \Vdash A \quad \Gamma; \Delta_2, B \Vdash C}{\Gamma; \Delta_1, \Delta_2, A \multimap B \Vdash C}\ {\multimap}\, L$$

$$\frac{\Gamma; \Delta \Vdash A \quad \Gamma; \Delta \Vdash B}{\Gamma; \Delta \Vdash A \& B}\ \& R \qquad \frac{\Gamma; \Delta, A_i \Vdash C}{\Gamma; \Delta, A_1 \& A_2 \Vdash C}\ \& L_i$$

$$\frac{}{\Gamma; \Delta \Vdash \top}\ \top R \qquad \text{no } \top L \text{ rule}$$

$$\frac{\Gamma; \Delta \Vdash A_i}{\Gamma; \Delta \Vdash A_1 \oplus A_2}\ {\oplus}R_i \qquad \frac{\Gamma; \Delta, A \Vdash C \quad \Gamma; \Delta, B \Vdash C}{\Gamma; \Delta, A \oplus B \Vdash C}\ {\oplus}L$$

$$\text{no } \mathbf{0}R \text{ rule} \qquad \frac{}{\Gamma; \Delta, \mathbf{0} \Vdash C}\ \mathbf{0}L$$

$$\frac{\Gamma; \cdot \Vdash A}{\Gamma; \cdot \Vdash !A}\ !R \qquad \frac{\Gamma, A; \Delta \Vdash C}{\Gamma; \Delta, !A \Vdash C}\ !L$$

Figure 1: Dyadic calculus for intuitionistic linear logic