Click and coLLecT An Interactive Linear Logic Prover

Etienne Callies, Olivier Laurent

From the proceedings of Trends in Linear Logic and Applications 2021 (TLLA 2021, eds E. Pimentel, Ch. Retoré)
https://hal.archives-ouvertes.fr/TLLA2021

Click and coLLecT
An Interactive Linear Logic Prover

Etienne Callies*
etienne.callies@gmail.com

Olivier Laurent*
olivier.laurent@ens-lyon.fr

Abstract

We describe the 'Click and coLLecT' system, an interactive tool for building linear logic proofs online.

1 Introduction

The lack of pedagogical material and tools for learning linear logic (LL) [Gir87] is mostly recognized by the community. Directly inspired by Logitext[1], we develop "C1ick ⅋ c⊗LLec⊥", an interactive tool for building linear logic proofs in the sequent calculus. It is available online:

`https://click-and-collect.linear-logic.org/`

It can be used on a computer or on smartphones (even if interaction is then not yet optimal).

The propositional part of Logitext relies on a set of reversible rules, such that choosing a formula in a sequent uniquely determines the rule to apply. This is not possible with linear logic since for example the introduction rules for $\oplus$ cannot be chosen to be reversible.

In the development of the tool, we address two main audiences:

- *beginners* who want to learn the rules of the sequent calculus for linear logic. Doing so on paper is often tedious since there are many rules (more than 10 rules for full propositional LL) and they can be easily mixed up (consider for example $\otimes$ and &). The idea here is to allow users to practice the rules even without knowing them precisely since the tool is in charge of applying rules on the request by the user to destruct a connective. A beginner will certainly

*Univ Lyon, EnsL, UCBL, CNRS, LIP, F-69342, LYON Cedex 07, France.
This work was supported by the IRN Linear Logic, and by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

[1] `http://logitext.mit.edu/`

$$\frac{}{\vdash A, A^{\perp}}\ ax \qquad \frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ \otimes \qquad \frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, A \parr B, \Delta}\ \parr$$

$$\frac{\vdash \Gamma, A, \Delta \quad \vdash \Gamma, B, \Delta}{\vdash \Gamma, A \,\&\, B, \Delta}\ \& \qquad \frac{\vdash \Gamma, A, \Delta}{\vdash \Gamma, A \oplus B, \Delta}\ \oplus_1 \qquad \frac{\vdash \Gamma, B, \Delta}{\vdash \Gamma, A \oplus B, \Delta}\ \oplus_2$$

$$\frac{}{\vdash 1}\ 1 \qquad \frac{\vdash \Gamma, \Delta}{\vdash \Gamma, \perp, \Delta}\ \perp \qquad \frac{}{\vdash \Gamma, \top, \Delta}\ \top$$

$$\frac{\vdash ?\Gamma, A, ?\Delta}{\vdash ?\Gamma, !A, ?\Delta}\ ! \qquad \frac{\vdash \Gamma, \Delta}{\vdash \Gamma, ?A, \Delta}\ ?w \qquad \frac{\vdash \Gamma, A, \Delta}{\vdash \Gamma, ?A, \Delta}\ ?d \qquad \frac{\vdash \Gamma, ?A, ?A, \Delta}{\vdash \Gamma, ?A, \Delta}\ ?c$$

Figure 1: Basic Rules

have surprises when seeing that the rule executed by the system is not exactly what he might have expected, but this way he will be able to learn what lies behind each connective.

- *advanced users* who know the LL system perfectly well, but are interested in building proofs quicker than on paper, or for the purpose of using them in LaTeX or Coq while relying on a more comfortable input syntax.

As opposed to Logitext which performs on-the-fly interactions with Coq to validate rule applications, we just rely on the tool itself, assuming there is no bug in the rule construction. This is lighter since there is no interaction with a third-party tool, but clearly not reasonable as a certificate for the validity of the induced proofs. To go further, either the user has to check each rule application by himself, or he can rely on the proposed Coq export which allows to validate proofs through a minimal (deep) encoding of LL in Coq (see Section 5.1).

After a presentation in Section 2 of the logical system on which C1ick ⅋ c⊗LLec⊥ relies, we give the main lines of the interaction with C1ick ⅋ c⊗LLec⊥ in Section 3 (with a few screenshots). Section 4 describes features designed to help proof construction, and Section 5 gives the list of provided export functionalities. A few words about implementation are given in Section 6.

2 Logic Specification

We consider classical propositional linear logic, and for simplification reasons, both on interface side and code side, we work only with one-sided sequents. The

rules we consider are presented on Figure 1. It is important to notice that the induced system is *not* complete for LL provability. For example, $\vdash A \otimes B, B^\perp \parr A^\perp$ is not provable. Some exchange rule is necessary.

One possibility is to add the usual exchange rule (with σ any permutation):

$$\frac{\vdash \Gamma}{\vdash \Gamma\sigma}\ ex_\sigma$$

But it is also possible to put it below ($\otimes$) rules only, thus leading to the following generalized ($\otimes_\sigma$) rule, which makes $\vdash A \otimes B, B^\perp \parr A^\perp$ provable:

$$\frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash (\Gamma, A \otimes B, \Delta)\sigma}\ \otimes_\sigma \qquad\qquad \dfrac{\dfrac{\dfrac{}{\vdash A^\perp, A}\ ax \qquad \dfrac{}{\vdash B, B^\perp}\ ax}{\vdash A \otimes B, B^\perp, A^\perp}\ \otimes_{(231)}}{\vdash A \otimes B, B^\perp \parr A^\perp}\ \parr$$

Lemma 1 (Completeness)**.** *A sequent $\vdash \Gamma$ is provable in* LL *if and only if it is provable using the rules of Figure 1 with the ($\otimes$) rule replaced by the ($\otimes_\sigma$) rule.*

Proof. Let us consider LL provability given by the rules of Figure 1 together with the (ex_σ) rule. One can check that any rule, but the ($\otimes$) one, can be commuted down with the (ex_σ) rule. This is enough to conclude. □

Another property of the system described in Figure 1 is that, given a formula A in a sequent $\vdash \Gamma, A, \Delta$, there is at most one possible rule having this formula as main conclusion ($!A$ being considered as the unique main conclusion of the ($!$) rule), except for:

- the (ax) rule which can be applied as soon as a sequent has the shape $\vdash A, A^\perp$;
- the ($\oplus_i$) rules which can be both applied to get $A \oplus B$;
- the ? formulas for which three rules are possible to introduce $?A$: ($?w$), ($?d$) and ($?c$).

3 User Interface

Based on the properties presented in Section 2, the following features provided by the C1ick $\parr$ c$\otimes$LLec$\perp$ system are complete for building LL proofs:

- allowing to drag and drop formulas in a sequent to change the order (this is in fact necessary only in the conclusion of ($\otimes$) rules);
- clicking on the $\vdash$ symbol to apply the (ax) rule;
- clicking on one of the two immediate sub-formulas of a $\oplus$ formula to apply the corresponding ($\oplus_i$) rule;

- clicking on a ? to apply the (?w) rule, double-clicking to apply the (?c) rule, or clicking on its immediate sub-formula to apply a (?d) rule;
- otherwise, clicking on a formula to apply the unique rule associated to its main connective.

Exchanges are thus not displayed as explicit rules: just in-place permutation of formulas through drag and drop. However, internally we have an explicit representation of these rules on which we rely for exports in particular. To make the user's life easier, exchange rules are accepted in any sequents, not just in those below $\otimes$ rules.

3.1 Input Sequent

The tool input was designed to be as simple as possible: it is composed of a text field and a PROVE button to type and submit a sequent, according to a syntax that accepts both ASCII (*, |, &, +) and UTF-8 ($\otimes$, $\parr$, &, $\oplus$) symbols:

It is possible to input formulas, comma-separated lists of formulas (understood as one-sided sequents), or two-sided sequents relying on symbol $\vdash$ (or |-). Two sided-sequents are internally turned into one-sided ones (with negation restricted to atoms) before proof starts.

3.2 Interactive Prover

By clicking on PROVE, we obtain a one-sided sequent we can interact with using the mouse. The interactive construction of the proofs goes bottom-up starting with the user input as conclusion sequent for the proof to be built:

$$\vdash A \otimes A^{\perp}, A \parr A^{\perp}$$

Clicking on a connective ($\parr$ for example here) applies the associated rule:

$$\dfrac{\vdash A \otimes A^{\perp}, A, A^{\perp}}{\vdash A \otimes A^{\perp}, A \parr A^{\perp}}\,\parr$$

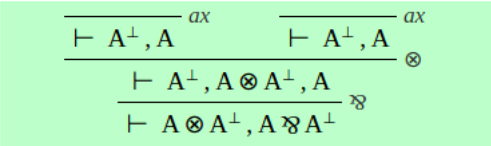
Before applying the ($\otimes$) rule, we manually perform an exchange rule, using drag and drop with the mouse:

$$\dfrac{\vdash A^{\perp}, A \otimes A^{\perp}, A}{\vdash A \otimes A^{\perp}, A \parr A^{\perp}}\,\parr$$

Then we can apply the ($\otimes$) rule:

$$\dfrac{\dfrac{\vdash A^{\perp}, A \qquad \vdash A^{\perp}, A}{\vdash A^{\perp}, A \otimes A^{\perp}, A}\,\otimes}{\vdash A \otimes A^{\perp}, A \parr A^{\perp}}\,\parr$$

Finally we can click on literals or on the $\vdash$ symbols to apply (ax) rules. When the proof is completed, a green background is displayed:



3.3 Tutorial and Exercises

A tutorial is provided to learn interactions step by step. This includes in particular the case of connectives $\oplus$ and ? for which there is more than one possible rule to apply. It is concluded with a short list of sequents to prove as final exercises. A larger list of standard LL principles can also be used to practice.

4 Proof Features

Some additional features are provided as support for proof construction.

4.1 On-The-Fly Provability Checks

The provability of propositional LL is undecidable [LMSS92]. Nevertheless it is possible to define some decidable checks which guarantee the non-provability of a given sequent. In order to help the user understanding he is going in a wrong direction, we have implemented a few such tests:

- Mapping $\otimes$ and & to $\wedge$, $\parr$ and $\oplus$ to $\vee$, etc. and erasing the exponential connectives turns any provable LL sequent into a sequent provable in classical logic. As a consequence, the non-provability of the obtained classical sequent in classical logic entails the non-provability of the starting one in LL. We have implemented a very simple provability checker for classical logic. More efficient SAT solvers could be used in the future if more efficiency is required.

- Phase models [Gir87] are a complete semantics for LL provability. A few simple ones based on the monoid $(\mathbb{Z},+,0)$ are implemented. They mostly challenge MLL formulas.

- To be more accurate on MALL formulas, we also have a check that additive slices are well balanced: appropriate equilibrium between the number of X and of $X^\perp$ for each atom X.

As soon as one of these checks detects a non-provable sequent, the $\vdash$ symbol is turned into a red one to inform the user that his current trial is a dead-end. He can undo part of the proof construction by clicking on a previous rule-name to

backtrack, or on a connective lower in the proof to apply a new rule instead of an existing one.

4.2 Auto-Reverse Mode

For more advanced users, it is often useful to ignore reversible rules which are known not to impact provability (so that they can be applied systematically during proof construction). As an option, an *auto-reverse* mode is provided which applies a bunch of reversible rules automatically at each proof step. This includes ($\top$), ($\bot$), ($\parr$) and (&) rules. But also some other rules under appropriate conditions: (!) rule in ? context, (1) rule with an empty context, ($\otimes$) rule with an empty context, or (*ax*) rule for pairs of dual atomic formulas.

4.3 Automated Proving

As already mentioned in Section 4.1, LL provability is undecidable. It is nevertheless semi-decidable and propositional MALL provability is decidable. We can thus make a call to an (external) automated prover to try closing a proof (by double-clicking on the $\vdash$ symbol). We use APLL which provides a complete algorithm (based on backward focused proof search) for MALL and is terminating but incomplete for full LL (through bounded proof search for exponentials). It is important for us that APLL provides an explicit proof as output so that we can display it to the user. A number of simplifications (exchange eliminations along Lemma 1, weakening simplifications, detection of redundancies) of the generated proof are implemented to display easier-to-read proofs (otherwise the structure of focused proofs makes them often unnatural for a human reader).

4.4 Cut Rule

It is possible to consider proofs with cuts by activating a dedicated option. It makes commas becoming active: a click on a comma requests a cut-formula A through a pop-up window and applies a cut rule of the following shape (where the comma between Γ and Δ is the clicked one):

$$\frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ cut$$

4.5 Notations

When dealing with big formulas or when some of them have a specific meaning (like $!(X \multimap X) \multimap !(X \multimap X)$ for Church natural numbers), it is natural to introduce short-names as notations. If such a notation is declared, it becomes possible to unfold it by clicking on the alias (N below):

$$\frac{\vdash \Gamma, ?(X \otimes X^{\perp}) \parr !(X^{\perp} \parr X), \Delta}{\vdash \Gamma, N, \Delta}\ def \qquad N ::= !(X \multimap X) \multimap !(X \multimap X)$$

By accepting also cyclic notations, this mechanism allows to deal with recursive definitions of formulas for free. The typical example being $o ::= !o \multimap o$ used for the encoding of the untyped λ-calculus in LL. Even if recursive formulas are then introduced, proofs remain finite so far.

5 Exports

5.1 Coq Export

The tool is supposed to apply only valid rules, so that when the green background appears, the displayed proof should be a valid LL proof. However we do not necessarily want to rely on the tool concerning validity of the output, especially when automated proving features are used. In practice, none of the internal features has been formally verified. Nevertheless, it is possible to get a strong certificate through on-demand generation of a Coq file. Users can compile this file to certify their proof is valid, and also reuse the Coq code as a lemma for a larger development. It relies on the NanoYalla package which is extracted from the Yalla library and compatible with it (in particular it is possible to rely on the cut-elimination proof provided by Yalla).

5.2 Proof Sharing

The internal representation of a proof can be encoded into the URL. In this way it can be reopened later or shared with other users.

5.3 ASCII and UTF-8 Exports

Simple textual exports to ASCII and UTF-8 with appropriate vertical alignment of sub-proofs (better displayed with fixed-width fonts) give a direct way to re-use the constructed proofs.

5.4 From LaTeX Export to PDF and PNG

An export of the proof in LaTeX is also possible, in order to be inserted in an article. It relies on the cmll and ebproof packages. Since LaTeX export is often aimed to be converted to PDF in the end, Click ⅋ c⊗LLec⊥ can do it for the users. It also proposes an export as an image in PNG format from this generated PDF version.

The textual exports (LaTeX, ASCII, UTF-8, etc.) are provided both with implicit and explicit exchange rules.

6 Implementation

The code is open source and can be found on GitHub:

`https://github.com/etiennecallies/click-and-collect`

It splits between a front-end part and a back-end part, communicating through an API relying on the JSON format, and which could be exploited with other linear logic tools. The front-end is designed as simple as possible. It is a single static HTML page, with javascript using jquery. The javascript only displays proofs and interacts with the back-end. The back-end is written in OCaml which is known to be well adapted to the definition of types for proofs and to proof manipulations.

7 Conclusion

The current version of C1ick $\parr$ c$\otimes$LLec$\bot$ allows users to build proofs in the standard one-sided sequent calculus for classical propositional LL. It comes with features for helping in proof construction (Section 4) and various possible exports of (possibly open) proofs (Section 5).

The GitHub wiki describes some extensions under consideration (with also some discussions through issues):

`https://github.com/etiennecallies/click-and-collect/wiki`

Let us mention here just a few key possible extensions for proof construction:

- consider an alternative version dedicated to intuitionistic linear logic. This requires more thought on the interface to deal with rules like the right introduction of $\otimes$ which requires to select a split point for the left context:

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \otimes R$$

- incorporating (first and second-order) quantifiers is also a natural direction to be considered.

In a slightly different direction, we are considering the introduction of a *proof transformation* mode which would freeze the proof under construction and offer tools for local or global transformations: small-step cut-elimination, axiom expansion, big-step cut-elimination, substitution, etc.

References

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[LMSS92] Patrick Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56(1–3):239–311, 1992.