École Normale Supérieure

Département d'Informatique

Rapport de Stage de L3

Recherche de preuve automatique en logique linéaire

Automated proof search in Linear Logic

Wu Jui-Hsuan

Sous la direction de

Olivier LAURENT[1] & Youakim BADR[2]

[1]LIP, CNRS - ENS Lyon

[2]LIRIS, INSA Lyon

11 Juin 2018 – 3 Août 2018

1 Introduction

Linear logic [4] was first introduced by Jean-Yves Girard as a refinement of classical and intuitionistic logic in 1987. It has become influential in fields such as programming languages and linguistics because of its resource sensitivity.

My internship aims at developing an automated prover for linear logic. My work contributes to the project **Logique linéaire pour l'Internet des Objets (LLipIdO)**, which aims at developping a tool using linear logic for describing the assemblage of components using consumable and limited resources. Once the components of a system described by formulas of linear logic, their assemblage can be obtained by a proof search.

During the internship, I implemented a prover for linear logic and intuitionistic linear logic using different proof search methods (backward, forward). Both of them are based on the notion of focusing introduced by J.-M. Andreoli [1].

The first prover starts from the goal sequent and uses backward reasoning (i.e. applying rules backwards), and the second one uses the focused inverse method [2], it starts from axioms and constructs new provable sequents from the database of provable sequents and add them into it. Asynchronous connectives (i.e. connectives with invertible rules) are decomposed in the active phase, and synchronous connectives (i.e. connectives with non-invertible rules) are decomposed in the focal phase. In each phase, a sequence of decompositions of connectives of the same nature is regarded as one step.

2 Linear Logic and Intuitionistic Linear Logic

2.1 Linear Logic

The language of formulas of propositional linear logic is defined inductively as follows:

$$A ::= X \mid A^{\perp} \mid A \otimes A \mid A \oplus A \mid A \,\&\, A \mid A \parr A \mid !A \mid ?A \mid \top \mid \perp \mid 1 \mid 0$$

where X is the set of atoms.

One can translate formulas of LL into formulas of classical logic:

$$\begin{array}{cc} X \longrightarrow X & \perp \longrightarrow \neg \\ \otimes, \& \longrightarrow \wedge & \parr, \oplus \longrightarrow \vee \\ 1, \top \longrightarrow \mathrm{T} & 0, \perp \longrightarrow \mathrm{F} \\ !, ? \longrightarrow \text{erased} & \end{array}$$

We can prove that if a formula is provable in LL, then its translation using the above correspondence is provable in classical logic. However, the converse is not true. Take the law of excluded middle as an example, for every atom X, $X \vee \neg X$ is provable in classical logic, one of its possible translations $X \parr X^{\perp}$ is provable in LL while the other one $X \oplus X^{\perp}$ is not. This also shows that in intuitionistic linear logic, the law of excluded middle does not exist.

We distinguish three kinds of connectives: Additives $(\oplus, \&, \top, 0)$, Multiplicatives $(\otimes, \parr, \perp, 1)$ and Exponentials $(!, ?)$.
In the following, we use sequent calculus instead of Hilbert-style systems to define linear logic. A sequent is written $\Gamma \vdash \Delta$ where Γ and Δ are lists of formulas and the order of formulas in a list does not count (We can still take the order into consideration by considering additional permutation rules). When we write $A_1, A_2, ..., A_n \vdash B_1, B_2, ..., B_m$, the commas on the left-hand side will be interpreted as $\otimes$ and the commas on the right-hand side as $\parr$. Intuitively, the sequent $\Gamma \vdash \Delta$ means that the conjunction of Γ entails the disjunction of Δ.

We now present the inference rules of linear logic:

Identity

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash \Delta, A \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ cut$$

Negation

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

Additives

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}\ \oplus L \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \oplus B}\ \oplus R_1 \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \oplus B}\ \oplus R_2 \qquad \frac{}{\Gamma, 0 \vdash \Delta}\ 0L$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}\ \&L_1 \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}\ \&L_2 \qquad \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \,\&\, B}\ \&R \qquad \frac{}{\Gamma \vdash \Delta, \top}\ \top R$$

Multiplicatives

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \otimes L \qquad \frac{\Gamma \vdash \Delta, A \quad \Gamma' \vdash \Delta', B}{\Gamma, \Gamma' \vdash \Delta, \Delta', A \otimes B}\ \otimes R \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ \parr L \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \parr B}\ \parr R$$

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

Exponentials

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\ wkL \qquad \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ contL \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ derL \qquad \frac{!\Gamma \vdash ?\Delta, B}{!\Gamma \vdash ?\Delta, !B}\ !R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, ?A}\ wkR \qquad \frac{\Gamma \vdash \Delta, ?A, ?A}{\Gamma \vdash \Delta, ?A}\ contR \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, ?A}\ derR \qquad \frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\ ?L$$

Fig. 1. The sequent calculus system for linear logic.

First note that the above sequent calculus system is quite symmetric: For every right introduction rule, there exists one left introduction rule for the dual connective that has exactly the same structure. Moreover, we have: $\Gamma, A \vdash \Delta$ is provable if and only if $\Gamma \vdash \Delta, A^\perp$ is provable. Consequently, $\Gamma \vdash \Delta$ is provable if and only if $\vdash \Delta, \Gamma^\perp$ where $(A_1, A_2, ..., A_n)^\perp = A_1^\perp, A_2^\perp, ..., A_n^\perp$.
We can thus come up with another sequent calculus system: one-sided sequent calculus. Its inference rules are similar to those of the above system.

We first define the negation by De Morgan's law: $(X)^\perp = X^\perp$, $(X^\perp)^\perp = X$, $(A \otimes B)^\perp = A^\perp \parr B^\perp$, $(A \parr B)^\perp = A^\perp \otimes B^\perp$, $(1)^\perp = \perp$, $(\perp)^\perp = 1$
$(A \oplus B)^\perp = A^\perp \,\&\, B^\perp$, $(A \,\&\, B)^\perp = A^\perp \oplus B^\perp$, $(0)^\perp = \top, (\top)^\perp = 0$, $(!A)^\perp = ?(A^\perp)$, $(?A)^\perp = !(A^\perp)$
Observe that $(\cdot)^\perp$ is an involution, i.e., $A^{\perp\perp} = A$ for all propositions A. $A^\perp$ is also called the linear negation of A.

We now present the inference rules of one-sided linear logic:

Identity

$$\frac{}{\vdash A, A^\perp}\ ax \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma', A^\perp}{\vdash \Gamma, \Gamma'}\ cut$$

Additives

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

Multiplicatives

$$\frac{\vdash \Gamma, A \quad \vdash \Gamma', B}{\vdash \Gamma, \Gamma', A \otimes B}\ \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ \parr \qquad \frac{}{\vdash 1}\ 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\ \bot$$

Exponentials

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ wk \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ cont \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ der \qquad \frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ !$$

Fig. 2. One-sided sequent calculus system for linear logic

We also define the linear implication $A \multimap B := A^{\perp} \parr B$.

2.2 Intuitionistic Linear Logic

In intuitionistic linear logic (ILL), all sequents should have exactly one formula on the right hand side, so we represent the inference rules using two-sided sequents.
The connectives $\parr$, $\bot$, ? are not available anymore, but the linear implication ($\multimap$) is. The language of formulas of intuitionistic linear logic can thus be defined inductively as follows:

$$A ::= X \mid A \otimes A \mid A \oplus A \mid A \,\&\, A \mid !A \mid \top \mid A \multimap A \mid 1 \mid 0$$

We now present the inference rules of intuitionistic linear logic:

Identity

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A \quad \Gamma', A \vdash C}{\Gamma, \Gamma' \vdash C}\ cut$$

Additives

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus R_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus R_2 \qquad \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus L \qquad \frac{}{\Gamma, 0 \vdash C}\ 0\mathrm{L}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \& R \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \,\&\, B \vdash C}\ \& L_1 \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \,\&\, B \vdash C}\ \& L_2 \qquad \frac{}{\Gamma \vdash \top}\ \top R$$

Multiplicatives

$$\frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\ \otimes R \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ \otimes L \qquad \frac{}{\vdash 1}\ 1R \qquad \frac{\Gamma \vdash C}{\Gamma, 1 \vdash C}\ 1L$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap R \qquad \frac{\Gamma \vdash A \quad \Gamma', B \vdash C}{\Gamma, \Gamma', A \multimap B \vdash C}\ \multimap L$$

Exponentials

$$\frac{\Gamma \vdash C}{\Gamma, !A \vdash C}\ wk_L \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}\ cont_L \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\ der_L \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\ !R$$

Fig. 3. The sequent calculus system for intuitionistic linear logic.

Observe that the four exponential rules give rise to the notion of unrestricted resources. The wk_L rule plays the role of the usual weakening rule. The contraction and dereliction rules show that we can create a fresh copy of A and the $!R$ rule shows that if we can produce a resource A with only unrestricted resources, then we can produce the corresponding unrestricted resource !A.

Another fact to know is that every provable sequent has at least a proof using no *cut* rule. This is true in both LL and ILL and a such proof is called a cut-free proof.

2.3 Focusing

In this section, we summarize briefly the main results from [1].
A literal is either an atom or the dual (negation) of an atom. In the following, we suppose that all formulas are in *negation normal form*, i.e. containing no occurrence of $\multimap$ and all negations have atomic scope.

In LL, we distinguish two kinds of connectives: *asynchronous connectives* and *synchronous connectives*. The asynchronous connectives are $\&$, $\parr$, $\top$ and $\bot$, which have invertible right introduction rules, and also ? while the synchronous connectives are their corresponding duals: $\oplus$, $\otimes$, 0, 1 and !. A formula is called asynchronous (resp. synchronous) if its main connective is asynchronous (resp. synchronous).

In [1], Andreoli introduced a triadic focused proof system LLF that consists of two kinds of sequents ($\vdash \Theta : \Gamma \Uparrow L$ and $\vdash \Theta : \Gamma \Downarrow F$) and four types of rules (Introduction, Identity, Reaction and Decision). This focused system helps to reduce inessential non-determinism in the proof search and this property will be illustrated in the section about the backward proof search.
In a sequent $\vdash \Theta : \Gamma \Uparrow L$, which corresponds to an active (asynchronous) phase, Θ and Γ are multisets of formulas and L is a list of formulas. In practice, Θ can be regarded as a set since there is no need to keep several copies of a formula in Θ thanks to the contraction rule. This is simply an encoding of the usual sequent $\vdash ?\Theta, \Gamma, L$.
In a sequent $\vdash \Theta : \Gamma \Downarrow F$, which corresponds to a focusing (synchronous) phase. Θ is a multiset of formulas and Γ is a multiset of synchronous formulas and literals and F is the formula under focus.
We use $\cdot$ to denote empty multisets or lists in the following.

Theorem 1 Proving $\vdash \Delta$ in LL is equivalent to proving $\vdash \cdot : \cdot \Uparrow \Delta$ in LLF.

Introduction

$$\frac{\vdash \Theta : \Gamma \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, \bot}\ \bot \qquad \frac{}{\vdash \Theta : \Gamma \Uparrow L, \top}\ \top \qquad \frac{}{\vdash \Theta : \cdot \Downarrow 1}\ 1 \qquad \frac{\vdash \Theta : \cdot \Uparrow F}{\vdash \Theta : \cdot \Downarrow\ !F}\ ! \qquad \frac{\vdash \Theta, F : \Gamma \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, ?F}\ ?$$

$$\frac{\vdash \Theta : \Gamma \Uparrow L, F \qquad \vdash \Theta : \Gamma \Uparrow L, G}{\vdash \Theta : \Gamma \Uparrow L, F \,\&\, G}\ \& \qquad \frac{\vdash \Theta : \Gamma \Uparrow L, F, G}{\vdash \Theta : \Gamma \Uparrow L, F \parr G}\ \parr \qquad \frac{\vdash \Theta : \Gamma \Downarrow F \qquad \vdash \Theta : \Gamma' \Downarrow G}{\vdash \Theta : \Gamma, \Gamma' \Downarrow F \otimes G}\ \otimes$$

$$\frac{\vdash \Theta : \Gamma \Downarrow F_1}{\vdash \Theta : \Gamma \Downarrow F_1 \oplus F_2}\ \oplus_1 \qquad \frac{\vdash \Theta : \Gamma \Downarrow F_2}{\vdash \Theta : \Gamma \Downarrow F_1 \oplus F_2}\ \oplus_2$$

Identity

$$\frac{}{\vdash \Theta : K^{\perp} \Downarrow K}\ I_1 \qquad \frac{}{\vdash \Theta, K^{\perp} : \cdot \Downarrow K}\ I_2 \qquad \text{where } K \text{ is a positive literal}$$

Reaction

$$\frac{\vdash \Theta : \Gamma, S \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, S}\ R\Uparrow$$

$$\frac{\vdash \Theta : \Gamma \Uparrow N}{\vdash \Theta : \Gamma \Downarrow N}\ R\Downarrow \qquad \text{where } N \text{ is neither synchronous nor a positive atom}$$

Decision

$$\frac{\vdash \Theta : \Gamma \Downarrow F}{\vdash \Theta : \Gamma, F \Uparrow} D_1 \qquad \frac{\vdash \Theta, F : \Gamma \Downarrow F}{\vdash \Theta, F : \Gamma \Uparrow} D_2 \quad \text{where } F \text{ is not a negative literal}$$

Fig. 4. The focused proof system LLF for linear logic.

In the $\top$ rule, Γ should contain only synchronous formulas and in the D_1 rule, F should be synchronous.

We can prove by induction on rules that Γ contains only synchronous formulas or literals for all the sequents $\vdash \Theta : \Gamma \Uparrow L$ or $\vdash \Theta : \Gamma \Downarrow F$ in a proof in LLF.

3 Backward Proof Search

A backward proof search starts from the goal sequent and matches the goal sequent with the conclusion of a rule and then tries to find a proof for each of the premises of this rule. For example, to prove $\vdash (A \otimes B) \oplus C, A^\perp \parr B^\perp$ in one-sided LL, we can first match the $\oplus_1$ rule, then we try to prove $\vdash A \otimes B, A^\perp \parr B^\perp$. We now apply the $\parr$ rule, and we need to prove $\vdash A \otimes B, A^\perp, B^\perp$. Eventually, we apply the $\otimes$ rule. Since $\vdash A, A^\perp$ and $\vdash B, B^\perp$ are axioms, $\vdash (A \otimes B) \oplus C, A^\perp \parr B^\perp$ is provable, and the proof can be extracted by the above reasoning. Some non-determinism can be found in the above reasoning:

- In the first step, we can also choose to apply the $\parr$ rule.
- When we apply the $\otimes$ rule, we need to decompose the "context" $A^\perp, B^\perp$ and the only decomposition that leads to a proof is the one described above.

Note that in MALL (fragment of LL without exponentials), for every sequent s, we have a finite number of rules whose conclusion matches s and for every rule, its conclusion always has a strictly bigger size than any of its premises. Additionally, in a non-deterministic rule such as the rule $\otimes$, we have a finite number of choices since contexts are always finite. As a result, the search space of a backward proof search in MALL is finite since every node has a finite number of children and the depth of this search tree is finite. In other words, any backward proof search in MALL will terminate in finite time.

However, it is not the case in LL. The size of goal sequents will increase when the contraction rule is applied and this has some connections with the undecidability of LL. In order to have an algorithm that always terminates, we should bound the search space in a more delicate way.

For LL, we use LLF for our proof search. We now prove $\vdash (A \otimes B) \oplus C, A^\perp \parr B^\perp$ in LL using the LLF system as an example.
By **theorem 1**, we need to prove $\vdash \cdot : \cdot \Uparrow (A \otimes B) \oplus C, A^\perp \parr B^\perp$ in LLF.
We first apply the $\parr$ rule and we need to prove $\vdash \cdot : \cdot \Uparrow (A \otimes B) \oplus C, A^\perp, B^\perp$. Now we apply twice $R \Uparrow$ consecutively, and we now need to prove $\vdash \cdot : B^\perp, A^\perp \Uparrow (A \otimes B) \oplus C$.
We pass to the focusing phase by applying the $R \Downarrow$ rule, then we apply the $\oplus_1$ rule and $\otimes$ rule, and we obtain two sequents to prove: $\vdash \cdot : A^\perp \Downarrow A$ and $\vdash \cdot : B^\perp \Downarrow B$ which are provable by applying the I_1 rule.
Note that the first non-determinism described in the paragraph about the backward proof search in LL does not appear in LLF.

For ILL, we use the proof system ILLF (generalized from [3]) described below.
Since the sequents are two sided, we need to distinguish left and right focusing (resp. active) phases. We first define the following meta-variables:

meta-variable	meaning
P	left-synchronous $(\&, \top, \multimap, X)$
Q	right-synchronous $(\otimes, \oplus, !, 1, 0, X)$
T	left-asynchronous $(\otimes, \oplus, !, 1, 0)$
R	right-asynchronous $(\&, \top, \multimap)$

The focused system ILLF consists of three kinds of sequents: left-focal sequents of the form $\Theta\ ;\ \Gamma\ ;\ A \ll Q$ (A under focus), right-focal sequents of the form $\Theta\ ;\ \Gamma \gg A$, and active sequents of the form $\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow C$. As usual, Θ indicates the unrestricted zone, Γ contains only left-synchronous formulas and Ω is an ordered list of formulas. Any sequent will be reduced to the form $\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow Q$ after the active phase, which we call *neutral sequents*.
Proving $\Gamma \vdash C$ in ILL is equivalent to proving $\cdot\ ;\ \cdot\ ;\ \Gamma \Longrightarrow C$ in ILLF

We now present the inference rules of ILLF:

$$\boxed{\Theta\ ;\ \Gamma \gg A \quad \text{right-focal}}$$

$$\frac{\Theta\ ;\ \Gamma \gg A \qquad \Theta\ ;\ \Gamma' \gg B}{\Theta\ ;\ \Gamma, \Gamma' \gg A \otimes B}\otimes R \qquad \frac{\Theta : \Gamma \gg A_i}{\Theta : \Gamma \gg A_1 \oplus A_2}\oplus R_i \quad i \in \{1,2\} \qquad \frac{}{\Theta\ ;\ \cdot \gg 1}\,1R \qquad \frac{\Theta\ ;\ \cdot\ ;\ \cdot \Longrightarrow A}{\Theta\ ;\ \cdot \gg\, !A}\,!R$$

$$\boxed{\Theta\ ;\ \Gamma\ ;\ A \ll Q \quad \text{left-focal}}$$

$$\frac{}{\Theta\ ;\ \cdot\ ;\ X \ll X}\ init \qquad \frac{\Theta\ ;\ \Gamma\ ;\ A_i \ll C}{\Theta\ ;\ \Gamma\ ;\ A_1 \,\&\, A_2 \ll C}\,\&L_i \quad i \in \{1,2\} \qquad \frac{\Theta\ ;\ \Gamma\ ;\ B \ll C \qquad \Theta\ ;\ \Gamma' \gg A}{\Theta\ ;\ \Gamma, \Gamma'\ ;\ A \multimap B \ll C} \multimap_L$$

$$\boxed{\text{focus}}$$

$$\frac{\Theta\ ;\ \Gamma\ ;\ P \ll Q}{\Theta\ ;\ \Gamma, P\ ;\ \cdot \Longrightarrow Q}\ lf \qquad \frac{\Theta, A\ ;\ \Gamma\ ;\ A \ll Q}{\Theta, A\ ;\ \Gamma\ ;\ \cdot \Longrightarrow Q}\ copy \qquad \frac{\Theta\ ;\ \Gamma \gg T}{\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow T}\ rf$$

$$\boxed{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow R \quad \text{right-active}}$$

$$\frac{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A \qquad \Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow B}{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A \,\&\, B}\,\&R \qquad \frac{}{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow \top}\,\top R \qquad \frac{\Theta\ ;\ \Gamma\ ;\ \Omega, A \Longrightarrow B}{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A \multimap B} \multimap R$$

$$\boxed{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow Q \quad \text{left-active}}$$

$$\frac{\Theta\ ;\ \Gamma\ ;\ \Omega, A, B \Longrightarrow Q}{\Theta\ ;\ \Gamma\ ;\ \Omega, A \otimes B \Longrightarrow Q}\otimes L \qquad \frac{\Theta\ ;\ \Gamma\ ;\ \Omega, A \Longrightarrow Q \qquad \Theta\ ;\ \Gamma\ ;\ \Omega, B \Longrightarrow Q}{\Theta\ ;\ \Gamma\ ;\ \Omega, A \oplus B \Longrightarrow Q}\oplus_L \qquad \frac{\Theta, A\ ;\ \Gamma\ ;\ \Omega \Longrightarrow Q}{\Theta\ ;\ \Gamma\ ;\ \Omega, !A \Longrightarrow Q}\,!L$$

$$\frac{\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow Q}{\Theta\ ;\ \Gamma\ ;\ \Omega, 1 \Longrightarrow Q}\,1L \qquad \frac{}{\Theta\ ;\ \Gamma\ ;\ \Omega, 0 \Longrightarrow Q}\,0L \qquad \frac{\Theta\ ;\ \Gamma, P\ ;\ \Omega \Longrightarrow Q}{\Theta\ ;\ \Gamma\ ;\ \Omega, P \Longrightarrow Q}\ act$$

$$\boxed{\text{blur}}$$

$$\frac{\Theta\ ;\ \Gamma\ ;\ T \Longrightarrow Q}{\Theta\ ;\ \Gamma\ ;\ T \ll Q}\ lb \qquad \frac{\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow P}{\Theta\ ;\ \Gamma \gg P}\ rb$$

Fig. 5. The focused proof system ILLF for intuitionistic linear logic.

3.1 Design of the prover

We use the focused proof system LLF in order to reduce inessential non-determinism in the proof search.

Besides the standard design, we propose the following implementation techniques:

3.1.1 Control of the D_2 rule

The main difficulty lies in the choice made in the decision rule D_2 (see Fig. 4). In D_2, we choose a formula from the unrestricted zone Θ and put it under focus. If we use a classic depth-first design, we will first visit the branch in which we choose the same formula when applying the rule D_2. In most cases, this cannot find the proof efficiently. If we use a breadth-first design based on the use of queues, then the proof extration step will be much more difficult than the recursive depth-first approach.

Therefore, we decide to choose the formula in a round-robin style. To do that, we use a list "$select\_d2$" of candidates for the rule D_2. When we apply the rule D_2, we choose the head of the list as the formula under focus and then delete it from the list. Once the list is empty, we refill it with the elements of Θ. In order to prevent the non-termination of our algorithm, knowing that LL is undecidable, we also use a variant max_d2 such that either max_d2 or the length of $select\_d2$ decreases when we apply the rule D_2 (In fact, the max_d2 is decreased by 1 when the list $select\_d2$ is refilled). This gives us a simple way of bounding the number of applications of D_2, even if we cannot interpret the explicit meaning of max_d2.

3.1.2 Optimization : order of branches

When we apply the rule & or $\otimes$ in LLF, we have two branches to prove. In cases where one of the branches is not provable, the decision on the first branch to prove have a huge influence on the execution time of the algorithm. A naive idea is to choose the branch with a smaller main formula (the formula obtained from the decomposition of the connective) in terms of size. Another idea which is more sophiscated is to choose the formula with the main formula of a smaller "whynot-height". The whynot-height $wn_h(F)$ of a formula F is defined inductively as follows:

$wn_h(1) = wn_h(0) = wn_h(\top) = wn_h(\bot) = wn_h(X) = wn_h(X^\bot) = 0$ where X is an atom.

$wn_h(?F) = 1 + wn_h(F), wn_h(!F) = wn_h(F), wn_h(F \diamond G) = max(wn_h(F), wn_h(G)), \quad \forall \diamond \in \{\otimes, \oplus, \&, \parr\}$

In fact, as observed in the previous paragraph, the main factor that will slow down our algorithm is the D_2 rule , and we can see that the "D_2 depth" (the number of applications of D_2) of a single branch corresponds approximately to the whynot-height of a formula in the goal sequent. Therefore, by choosing the branch with smaller whynot-height, we are likely to visit the branch that can be proved or disproved faster than the other branch.

4 Forward Proof Search: The focused inverse method

The inverse method [7] is a saturation-based proving technique that starts from some axioms in a database and then moves forward by applying rules to a collection of sequents in the database and adding the results obtained into the database until the goal sequent has been derived or the database has been saturated.

4.1 The focused inverse method

In [2], Chaudhuri adapted the idea of focusing to the inverse method. The main idea is to define "big step" inference rules that correspond to the combination of two phases (one focusing and one active). Each derived rule starts from a neutral sequent from which a synchronous formula is selected under focus, then continues with a sequence of focusing steps until the formula under focus becomes asynchronous. This is followed by a sequence of applications of active rules and we obtain a collection of neutral sequents, regarded as leaves of this derivation tree.

In the following paragraph, we will summarize the main ideas and results from [2].

4.1.1 Forward sequent calculus

We want to construct an appropriate sequent calculus for forward proof search in ILL.
In order to deal with rules with arbitrary linear zone (Γ ; Ω in ILLF, see Fig. 5.), such as $\top R$ and $0L$, Chaudhuri introduced the definition of forward sequents.

Definition 1 (forward sequents)

A forward sequent has one of the following forms:

- strong sequent : Θ ; $[\Gamma]_0 \longrightarrow C$
- weak sequent : Θ ; $[\Gamma]_1 \longrightarrow \gamma$

Here, C is a formula and γ is of the form $\cdot$ or C.

Intuitively, weak sequents are used to deal with arbitrary linear zone. For example, the conclusion of the $\top R$ rule of ILLF can be rewritten as Θ ; $[\cdot]_1 \longrightarrow \top$. In fact, this can even be rewritten as $\cdot$; $[\cdot]_1 \longrightarrow \top$ since the unrestricted zone can be easily manipulated by the exponenetial rules.

We first define some operations for linear contexts with weakness flag ($[\Gamma]_w, w = 0, 1$) :

Definition 2 (Subsumption) We define the subsumption relation between forward sequents as follows,

$$(\Theta \ ; \ [\Gamma]_0 \longrightarrow C) \prec (\Theta' \ ; \ [\Gamma]_1 \longrightarrow C) \qquad \text{if } \Theta \subseteq \Theta'$$
$$(\Theta \ ; \ [\Gamma]_1 \longrightarrow \gamma) \prec (\Theta' \ ; \ [\Gamma']_w \longrightarrow \gamma') \qquad \text{if } \Theta \subseteq \Theta', \Gamma \subseteq \Gamma', \text{ and } \gamma \subseteq \gamma'$$

Definition 3 (Additive composition) Given two linear contexts with weakness flag, $[\Gamma]_w$ and $[\Gamma']_{w'}$, we define their additive composition as follows:

$$[\Gamma]_w + [\Gamma']_{w'} = \begin{cases} [\Gamma]_0 & \text{if } w = w' = 0 \text{ and } \Gamma = \Gamma' \\ [\Gamma]_0 & \text{if } w = 0,\ w' = 1 \text{ and } \Gamma' \subseteq \Gamma \\ [\Gamma']_0 & \text{if } w = 1,\ w' = 0 \text{ and } \Gamma \subseteq \Gamma' \\ [\Gamma \sqcup \Gamma']_1 & \text{if } w = w' = 1 \end{cases}$$

where $\Gamma \sqcup \Gamma'$ denotes the least upper bound of Γ and Γ', i.e., for every formula F, the number of its occurrences in $\Gamma \sqcup \Gamma'$ is equal to the maximum of the number of its occurrences in Γ and the number of its occurrences in Γ'.
Note that this function is partial.

Definition 4 (Multiplicative composition) Given two linear contexts with weakness flag, $[\Gamma]_w$ and $[\Gamma']_{w'}$, we define their multiplicative composition as follows: $[\Gamma]_w \times [\Gamma']_{w'} = [\Gamma \cup \Gamma']_{w \vee w'}$
Here, $\Gamma \cup \Gamma'$ denotes the sum of the multisets Γ and Γ'.

Definition 5 (Insertion) Given a linear context with weakness flag $[\Gamma]_w$ and a formula A, we define the result context after insertion of A as follows: $[\Gamma]_w, A = [\Gamma, A]_w$

For any given formula, we want to construct a derived inference for the formula corresponding to a pair of focusing and active phases. Every formula is interpreted as a relation between sequences of forward sequents (corresponding to premises of the derived rules) and forward sequents (corresponding to the conclusions).
These relations are represented in an applicative style: $R[x] \hookrightarrow y$ for $(x, y) \in R$.

The derived rule for positive formulas is

$$\frac{s_1\ s_2 \cdots s_n\ (foc^+(Q)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Theta\ ;\ D \longrightarrow \cdot)}{\Theta\ ;\ D \longrightarrow Q}\ foc^+$$

Similarly, for negative formulas, we have two rules:

$$\frac{s_1\ s_2 \cdots s_n\ (foc^-(P)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Theta\ ;\ D \longrightarrow Q)}{\Theta\ ;\ D, P \longrightarrow Q}\ foc^-$$

$$\frac{s_1\ s_2 \cdots s_n\ (foc^-(A)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Theta\ ;\ D \longrightarrow Q)}{\Theta, A\ ;\ D \longrightarrow Q}\ !foc^-$$

Intuitively, the relation $foc^+(A)$ corresponds to the right-focal phase, $foc^-(A)$ to the left-focal phase, and $act(s)$ to the active phases. Their definitions are given in detail as follows:

right-focal

$$\frac{foc^+(A)[\Sigma_1] \hookrightarrow \Theta_1\ ;\ D_1 \longrightarrow \cdot \qquad foc^+(B)[\Sigma_2] \hookrightarrow \Theta_2\ ;\ D_2 \longrightarrow \cdot}{foc^+(A \otimes B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Theta_1, \Theta_2\ ;\ D_1 \times D_2 \longrightarrow \cdot}\ \otimes F \qquad \frac{foc^+(A_i)[\Sigma] \hookrightarrow s}{foc^+(A_1 \oplus A_2)[\Sigma] \hookrightarrow s}\ \oplus F_i$$

$$\frac{}{foc^+(1)[\cdot] \hookrightarrow \cdot\ ;\ [\cdot]_0 \longrightarrow \cdot}\ 1F \qquad \frac{act(\cdot\ ;\ \cdot\ ;\ \cdot \Longrightarrow A)[\Sigma] \hookrightarrow \Theta\ ;\ [\cdot]_w \longrightarrow \cdot}{foc^+(!A)[\Sigma] \hookrightarrow \Theta\ ;\ [\cdot]_0 \longrightarrow \cdot}\ !F \qquad \frac{act(\cdot\ ;\ \cdot\ ;\ \cdot\ ;\ R)[\Sigma] \hookrightarrow s}{foc^+(R)[\Sigma] \hookrightarrow s}\ FA^+$$

left-focal

$$\frac{}{foc^-(X)[\cdot] \hookrightarrow \cdot\ ;\ [\cdot]_0 \longrightarrow X}\ \text{init} \qquad \frac{foc^-(B)[\Sigma_1] \hookrightarrow \Theta_1\ ;\ D_1 \longrightarrow \gamma \qquad foc^+(A)[\Sigma_2] \hookrightarrow \Theta_2\ ;\ D_2 \longrightarrow \cdot}{foc^-(A \multimap B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Theta_1, \Theta_2\ ;\ D_1 \times D_2 \longrightarrow \gamma}\ \multimap F$$

$$\frac{foc^-(A_i)[\Sigma] \hookrightarrow s}{foc^-(A_1\ \&\ A_2)[\Sigma] \hookrightarrow s}\ \&F_i \qquad \frac{act(\cdot\ ;\ \cdot\ ;\ L \Longrightarrow \cdot)[\Sigma] \hookrightarrow s}{foc^-(L)[\Sigma] \hookrightarrow s}\ FA^-$$

active

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A)[\Sigma_1] \hookrightarrow \Theta_1\ ;\ D_1 \longrightarrow \cdot \qquad act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow B)[\Sigma_2] \hookrightarrow \Theta_2\ ;\ D_2 \longrightarrow \cdot}{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A\ \&\ B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Theta_1, \Theta_2\ ;\ D_1 + D_2 \longrightarrow \cdot}\ \&A$$

$$\frac{}{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow \top)[\cdot] \hookrightarrow \cdot\ ;\ [\cdot]_1 \longrightarrow \cdot}\ \top A$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A \Longrightarrow \xi)[\Sigma_1] \hookrightarrow \Theta_1\ ;\ D_1 \longrightarrow \gamma_1 \qquad act(\Theta\ ;\ \Gamma\ ;\ \Omega, B \Longrightarrow \xi)[\Sigma_2] \hookrightarrow \Theta_2\ ;\ D_2 \longrightarrow \gamma_2}{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A \oplus B \Longrightarrow \xi)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Theta_1, \Theta_2\ ;\ D_1 + D_2 \longrightarrow \gamma_1 \cup \gamma_2}\ \oplus A$$

$$\frac{}{act(\Theta\ ;\ \Gamma\ ;\ \Omega, 0 \Longrightarrow \xi)[\cdot] \hookrightarrow \cdot\ ;\ [\cdot]_1 \longrightarrow \cdot}\ 0A$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A \Longrightarrow B)[\Sigma] \hookrightarrow s}{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A \multimap B)[\Sigma] \hookrightarrow s}\ \multimap A$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow B)[\Sigma] \hookrightarrow \Theta'\ ;\ [\Gamma']_1 \longrightarrow \gamma}{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow A \multimap B)[\Sigma] \hookrightarrow \Theta'\ ;\ [\Gamma']_1 \longrightarrow \gamma}\ \multimap A'$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A, B \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A \otimes B \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ \otimes A$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A_i \Longrightarrow \xi)[\Sigma] \hookrightarrow \Theta'\ ;\ [\Gamma']_1 \longrightarrow \gamma}{act(\Theta\ ;\ \Gamma\ ;\ \Omega, A_1 \otimes A_2 \Longrightarrow \xi)[\Sigma] \hookrightarrow \Theta'\ ;\ [\Gamma']_1 \longrightarrow \gamma}\ \otimes A'$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{act(\Theta\ ;\ \Gamma\ ;\ \Omega, 1 \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ 1A$$

$$\frac{act(\Theta, A\ ;\ \Gamma\ ;\ \Omega \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{act(\Theta\ ;\ \Gamma\ ;\ \Omega, !A \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ !A$$

$$\frac{act(\Theta\ ;\ \Gamma\ ;\ \Omega, P \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{act(\Theta\ ;\ \Gamma, P\ ;\ \Omega \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ act$$

$$\frac{\xi \subseteq \gamma}{act(\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow \xi)[\Theta, \Theta'\ ;\ [\Gamma, \Gamma']_w \longrightarrow \gamma] \hookrightarrow \Theta'\ ;\ [\Gamma']_w \longrightarrow \gamma \backslash \xi}\ match$$

Fig. 6. Forward derived rules for ILL : focal and active phases

In the $\oplus A$ rule, $\gamma_1 \cup \gamma_2$ is defined as follows:

$$\gamma_1 \cup \gamma_2 = \begin{cases} \gamma_1 & if\ \gamma_2 = \cdot \\ \gamma_2 & if\ \gamma_1 = \cdot \\ \gamma_1 & if\ \gamma_1 = \gamma_2 \end{cases}$$

Note that this definition is partial.

In the *match* rule, $\xi \subseteq \gamma$ if $\xi = \cdot$ or $\xi = \gamma$, and

$$\gamma \backslash \xi = \begin{cases} \gamma & if\ \xi = \cdot \\ \cdot & if\ \xi = \gamma \end{cases}$$

We have the two following theorems:

Theorem 2 (Soundness)

1. If $\Theta\ ;\ [\Gamma]_0 \longrightarrow C$ is derivable, then $\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow C$ is derivable.
2. If $\Theta\ ;\ [\Gamma]_1 \longrightarrow \gamma$ is derivable, then $\Theta\ ;\ \Gamma'\ ;\ \cdot \Longrightarrow C$ is derivable for every $\Gamma' \supseteq \Gamma$ and $C \supseteq \gamma$.

Theorem 3 (Completeness) If $\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow Q$, then for some derivable $\Theta'\ ;\ [\Gamma']_w \longrightarrow \gamma$, we have,

$$(w = 0, \Theta' \subseteq \Theta, \Gamma' = \Gamma\ and\ \gamma = Q)\ \text{or}\ (w = 1, \Theta' \subseteq \Theta, \Gamma' \subseteq \Gamma\ and\ \gamma \subseteq Q)$$

4.1.2 The focused inverse method

What remains is to implement a proof search strategy that uses this forward calculus. An essential issue in the inverse method is to determine the propositions for which we need to derive inference rules in order to guarantee the completeness. Since this calculus of derived rules contains only neutral sequents, we only need to consider the propositions that occur in neutral sequents, called *frontier proposition*. To determine the frontier propositions, we define two functions, f (focal) and a (active) from signed propositions to sets of frontier propositions as follows:

$$
\begin{array}{cc}
f(X)^- = \emptyset & f(X)^+ = \{X^+\} \\
a(X)^- = \{X^-\} & a(X)^+ = \{X^+\} \\
f(A \otimes B)^- = a(A \otimes B)^- & f(A \otimes B)^+ = f(A)^+ \cup f(B)^+ \\
a(A \otimes B)^- = a(A)^- \cup a(B)^- & a(A \otimes B)^+ = f(A \otimes B)^+ \cup \{(A \otimes B)^+\} \\
f(A \oplus B)^- = a(A \oplus B)^- & f(A \oplus B)^+ = f(A)^+ \cup f(B)^+ \\
a(A \oplus B)^- = a(A)^- \cup a(B)^- & a(A \oplus B)^+ = f(A \oplus B)^+ \cup \{(A \oplus B)^+\} \\
f(A \,\&\, B)^- = f(A)^- \cup f(B)^- & f(A \,\&\, B)^+ = a(A \,\&\, B)^+ \\
a(A \,\&\, B)^- = f(A \,\&\, B)^- \cup \{(A \,\&\, B)^-\} & a(A \,\&\, B)^+ = a(A)^+ \cup a(B)^+ \\
f(A \multimap B)^- = f(A)^+ \cup f(B)^- & f(A \multimap B)^+ = a(A \multimap B)^+ \\
a(A \multimap B)^- = f(A \multimap B)^- \cup \{(A \multimap B)^-\} & a(A \multimap B)^+ = a(A)^- \cup a(B)^+ \\
f(!A)^- = a(!A)^- & f(!A)^+ = a(A)^+ \\
a(!A)^- = f(A)^- \cup \{(A)^-_!\} & a(!A)^+ = f(A)^+ \cup \{(!A)^+\} \\
f(1)^- = f(1)^+ = a(1)^- = a(1)^+ = \emptyset & f(0)^- = f(0)^+ = a(0)^- = a(0)^+ = \emptyset \\
f(\top)^- = f(\top)^+ = a(\top)^- = a(\top)^+ = \emptyset &
\end{array}
$$

Here, $(A)^-_!$ means that when we talk about the derived rule, it is the $!foc^-$ rule that is chosen.

Definition 6 (frontier) Given a goal sequent $\Theta\ ;\ \Gamma\ ;\ \cdot \Longrightarrow Q$, its frontier contains,

- all the propositions in Θ, Γ, Q,
- the collection $f(A)^-$ for all $A \in \Theta, \Gamma$, and
- the collection $f(Q)^+$.

Lemma 1 (neutral subformula property) In any backward focused proof, all neutral sequents consist only of frontier propositions of the goal sequent.

Before generating derived rules, we precalculate the frontier proposition of the goal sequents. There is no need to generate axioms (*init*) separately, as the derived rules of negative atoms in the frontier give directly the necessary initial sequents.

Each rule is applied to sequents selected from the current database, and we will get a new sequent if the rule applies succesfully. Before inserting this new sequent into the database, we need to check if this sequent is actually subsumed by some sequent already in the database or if it subsumes some sequents already in the database. In this cases, the subsumed sequents will not be kept in the database in order to avoid redundancy.

All these above results are explained and proved rigorously in [2]. In fact, we can assign a bias to each atom, but all atoms are considered right-biased here.

4.1.3 Some implementation details

K. Chaudhuri have given one implementation technique about the unrestricted context (the $!foc^-$ rule) : if the proposition $A^-_!$ considered occurs in the unrestricted zone in the goal sequent $\Theta_0\ ;\ \Gamma_0\ ;\ \gamma$, then it need not to be added into the unrestricted zone in the conclusion.
This technique comes from the fact that in a backward proof, the unrestricted zone Θ_0 is shared in every branch of the proof.

We have come up with another simple technique concerning the positive proposition C considered in the foc^+ rule. If all occurrences of C in the sequent are in the form of $!C$, then we can directly add $!C$ in the conclusion. To implement this in an efficient way, we precalculate a table *tbl* that associates every subformula F of the goal sequent with $!^kF$ where k is the minimal non-negative integer such

that not all occurrences of $!^k F$ in the goal sequent are in the form of $!^{k+1} F$. When we consider the proposition C when applying the foc^+ rule, we will add $tbl(C)$ to the right-hand side of the conclusion.

When we apply the derived rules, we first need to select a list of sequents (not necessarily distinct) from the database. Once the database is not empty, there is an infinite number of ways of selecting this list. In practice, we will set an upper bound on the number of times a sequent appears in this list of sequents. We will lose the completeness by doing so, but the soundness is conserved.

4.1.4 Proposing a generalization of the forward calculus for LL

We have generalized this forward calculus for ILL to another one for LL.
We present briefly our construction of these relations:

$$\boxed{\text{focal}}$$

$$\frac{foc(A)[\Sigma_1] \hookrightarrow \Theta_1 \ ; \ D_1 \qquad foc(B)[\Sigma_2] \hookrightarrow \Theta_2 \ ; \ D_2}{foc(A \otimes B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Theta_1, \Theta_2 \ ; \ D_1 \times D_2} \otimes F$$

$$\frac{foc(A_i)[\Sigma] \hookrightarrow s}{foc(A_1 \oplus A_2)[\Sigma] \hookrightarrow s} \oplus F_i$$

$$\frac{act(\cdot \ ; \ \cdot \ ; \ A)[\Sigma] \hookrightarrow \Theta \ ; \ [\cdot]_w}{foc(!A)[\Sigma] \hookrightarrow \Theta \ ; \ [\cdot]_0} \ !F$$

$$\frac{}{foc(1)[\cdot] \hookrightarrow \cdot \ ; \ [\cdot]_0} \ 1F$$

$$\frac{}{foc(X)[\cdot] \hookrightarrow \cdot \ ; \ [X^\perp]_0} \ init$$

$$\frac{act(\cdot \ ; \ \cdot \ ; \ L)[\Sigma] \hookrightarrow s}{foc(L)[\Sigma] \hookrightarrow s} \ FA \quad \text{where } L \text{ is asynchronous}$$

$$\boxed{\text{active}}$$

$$\frac{act(\Theta \ ; \ \Gamma \ ; \ L, A)[\Sigma_1] \hookrightarrow \Theta_1 \ ; \ D_1 \qquad act(\Theta \ ; \ \Gamma \ ; \ L, B)[\Sigma_2] \hookrightarrow \Theta_2 \ ; \ D_2}{act(\Theta \ ; \ \Gamma \ ; \ L, A \,\&\, B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Theta_1, \Theta_2 \ ; \ D_1 + D_2} \ \&A$$

$$\frac{act(\Theta \ ; \ \Gamma \ ; \ L, A, B)[\Sigma] \hookrightarrow s}{act(\Theta \ ; \ \Gamma \ ; \ L, A \parr B)[\Sigma] \hookrightarrow s} \ \parr A$$

$$\frac{act(\Theta \ ; \ \Gamma \ ; \ L, A_i)[\Sigma] \hookrightarrow \Theta' \ ; \ [\Gamma']_1}{act(\Theta \ ; \ \Gamma \ ; \ L, A_1 \parr A_2)[\Sigma] \hookrightarrow \Theta' \ ; \ [\Gamma']_1} \ \parr A_i$$

$$\frac{}{act(\Theta \ ; \ \Gamma \ ; \ L, \top)[\Sigma] \hookrightarrow \cdot \ ; \ [\cdot]_1} \ \top A$$

$$\frac{act(\Theta \cup \{A\} \ ; \ \Gamma \ ; \ L)[\Sigma] \hookrightarrow s}{act(\Theta \ ; \ \Gamma \ ; \ L, ?A)[\Sigma] \hookrightarrow s} \ ?A$$

$$\frac{act(\Theta \ ; \ \Gamma \ ; \ L)[\Sigma] \hookrightarrow s}{act(\Theta \ ; \ \Gamma \ ; \ L, \perp)[\Sigma] \hookrightarrow s} \ \perp A$$

$$\frac{act(\Theta \ ; \ \Gamma, R \ ; \ L)[\Sigma] \hookrightarrow s}{act(\Theta \ ; \ \Gamma \ ; \ L, R)[\Sigma] \hookrightarrow s} \ act \quad \text{where } R \text{ is synchronous}$$

$$\frac{}{act(\Theta \ ; \ \Gamma \ ; \ \cdot)[\Theta, \Theta' \ ; \ \Gamma, \Gamma'] \hookrightarrow \Theta' \ ; \ \Gamma'} \ match$$

Fig. 7. Forward derived rules for LL : focal and active phases

5 Implementation of the prover

5.1 General design

The prover is implemented in OCaml. The unrestricted zone of a sequent is represented as a set. In backward search, the linear zone is implemented using a list while it is implemented as a map in the forward search.
Further details can be found on https://github.com/wujuihsuan2016/LL_prover

5.2 Input Syntax

A valid test file consists of a sequent and comments delimited by (* and *). A sequent is of the form $\Gamma \vdash \Delta$, where Γ and Δ are comma-separated lists of formulas. A formula is represented by the following syntax :

LL symbol	representation
X	X
$X^{\perp}$	$X\hat{}$
$\otimes$	$*$
$\parr$	|
&	&
$\oplus$	+
$\multimap$	-o

LL symbol	representation
!	!
?	?
$\top$	top
$\perp$	bot
1	1
0	0

5.3 Outputs (only available for the backward search)

- LaTeX Export : The prover gives users the possibility to print proofs in a more readable style by using the LaTeX libraries cmll and ebproof. Users are allowed to generate proofs in focused systems but for users who are not familiar with the notion of focusing, it is possible to generate proofs in the usual LL (or ILL) system as well.
- Coq Export : In order to guarantee the validity of proofs produced, users are allowed to generate a corresponding certificate by using the Coq library Yalla. Miniyalla, a light version of Yalla, which contains all necessary elements for our exports to coq, is available on https://github.com/wujuihsuan2016/LL_prover.

6 Experiments

We have tested the prover with sequents gathered from [6], which are obtained from translations from intuitionistic logic into linear logic.

We now present some of our results:

Tests from Girard's translation				
Test	LL (B)	LL (F)	ILL (B)	ILL (F)
1	0.003	0.203	0.003	0.044
2	2.507	0.706	0.047	0.355
3	0.043	0.428	0.020	0.098
4	1.738	0.574	0.020	0.100
5	0.029	0.700	0.023	0.096
6	2.446	0.648	0.046	0.198
7	0.036	0.835	0.020	0.165
8	4.798	0.608	0.019	0.261
9	6.791	0.922	0.020	0.090
10	0.024	0.410	0.007	0.098
11	0.012	0.386	0.014	0.130
12	0.013	0.274	0.010	0.046
13	0.020	0.426	0.045	0.287
14	0.024	0.870	0.144	0.305
15	3.128	0.709	0.042	0.806
16	0.011	0.356	0.019	0.113
17	0.264	0.434	0.032	0.072
18	0.020	0.573	0.034	0.076
19	0.303	0.446	0.031	0.074
20	0.004	0.454	0.009	0.020
21	0.252	0.703	0.050	0.092
22	199.2	1.307	0.757	0.279
23	0.112	2.764	8.154	1.400
24	0.082	2.149	1.288	0.609
25	3.202	3.736	66.19	2.025
26	0.093	3.402	0.072	0.265

Tests from multiplicative translation		
Test	LL (B)	LL (F)
1	0.003	0.184
2	0.015	0.913
3	0.015	0.440
4	0.013	0.432
5	0.018	0.801
6	0.017	1.742
7	0.011	0.888
8	0.015	1.130
9	0.014	0.976
10	0.007(N)	0.533(-)
11	0.007(N)	0.552(-)
12	0.004(N)	0.376(-)
13	0.011	0.805
14	0.009	1.923
15	0.039	3.594
16	0.017(N)	1.304(-)
17	0.037(N)	1.444(-)
18	0.038(N)	1.712(-)
19	0.039(N)	1.476(-)
20	0.017	0.574(bound = 2)
21	0.057	3.610
22	0.259	794.7
23	0.057	566.3
24	0.025	157.9
25	0.036	734.7
26	0.068	>30000

Tab. 1. Results. For the inverse method (forward search, F), the upper bound (see 4.1.3) is set to be 1 and for the backward search (B), the bound (max_d2, see 3.1.1) is set to be 4. In the result, (N) indicates that the sequent is not provable and (-) indicates that no proof is found. Execution times are measured in miliseconds.

We can observe that for the tests from Girard's translation, if it takes a long time to prove backwards then it takes much less time to prove the sequent by using the inverse method (test 22 for LL, test 25 for ILL, etc.).
However, for some tests from the multiplicative translation, the inverse method ends up with a long execution time (test 26, test 22, etc.). This is resulted from the redundancy of the database. Since in every round, we need to consider all possible sub-multisets of the database (with maximal multiplicity equal to the bound), the execution time is exponenetial in the size of the database. We should therefore look for a way of reducing this redundancy in order to make the inverse method more efficient.

7 Application : Conservativity of LL over ILL

One simple application of our prover is the study of the conservativity of LL over ILL. We want to find counterexamples of the conservativity, i.e., formulas of ILL that are provable in LL but are not provable in ILL. In [5], O. Laurent found some necessary conditions that a candidate of counterexample should satisfy. This helps us to reduce the number of times we need to execute our prover.
We briefly present our result here: There are 11 shortest exponential-free counterexamples of size 9:

$$\begin{array}{|c|c|}
\hline
((X_0 \multimap X_1) \multimap 0) \multimap (X_0 \otimes \top) & (((\top \multimap X_0) \multimap X_1) \multimap 0) \multimap X_0 \\
((X_0 \multimap X_1) \multimap 0) \multimap (\top \otimes X_0) & (((\top \multimap X_0) \multimap 0) \multimap 0) \multimap X_0 \\
((X_0 \multimap 0) \multimap 0) \multimap (X_0 \otimes \top) & (((\top \multimap X_0) \multimap 1) \multimap 0) \multimap X_0 \\
((X_0 \multimap 0) \multimap 0) \multimap (\top \otimes X_0) & (((\top \multimap 1) \multimap X_0) \multimap 0) \multimap 1 \\
((X_0 \multimap 1) \multimap 0) \multimap (X_0 \otimes \top) & (((\top \multimap 1) \multimap 0) \multimap 0) \multimap 1 \\
((X_0 \multimap 1) \multimap 0) \multimap (\top \otimes X_0) & \\
\hline
\end{array}$$

There are two main counterexamples : $((X_0 \multimap X_1) \multimap 0) \multimap (X_0 \otimes \top)$ and $(((\top \multimap X_0) \multimap X_1) \multimap 0) \multimap X_0$ and the others are either their equivalents or instances of them.
Another interesting fact is that $(((\top \multimap 1) \multimap 0) \multimap 0) \multimap 1$ is a counterexample without atoms.

8 Conclusion and Future work

In my work, I developed an automated prover for LL and ILL. I also contributed to the proof search techniques with the following :

1. the backward proof search : the optimization on the order of branches and the idea of applying the D_2 (copy) rule in a round-robin style.
2. the forward proof search : the focused inverse method applied to LL

There are still some issues to improve in the future, including

- In some cases, there are too many sequents inserted into the database during the forward search, which implies a long execution time. In order to deal with this problem, we may need to find additional optimizations in order to reduce the number of unuseful sequents in the database.
- In this version of our prover, the relations foc^-, foc^+ and act are calculated in an applicative style which loses informations about the rules applied. This makes the proof extraction even more difficult. A possible approach is to memorize the proof of every sequent when it is generated by a derived rule.
- Give proofs of the soundness and completeness of the forward system for LL.
- Extend benchmark tests and give more detailed analysis of the execution times for the different methods.
- Develop a web interface for the prover using the framework Eliom and js_of_ocaml.

References

[1] J.-M. Andreoli. Logic programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3):297-347, 1992.

[2] K. Chaudhuri. The focused inverse method for linear logic, Ph.D. Thesis, Carnegie Mellon University, Technical Report CMU-CS-06-162, 2006.

[3] K. Chaudhuri and F. Pfenning. Focusing the inverse method for linear logic. In Luke Ong, editor, *Proceedings of CSL 2005*, pages 200-215. Springer-Verlag LNCS 3634, 2005.

[4] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50:1-102, 1987.

[5] O. Laurent. Around Classical and Intuitionistic Linear Logics. In *Proceedings of 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018.

[6] C. Olarte, V. de Paiva, E. Pimentel and G. Reis. Benchmarking Linear Logic Translations. https://github.com/carlosolarte/Benchmarking-Linear-Logic

[7] T. Tammet. Resolution, inverse method and the sequent calculus. In *Proceedings of the 5th Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 65-83. Springer-Verlag LNCS 1289, 1997.