The ILLTP Library for Intuitionistic Linear Logic

Carlos Olarte
Universidade Federal do Rio Grande do Norte, Brazil
carlos.olarte@gmail.com

Valeria de Paiva
Nuance Communications, USA
valeria.depaiva@gmail.com

Elaine Pimentel*
Universidade Federal do Rio Grande do Norte, Brazil
elaine.pimentel@gmail.com

Giselle Reis†
Carnegie Mellon University, Qatar
giselle@cmu.edu

Benchmarking automated theorem proving (ATP) systems using standardized problem sets is a well-established method for measuring their performance. However, the availability of such libraries for non-classical logics is very limited. In this work we propose a library for benchmarking Girard's (propositional) intuitionistic linear logic. For a quick bootstrapping of the collection of problems, and for discussing the selection of relevant problems and understanding their meaning as linear logic theorems, we use translations of the collection of Kleene's intuitionistic theorems in the traditional monograph "Introduction to Metamathematics". We analyze four different translations of intuitionistic logic into linear logic and compare their proofs using a linear logic based prover with focusing. In order to enhance the set of problems in our library, we apply the three provability-preserving translations to the propositional benchmarks in the ILTP Library. Finally, we generate a comprehensive set of reachability problems for Petri nets and encode such problems as linear logic sequents, thus enlarging our collection of problems.

1 Introduction

Benchmarking automated theorem proving (ATP) systems using standardized problem sets is a well-established method for measuring their performance. However, the availability of such libraries for *non-classical* logics is very limited. For intuitionistic logic, several small collections of formulas have been published and used for testing ATP systems. Raths, Otten and Kreitz [24] consolidated and extended these small sets to provide the ILTP Library[1] for first-order and propositional intuitionistic logic. For modal systems there are at least two libraries of theorems [31, 23].

In this paper, we provide a library for benchmarking Girard's linear logic [11]. Linear logic is a substructural logic that is a refinement of classical and intuitionistic logic, combining the dualities of the former with many of the constructive properties of the latter. Ideas from linear logic have been influential in fields such as programming languages, game semantics, quantum physics, as well as linguistics, particularly because of its emphasis on resource-boundedness, duality, and interaction. In particular, linear logic has had an important role as a logical framework for specifying and reasoning about logical and computational systems (the list is long; some examples are [6, 19, 8, 22]). As a consequence, several provers have been built for linear logic for different purposes (listing some: Sympli [2], llprover[3], LL

*Olarte and Pimentel are supported by CAPES, CNPq and the project FWF START Y544-N23.
†This paper was made possible by grant NPRP 7-988-1-178 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

[1] http://iltp.de/
[2] https://github.com/chaudhuri/sympli
[3] http://bach.istc.kobe-u.ac.jp/llprover/

T. Ehrhard, M. Fernández, V. de Paiva, L. Tortora de Falco (Eds.):
Linearity-TLLA 2018
EPTCS 292, 2019, pp. 118–132, doi:10.4204/EPTCS.292.7

prover[4], linTAP[5], LL prover explorer[6], Lolli[7], Alcove[8]). However, so far, there has been no discussion about the *efficiency* or *adequacy* of these provers. In this work we present a comprehensive collection of propositional tests. Moreover, with the aim of comparing proofs obtained from different translations of intuitionistic problems, we use a prototypical prover, based on focusing, to run some experiments.

When designing a benchmark, one has to carefully decide on a set of formulas that is *meaningful* in, at least, three ways: (1) the formulas should be able to distinguish several different characteristics of the logical systems and provers; (2) the set should contain important theorems and paradigmatic formulas (non necessarily provable); and (3) the set should be large enough, so to serve as a comparison for different provers and systems. We first concentrate on (1) and (2), comparing translations of a (small) set of intuitionistic formulas into linear logic. This will enable the discussion of other possible approaches to follow. We will then deal with (3), by introducing a library with more than 4.000 problems, including different translations of intuitionistic formulas as well as translations of reachability problems in Petri nets as linear logic sequents.

Starting with (1), it turns out that propositional linear logic (LL) has many aspects that need to be considered. For example, one could adopt its classical (CLL) or intuitionistic (ILL) versions. Hence one important task would be to determine the difference in provability between them, and this is already far from trivial. While it is possible to differentiate the syntax of formulas and the presentation of the inference rules by the standard restriction on the right context to having at most one formula in ILL[9], FILL [2] is a multiple-conclusion system with the same connectives and rules as CLL, but restricting the form of the application of such rules. Restricting ourselves to formulas with the same syntax in classical and intuitionistic versions of LL, the first interesting question would be which formulas are provable in CLL but not in ILL. This is the same issue *e.g.* when building a benchmark for intuitionistic logic versus the existing ones for classical logic. But the linear case is far more complicated, since the lack of the structural rules of weakening and contraction in both ILL and CLL makes these systems "closer" to each other than in the case of classical and intuitionistic logics. Indeed, only very recently [13] the first conservativity results presented in [26] were generalized.

Another important aspect to be taken into consideration is *focusing* [1]. It turns out that both ILL and CLL admit complete focused proof systems, and provers can be built using proof search strategies based on this discipline, which reduces the proof search space. This has an immediate effect on the proposal of formulas composing a possible benchmark library, since the amount of exponentials in a formula can make a significant difference on the performance of provers.

Concerning (2), there is no consensus in the community on a set of "principal" theorems that should be used as a test for LL-based theorem provers. In this work, we will start the discussion by considering the translation of a fragment of Kleenes basic list of intuitionistic logic (IL). The first challenge is to understand *how* these intuitionistic theorems should be interpreted in LL. A first answer would be: use one of the well known translations of IL into LL. This naive approach has, at least, two problems. First, it is not adequate to elect *one* translation, since different translations have very different computational behaviors, as it will be clear in Section 2.2. Second, some translations would not give the best interpreta-

[4] https://github.com/wujuihsuan2016/LL_prover

[5] http://www.leancop.de/lintap/

[6] https://github.com/andykitchen/linear-logic

[7] http://www.lix.polytechnique.fr/~dale/lolli/

[8] http://cic.puj.edu.co/~caolarte/alcove2/

[9] We note that in the literature there are two versions of ILL, having *at most* or *exactly* one formula in the right context. This is similar to the problem of considering intuitionistic/minimal logics. Since in this work we will use a multiplicative fragment of ILL, we adopted the version of ILL having $\bot$ in the grammar.

tion of linear logic formulas. As a simple example, $A \to A$ should most probably be translated as $A \multimap A$, without exponentials since this is equivalent, as a theorem, to the identity. But *any* sound translation from IL to LL adds exponentials to implicational formulas. Hence none of them would preserve the axiom's interpretation.

To achieve desiderata (3), we start by applying four different translations from IL to LL using Kleene's collection of IL theorems. One of the translations is not validity perserving, resulting in 22 non-provable formulas. We propose provable versions (not following any systematic translation) for those. This gives a initial set of 271 sequents for our library. Next, we apply to all the propositional formulas of the ILTP library the three provability preserving translations considered in this work. This whole set will not only provide some interesting insights on different behaviors of LL formulas coming from different translations in the literature, but also present a significant set of 1.086 formulas for benchmarking linear logic based provers. Finally, to enlarge the number of problems in our library, we have generated automatically several reachability problems from the large collection of Petri nets[10], available in the Petri Nets Repository[11]. Then, by using the standard translation of Petri nets as ILL formulas [17, 4], we add to our benchmark 3.137 formulas of different levels of difficulty. All these 4.494 formulas will constitute our ILLTP library for intuitionistic linear logic, available at:

`https://github.com/meta-logic/lltp`.

Outline. The rest of the paper is organized as follows. Section 2 presents LL, focusing, translations and decorations; Section 3 explains the different parts that make ILLTP, and some experiment results; Section 4 concludes the paper and presents some future research directions.

2 Linear Logic

Although we assume that the reader is familiar with linear logic, we review some of its basic proof theory (see [30] for more details).

Linear logic is a substructural logic proposed by Girard [11] as a refinement of classical and intuitionistic logic. Formulas for propositional linear logic (LL) are built from the following grammar

$$F ::= p \mid 1 \mid 0 \mid \top \mid \bot \mid F \otimes F \mid F \parr F \mid F \,\&\, F \mid F \oplus F \mid F \multimap F \mid ?F \mid !F$$

where atomic formulas p or their negations $p^\perp = p \multimap \bot$ are called *literals*. The logical connectives for LL can be divided into the following groups: the *multiplicative* version of conjunction, true, disjunction, and false, which are written as $\otimes$, 1, $\parr$, $\bot$, respectively; and the *additive* version of these connectives, which are written as &, $\top$, $\oplus$, 0, respectively; and the *exponentials* ! and ?.

In this work we will concentrate on intuitionistic linear logic (ILL) [11], having sequents of the form $\Gamma \vdash \Delta$ where Δ has at most one formula and Γ is a multiset of formulas built from the grammar above, except for the connective $\parr$ and the exponential ?. The rules of ILL are depicted in Figure 1.

2.1 Focusing

Andreoli introduced in [1] a notion of normal form for cut-free derivations in linear logic. The connectives of LL can be divided into two classes: *negative* ($\parr$, $\bot$, &, $\top$, and ?) and *positive* ($\otimes$, 1, $\oplus$, 0, and !). Note that the dual of a negative connective is positive and vice-versa. In general, the right introduction

[10] We thank Frank Pfenning, who suggested the use of Petri nets during the TLLA/Linearity 2018 meeting.

[11] `https://pnrepository.lip6.fr/`

$$\frac{}{p \vdash p}\ \mathsf{init} \qquad \frac{}{\vdash 1}\ 1_R \qquad \frac{}{\Gamma \vdash \top}\ \top_R \qquad \frac{\Gamma \vdash}{\Gamma \vdash \bot}\ \bot_R \qquad \frac{\Gamma \vdash \Delta}{\Gamma, 1 \vdash \Delta}\ 1_L \qquad \frac{}{\bot \vdash}\ \bot_L \qquad \frac{}{\Gamma, 0 \vdash \Delta}\ 0_L$$

$$\frac{\Gamma \vdash F \quad \Gamma \vdash G}{\Gamma \vdash F \,\&\, G}\ \&_R \qquad \frac{\Gamma_1 \vdash F \quad \Gamma_2 \vdash G}{\Gamma_1, \Gamma_2 \vdash F \otimes G}\ \otimes_R \qquad \frac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \oplus F_2}\ \oplus_{R_i} \qquad \frac{\Gamma, F_i \vdash \Delta}{\Gamma, F_1 \,\&\, F_2 \vdash \Delta}\ \&_{L_i}$$

$$\frac{\Gamma, F, G \vdash \Delta}{\Gamma, F \otimes G \vdash \Delta}\ \otimes_L \qquad \frac{\Gamma, F \vdash \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \oplus G \vdash \Delta}\ \oplus_L \qquad \frac{\Gamma, F \vdash G}{\Gamma \vdash F \multimap G}\ \multimap_R \qquad \frac{\Gamma_1 \vdash F \quad \Gamma_2, G \vdash \Delta}{\Gamma_1, \Gamma_2, F \multimap G \vdash \Delta}\ \multimap_L$$

$$\frac{!\Gamma \vdash F}{!\Gamma \vdash\, !F}\ !_R \qquad \frac{\Gamma, !F, !F \vdash \Delta}{\Gamma, !F \vdash \Delta}\ \mathsf{cont}_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma, !F \vdash \Delta}\ \mathsf{weak}_L \qquad \frac{\Gamma, F \vdash \Delta}{\Gamma, !F \vdash \Delta}\ \mathsf{der}_L$$

Figure 1: System ILL. In $\oplus_{R_i}$ and $\&_{L_i}$, $F_i \in \{F_1, F_2\}$.

rules for negative connectives are invertible, meaning that the conclusion of any of these introduction rules is equivalent to its premises. The right introduction rules for the positive connectives are not necessarily invertible, and may require a choice or a context restriction on the application of rules. The notions of negative and positive polarities are extended to formulas in the natural way by considering the outermost connective, *e.g.*, $A \oplus B$ is positive while $A \,\&\, B$ is negative. Any bias can be assigned to atomic LL formulas.

A focused proof is organized around two "phases" of proof construction: the *negative phase* for introducing negative connectives on the succedent or positive ones in the antecedent, and the *positive phase* for the positive connectives on the succedent or negative ones in the antecedent. In the focusing discipline, formulas are decomposed eagerly in the negative phase, until only positive formulas on the succedent or negative on the antecedent are left. Then one of them is non-deterministically chosen to be focused on.

The focused system ILLF for intuitionistic linear logic is presented in Figure 2. There are three kinds of arrows in this proof system and a pair of contexts to the left of the arrows:

- Θ is a set of antecedent formulas whose main connective is a bang, being hence the unbounded (classical) context;
- Γ is a multiset of linear formulas, behaving as the bounded (linear) context;
- sequents of the form $\Theta : \Gamma - _F \rightarrow$ or $\Theta : \Gamma \xrightarrow{F} \Delta$ belong to the positive phase and introduce the logical connective of the "focused" formula F: building proofs of such sequents may require non-invertible proof steps to be taken;
- sequents of the form $\Theta : \Gamma \longrightarrow \Delta$ belong to the negative phase and decompose the multisets Γ, Δ in such a way that only inference rules over negative formulas on the right or positive ones on the left are applied.

The structural rules D_{L1}, D_{L2} and D_R make the transition between negative and positive phases. The *positive* phase begins by choosing a non-atomic fomula N_a from the classical context or a negative left formula N (resp. a positive right formula P) on which to focus using D_{L1}, D_{L2} (resp. D_R). Atoms in ILLF always have a *positive* bias[12]. Positive rules are applied to $N, N_a/P$ until either 1 or an atom is encountered in the right, or $\bot$ is on the left (and, in such cases, the proof must end by applying the

[12] Although ILL can allow any polarity assignment for atomic formulas, we will present here the system that was actually implemented.

respective axiom), or the promotion rule (!) is applied, or yet a positive left (respect. negative right) subformula is encountered, when the proof switches to the negative phase by applying R_L (resp. R_R).

An observation is in order: the system presented in Figure 2 induces a *weak focusing*, in the sense that, a priori, the positive phase could start even with the presence, in one of the contexts, of formulas amenable to negative rule applications. However, our rewrite-based implementation (see Section 3.3) forces the strong focusing approach: rules in the negative phase are exhaustively applied until a normal form is found, i.e., in all the resulting subgoals, there are no positive formulas in the linear context, nor a negative formula in the succedent.

Negative Phase

$$\frac{\Theta : \Gamma, F, G \longrightarrow \Delta}{\Theta : \Gamma, F \otimes G \longrightarrow \Delta}\ \otimes_L \qquad \frac{\Theta : \Gamma, F \longrightarrow G}{\Theta : \Gamma \longrightarrow F \multimap G}\ \multimap_R \qquad \frac{\Theta : \Gamma \longrightarrow \Delta}{\Theta : \Gamma, 1 \longrightarrow \Delta}\ 1_L \qquad \frac{\Theta : \Gamma \longrightarrow}{\Theta : \Gamma \longrightarrow \bot}\ \bot_R$$

$$\frac{}{\Theta : \Gamma \longrightarrow \top}\ \top_R \qquad \frac{}{\Theta : \Gamma, 0 \longrightarrow \Delta}\ 0_L \qquad \frac{\Theta, F : \Gamma \longrightarrow \Delta}{\Theta : \Gamma, !F \longrightarrow \Delta}\ !_L$$

$$\frac{\Theta : \Gamma \longrightarrow F \quad \Theta : \Gamma \longrightarrow G}{\Theta : \Gamma \longrightarrow F \,\&\, G}\ \&_R \qquad \frac{\Theta : \Gamma, F \longrightarrow \Delta \quad \Theta : \Gamma, H \longrightarrow \Delta}{\Theta : \Gamma, F \oplus H \longrightarrow \Delta}\ \oplus_L$$

Positive Phase

$$\frac{\Theta : \Gamma_1 -_F\!\rightarrow \quad \Theta : \Gamma_2 -_G\!\rightarrow}{\Theta : \Gamma_1, \Gamma_2 -_{F \otimes G}\!\rightarrow}\ \otimes_R \qquad \frac{\Theta : \Gamma_1 -_F\!\rightarrow \quad \Theta : \Gamma_2 \xrightarrow{G} \Delta}{\Theta : \Gamma_1, \Gamma_2 \xrightarrow{F \multimap G} \Delta}\ \multimap_L \qquad \frac{\Theta : \Gamma -_{F_i}\!\rightarrow}{\Theta : \Gamma -_{F_1 \oplus F_2}\!\rightarrow}\ \oplus_{R_i}$$

$$\frac{\Theta : \Gamma \xrightarrow{F_i} \Delta}{\Theta : \Gamma \xrightarrow{F_1 \& F_2} C}\ \&_{L_i} \qquad \frac{}{\Theta : \cdot -_1\!\rightarrow}\ 1_R \qquad \frac{}{\Theta : \cdot \xrightarrow{\bot}}\ \bot_L \qquad \frac{\Theta : \cdot \longrightarrow F}{\Theta : \cdot -_{!F}\!\rightarrow}\ !_R$$

$$\frac{}{\Theta : \Gamma -_p\!\rightarrow}\ I_R \text{ given } p \in (\Theta \cup \Gamma) \text{ and } \Gamma \subseteq \{p\}$$

Structural Rules

$$\frac{\Theta, N_a : \Gamma \xrightarrow{N_a} \Delta}{\Theta, N_a : \Gamma \longrightarrow \Delta}\ D_{L1} \qquad \frac{\Theta : \Gamma \xrightarrow{N} \Delta}{\Theta : \Gamma, N \longrightarrow \Delta}\ D_{L2} \qquad \frac{\Theta : \Gamma -_P\!\rightarrow}{\Theta : \Gamma \longrightarrow P}\ D_R$$

$$\frac{\Theta : \Gamma, P \longrightarrow \Delta}{\Theta : \Gamma \xrightarrow{P} \Delta}\ R_L \qquad \frac{\Theta : \Gamma \longrightarrow N}{\Theta : \Gamma -_N\!\rightarrow}\ R_R$$

Figure 2: System ILLF: a focused proof system for ILL. Here, p is an atomic formula (we will assume positive bias for atoms); N is a negative formula; P is a positive formula; N_a is a non-atomic formula. In $\oplus_{R_i}$ and $\&_{L_i}$, $F_i \in \{F_1, F_2\}$.

Finally, we observe that, although focusing is a complete strategy that decreases considerably the search space, it is not enough for avoiding loops on proof search. Let us give two simple examples. A focused proof of the sequent $\cdot : !A, !(!A \multimap B) \longrightarrow 0$ starts by storing $A, !A \multimap B$ in the classical context and then it focuses on $!A \multimap B$ continuously, adding various copies of B in the linear context. Since the sequents produced in this process are all different, no loop detection will work for this case. One way of avoiding this kind of behavior would be by restricting the number of focusing steps over the same formula in the classical context. This is a sound, but not complete, strategy. Another way to deal with this problem would be to introduce a heuristics in order to stop the computation or to modify the

sequents, depending on the shape of formulas in the context. In this particular example, it is enough to observe that sequents of the form $\Theta, A, A \multimap B : \Gamma \longrightarrow C$ can be re-written as $\Theta, A, B, A \multimap B : \Gamma \longrightarrow C$, and continue by analyzing the formula B.

Another pathological behavior may arise with a sequent of the shape $\Theta, P_1 \,\&\, P_2 : \Gamma \longrightarrow C$ where P_i is a positive formula. In this case, if we decide to focus on the formula $P_1 \,\&\, P_2$, we are forced to choose one of the P_i and then, the positive phase ends by adding yet another copy of P_i into the linear context. In this particular case, we may use the well known equivalence $!(F \,\&\, G) \equiv !F \otimes !G$ and rewrite the above sequent into a simpler one: $\Theta, P_1, P_2 : \Gamma \longrightarrow C$. If P_1 and P_2 are atomic formulas, this simplification is quite useful since those atoms will remain in the classical context (remember that it is not possible to focus on them) and they will never be added to the linear context.

In a broader sense, the analysis of proof theoretic properties of LL must guide the design of more advanced/efficient proof search procedures. For instance, invertibility and permutability of rules lead to systems that reduce the non-determinism [10, 1] as already mentioned. Moreover, the resource management problem has been tackled by delaying, as much as possible, the decision on how to split the linear context [5, 16]. On the other hand, systems for forward reasoning have been also explored in the context of LL (see e.g., [20, 28, 29, 7]), usually showing a better performance that those based on backward reasoning [28, 7]. Since our goal with this work is to present an initial library for LL and not to test particular theorem provers or automated reasoning techniques, we will not focus our attention on these issues. We just note that no silver-bullet will work in all the cases since, as it is well known, propositional linear logic is undecidable [15].

2.2 Translations and Decorations

A naive approach for building a set of test formulas for LL based provers would be to use one of the well known translations of intuitionistic (or classical) formulas into LL. Since there are several ways of translating a formula from IL to LL, we asked ourselves which one would be the best option, if any. Each translation characterizes a different linear view of intuitionistic formulas and it is interesting and relevant to establish a comparison between them. We analyze four: a multiplicative translation, the original Girard's translation, Girard's positive translation and Miller and Liang's $0/1$ translation.

The multiplicative translation trivially substitutes the intuitionistic connectives by their multiplicative linear version

$$\begin{array}{rclrclrcl} (p)^m & = & p & (A \to B)^m & = & A^m \multimap B^m & (A \wedge B)^m & = & A^m \otimes B^m \\ (t)^m & = & 1 & (A \vee B)^m & = & A^m \parr B^m & (f)^m & = & \bot \end{array}$$

Translation of sequents is given by $(\Gamma \vdash A)^m = \Gamma^m \vdash A^m$. Observe that this translation *does not* preserve provability: for instance, diagonals $A \otimes A \to A$ exist in IL, but not in LL.

Girard's translation [11], also known as *call-by-name* translation, is the most well known translation of IL into LL, defined as

$$\begin{array}{rclrclrcl} (p)^g & = & p & (A \to B)^g & = & !A^g \multimap B^g & (A \wedge B)^g & = & A^g \,\&\, B^g \\ (t)^g & = & \top & (A \vee B)^g & = & !A^g \oplus !B^g & (f)^g & = & 0 \end{array}$$

Sequents are translated as $(\Gamma \vdash A)^g = !\Gamma^g \vdash A^g$. Girard's translation preserves provability but is not a *decoration* in the sense of [9], namely, a proof of A in IL is transformed into a proof of A^g in LL which is not isomorphic to the original one.

Girard proposed in the same paper [11] another translation, known as *call-by-value* translation. Henceforth, we will call this translation *positive*, since IL formulas become positive LL formulas.

$$\begin{array}{rclrclrcl} (p)^p & = & !\,p & (A \to B)^p & = & !(A^p \multimap B^p) & (A \wedge B)^p & = & A^p \otimes B^p \\ (t)^p & = & 1 & (A \vee B)^p & = & A^p \oplus B^p & (f)^p & = & 0 \end{array}$$

Sequents are translated as $(\Gamma \vdash A)^p = \Gamma^p \vdash A^p$. It is easy to see that the positive translation is a decoration: proofs of A^p in LL are isomorphic to proofs of A in IL (see [9] for details).

Another interesting translation is the $0/1$ translation [14], which distinguishes the polarity of formulas in a sequent.

$$\begin{array}{rclrclrcl} (p)^0 & = & p & (A \to B)^0 & = & !A^1 \multimap !B^0 & (A \wedge B)^0 & = & !A^0 \,\&\, !B^0 \\ (t)^0 & = & \top & (A \vee B)^0 & = & !A^0 \oplus !B^0 & (f)^0 & = & 0 \end{array}$$

$$\begin{array}{rclrclrcl} (p)^1 & = & p & (A \to B)^1 & = & !(!A^0 \multimap B^1) & (A \wedge B)^1 & = & !(A^1 \,\&\, B^1) \\ (t)^1 & = & 1 & (A \vee B)^1 & = & !A^1 \oplus !B^1 & (f)^1 & = & 0 \end{array}$$

The translation of sequents is given by $(\Gamma \vdash A)^{0/1} = !\Gamma^0 \vdash A^1$. Using this translation, *focused proofs* in ILLF are in bijective correspondence with proofs in IL. In a loose sense, this can be considered a decoration if the isomorphism is interpreted "modulo focusing". In the focusing context, this is referred to as *adequacy on the level of derivations* [21].

Basically these four translations differ on their use of bangs and polarization of atoms. The multiplicative translation introduces no bangs; Girard's translation forces backchaining proofs; the positive translation sets the global preference to be forward-chaining and atoms have positive polarity; the $0/1$ translation is asymmetric, and it does not impose restrictions on atoms. Of course, less bangs implies shorter proofs. For instance, consider the IL sequent $A \to B, B \to C \vdash A \to C$ and the corresponding proofs using the four translations in Figure 3.

3 The ILLTP Library

The ILLTP library is composed of problems from three sources: Kleene's "Introduction to Metamathematics", ILTP (a library of problems for intuitionistic theorem provers), and Petri nets from the Model Checking Contest [13]. We use a syntax similar to TPTP [27][14], where axioms (formulas on the left side of the sequent) and conjectures (the formula on the right side of the sequent) are specified as `fof(name, axiom, F)` and `fof(name, conjecture, F)` respectively. The name chosen is not important. The denominations `axiom` and `conjecture` are fixed. `F` is a linear logic formula built from the connectives: `*`, `&`, `+`, `|`, `-o`, `!`, `?`, corresponding to $\otimes$, $\&$, $\oplus$, $\parr$, $\multimap$, !, ?, respectively[15].

3.1 Kleene's problems

Kleenes traditional book "Introduction to Metamathematics" [12] has a collection of interesting intuitionistic theorems. They are rather straightforward, thus they would not be especially useful for testing the efficiency of a prover. Instead, they can be regarded as a minimal set of intuitionistic theorems that a

[13] `https://pnrepository.lip6.fr/mcc/`

[14] `http://www.tptp.org`

[15] Although our library is used here for ILL, the format of our files and translations support LL formulas as well, so that our library can be extended to the classical case as well.

Multiplicative translation

$$\dfrac{\dfrac{}{\cdot : A \vdash A} \quad \dfrac{\dfrac{}{\cdot : B \vdash B} \quad \dfrac{}{\cdot : C \vdash C}}{\cdot : B, B \multimap C \vdash C}\ \multimap}{\dfrac{\cdot : A, A \multimap B, B \multimap C \vdash C}{\cdot : A \multimap B, B \multimap C \vdash A \multimap C}\ \star}\ \multimap$$

Call-by-name

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, !(A) \multimap B, !(B) \multimap C : \cdot \vdash A}}{A, !(A) \multimap B, !(B) \multimap C : \cdot \vdash !(A)}\ ! \quad \dfrac{}{A, !(A) \multimap B, !(B) \multimap C : B \vdash B}\ \multimap}{A, !(A) \multimap B, !(B) \multimap C : !(A) \multimap B \vdash B}\ D_C}{A, !(A) \multimap B, !(B) \multimap C : \cdot \vdash B}}{A, !(A) \multimap B, !(B) \multimap C : \cdot \vdash !(B)}\ ! \quad \dfrac{}{A, !(A) \multimap B, !(B) \multimap C : C \vdash C}\ \multimap}{\dfrac{A, !(A) \multimap B, !(B) \multimap C : !(B) \multimap C \vdash C}{A, !(A) \multimap B, !(B) \multimap C : \cdot \vdash C}\ D_C}{\cdot : !(!(A) \multimap B), !(!(B) \multimap C) \vdash !(A) \multimap C}\ \star$$

Call-by-value

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash A}}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(A)}\ ! \quad \dfrac{}{A, B, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash B}\ \star}{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(B) \vdash B}\ \multimap}{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(A) \multimap !(B) \vdash B}\ D_C}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash B}}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(B)}\ ! \quad \dfrac{\dfrac{}{A, C, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash C}}{A, C, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(C)}\ ! \quad \dfrac{}{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(C) \vdash !(C)}\ \star}{\dfrac{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(B) \multimap !(C) \vdash !(C)}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(C)}\ D_C}\ \multimap}{\dfrac{!(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(A) \multimap !(C)}{!(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(!(A) \multimap !(C))}\ \star}\ !}{\cdot : !(!(A) \multimap !(B)), !(!(B) \multimap !(C)) \vdash !(!(A) \multimap !(C))}\ \star$$

0/1

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash A}}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(A)}\ ! \quad \dfrac{}{A, B, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash B}\ \star}{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(B) \vdash B}\ \multimap}{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(A) \multimap !(B) \vdash B}\ D_C}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash B}}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(B)}\ ! \quad \dfrac{\dfrac{}{A, C, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash C}}{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(C) \vdash C}\ \star}{\dfrac{A, !(A) \multimap !(B), !(B) \multimap !(C) : !(B) \multimap !(C) \vdash C}{A, !(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash C}\ D_C}\ \multimap}{\dfrac{!(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(A) \multimap C}{!(A) \multimap !(B), !(B) \multimap !(C) : \cdot \vdash !(!(A) \multimap C)}\ \star}\ !}{\cdot : !(!(A) \multimap !(B)), !(!(B) \multimap !(C)) \vdash !(!(A) \multimap C)}\ \star$$

Figure 3: Proof of the sequent $A \to B, B \to C \vdash A \to C$ using the four translations. The formulas in blue represent the classical context and the $\star$ rule condenses all the rules in the negative phase.

sound prover should be able to derive. As such, they can be valuable to uncover bugs and unsoundness. Our first goal is to set up a similar set for LL.

The main challenge is to understand how these intuitionistic theorems should be interpreted in LL. Deciding whether to translate intuitionistic disjunction as the multiplicative disjunction $\parr$ of linear logic or the additive disjunction $\oplus$ changes the target system under consideration, thus we prefer to not consider the intuitionistic disjunction to begin with. Hence we will start by considering what we call the *rudimentary fragment of* IL, which is very well-behaved: semantically this fragment corresponds to cartesian closed categories. Figure 4 shows the 61 theorems considered.

Applying each translation defined in Section 2.2 to these 61 sequents gives rise to 244 different ILL sequents. As already noted, provability is not preserved in the multiplicative translation. The reason for that, other than the obvious absence of structural rules in the left context, is that the multiplicative false $\bot$ is relevant, so while $0 \vdash B$ for any B, $A \otimes (A \multimap \bot) \nvdash B$ in LL. The other three translations fix this by systematically adding bangs and additive connectives.

The multiplicative translation of all sequents in Kleene's list is in Appendix A, including the 22 ones that are not provable (indicated by $\nvdash$). In Figure 5 we present an alternative translation for them, using a small number of bangs and/or additives.

3.2 ILTP problems

The problems from the ILTP library for intuitionistic theorem provers [24] were parsed and translated into linear logic using the three provability preserving translations presented in Section 2.2. Due to the number and size of problems, it is unpractical to apply the multiplicative translation and find the unprovable sequents to propose an alternative translation as it was done with the previous set of problems.

The original library contains 274 problems (including theorems and non-theorems), separated into three categories: LCL (logical calculi), SYJ and SYN (syntactic problems that have no obvious interpretation in intuitionistic and classical logic, respectively). Their translation resulted in 822 linear logic problems. Moreover, we also considered the remaining Kleene's theorems in [12] (i.e., those out of the rudimentary fragment) resulting in a total of 1.086 problems in this collection.

3.3 Comparing translations

We specified IL and different fragments of linear logic in rewriting logic (RW, see e.g. [18]) and implemented in Maude[16] a very basic prover for IL as well as for ILLF and LLF. The use of RW leads to a clear separation between deterministic inference rules that can be eagerly applied (as those in the negative phase) and non-deterministic inference rules where backtracking may be needed (as those in the positive phase). Moreover, the minimal distance between the represented logic (IL, ILLF and LLF) and its specification in RW allowed us to quickly implement a good prototypical tool useful for our tests. Although more efficient provers can be built by e.g., including sophisticated heuristics and specialized data structures, our prototypical implementations were enough to compare the different translations.

Using these provers, we generated LaTeX proofs for the intuitionistic problems from Kleene and ILTP, plus for all ILL translations of the sequents, when provable. These results can be found in the PDFs at

`https://github.com/carlosolarte/Linear-Logic-Prover-in-Maude`

in the directory `output`.

[16] `http://maude.cs.uiuc.edu`

1. $\vdash A \to A$
2. $A \to B, B \to C \vdash A \to C$
3. $A \to (B \to C) \vdash B \to (A \to C)$
4. $A \to (B \to C) \vdash A \wedge B \to C$
5. $A \wedge B \to C \vdash A \to (B \to C)$
6. $A \to B \vdash (B \to C) \to (A \to C)$
7. $A \to B \vdash (C \to A) \to (C \to B)$
8. $A \to B \vdash A \wedge C \to B \wedge C$
9. $A \to B \vdash C \wedge A \to C \wedge B$
10. $\neg A \vdash A \to B$
11. $A \vdash \neg A \to B$
12. $B \vdash A \to B$
13. $A \to B \vdash \neg B \to \neg A$
14. $A \to \neg B \vdash \neg\neg B \to \neg A$
15. $A \to B, B \to A \vdash A \leftrightarrow B$
16. $A \leftrightarrow B \vdash A \to B$
17. $A \leftrightarrow B \vdash B \to A$
18. $A \leftrightarrow B, A \vdash B$
19. $A \leftrightarrow B, B \vdash A$
20. $\vdash A \leftrightarrow A$
21. $A \leftrightarrow B \vdash B \leftrightarrow A$
22. $A \leftrightarrow B, B \leftrightarrow C \vdash A \leftrightarrow C$
23. $A \to (B \to C), \neg\neg A, \neg\neg B \vdash \neg\neg C$
24. $\neg\neg(A \to B) \vdash \neg\neg A \to \neg\neg B$
25. $\neg\neg(A \to B), \neg\neg(B \to C) \vdash \neg\neg(A \to C)$
26. $\vdash \neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$
27. $\vdash \neg\neg(A \leftrightarrow B) \leftrightarrow (\neg\neg(A \to B) \wedge \neg\neg(B \to A))$
28. $A \leftrightarrow B \vdash (A \to C) \leftrightarrow (B \to C)$
29. $A \leftrightarrow B \vdash (C \to A) \leftrightarrow (C \to B)$
30. $A \leftrightarrow B \vdash (A \wedge C) \leftrightarrow (B \wedge C)$
31. $A \leftrightarrow B \vdash (C \wedge A) \leftrightarrow (C \wedge B)$
32. $A \leftrightarrow B \vdash \neg A \leftrightarrow \neg B$
33. $\vdash ((A \wedge B) \wedge C) \leftrightarrow (A \wedge (B \wedge C))$
34. $\vdash (A \wedge B) \leftrightarrow (B \wedge A)$
35. $\vdash (A \wedge A) \leftrightarrow A$
36. $A \vdash (A \to B) \leftrightarrow B$
37. $B \vdash (A \to B) \leftrightarrow B$
38. $\neg A \vdash (A \to B) \leftrightarrow \neg A$
39. $\neg B \vdash (A \to B) \leftrightarrow \neg A$
40. $B \vdash (A \wedge B) \leftrightarrow A$
41. $\neg B \vdash (A \wedge B) \leftrightarrow B$
42. $\vdash A \to \neg\neg A$
43. $\vdash \neg\neg\neg A \leftrightarrow \neg A$
44. $\vdash \neg(A \wedge \neg A)$
45. $\vdash \neg(A \leftrightarrow \neg A)$
46. $\vdash \neg\neg(\neg\neg A \to A)$
47. $\vdash (A \wedge (B \wedge \neg B)) \leftrightarrow (B \wedge \neg B)$
48. $\vdash (A \to B) \to \neg(A \wedge \neg B)$
49. $\vdash (A \to \neg B) \leftrightarrow (\neg(A \wedge B))$
50. $\vdash (\neg(A \wedge B)) \leftrightarrow (\neg\neg A \to \neg B)$
51. $\neg\neg B \to B \vdash (\neg\neg A \to B) \leftrightarrow (A \to B)$
52. $\neg\neg B \to B \vdash (A \to B) \leftrightarrow (\neg(A \wedge \neg B))$
53. $\vdash (\neg\neg A \to B) \to \neg(A \wedge \neg B)$
54. $\vdash (A \wedge B) \to \neg(A \to \neg B)$
55. $\vdash (A \wedge \neg B) \to \neg(A \to B)$
56. $\vdash \neg\neg A \wedge B \to \neg(A \to \neg B)$
57. $\vdash (\neg\neg A \wedge \neg B) \leftrightarrow \neg(A \to B)$
58. $\vdash \neg(A \to B) \leftrightarrow \neg\neg(A \wedge \neg B)$
59. $\vdash \neg\neg(A \to B) \leftrightarrow \neg(A \wedge \neg B)$
60. $\vdash \neg(A \wedge \neg B) \leftrightarrow (A \to \neg\neg B)$
61. $\vdash (A \to \neg\neg B) \leftrightarrow (\neg\neg A \to \neg\neg B)$

Figure 4: Kleene's theorems collected from [12] (page 113 onwards). Only the $(\to, \wedge)$ fragment is considered. Bi-implication is defined as $A \leftrightarrow B = (A \to B) \wedge (B \to A)$.

10. $A \multimap 0 \vdash A \multimap B$
11. $A \vdash (A \multimap 0) \multimap B$
12. $B \vdash\ !A \multimap B$
16. $(A \multimap B) \otimes\ !(B \multimap A) \vdash A \multimap B$
17. $\ !(A \multimap B) \otimes (B \multimap A) \vdash B \multimap A$
18. $A \multimap\!\circ\ B, A \vdash B \otimes (B \multimap A)$
19. $A \multimap\!\circ\ B, B \vdash A \otimes (A \multimap B)$
26. a. $\vdash ((A \,\&\, B)^{\perp\perp}) \multimap (A^{\perp\perp} \,\&\, B^{\perp\perp})$
    b. $\vdash (A^{\perp\perp} \otimes B^{\perp\perp}) \multimap (A \otimes B)^{\perp\perp}$
27. a. $\vdash (!(A \multimap B) \otimes\ !(B \multimap A))^{\perp\perp} \multimap [(A \multimap B)^{\perp\perp} \,\&\, (B \multimap A)^{\perp\perp}]$
    b. $\vdash (A \multimap B)^{\perp\perp} \otimes (B \multimap A)^{\perp\perp} \multimap (A \multimap\!\circ\ B)^{\perp\perp}$
35. $\vdash (!A \otimes\ !A) \multimap\!\circ\ !A$
36. $A \vdash ((A \multimap B) \multimap B) \otimes (B \multimap (!A \multimap B))$
37. $B \vdash (!(A \multimap B) \multimap B) \otimes (B \multimap (!A \multimap B))$
38. $A^{\perp} \vdash (!(A \multimap B) \multimap A^{\perp}) \otimes ((A \multimap 0) \multimap (A \multimap B))$
39. $B \multimap 0 \vdash (A \multimap B) \multimap\!\circ\ (A \multimap 0)$
40. $B \vdash ((A \otimes\ !B) \multimap A) \otimes (A \multimap (A \otimes B))$
41. $B \multimap 0 \vdash ((!A \otimes B) \multimap B) \otimes (B \multimap (A \otimes B))$
45. $\vdash (!(A \multimap A^{\perp}) \otimes ((!A)^{\perp} \multimap\ !A))^{\perp}$
46. $\vdash (!(!(A^{\perp} \multimap 0)) \multimap A)^{\perp})^{\perp}$
47. $\vdash A \otimes (B \otimes (B \multimap 0)) \multimap\!\circ\ (B \otimes (B \multimap 0))$
57. a. $\vdash (A^{\perp\perp} \otimes B^{\perp}) \multimap (A \multimap B)^{\perp}$
    b. $\vdash (!A \multimap B)^{\perp} \multimap ((A \multimap 0)^{\perp} \,\&\, B^{\perp})$
58. a. $\vdash !((!A \multimap B)^{\perp}) \multimap ((A \otimes B^{\perp}) \multimap 0)^{\perp}$
    b. $\vdash (A \otimes B^{\perp})^{\perp\perp} \multimap (A \multimap B)^{\perp}$
59. a. $\vdash (A \multimap B)^{\perp\perp} \multimap (A \otimes B^{\perp})^{\perp}$
    b. $\vdash ((A \otimes B^{\perp}) \multimap 0) \multimap (!(!A \multimap B)^{\perp})^{\perp}$

Figure 5: Alternative translation for Kleene theorems

	Encoding 0/1				Call-by-Name				Call-by-Value			
	KLE	LCL	SYJ	SYN	KLE	LCL	SYJ	SYN	KLE	LCL	SYJ	SYN
Num. of Problems	88	2	248	20	88	2	248	20	88	2	248	20
Unsolved (timeouts)	11	2	209	9	12	2	203	9	9	2	209	9
Solved (Theorems)	76	0	39	7	76	0	45	6	76	0	39	7
Non-Theorems	1	0	0	4	0	0	0	5	3	0	0	0
Min Time	17	-	18	16	17	-	17	16	16	-	17	17
Avg Time	798.78	-	34039.10	105.81	250.43	-	20319.98	96.27	494.49	-	33884.36	77.18
Max Time	41548	-	276886	458	4186	-	174778	383	24427	-	265778	321

Table 1: Comparison of translations. Times are measured in miliseconds. Test run on a QEMU Virtual CPU, 2GHz, 64 bits, 6GB of RAM running Ubuntu. Timeout set to 5 minutes.

The proofs are grouped by intuitionistic problem, so one can compare the shape and size of the original intuitionistic proofs and their linear versions resulted from each translation. A summary of the results is presented in Table 1.

3.4 Petri nets problems

Petri nets (see e.g. [25]) are a model for distributed systems that generalize automata and hence also transition systems. Local states in the net are called *places* (denoted graphically as a circle) and the global state is a multiset of local states called a *marking*. A certain number n of copies of a local state s in a marking M is denoted as n tokens (graphically bullets) inside the circle s. Transitions produce and consume tokens, representing transformations in the local states.

Abstractly, we can see a Petri net as a multiset rewrite system and thus, there is a natural connection with the multiplicative fragment of linear logic [3]. More precisely, a transition system over the set of places S is a set of rewrite rules of the form $r : \bullet t \rhd t\bullet$ where $\bullet t$ and $t\bullet$ are multisets on S, usually called preset and postset. As expected, the rule r is enabled in a marking (global state) M if $\bullet t$ is a multisubset of M. Then, M reduces to M', notation $M \rhd M'$, where $M' = M \setminus \bullet t \uplus t\bullet$. Let $\rhd^*$ be the reflexive and transitive closure of $\rhd$. We say that M' is reachable from M if $M \rhd^* M'$.

Let $s \in S$ be a place and M, M' be a multisets on S. Then, we can encode places, markings and rules

as follows [3]:

$$\begin{array}{lcllcl} (s)^{pt} & = & \mathtt{s} & (\{\})^{pt} & = & 1 \\ (\{s\} \uplus M)^{pt} & = & (s)^{pt} \otimes (M)^{pt} & (r : M \rhd M')^{pt} & = & !((M)^{pt} \multimap (M')^{pt}) \end{array}$$

where $\mathtt{s}$ is an atomic proposition. Given a set of transitions $\mathscr{R}$, the reachability problem whether $M \rhd^* M'$ can be encoded as the ILL sequent $\bigotimes_{r \in \mathscr{R}} (r)^{pt} \vdash (M)^{pt} \multimap (M')^{pt}$.

The Petri Nets Repository[17] contains a large collection of Petri nets in PNML (Petri Net Markup Language[18]) syntax, an XML with tags for places and transitions (or "arcs"), and a way to represent the initial marking. We wrote scripts that translate PNML into a simple Maude program that can perform transitions in the net, transforming the markings. This program was used to generate final markings of the Petri nets after 1, 5, 10, 20, 50 and 100 transitions. Each final marking, together with the Petri net specification, is a reachability problem in linear logic, which was written in the ILLTP format (described at the beginning of Section 3) using the translation above. It is worth noticing that some nets are cyclic (deadlock free transition systems), hence it is possible that the marking after n transition can be also obtained after $m < n$ transitions. This procedure allowed us to generate 3.137 new problems.

4 Conclusion

In this work we proposed an initial set of problems as a library for benchmarking linear logic based provers. Starting with the $(\rightarrow, \wedge)$-fragment of Kleene's theorems, we generated 244 different ILL sequents using 4 automatic translations: *multiplicative*, *call-by-name*, *call-by-value* and *0/1*. The first translation is the only one that does not preserve provability. For each of those 22 ILL sequents that are not provable via the multiplicative translation, we proposed an alternative provable sequent with a small set of additives and bangs added. This makes these particular sequents amenable to the use of all the power of focusing theorem provers. In fact, the excess of bangs in formulas tends to neutralize the efficacy of focusing, due to the positive/negative behavior of the exponentials. Thus our initial proposal for a suitable benchmark for ILL has 271 formulas, testing aspects like provability and focusing.

It is worth noticing that (1) we include $\bot$ in the grammar of ILL; (2) all the sequents of our collection can also serve as tests in CLL. The decision in (1) was motivated by the fact that the resulting sequents fall into the multiplicative fragment of ILL. However, observe that one could clearly exchange $\bot$ for 0 in the multiplicative translation (that would not be multiplicative anymore) and still obtain a significant set of 23 formulas not provable via this new translation.

We then moved in the direction of increasing the amount of formulas in our library, still using the translation approach. For that, we took all the 274 formulas from the ILTP Library and applied the three sound translations from IL to LL considered in this work. At this point, our library had already 1.093 formulas to be tested. For an initial experiment with this first proposed set of sequents, we implemented a focused-based prover for ILL. This allowed us to compare well known LL translations for IL formulas, not only considering the time needed for proving a sequent: our results give a comprehensive collection of proofs where it is possible to visualize the different shapes (and sizes) of such proofs.

Finally, following Frank Pfenning's suggestion, we have generated automatically several reachability problems from the large collection of Petri nets. This not only moves us out of intuitionistic logic

[17] https://pnrepository.lip6.fr/

[18] http://pnml.lip6.fr/

examples, but it also considers a serious amount of problems that are actually used for testing Petri nets model checkers.

It should be noted that these are only the first steps on the direction of benchmarking linear logic. For future work, we intend to test different provers already available online and to propose a large collection of formulas dedicated to classical linear logic CLL.

References

[1] Jean-Marc Andreoli (1992): *Logic Programming with Focusing Proofs in Linear Logic*. *J. Log. Comput.* 2(3), pp. 297–347, doi:10.1093/logcom/2.3.297.

[2] Torben Braüner & Valeria de Paiva (1996): *Cut-Elimination for Full Intuitionistic Linear Logic*. Technical Report, BRICS Report Series.

[3] Iliano Cervesato (1994): *Petri Nets as Multiset Rewriting Systems in a Linear Framework*. Technical Report, Dipartimento di Informatica. Universitá di Torino.

[4] Iliano Cervesato (1995): *Petri Nets and Linear Logic: a Case Study for Logic Programming*. In María Alpuente & Maria I. Sessa, editors: *1995 Joint Conference on Declarative Programming, GULP-PRODE'95, Marina di Vietri, Italy, September 11-14, 1995*, pp. 313–320.

[5] Iliano Cervesato, Joshua S. Hodas & Frank Pfenning (2000): *Efficient resource management for linear logic proof search*. *Theor. Comput. Sci.* 232(1-2), pp. 133–163, doi:10.1016/S0304-3975(99)00173-5.

[6] Iliano Cervesato & Frank Pfenning (2002): *A Linear Logical Framework*. *Information & Computation* 179(1), pp. 19–75, doi:10.1006/inco.2001.2951.

[7] Kaustuv Chaudhuri & Frank Pfenning (2005): *A Focusing Inverse Method Theorem Prover for First-Order Linear Logic*. In Robert Nieuwenhuis, editor: *Automated Deduction - CADE-20*, *LNCS* 3632, Springer, pp. 69–83, doi:10.1007/11532231_6.

[8] Kaustuv Chaudhuri & Giselle Reis (2015): *An Adequate Compositional Encoding of Bigraph Structure in Linear Logic with Subexponentials*. In: *LPAR-20*, pp. 146–161, doi:10.1007/978-3-662-48899-7_11.

[9] Vincent Danos, Jean-Baptiste Joinet & Harold Schellinx (1995): *On the linear decoration of intuitionistic derivations*. *Arch. Math. Log.* 33(6), pp. 387–412, doi:10.1007/BF02390456.

[10] Didier Galmiche & Guy Perrier (1994): *Foundations of Proof Search Strategies Design in Linear Logic*. In Anil Nerode & Yuri V. Matiyasevich, editors: *Proc. of LFCS'94*, *LNCS* 813, Springer, pp. 101–113, doi:10.1007/3-540-58140-5_11.

[11] Jean-Yves Girard (1987): *Linear Logic*. *Theoretical Computer Science* 50, pp. 1–102, doi:10.1016/0304-3975(87)90045-4.

[12] S. Kleene (1952): *Introduction to Metamathematics*. Van Nostrand, New York.

[13] Olivier Laurent (2018): *Around Classical and Intuitionistic Linear Logics*. In Anuj Dawar & Erich Grädel, editors: *Proceedings of LICS*, ACM, pp. 629–638, doi:10.1145/3209108.3209132.

[14] Chuck Liang & Dale Miller (2009): *Focusing and polarization in linear, intuitionistic, and classical logics*. *Theor. Comput. Sci.* 410(46), pp. 4747–4768, doi:10.1016/j.tcs.2009.07.041.

[15] Patrick Lincoln, John C. Mitchell, Andre Scedrov & Natarajan Shankar (1992): *Decision Problems for Propositional Linear Logic*. *Annals of Pure and Applied Logic* 56(1-3), pp. 239–311, doi:10.1016/0168-0072(92)90075-B.

[16] Pablo López & Ernesto Pimentel (1999): *Resource Management in Linear Logic Search Revisited*. In Harald Ganzinger, David A. McAllester & Andrei Voronkov, editors: *LPAR'99*, *LNCS* 1705, Springer, pp. 304–319, doi:10.1007/3-540-48242-3_19.

[17] Narciso Martí-Oliet & José Meseguer (1989): *From Petri Nets to Linear Logic*. In David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts & Axel Poigné, editors: *Category Theory and Computer Science*, *LNCS* 389, Springer, pp. 313–340, doi:10.1007/BFb0018359.

[18] José Meseguer (2012): *Twenty years of rewriting logic*. *J. Log. Algebr. Program.* 81(7-8), pp. 721–781, doi:10.1016/j.jlap.2012.06.003.

[19] Dale Miller & Elaine Pimentel (2013): *A formal framework for specifying sequent calculus proof systems*. *Theor. Comput. Sci.* 474, pp. 98–116, doi:10.1016/j.tcs.2012.12.008.

[20] Grigori Mints (1993): *Resolution calculus for the first order linear logic*. *Journal of Logic, Language and Information* 2(1), pp. 59–83, doi:10.1007/BF01051768.

[21] Vivek Nigam & Dale Miller (2010): *A Framework for Proof Systems*. *Journal of Automated Reasoning* 45(2), pp. 157–188, doi:10.1007/s10817-010-9182-1.

[22] Carlos Olarte & Elaine Pimentel (2017): *On concurrent behaviors and focusing in linear logic*. *Theor. Comput. Sci.* 685, pp. 46–64, doi:10.1016/j.tcs.2016.08.026.

[23] Thomas Raths & Jens Otten (2012): *The QMLTP problem library for first-order modal logics*. In: *International Joint Conference on Automated Reasoning*, pp. 454–461, doi:10.1007/978-3-642-31365-3_35.

[24] Thomas Raths, Jens Otten & Christoph Kreitz (2007): *The ILTP problem library for intuitionistic logic*. *Journal of Automated Reasoning* 38(1-3), pp. 261–271, doi:10.1007/s10817-006-9060-z.

[25] Wolfgang Reisig (1985): *Petri Nets: An Introduction*. *EATCS Monographs on Theoretical Computer Science* 4, Springer, doi:10.1007/978-3-642-69968-9.

[26] Harold Schellinx (1991): *Some Syntactical Observations on Linear Logic*. *J. Log. Comput.* 1(4), pp. 537–559, doi:10.1093/logcom/1.4.537.

[27] G. Sutcliffe (2017): *The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0*. *Journal of Automated Reasoning* 59(4), pp. 483–502, doi:10.1007/s10817-017-9407-7.

[28] Tanel Tammet (1994): *Proof Sytrategies in Linear Logic*. *J. Autom. Reasoning* 12(3), pp. 273–304, doi:10.1007/BF00885763.

[29] Tanel Tammet (1997): *Resolution, Inverse Method and the Sequent Calculus*. In Georg Gottlob, Alexander Leitsch & Daniele Mundici, editors: *KGC'97*, *LNCS* 1289, Springer, pp. 65–83, doi:10.1007/3-540-63385-5_33.

[30] Anne S. Troelstra (1992): *Lectures on Linear Logic*. CSLI Lecture Notes 29, Center for the Study of Language and Information, Stanford, California.

[31] Max Wisniewski, Alexander Steen & Christoph Benzmüller (2016): *TPTP and Beyond: Representation of Quantified Non-Classical Logics*. In Christoph Benzmüller & Jens Otten, editors: *Proceedings of ARQNL@IJCAR*, *CEUR Workshop Proceedings* 1770, CEUR-WS.org, pp. 51–65.

A Multiplicative translation of Kleene's list

1. $\vdash A \multimap A$ (identity)
2. $A \multimap B, B \multimap C \vdash A \multimap C$ (transitivity of implication)
3. $A \multimap (B \multimap C) \vdash B \multimap (A \multimap C)$ (exchange of premises)
4. $A \multimap (B \multimap C) \vdash A \otimes B \multimap C$ (uncurrying)
5. $A \otimes B \multimap C \vdash A \multimap (B \multimap C)$ (currying)
6. $A \multimap B \vdash (B \multimap C) \multimap (A \multimap C)$ (precomposing maps)
7. $A \multimap B \vdash (C \multimap A) \multimap (C \multimap B)$ (post-composing maps)
8. $A \multimap B \vdash A \otimes C \multimap B \otimes C$ (tensor is a bifunctor)
9. $A \multimap B \vdash C \otimes A \multimap C \otimes B$ (tensor is a bifunctor)
10. $A^\perp \nvdash A \multimap B$
11. $A \nvdash A^\perp \multimap B$
12. $B \nvdash A \multimap B$ but (projection is non-linear)
13. $A \multimap B \vdash B^\perp \multimap A^\perp$ (linear negation is contravariant)
14. $A \multimap B^\perp \vdash B^\perp \multimap A^\perp$
15. $A \multimap B, B \multimap A \vdash A \circ\!\!-\!\!\circ B$
16. $A \circ\!\!-\!\!\circ B \nvdash A \multimap B$ (cannot throw away $B \multimap A$)
17. $A \circ\!\!-\!\!\circ B \nvdash B \multimap A$ (cannot throw away $A \multimap B$)
18. $A \circ\!\!-\!\!\circ B, A \nvdash B$
19. $A \circ\!\!-\!\!\circ B, B \nvdash A$
20. $\vdash A \circ\!\!-\!\!\circ A$
21. $A \circ\!\!-\!\!\circ B \vdash B \circ\!\!-\!\!\circ A$
22. $A \circ\!\!-\!\!\circ B, B \circ\!\!-\!\!\circ C \vdash A \circ\!\!-\!\!\circ C$
23. $A \multimap (B \multimap C), A^{\perp\perp}, B^{\perp\perp} \vdash C^{\perp\perp}$
24. $(A \multimap B)^{\perp\perp} \vdash A^{\perp\perp} \multimap B^{\perp\perp}$ (double negation is a functor)
25. $(A \multimap B)^{\perp\perp}, (B \multimap C)^{\perp\perp} \vdash (A \multimap C)^{\perp\perp}$
26. $\nvdash (A \otimes B)^{\perp\perp} \circ\!\!-\!\!\circ A^{\perp\perp} \otimes B^{\perp\perp}$
27. $\nvdash (A \circ\!\!-\!\!\circ B)^{\perp\perp} \circ\!\!-\!\!\circ (A \multimap B)^{\perp\perp} \otimes (B \multimap A)^{\perp\perp}$
28. $A \circ\!\!-\!\!\circ B \vdash (A \multimap C) \circ\!\!-\!\!\circ (B \multimap C)$
29. $A \circ\!\!-\!\!\circ B \vdash (C \multimap A) \circ\!\!-\!\!\circ (C \multimap B)$
30. $A \circ\!\!-\!\!\circ B \vdash (A \otimes C) \circ\!\!-\!\!\circ (B \otimes C)$
31. $A \circ\!\!-\!\!\circ B \vdash (C \otimes A) \circ\!\!-\!\!\circ (C \otimes B)$
32. $A \circ\!\!-\!\!\circ B \vdash A^\perp \circ\!\!-\!\!\circ B^\perp$
33. $\vdash ((A \otimes B) \otimes C) \circ\!\!-\!\!\circ (A \otimes (B \otimes C))$.
34. $\vdash A \otimes B \circ\!\!-\!\!\circ B \otimes A$
35. $\nvdash A \otimes A \circ\!\!-\!\!\circ A$ ($\otimes$ is not idempotent)
36. $A \nvdash (A \multimap B) \circ\!\!-\!\!\circ B$
37. $B \nvdash (A \multimap B) \circ\!\!-\!\!\circ B$
38. $A^\perp \nvdash (A \multimap B) \circ\!\!-\!\!\circ A^\perp$
39. $B^\perp \nvdash (A \multimap B) \circ\!\!-\!\!\circ A^\perp$
40. $B \nvdash (A \otimes B) \circ\!\!-\!\!\circ A$
41. $B^\perp \nvdash ((!A \otimes B) \multimap B) \otimes (B \multimap (A \otimes B))$
42. $\vdash A \multimap A^{\perp\perp}$
43. $\vdash A^{\perp\perp\perp} \circ\!\!-\!\!\circ A^\perp$
44. $\vdash (A \otimes A^\perp)^\perp$
45. $\nvdash (A \circ\!\!-\!\!\circ A^\perp)^\perp$
46. $\nvdash (((A \multimap 0) \multimap 0) \multimap A)^{\perp\perp}$
47. $\nvdash A \otimes (B \otimes B^\perp) \circ\!\!-\!\!\circ (B \otimes B^\perp)$
48. $\vdash (A \multimap B) \multimap (A \otimes B^\perp)^\perp$
49. $\vdash (A \multimap B^\perp) \circ\!\!-\!\!\circ (A \otimes B)^\perp$
50. $\vdash (A \otimes B)^\perp \circ\!\!-\!\!\circ (A^{\perp\perp} \multimap B^\perp)$
51. $B^{\perp\perp} \multimap B \vdash (A^{\perp\perp} \multimap B) \circ\!\!-\!\!\circ (A \multimap B)$
52. $B^{\perp\perp} \multimap B \vdash (A \multimap B) \circ\!\!-\!\!\circ (A \otimes B^\perp)^\perp$
53. $\vdash (A^{\perp\perp} \multimap B) \multimap (A \otimes B^\perp)^\perp$
54. $\vdash A \otimes B \multimap (A \multimap B^\perp)^\perp$
55. $\vdash A \otimes B^\perp \multimap (A \multimap B)^\perp$
56. $\vdash (A^{\perp\perp} \otimes B) \multimap (A \multimap B^\perp)^\perp$
57. $\nvdash (A^{\perp\perp} \otimes B^\perp) \circ\!\!-\!\!\circ (A \multimap B)^\perp$
58. $\nvdash (A \multimap B)^\perp \circ\!\!-\!\!\circ (A \otimes B^\perp)^{\perp\perp}$
59. $\nvdash (A \multimap B)^{\perp\perp} \circ\!\!-\!\!\circ (A \otimes B^\perp)^\perp$
60. $\vdash (A \otimes B^\perp)^\perp \circ\!\!-\!\!\circ (A \multimap B^{\perp\perp})$
61. $\vdash (A \multimap B^{\perp\perp}) \circ\!\!-\!\!\circ (A^{\perp\perp} \multimap B^{\perp\perp})$