A functional correspondence between natural deduction and sequent calculus

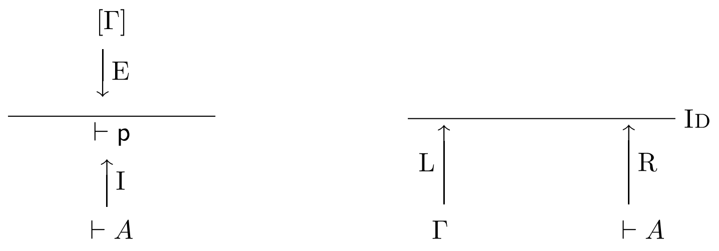
Matthias Puech[1][2]

[1] Univ Paris Diderot, Sorbonne Paris Cité,
PPS, UMR 7126 CNRS, PiR2, INRIA Paris-Rocquencourt, F-75205 Paris, France
[2] Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna, Bologna, Italy

1 Introduction

A typical introductory course to proof theory starts by presenting the two calculi introduced by Gentzen [4]: first *natural deduction*, that shows how to define the meaning of each logical connectives by introductions and eliminations, and then *sequent calculus*, as an equivalent refinement of the latter, making it easier to search for proofs of a proposition. Normal natural deductions admit a *bidirectional* reading: introduction rules are read bottom-up, from the goal sequent until it is fully decomposed in all its atomic parts; elimination rules are read top-down, from the hypothesis down to atoms (Fig. 1a). This obliges the reader (or the proof search procedure) to read (or generate proof trees) in two directions. Sequent calculus is then presented as a response to this cumbersome bidirectionality, by turning all elimination subproofs upside down: introductions are renamed "right rules", upside-down eliminations become "left rules" and operate on formulae in the environment Γ; their meeting point is the identity rule (Fig. 1b).

Standing on the other side of the Curry-Howard looking glass, it is well-known since Herbelin [5] that (a restriction of) intuitionistic sequent calculus can be viewed as a type system for the $\overline{\lambda}$-*calculus*, a language in which consecutive eliminations are reversed wrt. the usual λ-calculus. A $\overline{\lambda}$-term is closer to an abstract machine state than a λ-term. In the minimal implicative fragment, Espìrito Santo explains their syntactic difference by the "associativity" of application [7]: the λ-calculus' is left-associative $(\ldots((M_1\,M_2)\,M_3)\,\ldots)\,M_n$, whereas the $\overline{\lambda}$-calculus' is right-associative $M_1 \star (M_2 :: (M_3 :: (\ldots :: M_n)\ldots))$, allowing a useful direct access to the *head* of an application spine. For this reason, it is sometimes referred to as a *spine calculus* [1]. The restriction of sequent calculus in question, known as LJT, can be raised [7]; it also admits a dual, known as LJQ [3], and can be extended to classical logic [2] and full predicate logic [6]. In all cases, the respective authors show that translating a natural deduction-style



(a) Natural deduction (b) Sequent calculus

Figure 1: From natural deductions to sequent calculus proofs

proof into a sequent calculus-style one is a matter of reversing the "associativity" of consecutive eliminations.

We show here a way to systematically derive, not a particular proof, but the *inference system* of intuitionistic sequent calculus itself from the rules of intuitionistic natural deduction, and moreover by means of off-the-shelf tools from functional programming language compilation: take natural deduction, presented *in the metalanguage* as the bidirectional type checker for λ-terms, turn it into an equivalent two-passes, first-order tail-recursive program suitable for a low-level implementation and you will get a type checker for $\overline{\lambda}$-terms. In other words, and rewriting history, if Gentzen had been a functional programmer, he could have invented the sequent calculus by *compiling* the calculus of natural deductions.

2 A functional correspondence

- partial *CPS-transformation*, translating recursive calls corresponding to eliminations into higher-order, tail-recursive calls,
- *defunctionalization*, turning this higher-order program into a first-order one and introducing the type of *reversed* eliminations,
- *extrusion*, which isolates the pass that reverses an input λ-term into a $\overline{\lambda}$-term from the pass doing the actual type checking.

References

[1] I. Cervesato and F. Pfenning. Linear higher-order pre-unification. In *Logic in Computer Science, 1997. LICS'97. Proceedings., 12th Annual IEEE Symposium on*, pages 422–433. IEEE, 1997.

[2] P.L. Curien and H. Herbelin. The duality of computation. In *ACM sigplan notices*, volume 35, pages 233–243. ACM, 2000.

[3] V. Danos, J.B. Joinet, and H. Schellinx. LKQ and LKT: sequent calculi for second order logic based upon dual linear decompositions of classical implication. *Advances in Linear Logic*, 222:211–224, 1995.

[4] G. Gentzen. Untersuchungen über das logische schließen. i. *Mathematische Zeitschrift*, 39(1):176–210, 1935.

[5] H. Herbelin. A λ-calculus structure isomorphic to gentzen-style sequent calculus structure. In *Computer Science Logic*, pages 61–75. Springer, 1995.

[6] H. Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de lambda-termes et comme calcul de stratégies gagnantes*. PhD thesis, Université Paris-Diderot-Paris VII, 1995.

[7] José Espírito Santo. Completing herbelin's programme. In Simona Ronchi Della Rocca, editor, *TLCA*, volume 4583 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2007.