Focusing proofs and logics for models of computation

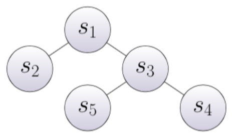
Aleksandra Samonek (UCLouvain)

Keywords: proof theory, linear logic, focusing, computation, logic programming

Linear logic (LL) is particularly successful in expressing the abstract models of computational processes. The completeness of focusing proofs was first shown for LL by [1] and with the following basic principle:

computation = proof search.

In [1] Andreoli observed that the chronology of the computational process is well expressed using the sequent calculus of G. Gentzen. Gentzen-style sequents can be used to easily formalize the history of execution of a computational process during a certain time interval. A sequent system describes the correct inferences in proofs. This description corresponds to allowed process state transitions. Consider the following graph:



The point s_1 represents the state of the computational process at the beginning of the time interval. It corresponds to the root of the tree, or the conclusion of a sequent. s_2 and s_3 are the nodes of the tree representing the intermediate states of the process, while s_4 and s_5 are the leaves of the tree and represent the resulting states at the end of the time interval or the hypotheses. A state is represented not by an atom (as would be default in classical logic), but as a sequent (a multiset of formulae). Unordered multisets allow concurrent access to formulae of the sequent.

However, the most troublesome issue with such Gentzen-style proof is that they proceed very slowly, due to the number of redundant options which a search function needs to consider before finding an appropriate form of the proof. Indded, proofs in a Gentzen-style sequent calculus for LL (and other logics as well) can be *redundant*, meaning that two proofs can differ syntactically although they are identical up to some irrelevant ordering or simplification of application of inference rules (IRs). Consequently, the search procedure makes (computationally) costly choices which turn out to be *irrelevant*.

Focusing proofs is one of the methods used to reduce this redundancy and consequently, speed up the proof search. More specifically, focusing is a strategy in proof searching in which the searching procedure alternates between two phases:

1. **an inversion phase** (when the invertible inference rules are applied exhaustively) and
2. **a chaining phase** (when a selected formula is decomposed as much as possible using non-invertible rules).

In LL **synchronous connectives** are such that the right-introduction inference rules for those connectives are (generally) not invertible, the opposite for **asynchronous connectives**. In focusing proofs the synchronous/asynchronous classification is extended to atoms. This assignment of positive (synchronous) bias or negative (asynchronous) bias is arbitrary and influences the shape and the number of focused proofs, but not the fact of whether a focused proof for a given formula exists in general. For a variety of logics, LL among them, **focusing is complete** and provides a foundation for **developing logics into programming languages**.

Various focusing proofs methods and results have been developed in proof theory and theoretical computer science. However, so far no implementation of focusing proofs to automated theorem proving has been presented. In particular, [1] showed a first focused proof system for a full logic (LLF), which was complete wrt its logic and tractable. Then [2, 3] used Andreoli's completeness result to design and formalize certain logic programming languages. [4] developed focusing proof systems for classical logic ($LKT/LKQ/LK^{\eta}$). [5] developed LJQ which permits the so called forward-chaining in proofs. [6] used both both forward chaining and backward chaining in proofs for full INT (LKF). [7] showed a modal proof of focalization *via* focalization graphs. Finally, [8]: proposed a method for automatic generation of certain focused proof systems *via* permutation graphs based on [7].

During this talk I will (i.) demostrate **how a logic may be turned into a programming language** and (ii) how it can be used to **design a focused proof system** based on the result in [1] and (iii.) discuss how this method of speeding up proof search relates to methods in computer science for **theorem proving optimization**, like using neural networks for premise selection (*cf.* [9]).

References

[1] J.-M. Andreoli, "Logic programming with focusing proofs in linear logic," *Journal of Logic and Computation*, vol. 2, no. 3, pp. 297–347, 1992.

[2] J.-M. Andreoli and R. Pareschi, "Linear objects:logical processes with built-in inheritance," *New Generation Computing*, vol. 9, no. 3-4, pp. 445–473, 1991.

[3] D. Miller, "A multiple-conclusion specification logic," *Theoretical Computer Science*, vol. 165, no. 1, pp. 201–232, 1996.

[4] V. Danos, J.-B. Joinet, and H. Schellinx, "The structure of exponentials: Uncovering the dynamics of linear logic proofs," in *Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pp. 159–171, Springer, 1993.

[5] 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.

[6] C. Liang and D. Miller, "Focusing and polarization in linear, intuitionistic, and classical logics," *Theoretical Computer Science*, vol. 410, no. 46, pp. 4747–4768, 2009.

[7] D. Miller and A. Saurin, "From proofs to focused proofs: a modular proof of focalization in linear logic," in *International Workshop on Computer Science Logic*, pp. 405–419, Springer, 2007.

[8] V. Nigam, G. Reis, and L. Lima, "Towards the automated generation of focused proof systems," *arXiv preprint arXiv:1511.04177*, 2015.

[9] A. A. Alemi, F. Chollet, G. Irving, C. Szegedy, and J. Urban, eds., *DeepMath - Deep Sequence Models for Premise Selection*, 2016.