Focused proof systems and their automatic generation

Aleksandra Samonek

UCLouvain

aleksandra.samonek@uclouvain.be

Focusing

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).

For a variety of logics **focusing is complete** and provides a foundation for **developing logics into programming languages**.

Focusing in linear logic (LL)

Logic programming

LL is particularly good for providing abstract models of computational processes. The completeness of focusing proofs was first shown for LL by [Andreoli, 1992] and with the following basic principle:

computation = proof search.

Proofs in a Gentzen-style sequent calculus for LL (and other logics as well) can be *redundant*, *i. e.* 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*.

Solution for linear logic [Andreoli, 1992]

Define a subclass of proofs (called **focusing proofs** FP) that is:

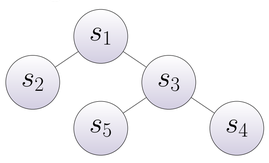
1. **complete** (any formula derivable in LL has a FP) and
2. **tractable** (many irrelevant choices are eliminated when the search procedure is aimed at a FP).

Why Gentzen-style sequents for focused proofs?

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.
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.

A tree-like representation of computation

Consider the following tree:



* 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.
* 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.

Asynchronous *vs* synchronous connectives in LL

In linear logic, **synchronous connectives** are such that the right-introduction IRs for those connectives are (generally) not invertible, the opposite for **asynchronous connectives**.
Synchronous connectives for LL: $\bot$, $\top$, $?$, $\&$, $\forall$, $\Im$.
Asynchronous connectives for LL (de Morgan duals of synchronous connectives):
$\mathbf{1}$, $\mathbf{0}$, $!$, $\oplus$, $\exists$, $\otimes$.

Assigning *bias* to atoms

In FP 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 FPs, but not the fact of whether a FP for a given formula exists in general.

Assigning *bias* to atoms

Example [Liang and Miller, 2009]

Consider the following Horn clause characterization of the Fibonacci series:

$$fib(0,0) \wedge fib(1,1) \wedge \forall n \forall f \forall f' (fib(n,f) \supset fib(n+1,f') \supset fib(n+2,f+f'))$$

We want to search for a FP of $fib(n, f_n)$.

* If we assign a negative bias to all atomic formulae, there exists only one FP of $fib(n, f_n)$. Moreover, the proof is "backwards chaining" and of exponential size.
* If we assign a positive bias to all atoms, there exists an infinite number of FPs of $fib(n, f_n)$, all of which are "forward chaining". The smallest of them is of size linear in n.

In **backward chaining** calculi all atoms have a negative bias, while in **forward chaining** calculi, all atoms have a positive bias.

Negative and positive phases in proof search

The following examples of invertible IR are due to [Miller 2014]:

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash A \supset B}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

$$\frac{\Gamma \vdash B[y/x]}{\Gamma \vdash \forall x.B}$$

Invertible IR in FP-search are applied exhaustively and in any order. This constitutes **the negative phase** of proof search, where formulae are interpreted as *processes* in a given *environment*. During a negative phase of proof search such formulae *evolve* without any communication with the environment (**asynchronously**, hence the naming of rules and connectives).

However, some IR are not generally invertible, like the following:

$$\frac{\Gamma \vdash B[t/x]}{\Gamma \vdash \exists x.B}$$

$$\frac{\Gamma_1 \vdash A \quad \Gamma 2 \vdash B}{\Gamma 1, \Gamma_2 \vdash A \wedge B}$$

Non-invertible IR are applied in a chain-like manner [Miller et al., 1987, Miller et al., 1989]. The process of applying them constitutes a positive phase of the proof search and some **backtracking and synchronization** with the environment are generally required.

Results obtained so far

– [Andreoli, 1992] showed a first focused proof system for a full logic (LLF), which was complete wrt its logic and tractable.

– [Andreoli and Pareschi, 1991, Miller, 1996] used Andreoli's completeness result to design and formalize certain logic programming languages.

– [Danos et al., 1993] developed focusing proof systems for classical logic ($LKT/LKQ/LK^{\text{îû}}$).

– [Herbelin, 1995] developed LJQ which permits forward-chaining in proofs.

– [Dyckhoff and Lengrand, 2007]: developed LJQ extension (LJQ').

– [Jagadeesan et al., 2005]: used both forward chaining and backward chaining in a subset of INT. λRCC allows for two polarities of atoms (agents and constraints are positive and goals are negative) and for mixing both forward chaining and backward chaining in a fragment of INT: forward chaining is used to model constraint propagation and backward chaining is used to model goal-directed search. Note: λRCC is not characterized as a FP system.

– [Liang and Miller, 2009] used both both forward chaining and backward chaining in proofs for full INT (LKF).

– LKF [Miller] provided focusing proofs for full classical logic.

– [Miller and Saurin, 2007] showed a modal proof of focalization *via* focalization graphs.

– [Nigam et al., 2015]: proposed a method for automatic generation of certain focused proof systems *via* permutation graphs based on [Miller and Saurin, 2007].

Can proof system focalization be automated?

Automating focalization means **formulating and proving the completeness** of focused-like proof systems in an automated fashion.
[Nigam et al., 2015] develop the permutation lemmas from [Miller and Saurin, 2007] in order to generalize the completeness proof based on those lemmas for the focused linear logic proof system.

Permutation Graphs

Let $\mathcal{S}$ be a commutative sequent calculus proof system in which non-atomic initial and cut rules are admissible and which allows contraction *iff* weakening is allowed.

Definition (Permutability). Let α and β be two IRs in $\mathcal{S}$. We say that α permutes up β ($\alpha \uparrow \beta$) if for every $\mathcal{S}$-derivation of a sequent S in which α operates on S and β operates on one or more of premises (but not on auxiliary formulae) of α, there exists another $\mathcal{S}$-derivation of S in which β operates on S and α operates on zero or more premises (but not on auxiliary formulae) of β. In this situation, we say conversely that β permutes down α ($\alpha \downarrow \beta$).

Definition 2 (Permutation graph). Let $\mathfrak{R}$ be the set of IR of $\mathcal{S}$.
We construct the (directed) permutation graph $P_{\mathcal{S}} = (V, E)$ for $\mathcal{S}$ by taking $V = \mathfrak{R}$ and $E = \{(\alpha, \beta) : \alpha \uparrow \beta\}$.

Definition 3 (Permutation cliques). Let $P_{\mathcal{S}}$ be a permutation graph of $\mathcal{S}$. Consider $P_{\mathcal{S}*} = (V*, E*)$, an undirected permutation graph obtained from $P_{\mathcal{S}} = (V, E)$ by taking $V* = V$ and $E* = \{(\alpha, \beta) : (\alpha, \beta) \in E \text{ and } (\beta, \alpha) \in E\}$. Then the permutation cliques of $\mathcal{S}$ are the maximal cliques (sets of vertices where all vertices are pairwise connected by one edge) of $P_{\mathcal{S}*}$.

(!) Note that permutation cliques for graphs have a role similar to that of **equivalence classes** for IR.

Definition 4 (Permutation partition). Let $P_{\mathcal{S}}$ be a permutation graph of $\mathcal{S}$. A permutation partition $\mathscr{P}$ is a partition of $P_{\mathcal{S}}$ such that each component is a complete graph. We will call each component of such partitions a permutation component (inferences in the same component permute over each other).

Definition 5 (Permutation partition hierarchy). Let $P_{\mathcal{S}}$ be a permutation graph of $\mathcal{S}$ and $\mathscr{P} = C_1, \ldots, C_n$ – a permutation partition. We say that $C_i \downarrow C_j$ *iff* for every inference $\alpha_i \in C_i$ and $\alpha_j \in C_j$ we have that $\alpha_i \downarrow \alpha j$, that is, $\alpha j \uparrow \alpha i$ or, equivalently $(\alpha_j, \alpha_i \in P_{\mathcal{S}}$.

Generating a focused proof system $\mathcal{S}^F$

Following [Nigam et al., 2015] we will now demonstrate a derivation of a focused proof system $\mathcal{S}^F$ from the permutation partitions of a given proof system $\mathcal{S}$ and enlist the conditions necessary for this derivation.

Definition 6 (Focusable permutation partition). Let $\mathcal{S}$ be a sequent calculus proof system and let $C_1, \ldots, C_n$ be a permutation partition of the IR in $\mathcal{S}$. We say that permutation partition is focusable if the following conditions are satisfied:

* $n = 2$ and $C_1 \downarrow C_2$;
* every rule in component C_2 has at most one auxiliary formula in each premise;
* every non-unary rule in component C_2 splits the context among the premises, meaning that no implicit copying of context formulae is allowed on branching rules.

We call C_1 the negative component and C_2 – the positive component and classify formula occurrences in a proof as negative and positive according to their introduction rules.

Definition 7 (Focused proof system). Let $\mathcal{S}$ be a sequent calculus proof system and $C_1 \downarrow C_2$ a focusable permutation partition of the rules in $\mathcal{S}$. Then we can define the focused system $\mathcal{S}^F$ in the following way:

Sequents. Sequents. sequents are of the shape $\Gamma, \Gamma' \vdash^p \Delta, \Delta'$, where $p \in \{+, -, 0\}$ indicates a positive, negative and neutral polarity sequents respectively. We will call Γ' and Δ' the active contexts.

Inference Rules. For each rule α in $\mathcal{S}$ belonging to the negative (positive) component, $\mathcal{S}^F$ will have a rule α with conclusion and premises being negative (positive) sequents and main and auxiliary formulae occurring in the active contexts.

Structural rules. The connection between the phases is done *via* the following structural rules:

* Selection rules move a formula F into the active context.
  If F is negative, then $p = -$.
  If F is positive, then there is no negative $F' \in \Gamma \cup \Delta$ and $p = +$.
* Store rules remove a formula F from the active context if F is negative and $p = +$ or if F is positive and $p = -$.
* The end rule removes the label $p = \{+, -\}$ of a sequent by setting it to 0 if the active contexts are empty.

$$\frac{\Gamma; F \vdash^p \Delta; \cdot}{\Gamma, F; \cdot \vdash^0 \Delta; \cdot}\ sel_l \qquad \frac{\Gamma; \cdot \vdash^p \Delta; F}{\Gamma; \cdot \vdash^0 \Delta, F; \cdot}\ sel_r$$

$$\frac{\Gamma, F; \Lambda \vdash^p \Delta; \Pi}{\Gamma; \Lambda, F \vdash^p \Delta; \Pi}\ st_l \qquad \frac{\Gamma; \Lambda \vdash^p \Delta, F; \Pi}{\Gamma; \Lambda \vdash^p \Delta; \Pi, F}\ st_r \qquad \frac{\Gamma; \cdot \vdash^0 \Delta; \cdot}{\Gamma; \cdot \vdash^p \Delta; \cdot}\ end$$

An $\mathcal{S}^F$-proof is characterized by sequences of inferences labeled with $+$ or $-$, called **phases**. We say that selection rules are responsible for starting a phase and the end rule finishes a phase. Between any two phases there is always a neutral state, denoted by a sequent labeled with 0.

Completeness conjecture

Conjecture 1. All proof systems derived from a focusable permutation partition are sound and complete. That is a sequent $\Gamma \vdash \Delta$ is provable in $\mathcal{S}$ *iff* the sequent $\Gamma; \cdot \vdash^0 \Delta; \cdot$ is provable in $\mathcal{S}^F$.

Literature

[Andreoli, 1992] Andreoli, J.-M. (1992).
Logic programming with focusing proofs in linear logic.
Journal of Logic and Computation, 2(3):297–347.

[Andreoli and Pareschi, 1991] Andreoli, J.-M. and Pareschi, R. (1991).
Linear objects:logical processes with built-in inheritance.
New Generation Computing, 9(3-4):445–473.

[Danos et al., 1993] Danos, V., Joinet, J.-B., and Schellinx, H. (1993).
The structure of exponentials: Uncovering the dynamics of linear logic proofs.
In *Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 159–171. Springer.

[Dyckhoff and Lengrand, 2007] Dyckhoff, R. and Lengrand, S. (2007).
Call-by-value λ-calculus and ljq.
Journal of Logic and Computation, 17(6):1109–1134.

[Herbelin, 1995] Herbelin, H. (1995).
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.

[Jagadeesan et al., 2005] Jagadeesan, R., Nadathur, G., and Saraswat, V. (2005).
Testing concurrent systems: An interpretation of intuitionistic logic.
In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 517–528. Springer.

[Liang and Miller, 2009] Liang, C. and Miller, D. (2009).
Focusing and polarization in linear, intuitionistic, and classical logics.
Theoretical Computer Science, 410(46):4747–4768.

[Miller, 1996] Miller, D. (1996).
A multiple-conclusion specification logic.
Theoretical Computer Science, 165(1):201–232.

[Miller et al., 1989] Miller, D., Nadathur, G., Pfenning, F., and Scedrov, A. (1989).
Uniform proofs as a foundation for logic programming.

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

[Miller et al., 1987] Miller, D. A., Nadathur, G., and Ščedrov, A. (1987).
Hereditary Harrop formulas and uniform proof systems.
University of Pennsylvania, School of Engineering and Applied Science, Department of Computer and Information Science.

[Nigam et al., 2015] Nigam, V., Reis, G., and Lima, L. (2015).
Towards the automated generation of focused proof systems.
arXiv preprint arXiv:1511.04177.