The subformula property in Intuitionistic sequents proof-search

Dominique Larchey-Wendling

LORIA – CNRS
Nancy, France
Overview of the talk

- We discuss proof/counter-model search IPL
- We deal with sequent calculi, old and new
- Presentation on the sub-formula property (SFP)
  - strict SFP, local rules (context untouched)
- Impact of the SFP (termination, complexity, indexation)
- Implementation issues
  - data structures for sequents and strategies
  - constant time rule application
- Transform the rules in the new system LSJ into local rules
Proof-search in the sequent calculus

- Left introduction rule for conjunction in IL (or CL)

\[
\frac{\Gamma, A, B \vdash \Delta}{\Gamma, [A \land B] \vdash \Delta} \quad [\land_L] \quad A \land B = \frac{\wedge}{A \wedge B}
\]

- \(A, B\) (direct) subformulas of the principal formula \([A \land B]\)

- Consequences of the SFP:
  - decreasing complexity: \(\text{size}(A) + \text{size}(B) < \text{size}(A \land B)\)
  - bounded set of formulae occurring in (backward) proof-search
  - guaranteed termination of proof-search (for CL, not IL)
The sub-formula property (SFP)

- Every formula introduced in backward proof-search is a sub-formula of the principal formula.
- The SFP does not ensure termination (Gentzen LJ):

\[
\begin{align*}
\Gamma, A \supset B \vdash A & \quad \Gamma, B \vdash C \\
\hline
\Gamma, [A \supset B] \vdash C & \quad [\supset_L] \\
\end{align*}
\]
The strict sub-formula property (SSFP)

• The principal formula is *removed* and *replaced* by some of its (direct or strict) subformulae, with no duplications, e.g.

\[
\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, [A \lor B] \vdash \Delta} [\lor_L] \quad A \lor B = \begin{array}{c} \lor \\ A \\ B \end{array}
\]

• In this case (e.g. CL), SSFP ensures termination:
  – size of sequents decreases from conclusions to premisses
  – proof-search depth linearly bounded by size of initial sequent
  – \( \mathcal{O}(n \log n) \) space proof-search algorithm
SFP/SSFP not necessary for termination

- LJT, contraction free sequent calculus for IL (Dyckhoff 92)

\[
\frac{\Gamma, B \supset C \vdash A \supset B \quad \Gamma, C \vdash D}{\Gamma, (A \supset B) \supset C \vdash D} \quad [\supset^4_L]
\]

- \( B \supset C \) is not a subformula of \((A \supset B) \supset C\)
- \( \text{size}(B \supset C) + \text{size}(A \supset B) \) is not lower than \( \text{size}((A \supset B) \supset C) \)
- but both \( B \supset C \) and \( A \supset B \) are strictly smaller than \((A \supset B) \supset C\)
- the well-founded multiset ordering ensures termination
Application of SFP: indexation

- Associate a number to each subformula
- Structurally different subformulas should have different indexes
- Structurally identical subformulas can have the same index
- Identical variables should have the same index
- Proof-search on indexes
Recognizing axioms (the naive way)

- Axioms are usually of the form

  \[
  \Gamma, A \vdash \Delta, \quad \text{or} \quad \Gamma \vdash \Delta \quad [\Gamma \cap \Delta \neq \emptyset]
  \]

- Complexity of naive implementation (e.g. lists):

  \[
  \text{size}(\Gamma) \times \text{size}(\Delta) \times \text{size(average formula)}
  \]

- Axioms should be tested at each step of proof-search
  - indeed, they might close/end the proof-search branch

- An efficient implementation of axioms recognition is thus crucial
Recognizing axioms (the indexed way)

- $\Gamma \vdash \Delta$ is indexed, e.g. $\vdash A_2 \land_3 B_4 \supset_1 A_3$
- $\Gamma$ (resp. $\Delta$) associated to a set of indexes (e.g. array of booleans)
- Each time $\Gamma$ (or $\Delta$) is modified, check for axiom (and mark)
- If $\Gamma \vdash \Delta$ not an axiom then $\Gamma - \{A\} \vdash \Delta$ not an axiom
  - $\Gamma, A \vdash \Delta$ axiom iff $A \in \Delta$ (e.g. $\Delta(A) = \text{true}$)
- To recognize axioms, check for the mark in constant time

\[
\begin{align*}
A_3, B_4 & \vdash A_3 & \text{Id because 3} \\
A_3 \land_2 B_4 & \vdash A_3 & \land_L \text{ and mark(3)} \\
\vdash A_2 \land_3 B_4 & \supset_1 A_3 & \supset_R
\end{align*}
\]
How to select the rule to apply?

- In the calculi we consider: select the principal formula

\[ A_1, \ldots, [A_i], \ldots, A_n \vdash B_1, \ldots, B_k \]

- Criteria for proof-search strategies:
  - lh/rh side, position in the list \( A_1, \ldots, A_n \)
  - outmost logical connective, complexity of the formula

- A “bad” choice may lead to failure:

\[
\begin{align*}
\text{fails} & \quad A \vdash A \\
& \quad A \lor B \vdash A \\
& \quad A \lor B \vdash [A \lor B] \\
& \quad \vdash \lor R \\
\end{align*}
\]

\[
\begin{align*}
\text{Id} & \quad A \vdash A \\
& \quad [A \lor B] \vdash \lor R \\
& \quad [A \lor B] \vdash \lor L \\
\end{align*}
\]

\[
\begin{align*}
\text{Id} & \quad B \vdash B \\
& \quad [A \lor B] \vdash \lor R \\
& \quad [A \lor B] \vdash \lor L \\
\end{align*}
\]
Representation and update of sequents

- \( \Gamma \) and \( \Delta \), both as lists and sets of indexes;
- Update in constant time:

\[
\frac{\Gamma \vdash \Delta_l, A_i, \Delta_r \quad \cdots}{\Gamma \vdash \Delta_l, [A_i \land R B_j], \Delta_r} [\land_R]
\]

- Remove the principal formula, insert one or two subformulae
- Beware *non-local rules* in STRIP (Larchey-W. et al. 2001)
  - all formulae (\( \cdot \)) \( \supset \) \( C \) removed when decomposing \((A \supset B) \supset C\)

\[
\frac{\cdots \quad \Gamma, C \vdash G}{\Gamma, [A \supset B] \supset C, D_1 \supset C, \ldots, D_k \supset C \vdash G} [\supset^4_L]
\]
Constant time proof-search step

\[ PS(\Gamma, [A \lor B] \vdash \Delta) = \]

1. replace \( A \lor B \) by \( A \), push \((A \leadsto A \lor B)\)
2. result = \( PS(\Gamma, A \vdash \Delta) \) (recursion)
3. pop \((A \leadsto A \lor B)\), replace \( A \) by \( A \lor B \)
4. if result = fail then return fail
5. replace \( A \lor B \) by \( B \), push \((B \leadsto A \lor B)\)
6. result = \( PS(\Gamma, B \vdash \Delta) \) (recursion)
7. pop \((B \leadsto A \lor B)\), replace \( B \) by \( A \lor B \)
8. return result

\[ \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, [A \lor B] \vdash \Delta} \]
Terminating proof-search for IPL

- From Gentzen (LJ) to Dyckhoff 92 (LJT) and Hudelmaier 93
- Dyckhoff & Pinto 96 (LJT/CRIP), Dyckhoff & Negri 2000
- Formalization: Weich 98 (Coq, extraction)
- Larchey-Wendling et al. 2001 (STRIP)
- Fiorino et al. 2000+ (tableaux variants of LJT)
- One of our longstanding problem: certified STRIP
- A new lead: the new system LSJ with SSFP
- Our contribution: optimize LSJ for indexed proof search
A sequent system for IPL with SSFP

“Contraction-free Linear Depth Sequent Calculi for IPL with the Subformula Property and Minimal Depth Counter-Models”

(Ferrari, Fiorentini and Fiorino, to appear in JAR)

- A new system LSJ with sequents of the form $\Theta \mid \Gamma \vdash \Delta$
- A finite refutation semantics: $T$ refutes $\Theta \mid \Gamma \vdash \Delta$ if

$$T \Vdash_s \Theta \text{ and } T \Vdash \Gamma \text{ and } T \not\Vdash \Delta$$

- Recover semantics for formulae: $T \not\Vdash A$ iff $T$ refutes $\emptyset \mid \emptyset \vdash A$
- A valid sequent has no refutation tree: $\emptyset \mid \emptyset \vdash A$ valid iff $A$ valid
Finite Kripke semantics for IPL

- \( \text{Var} = \) set of propositional variables
- A tree: \( \mathcal{T} = (S_T, [T_1, \ldots, T_k]) \), with \( S_T \subseteq \text{Var} \)
- A Kripke tree = monotonicity for all subtrees: \( S_T \subseteq S_{T_i} \)
- Subtree (\( \leq \)): \( T \leq T \) and \( T' \leq T_i \) implies \( T' \leq T \)
- Strict subtree (\( < \)): \( T' \leq T_i \) implies \( T' < T \)
- Monotonic Kripke semantics: \( \forall T' \leq T, \ T \models A \Rightarrow T' \models A \)

\[
\begin{align*}
\mathcal{T} \models A \supset B & \iff \forall T' \leq T, \ T' \models A \Rightarrow T' \models B \\
\mathcal{T} \models_s A & \iff \forall T' < T, \ T' \models A 
\end{align*}
\]

- This is a sound and complete semantics for IPL
The rules of LSJ (implicational fragment)

- Formulae in $\Theta$ are not active
- But they are activated by rightmost premise of $[\supset L]$ and $[\supset R]$.
- Strict sub-formula property (SSFP), but some rules are not local

\[
\frac{\Theta | \Gamma, A \vdash A, \Delta}{\Theta | \Gamma, A \vdash A, \Delta} \quad [\text{Id}] \quad \frac{\Theta | A, \Gamma \vdash B, \Delta \quad \emptyset | A, \Theta, \Gamma \vdash \Delta}{\Theta | \Gamma \vdash A \supset B, \Delta} \quad [\supset R]
\]

\[
\frac{\Theta | B, \Gamma \vdash \Delta \quad B, \Theta | \Gamma \vdash A, \Delta}{\Theta | A \supset B, \Gamma \vdash \Delta} \quad \frac{B | \Theta, \Gamma \vdash A}{\Theta | A \supset B, \Gamma \vdash \Delta} \quad [\supset L]
\]
Sound and completeness for LSJ

- Soundness for LSJ
  - if $\mathcal{T}$ refutes the conclusion of some rule then there exists $\mathcal{T}' \preceq \mathcal{T}$ that refutes one premisse of the rule
  - axioms have no refutation trees
  - hence no tree refutes a provable sequent
  - also impacts the depth of counter-models

- Completeness for LSJ
  - a dual refutation calculus RJ
  - extract a refutation tree from any (dual) proof in RJ
  - algorithm that builds either a LSJ-proof or (dual) RJ-proof
  - in the spirit of LJT/CRIP (Pinto & Dyckhoff 95)
LSJ rules are not local rules

- Formulas of $\Theta$ are moved in $\Gamma$
- Formulas of $\Delta$ are removed all together
- Hence rules touch the context

\[
\begin{array}{c}
\cdots \cdots \quad B \vdash \Theta, \Gamma \vdash A \\
\hline
\Theta \vdash A \supset B, \Gamma \vdash \Delta
\end{array}
\]

$[\supset_L]

\[
\begin{array}{c}
\cdots \cdots \quad \emptyset \vdash A, \Theta, \Gamma \vdash B \\
\hline
\Theta \vdash \Gamma \vdash A \supset B, \Delta
\end{array}
\]

$[\supset_R]$

- Our solution: refinement of LSJ into an indexed version
How to cope with $\Theta$ and $\Delta$: indexed sequents

- Let $n_1, \ldots, n_r, p_1, \ldots, p_k$ be non-negative integers

- Let $\Sigma = n_1 : A_1, \ldots, n_r : A_r$ and $\Omega = p_1 : B_1, \ldots, p_s : B_s$

- $\Sigma \vdash_n^\rho \Omega$ is an indexed sequent if
  - $n$ and $p$ are non-negative integers and
  - $n_i \leq n + 1$ and $p_j \leq p$ for any $i$, $j$

- Associated LSJ sequent $\Theta | \Gamma \vdash \Delta$ with
  
  $\Theta = \{ A_i \mid n_i = n + 1 \}$  $\Gamma = \{ A_i \mid n_i \leq n \}$  $\Delta = \{ B_j \mid p = p_j \}$

- We propose an indexed sequent calculus associated to LSJ
Indexed LSJ (part one)

\[
\Theta \mid \Gamma, A \vdash A, \Delta \quad \text{[Id]}
\]

\[
i : A, \Sigma \vdash_{n}^{p} \Omega, p : A \quad \text{with } i \leq n
\]
Indexed LSJ (part two)

\[
\begin{align*}
\Theta \mid A, \Gamma \vdash B, \Delta & \quad \emptyset \mid A, \Theta, \Gamma \vdash B \\
\Theta \mid \Gamma \vdash \Delta, A \supset B & \quad [\supset R]
\end{align*}
\]

\[
\begin{align*}
n : A, \Sigma \vdash^p_n \Omega, p : B & \quad n + 1 : A, \Sigma \vdash^{p+1}_{n+1} \Omega, p + 1 : B \\
\Sigma \vdash^p_n \Omega, p : A \supset B
\end{align*}
\]
Indexed LSJ (part three)

\[
\begin{align*}
\frac{\Theta | B, \Gamma \vdash \Delta \quad B, \Theta | \Gamma \vdash A, \Delta \quad B | \Theta, \Gamma \vdash A}{\Theta | A \supset B, \Gamma \vdash \Delta} & \quad \text{[\supset L]} \\
\end{align*}
\]

\[
\begin{align*}
i : B, \Sigma \vdash_p \Omega & \quad n + 1 : B, \Sigma \vdash_p \Omega, p : A & \quad n + 2 : B, \Sigma \vdash_{p+1} \Omega, p + 1 : A \\
\end{align*}
\]

\[
\begin{align*}
i : A \supset B, \Sigma \vdash_p \Omega & \quad \text{with } i \leq n
\end{align*}
\]
Properties of the indexed LSJ sequent calculus

- Has the SSFP, and thus terminates
- Sound and complete for IPL (as LSJ) (also counter-models)
- Local rules: context is preserved by rule application
- Each rule application implies a bounded number of operations
  - one removal, and one or two introductions
  - rules can be applied in constant time
- As with LSJ (unlike STRIP), manageable formalization (Coq)
Conclusion

- A new indexed sequent calculus for IPL based on LSJ
- Well suited for the implementation of proof-search (local SSFP)
- Soundness & completeness proved formally

Perspectives

- A certified indexed proof-search engine for IPL (Coq, extraction)
- Certified compilation of proof-search in IPL, potentially as efficient as STRIP