Lecture Notes on Certification

15-317: Constructive Logic
Frank Pfenning

Lecture 15
Tuesday, March 21, 2023

1 Introduction

We began this lecture by implementing the a decision procedure based on the inversion calculus with loop checking developed in Lecture 15. You can find the corresponding live code at loop.sml. There are no essentially new concepts here, just some live thoughts about the process of translating the rules into code which is difficult to convey in lecture notes.

The conceptual portion of this lecture is concerned with certification. Over the last few decades, decision procedures for provability and theorem provers for the predicate calculus and richer type theories have evolved using ever more effective and efficient methods of implementation. As a result, they have become increasingly hard to prove correct and correspondingly less trustworthy. Especially in theorem proving, this is an intolerable situation since we rely on logic specifically to get guarantees!

The way forward in this dilemma is *certification*. We design a theorem prover so it returns not just a yes-or-no answer, but a *certificate* that can be independently checked. In the simplest case this certificate is a *proof* for which we can have a straightforward proof checker, hopefully much simpler than the prover itself. The situation is more complicated if the prover returns "*no*", because showing that there is no proof is harder than exhibiting a proof. For systems in wide-spread use such as SAT solvers this has been a rich and interesting line of research. In more general systems, this has been the subject of study in *Logical Frameworks* [Pfenning, 2001]. In fact, the origin of the type system of ML (which stands for *metalanguage*) is to provide a small trustworthy core for a theorem prover [Gordon et al., 1978].

In this lecture we examine how to instrument the sequent calculus so it can calculate a proof term for natural deduction which we take as our "gold standard" and which can be checked by a small and simple bidirectional type-checker. Various refinements (like the inversion calculus, the focusing calculus, the contraction-free sequent calculus, and combinations and further refinements thereof) can produce proof objects based on the same ideas. You will explore this in a future homework assignment.

2 Implementing a Decision Procedure based on Inversion

Recall the inversion calculus from Lecture 15.

Antecedents and Succedents

Antecedents, not left invertible, unordered	Γ	$::=$	$A \supset B \mid P \mid \cdot \mid \Gamma_1, \Gamma_2$
Antecedents, ordered	Ω	$::=$	$A \cdot \Omega \mid \epsilon$
Succedent, not right invertible	C	$::=$	$A \vee B \mid \bot \mid P$

Judgments.

Right inversion	$\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} A$
Left inversion	$\Gamma \,;\, \Omega \xrightarrow{\mathsf{L}} C$
Choice	$\Gamma \,;\, \epsilon \xrightarrow{\mathsf{C}} C$

Rules.

Right Inversion.

$$\frac{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} A \quad \Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} B}{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} A \wedge B}\,\wedge R \qquad \frac{\Gamma \,;\, A \cdot \Omega \xrightarrow{\mathsf{R}} B}{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} A \supset B}\,\supset R \qquad \frac{}{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} \top}\,\top R$$

$$\frac{\Gamma \,;\, \Omega \xrightarrow{\mathsf{L}} A \vee B}{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} A \vee B}\,\mathsf{LR} \qquad \frac{\Gamma \,;\, \Omega \xrightarrow{\mathsf{L}} \bot}{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} \bot}\,\mathsf{LR} \qquad \frac{\Gamma \,;\, \Omega \xrightarrow{\mathsf{L}} P}{\Gamma \,;\, \Omega \xrightarrow{\mathsf{R}} P}\,\mathsf{LR}$$

Left Inversion.

$$\frac{\Gamma \,;\, A \cdot B \cdot \Omega \xrightarrow{\mathsf{L}} C}{\Gamma \,;\, (A \wedge B) \cdot \Omega \xrightarrow{\mathsf{L}} C}\,\wedge L \qquad \frac{\Gamma \,;\, A \cdot \Omega \xrightarrow{\mathsf{L}} C \quad \Gamma \,;\, B \cdot \Omega \xrightarrow{\mathsf{L}} C}{\Gamma \,;\, (A \vee B) \cdot \Omega \xrightarrow{\mathsf{L}} C}\,\vee L$$

$$\frac{}{\Gamma \,;\, \bot \cdot \Omega \xrightarrow{\mathsf{L}} C}\,\bot L \qquad \frac{\Gamma \,;\, \Omega \xrightarrow{\mathsf{L}} C}{\Gamma \,;\, \top \cdot \Omega \xrightarrow{\mathsf{L}} C}\,\top L$$

$$\frac{\Gamma, A \supset B \,;\, \Omega \xrightarrow{\mathsf{L}} C}{\Gamma \,;\, (A \supset B) \cdot \Omega \xrightarrow{\mathsf{L}} C}\,\mathsf{LL} \qquad \frac{\Gamma, P \,;\, \Omega \xrightarrow{\mathsf{L}} C}{\Gamma \,;\, P \cdot \Omega \xrightarrow{\mathsf{L}} C}\,\mathsf{LL}$$

$$\frac{\Gamma \,;\, \epsilon \xrightarrow{\mathsf{C}} C}{\Gamma \,;\, \epsilon \xrightarrow{\mathsf{L}} C}\,\mathsf{CL}$$

Choice.

$$\frac{\Gamma\ ;\ \epsilon \xrightarrow{\mathsf{R}} A}{\Gamma\ ;\ \epsilon \xrightarrow{\mathsf{C}} A \vee B}\ \vee R_1 \qquad \frac{\Gamma\ ;\ \epsilon \xrightarrow{\mathsf{R}} B}{\Gamma\ ;\ \epsilon \xrightarrow{\mathsf{C}} A \vee B}\ \vee R_2$$

$$\frac{}{\Gamma, P\ ;\ \epsilon \xrightarrow{\mathsf{C}} P}\ \mathsf{id}^* \qquad \frac{\Gamma, A \supset B\ ;\ \epsilon \xrightarrow{\mathsf{R}} A \quad \Gamma, [A \supset B]\ ;\ B \xrightarrow{\mathsf{R}} C}{\Gamma, A \supset B\ ;\ \epsilon \xrightarrow{\mathsf{C}} C}\ \supset L$$

Loop Checking.

$$\begin{array}{c}(\text{fail, if } \Gamma \subseteq \Gamma' \text{ and } C = C') \\ \Gamma\ ;\ \epsilon \xrightarrow{\mathsf{C}} C \\ \vdots \\ \Gamma'\ ;\ \epsilon \xrightarrow{\mathsf{C}} C'\end{array}$$

First, the basic data types.

```
1  datatype prop =
2           And of prop * prop
3         | Imp of prop * prop
4         | Or of prop * prop
5         | True
6         | False
7         | Var of string
8
9  type ctx = prop list
10 type seq = prop list * prop
11 type memo = seq list
```

We use a type `ctx = prop list` (for *context*) to represent both Γ and Ω. The memo table has to record all choice sequents on the current branch of search, so it is a list of pairs (Γ, C).

We are trying to pick the simplest and not the asymptotically most efficient representation. If we were to apply this decision procedure to large propositions we would want to make various optimizations, such as using some balanced binary search trees for contexts and the memo table. In fact, since we have to compare propositions we'd want to use some dynamic programming techniques to make this efficient.

However, I firmly believe a first implementation should stick as closely as possible to the abstract judgments. This will drastically increase the likelihood that it is correct. Optimizations can be applied later, as their necessity is discovered based on the concrete problem instances under consideration.

We represent each judgment by an ML function. Besides the components of the judgment, it also has to thread through the memo table. We further split the choice judgment into three. The first, `choose`, consults the memo table to see if it should fail, while `chooseR` and `chooseL` represent making a choice on the right or left. Many of these decisions are a

matter of style, not essential. Also, since Ω will always be empty, we simply omit it from the choice functions.

```
1  val invR : memo -> ctx -> ctx -> prop -> bool
2  val invL : memo -> ctx -> ctx -> prop -> bool
3  val choose : memo -> ctx -> prop -> bool
4  val chooseR : memo -> ctx -> prop -> bool
5  val chooseL : memo -> ctx -> prop -> bool
```

At this point you should consult the code on loop.sml. An interesting aspect of the code is that the known invariants on the propositions in certain antecedent and succedents are unknown to ML. So you will get some warnings from the compiler. In an extension of SML with refinement types such as Cidre [Davies, 1997] these warnings can be avoided; here they are just audited and annotated in the source.

3 Proof Terms for Restricted Sequents

For simplicity, we use the restricted sequent calculus because it is syntactically sparser. At the top level it seems straightforward: We have to go from

$$\longrightarrow A$$

to

$$\longrightarrow M : A$$

such that $M : A$ Actually, we want to be able to run a simple *algorithmic* type-checker on the resulting proof term, so instead we look for a term M with

$$\longrightarrow M : A \quad \text{with} \quad M \Leftarrow A$$

where $M \Leftarrow A$ means that M checks against type A (as introduced in Lecture 6). A key question is how we generalize this to a sequent with antecedents. For this, we have to remember how we constructed the sequent calculus: we took the rules of verifications, turning introduction rules into right rules and elimination rules into upside-down left rules.

Let's start with the introduction/right rules.

$$\frac{A\ \mathit{true} \quad B\ \mathit{true}}{A \wedge B\ \mathit{true}}\ \wedge I \qquad \frac{A\uparrow \quad B\uparrow}{A \wedge B\uparrow}\ \wedge I \qquad \frac{M \Leftarrow A \quad N \Leftarrow B}{\langle M, N\rangle \Leftarrow A \wedge B}\ \wedge I \qquad \frac{\Gamma \longrightarrow A \quad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B}\ \wedge R$$

From this picture it is pretty clear that we should assign

$$\frac{\Gamma \longrightarrow M : A \quad \Gamma \longrightarrow M : B}{\Gamma \longrightarrow \langle M, N\rangle : A \wedge B}\ \wedge R$$

Let's play through the correspondences for an elimination rule.

$$\frac{A \wedge B\ \mathit{true}}{A\ \mathit{true}}\ \wedge E_1 \qquad \frac{A \wedge B\downarrow}{A\downarrow}\ \wedge E_1 \qquad \frac{M \Rightarrow A \wedge B}{\mathbf{fst}\, M \Rightarrow A}\ \wedge E_1 \qquad \frac{\Gamma, A, B \longrightarrow C}{\Gamma, A \wedge B \longrightarrow C}\ \wedge L$$

Besides the fact that the elimination is turned upside-down in the left rule, we also note that in the restricted calculus we extract both A and B in the same rule. Forgetting about B for the moment, if we *had* a proof term M for $A \wedge B$ then **fst** M would be the proof term for A. Of course for B we need the second projection, so we have:

$$\frac{\Gamma, \mathbf{fst}\, M : A, \mathbf{snd}\, M : B \longrightarrow P : C}{\Gamma, M : A \wedge B \longrightarrow P : C}\ \wedge L$$

Summarizing what we have learned in the form of a conjecture: Assume we have

$$\overset{\mathcal{D}}{A_1, \ldots, A_n \longrightarrow C} \quad \text{and} \quad \overset{\Delta}{\underset{\mathcal{E}_1}{M_1 \Rightarrow A_1}} \ \ldots \ \overset{\Delta}{\underset{\mathcal{E}_n}{M_n \Rightarrow A_n}}$$

then for some C we can construct an annotated sequent

$$\overset{\mathcal{F}}{M_1 : A_1, \ldots, M_n : A_n \longrightarrow M : C} \quad \text{such that} \quad \overset{\Delta}{\underset{\mathcal{G}}{M \Leftarrow C}}$$

Here Δ stands for a collection of hypotheses $y_k \Rightarrow B_k$. The somewhat unexpected part of this is that the free variables in the M_i and M have to be accounted for in this separate collection of hypotheses Δ.

The annotation algorithm traverses $\mathcal{D}$, so the proof is by rule induction on the structure of $\mathcal{D}$. Despite its apparent syntactic complexity, the construction is not particularly difficult.

Using this as guidance and the relations between natural deductions, verification, bidirectional typing, and sequents, the remaining rules are not difficult to devise. We write $\hat{\Gamma}$ for a collection of antecedent $M_i : A_i$.

Implication.

$$\frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \supset B}\ \supset R \qquad \frac{\hat{\Gamma}, x : A \longrightarrow N : B}{\hat{\Gamma} \longrightarrow \lambda x.\, N : A \supset B}\ \supset R^x$$

We have annotated $\supset R$ with x because x must be fresh, that is, not already occur in the sequent.

$$\frac{\Gamma, A \supset B \longrightarrow A \quad \Gamma, B \longrightarrow C}{\Gamma, A \supset B \longrightarrow C}\ \supset L \qquad \frac{\hat{\Gamma}, M : A \supset B \longrightarrow N : A \quad \hat{\Gamma}, M\, N : B \longrightarrow P : C}{\hat{\Gamma}, M : A \supset B \longrightarrow P : C}\ \supset L$$

In our induction proof we have to apply to the induction hypothesis first on the first premise because that will yield the term N. Only then can we apply the the induction hypothesis to the second premise because only then can we construct $M\, N : B$.

Disjunction.

$$\frac{\Gamma \longrightarrow A}{\Gamma \longrightarrow A \vee B}\ \vee R_1 \qquad \frac{\Gamma \longrightarrow B}{\Gamma \longrightarrow A \vee B}\ \vee R_2$$

$$\frac{\hat{\Gamma} \longrightarrow M : B}{\hat{\Gamma} \longrightarrow \mathbf{inl}\, M : A \vee B}\ \vee R_1 \qquad \frac{\hat{\Gamma} \longrightarrow M : B}{\hat{\Gamma} \longrightarrow \mathbf{inr}\, M : A \vee B}\ \vee R_2$$

The left rules are slight different from before, because the corresponding change in the proof term occurs in the succedent. That's because the proof term is **case**.

$$\frac{\Gamma, A \longrightarrow C \quad \Gamma, B \longrightarrow C}{\Gamma, A \vee B \longrightarrow C}\ \vee L \qquad \frac{\hat{\Gamma}, u : A \longrightarrow N : C \quad \hat{\Gamma}, w : B \longrightarrow P : C}{\hat{\Gamma}, M : A \vee B \longrightarrow \mathbf{case}(M, u.\, N, w.\, P) : C}\ \vee L^{u,w}$$

Truth.

$$\frac{}{\Gamma \longrightarrow \top}\ \top R \qquad \frac{}{\hat{\Gamma} \longrightarrow \langle\,\rangle : \top}\ \top R$$

Falsehood.

$$\frac{}{\Gamma, \bot \longrightarrow C}\ \bot L \qquad \frac{}{\hat{\Gamma}, M : \bot \longrightarrow \mathbf{abort}\, M : C}\ \bot L$$

4 Summary

We have developed a version of the restricted sequent calculus that is annotated with proof terms. These proof terms act as certificates that can be checked independently in case a the given proposition is true.

$$\frac{\hat{\Gamma} \longrightarrow M : A \quad \hat{\Gamma} \longrightarrow M : B}{\hat{\Gamma} \longrightarrow \langle M, N\rangle : A \wedge B}\ \wedge R \qquad \frac{\hat{\Gamma}, \mathbf{fst}\, M : A, \mathbf{snd}\, M : B \longrightarrow P : C}{\hat{\Gamma}, M : A \wedge B \longrightarrow P : C}\ \wedge L$$

$$\frac{\hat{\Gamma}, x : A \longrightarrow N : B}{\hat{\Gamma} \longrightarrow \lambda x.\, N : A \supset B}\ \supset R^x \qquad \frac{\hat{\Gamma}, M : A \supset B \longrightarrow N : A \quad \hat{\Gamma}, M\, N : B \longrightarrow P : C}{\hat{\Gamma}, M : A \supset B \longrightarrow P : C}\ \supset L$$

$$\frac{\hat{\Gamma} \longrightarrow M : B}{\hat{\Gamma} \longrightarrow \mathbf{inl}\, M : A \vee B}\ \vee R_1 \quad \frac{\hat{\Gamma} \longrightarrow M : B}{\hat{\Gamma} \longrightarrow \mathbf{inr}\, M : A \vee B}\ \vee R_2 \quad \frac{\hat{\Gamma}, u : A \longrightarrow N : C \quad \hat{\Gamma}, w : B \longrightarrow P : C}{\hat{\Gamma}, M : A \vee B \longrightarrow \mathbf{case}(M, u.\, N, w.\, P) : C}\ \vee L^{u,w}$$

$$\frac{}{\hat{\Gamma} \longrightarrow \langle\,\rangle : \top}\ \top R \qquad \text{(no } \top L \text{ rule)}$$

$$\text{(no } \bot R \text{ rule)} \qquad \frac{}{\hat{\Gamma}, M : \bot \longrightarrow \mathbf{abort}\, M : C}\ \bot L$$

References

Rowan Davies. A practical refinement-type checker for Standard ML. In Michael Johnson, editor, *Algebraic Methodology and Software Technology Sixth International Conference (AMAST'97)*, pages 565–566, Sydney, Australia, December 1997. Springer-Verlag LNCS 1349. URL https://github.com/rowandavies/sml-cidre.

Michael J.C. Gordon, Robin Milner, L. Morris, Malcolm C. Newey, and Christopher P. Wadsworth. A metalanguage for interactive proof in LCF. In A. Aho, S. Zillen, and T. Szymanski, editors, *Conference Record of the 5th Annual Symposium on Principles of Programming Languages (POPL'78)*, pages 119–130, Tucson, Arizona, January 1978. ACM Press.

Frank Pfenning. Logical frameworks. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, chapter 17, pages 1063–1147. Elsevier Science and MIT Press, 2001. URL http://www.cs.cmu.edu/~fp/papers/handbook01.pdf.