First-order reasoning in the Calculus of Inductive Constructions

Pierre Corbineau

PCRI-LRI(CNRS UMR 8623)-INRIA Futurs
Bât. 490, Université Paris XI
91405 Orsay Cedex, France

Abstract. In this paper we present a contraction-free sequent calculus including inductive definitions for the first-order intuitionistic logic. We show that it is a natural extension to Dyckhoff's LJT calculus and we prove the contraction- and cut-elimination properties, thus extending Dyckhoff's result, in order to validate its use as a basis for proof-search procedures. Finally we describe the proof-search strategy used in our implementation as a tactic in the Coq proof assistant [2].

1 Introduction

Most basic intuitionistic predicate calculi using sequents [9, 4] include the structural rule of contraction or a left-introduction rule for the arrow in which the principal formula stays in the left premise :

$$\frac{\Gamma, A, A \vdash G}{\Gamma, A \vdash G}\ Contr \qquad \frac{\Gamma, A \to B \vdash A \quad \Gamma, B \vdash G}{\Gamma, A \to B \vdash G}\ L\to$$

Those rules have obvious bad properties if we use them in a bottom-up proof-search procedure since they can lead to loops in the proof-search process if not restricted.

In [5], Roy Dyckhoff described LJT, a calculus for the intuitionistic propositional logic without contraction. Instead he put forward that contraction could be shown admissible, i.e. it could be seen as an implicit rule in his system. Furthermore, he split the $L\to$ rule in several subcases depending on the formula being on the left of the arrow, and that way avoided the repetition of the principal formula in the premise.

In [6], together with Sara Negri he gave a direct proof of cut-elimination for this system, and for its extension to first-order quantifiers $\forall$ and $\exists$. Of course this extension did not have any termination property similar to that of LJT because of the rules about the universal quantifier.

The propositional part of the LJT sequent calculus has been implemented in the Coq proof assistant as a proof-search procedure : the `tauto` [1] tactic [12]. This procedure performs depth-first-search of proofs with optimization of search

[1] in Coq v8.0, tactic names start with lower case letters.

using reversibility of rules in the calculus. This tactic is also used as a goal simplification procedure called `intuition`. The approach used was successful so we wanted to extend it to first-order reasoning.

Moreover, two attempts at automating the predicate calculus in Coq were previously made : the first one was the implantation of a decision procedure for the direct predicate calculus [8, 1], a decidable restriction of the predicate calculus to its linear fragment, it led to the `linear` tactic [7] which was implemented in early versions of Coq. It has been discontinued since, because this fragment is not powerful enough.

The second attempt has been the port of the `jprover` module [14] from the Nuprl prover [10] to Coq. It is basically made of a classical tableau prover packed with a constraint solver to restrict it to intuitionistic logic. Similarly to `linear` this tactic behaves has a black box constructing a complete proof in one step. But it doesn't handle the case of $\forall x.P[x] \vdash \exists y.P[y]$ where the domain must be inhabited, and it has a very restricted view of logical connectives. Moreover its black-box behavior forbids its use as a goal simplification procedure.

Our purpose was to adapt Dyckhoff's system so that it could be used in a natural way for first-order intuitionistic proof-search in Coq. In order to do that we had to cope with the fact that in Coq only the implication $\rightarrow$ and the universal quantification $\forall$ are primitive constructions — they are two forms of dependent products — whereas standard logical connectives $\wedge$,$\vee$,$\perp$ and even the existential quantifier $\exists$ can be defined in terms of inductive definitions.

So we propose here a variant of Dyckhoff's LJT calculus where the primitive logical connectives are $\forall$,$\rightarrow$ and inductive definitions, viewing other connectives as particular cases of inductive definitions, but also allowing many more possible constructions.

In section 2, we first present our inductive definitions and the corresponding notion of first-order formula, and we show how this notion gives a natural extension of Dyckhoff's calculus. Then in section 3 we prove that our calculus enjoys both contraction- and cut-elimination properties. Finally in section 4 we discuss some proof-search strategy issues and present our implementation of a proof-search procedure based on this calculus.

2 A sequent calculus with inductive formulae

2.1 Introducing inductive formulae

In the following text we will use the notations $\overrightarrow{H_i}$, $\overrightarrow{H_i}\rightarrow X$, $\overrightarrow{x}$ and $\forall\overrightarrow{y_i}.X$ as shortcuts for $H_{i,1},\ldots,H_{i,p}$, $H_{i,1}\rightarrow(\ldots\rightarrow(H_{i,p}\rightarrow X))$, $x_1,\ldots,x_p$ and $\forall y_{i,1}\ldots\forall y_{i,p}.X$. Please note that the length of the sequences is always fixed *a priori*, and that the meaning of $\overrightarrow{H_i}$ depends on whether it is or not followed by an arrow. We suppose implication is right-associative and has higher priority than $\forall$. We will also use the $\{\_\}_i$ notation to mean either a sequence of formulae or a (finite) set of premises in a rule, where i ranges over the constructor indices or the hypotheses indices of an inductive formula.

To define our class of formula we start with a signature of first-order constants and predicates of fixed arity. Any term formed by the application of a n-ary predicate to n well-formed terms possibly containing variables will be called *atomic formula*, and the variables occurring in the n terms will be called the free variables of this formula.

Then we define compound formulae and inductive families mutually recursively, so let us begin with the inductive families. An inductive family is a triple $(I, \overrightarrow{X}, \{\mathcal{C}_1 : \tau_1; \mathcal{C}_2 : \tau_2; \ldots\})$ where I is the name of the inductive family, $\overrightarrow{X}$ a possibly empty list of formal parameters having a fixed arity and being either propositional or first-order parameters. $\mathcal{C}_i$ is the name and τ_i the type of the ith constructor, which is itself a formula.

Then we define our formula language inductively as follows: A formula is either an atomic formula or a compound formula. If A and B are formulae then so is $A \rightarrow B$, if $P[x]$ is a formula then so is $\forall x.P[x]$ and if $\overrightarrow{p}$ is a sequence of parameters whose arity and class (formula or term) fit those of the formal parameters of the I family, then $I(\overrightarrow{p})$ is a formula. Implication and universal quantification behave as usual regarding free and bound variables. The free variables in $I(\overrightarrow{p})$ are those in $\overrightarrow{p}$.

A constructor type must be a formula made of a (possibly empty) sequence of universal quantifications and implications and the head of that formula must be $I(\overrightarrow{X})$. The formal parameters must not be bound by the quantifiers. But all other free variables must be universally quantified.

Without loss of generality we will assume that constructor types are in weak prenex form, i.e. all dependent products outermost, thus being of the form $\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{X}, \overrightarrow{y_i}) \rightarrow I(\overrightarrow{X})$. We will call $\overrightarrow{H_i}$ the logical hypotheses of the constructor and $\overrightarrow{y_i}$ the first-order variables of the constructor.

We suppose that inductive families we consider are neither recursive nor mutually recursive, i.e. the relation defined by the use of an inductive family in the logical hypotheses of another one is well-founded.

Here we give a set of examples of inductive definitions defining standard connectives :

$$
\begin{gathered}
(\wedge, (A, B), \{pair : A \rightarrow B \rightarrow A \wedge B\}) \\
(\vee, (A, B), \{inj_l : A \rightarrow A \vee B;\ inj_r : B \rightarrow A \vee B\}) \\
(\bot, (), \{\}) \\
(\top, (), \{triv : \top\}) \\
(\exists, (H[\_]), \{witness : \forall y.H[y] \rightarrow \exists x.H[x]\})
\end{gathered}
$$

The $\wedge$ and $\vee$ inductive families have two propositional parameters of arity 0, $\top$ and $\bot$ have none, and $\exists$ has one propositional parameter of arity 1.

Given those definitions, the meaning of inductive formulae is that

$$
I(\overrightarrow{p}) \Leftrightarrow \bigvee_i (\exists \overrightarrow{y_i}. \bigwedge_j H_{i,j}(\overrightarrow{p}, \overrightarrow{y_i}))
$$

But since we use inductive type instead of just defining the connectives as plain identifiers, the elimination principles for the Calculus of Inductive Con-

$$\frac{}{\Gamma, P \vdash P}\ Ax$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ R\to \qquad \frac{\Gamma, P, B \vdash G}{\Gamma, P, P \to B \vdash G}\ La\to$$

$$\frac{\Gamma, A, B \to C \vdash B \quad \Gamma, C \vdash G}{\Gamma, (A \to B) \to C \vdash G}\ L\to\to$$

$$\frac{\Gamma \vdash A\,[x]}{\Gamma \vdash \forall x.A\,[x]}\ R\forall \qquad \frac{\Gamma, \forall x.A\,[x]\,, A\,[t] \vdash G}{\Gamma, \forall x.A\,[x] \vdash G}\ L\forall$$

$$\frac{\Gamma, (\forall x.A\,[x]) \to B \vdash \forall x.A\,[x] \quad \Gamma, B \vdash G}{\Gamma, (\forall x.A\,[x]) \to B \vdash G}\ L\forall\to$$

$$\frac{\{\Gamma \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})\}_j}{\Gamma \vdash I(\overrightarrow{p})}\ RI_i \qquad \frac{\{\Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash G\}_i}{\Gamma, I(\overrightarrow{p}) \vdash G}\ LI$$

$$\frac{\Gamma, \{\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \to B\}_i \vdash G}{\Gamma, I(\overrightarrow{p}) \to B \vdash G}\ LI\to$$

Fig. 1. The $LJTI$ calculus

structions give us primitive left- and right-introduction rules built in the calculus [13], instead of unfolding the definition and dealing with standard connectives.

Let us see somme more exotic examples : many specific predicates may be defined by non-recursive inductive definitions. For example we express that A satisfies the excluded-middle property using:

$$(\mathrm{Dec}, (A), \{istrue : A \to \mathrm{Dec}(A);\ isfalse : (A \to \bot) \to \mathrm{Dec}(A)\})$$

Another example could be to express the Euclidean division of two natural numbers. That is, $\mathrm{Eucl\_div}(a, b)$ gives both witnesses q,r and proofs of $r < b$ and $a = bq + r$. Eucl_div has two first-order parameters of arity 0 :

$$(\mathrm{Eucl\_div}, (a, b), \{EDintro : \forall q.\forall r.(r < b) \to (a = bq + r) \to \mathrm{Eucl\_div}(a, b)\})$$

2.2 The *LJTI* sequent calculus

From now on we will assume that t ranges over first order terms, x, y over first-order variables $A \ldots G$ over arbitrary formulae, P, Q over atomic formulae, x, y over first-order variables, and $\Gamma, \Gamma', \Gamma''$ over multisets of formulae. When we write $P\,[x]$ we assume that x is not free in $P\,[y]$ if $x \neq y$, and that any variable free in t is free in $P\,[t]$ (we allow the use of α-renaming in P).

Using the definition of inductive formula above we define the $LJTI$ sequent calculus in figure 1. Please note that using generic inductive definitions we have a smaller number of rules in our system than in LJT. Note that in axiom and $La\to$ rules P must be an atomic formula.

In the right introduction rule RI_i, i ranges over the constructor indices, so there is one such rule for each constructor, and in the left introduction rule LI

existential variables $\overrightarrow{y_i}$ must follow the eigenvariable condition, and so must x in the $R\forall$ rule. This means x and $\overrightarrow{y_i}$ must not occur free in the conclusion of those rules (or equivalently they must not occur free in Γ,G or $\overrightarrow{p}$).

For instance, if we try to apply this scheme to $\bot$, we get the following rules :

$$(no\ R\bot\ rule) \qquad \frac{}{\Gamma, \bot \vdash G}\ L\bot \qquad \frac{\Gamma \vdash G}{\Gamma, \bot \to A \vdash G}\ L\bot\to$$

You can check that the rules for $\bot$ match those for the standard connectives in [6] except for $L\bot$ which is a special case of weakening that is invertible (see rule (2) and rule (8)). For Eucl_div we would get :

$$\frac{\Gamma \vdash r < b \quad \Gamma \vdash a = bq + r}{\Gamma \vdash \mathrm{Eucl\_div}(a, b)}\ R\mathrm{Eucl\_div} \qquad \frac{\Gamma, r < b, a = bq + r \vdash G}{\Gamma, \mathrm{Eucl\_div}(a, b) \vdash G}\ L\mathrm{Eucl\_div}$$

$$\frac{\Gamma, \forall q.\forall r.(r < b) \to (a = bq + r) \to A \vdash G}{\Gamma, \mathrm{Eucl\_div}(a, b) \to A \vdash G}\ L\mathrm{Eucl\_div}\to$$

In the LEucl_div rule q and r mustn't be free in Γ or G nor in a or b.

3 Properties of the *LJTI* calculus

3.1 Inversion lemmata

We first give a series of lemmata about invertibility of rules, and admissibility of weakening.

We say that a rule is *admissible* in $LJTI$ if for every instance of the premise(s) that are derivable in $LJTI$, we get a derivation of the conclusion in $LJTI$. When there is only one premise in the rule, we say that this rule is *strongly admissible* if the derivation of the conclusion can be made shorter or of equal height than that of the premise, the height being 0 for an axiom and the maximum of the heights of the derivation of the premises plus one otherwise.

Lemma 1. *The following rules are strongly admissible in $LJTI$.*

$$\frac{\Gamma[x] \vdash G[x]}{\Gamma[t] \vdash G[t]} \quad (1) \qquad\qquad \frac{\Gamma \vdash G}{\Gamma, \Gamma' \vdash G}\ W \quad (2)$$

Proof.

rule (1) : by structural induction on the derivation tree, renaming eigenvariables by induction hypothesis.

rule (2) : By structural induction on the derivation tree, using rule (1) to rename eigenvariables where needed.

Lemma 2. *The following rules are strongly admissible in $LJTI$:*

$$\frac{\Gamma \vdash A{\rightarrow}B}{\Gamma, A \vdash B} \quad (3) \qquad \frac{\Gamma, \forall x.A\,[x] \rightarrow B \vdash G}{\Gamma, B \vdash G} \quad (6)$$

$$\frac{\Gamma, P{\rightarrow}B \vdash G}{\Gamma, B \vdash G} \quad (4) \qquad \frac{\Gamma, I(\overrightarrow{p}) \vdash G}{\Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{t}) \vdash G} \quad (7)$$

$$\frac{\Gamma, (C{\rightarrow}D){\rightarrow}B \vdash G}{\Gamma, B \vdash G} \quad (5) \qquad \frac{\Gamma, I(\overrightarrow{p}){\rightarrow}B \vdash G}{\Gamma, \{\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}B\}_i \vdash G} \quad (8)$$

Proof. By induction on the height of the derivation, using rule (1) to rename eigenvariables and for rule (7).

3.2 Admissibility of contraction

We first show that the generalized axiom rule is admissible in $LJTI$, and we obtain the admissibility of contraction which allows us to show the admissibility of generic $L{\rightarrow}$ rules used in standard sequent calculi.

To perform induction on formulae, we define a notion of weight which is given below :

$$\mathrm{wt}(P) = 1, P \text{ atomic}$$

$$\mathrm{wt}(A{\rightarrow}B) = 1 + \mathrm{wt}(A) + \mathrm{wt}(B)$$

$$\mathrm{wt}(\forall x.A\,[x]) = 1 + \mathrm{wt}(A\,[x])$$

$$\mathrm{wt}(I(\overrightarrow{p})) = \sum_i \mathrm{wt}(\mathcal{C}_i(\overrightarrow{p}))$$

$$\text{if } \mathcal{C}_i(\overrightarrow{p}) : \forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}I(\overrightarrow{p}) \text{ then}$$

$$\mathrm{wt}(\mathcal{C}_i(\overrightarrow{p})) = (2 \times \mathrm{length}(\overrightarrow{y_i})) + \sum_j 1 + \mathrm{wt}(H_{i,j}(\overrightarrow{p}))$$

This weight is lower than the one in [6] in the case of disjunction, but in fact Dyckhoff's proof is valid even with our weight. The essential fact about this weight is that the rules about inductive formulae that actually have premises, when read upward, replace their principal formula with strictly lighter formulae or remove them.

In our proofs, Ind steps mean that we use the induction hypothesis, we use the double bar to distinguish those steps from the others. Admissible rules are labeled by the lemma in which they were introduced.

Lemma 3. *These sequents are provable in LJTI, even if A is not atomic :*

$$\Gamma, A \vdash A \quad (9) \qquad \Gamma, A, A{\rightarrow}B \vdash B \quad (10)$$

Proof. 1. We prove by induction on $\mathrm{wt}(A)$ that for any Γ we have $\Gamma, A \vdash A$, by cases on the shape of A.

- If $A = I(\overrightarrow{p})$ then for all constructors $\mathcal{C}_i$ and logical hypotheses $H_{i,j}$ we have by induction hypothesis $\Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{y_i})$, so for each i we have $\Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash I(\overrightarrow{p})$ by RI_i. Since we can choose $\overrightarrow{y_i}$ that do not occur free in Γ nor in $\overrightarrow{p}$ we can use LI to obtain $\Gamma, I(\overrightarrow{p}) \vdash I(\overrightarrow{p})$.
- If $A = I(\overrightarrow{p}) \rightarrow B$, B being an arbitrary formula, for any constructor $\mathcal{C}_k$ and set of formulae Δ we have by induction hypothesis :

  $$\Gamma, \Delta, \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{z_k}) \rightarrow B \vdash \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{z_k}) \rightarrow B$$

  Using rule (3) for each $H_{k,j}$ we get :

  $$\Gamma, \Delta, \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{z_k}) \rightarrow B, \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{z_k}) \vdash B$$

  Now if we choose $\Delta = \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow B\}_i, \Delta'$ and Δ' is the sequence of formulae obtained by instantiating one or more of the $\overrightarrow{y_k}$ by the $\overrightarrow{z_k}$ in $\forall \overrightarrow{y_k}. \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{y_k}) \rightarrow B$. We can use $L\forall$ for each $z_{k,j}$ with that formula and the formulae in Δ' and we obtain for each constructor $\mathcal{C}_k$:

  $$\Gamma, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow B\}_i, \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{z_k}) \vdash B$$

  We can choose the $\overrightarrow{z_k}$ so that they are not free in Γ, $\overrightarrow{p}$ or B and from there we have :

  $$\dfrac{\dfrac{\dfrac{\{\Gamma, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow B\}_i, \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{z_k}) \vdash B\}_k}{\Gamma, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow B\}_i, I(\overrightarrow{p}) \vdash B}\ LI}{\Gamma, I(\overrightarrow{p}) \rightarrow B, I(\overrightarrow{p}) \vdash B}\ LI\rightarrow}{\Gamma, I(\overrightarrow{p}) \rightarrow B \vdash I(\overrightarrow{p}) \rightarrow B}\ R\rightarrow$$

- The other cases are handled similarly to [6].

2. We have $\Gamma, A \rightarrow B \vdash A \rightarrow B$ by rule (9), so we get $\Gamma, A, A \rightarrow B \vdash B$ by rule (3).

Lemma 4. *The following rule is admissible in LJTI :*

$$\frac{\Gamma \vdash D \quad \Gamma, B \vdash E}{\Gamma, D \rightarrow B \vdash E} \tag{11}$$

Proof. By induction on the height of the derivation d of the first premise and by cases on its last step.

- If it is by an axiom then D is atomic and $\Gamma = \Gamma', D$. We have :

  $$\frac{\Gamma', D, B \vdash E}{\Gamma', D, D \rightarrow B \vdash E}\ La\rightarrow$$

- If it is by RI_i then let $D = I(\overrightarrow{p})$:

  $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\{\Gamma \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})\}_j \quad \Gamma, B \vdash E}{\Gamma', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{t_i}) \rightarrow B \vdash E}\ \text{some } Ind}{\Gamma', \forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow B, \ldots, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{t_i}) \rightarrow B \vdash E}\ W}{\Gamma', \forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow B \vdash E}\ \text{some } L\forall}{\Gamma', \{\forall \overrightarrow{y_k}. \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{y_k}) \rightarrow B\}_k \vdash E}\ W}{\Gamma, I(\overrightarrow{p}) \rightarrow B \vdash E}\ LI\rightarrow$$

– If it is by LI then $\Gamma = \Gamma', I(\overrightarrow{p})$. We have for every constructor $\mathcal{C}_i$:

$$\dfrac{\Gamma', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash D \quad \begin{array}{c}\Gamma', I(\overrightarrow{p}), B \vdash E \\ \vdots \text{ rule } (7) \\ \Gamma', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}), B \vdash E\end{array}}{\Gamma', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}), D \rightarrow B \vdash E} \; Ind$$

If we choose $\overrightarrow{y_i}$ so that they are not free in Γ',$\overrightarrow{p}$,B or E, we use LI and get $\Gamma', I(\overrightarrow{p}), D \rightarrow B \vdash E$.

– If it is by $LI\rightarrow$ then $\Gamma = \Gamma', I(\overrightarrow{p}) \rightarrow C$. We have :

$$\dfrac{\dfrac{\Gamma', \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow C\}_i \vdash D \quad \begin{array}{c}\Gamma', I(\overrightarrow{p}) \rightarrow C, B \vdash E \\ \vdots \text{ rule } (8) \\ \Gamma', \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow C\}_i, B \vdash E\end{array}}{\Gamma', \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow C\}_i, D \rightarrow B \vdash E} \; Ind}{\Gamma', I(\overrightarrow{p}) \rightarrow C, D \rightarrow B \vdash E} \; LI\rightarrow$$

– See [6] for the other cases.

Lemma 5. *The following rule is admissible in $LJTI$:*

$$\frac{\Gamma, (C \rightarrow D) \rightarrow B \vdash E}{\Gamma, C, D \rightarrow B, D \rightarrow B \vdash E} \tag{12}$$

Proof. The interesting case where $(C \rightarrow D) \rightarrow B$ is principal is treated similarly to [6], using rule (11) .

Theorem 1. *The* Contraction *rule below is admissible in $LJTI$.*

$$\frac{\Gamma, A, A \vdash G}{\Gamma, A \vdash G} \; Contr$$

Proof. By lexicographic induction on $\mathrm{wt}(A)$ and the height of the derivation of the premise. If A is not principal in the last step deriving the premise, we use the induction hypothesis on the premise(s) of this step and apply the rule on the contracted premise. If A is principal, we do a case analysis on the shape of A.

– If A is an atomic formula P then the last rule is an axiom and $G = P$ so the conclusion is an axiom.
– If $A = I(\overrightarrow{p})$ then we have for each i:

$$\begin{array}{c}\Gamma, I(\overrightarrow{p}), \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash G \\ \vdots \text{ rule } (7) \\ \Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}), \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash G \\ \vdots \text{ some } Ind \\ \Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash G\end{array}$$

Since the $\overrightarrow{y_i}$ are not free in Γ or $\overrightarrow{p}$ nor in G, we can use LI to get $\Gamma, I(\overrightarrow{p}) \vdash G$.

– If $A = I(\overrightarrow{p}) \to C$ then we have :

$$\dfrac{\begin{array}{c} \Gamma, I(\overrightarrow{p}) \to C, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \to C\}_i \vdash G \\ \vdots \text{ rule (8)} \\ \Gamma, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \to C\}_i, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \to C\}_i \vdash G \\ \vdots \text{ some } Ind \\ \Gamma, \{\forall \overrightarrow{y_i}. \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \to C\}_i \vdash G \end{array}}{\Gamma, I(\overrightarrow{p}) \to C \vdash G} LI\to$$

– Otherwise we do as in [6] , using rule (12) when $A = (C \to D) \to B$.

Which closes our proof by induction.

Lemma 6. *The following rule is admissible in $LJTI$:*

$$\frac{\Gamma, A \to B \vdash A \quad \Gamma, B \vdash G}{\Gamma, A \to B \vdash G} \tag{13}$$

Proof.

$$\dfrac{\dfrac{\Gamma, A \to B \vdash A \quad \dfrac{\Gamma, B \vdash G}{\Gamma, A \to B, B \vdash G} W}{\Gamma, A \to B, A \to B \vdash A} \text{rule (11)}}{\Gamma, A \to B \vdash G} Contr$$

This last lemma shows us that the $LJTI$ calculus is complete with respect to the calculus which could be named LJI where the axiom rule would be the generalized one and all left arrow rules would be replaced by the one from the lemma.

3.3 Cut-Elimination theorem

The proof outline follows that of [6] except that with our notion of inductive formula there are fewer cases to consider.

Theorem 2. *The* Cut *rule below is admissible in $LJTI$.*

$$\frac{\Gamma \vdash A \quad \Gamma', A \vdash E}{\Gamma, \Gamma' \vdash E} Cut$$

Proof. By lexicographic induction on $\mathrm{wt}(A)$ and on the sum of the heights of the derivations of the premises :

If the first premise is an axiom, let $\Gamma = \Gamma'', A$:

$$\frac{\Gamma', A \vdash E}{\Gamma'', \Gamma', A \vdash E} W$$

If the second premise is an axiom, either $E \in \Gamma'$ or $A = E$ and the conclusion is an axiom.

Otherwise, neither premise is an axiom.

If A is not principal on the left, by cases on the last step of the left derivation :

- LI : We have

$$\dfrac{\dfrac{\{\Gamma'', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash A\}_i}{\Gamma'', I(\overrightarrow{p}) \vdash A}\,LI \qquad \Gamma', A \vdash E}{\Gamma'', I(\overrightarrow{p}), \Gamma' \vdash E}\,Cut$$

For each i we use the induction hypothesis :

$$\dfrac{\Gamma'', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash A \quad \Gamma', A \vdash E}{\Gamma'', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}), \Gamma' \vdash E}\,Cut$$

After renaming $\overrightarrow{y_i}$ if they occur free in Γ' or E, We use LI and obtain $\Gamma'', I(\overrightarrow{p}), \Gamma' \vdash E$.
- $LI\!\rightarrow$

$$\dfrac{\dfrac{\Gamma'', \{\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow C\}_i \vdash A}{\Gamma'', I(\overrightarrow{p}) \rightarrow C \vdash A}\,LI\!\rightarrow \qquad \Gamma', A \vdash E}{\Gamma'', I(\overrightarrow{p}) \rightarrow C, \Gamma' \vdash E}\,Cut$$

becomes :

$$\dfrac{\dfrac{\Gamma'', \{\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow C\}_i \vdash A \quad \Gamma', A \vdash E}{\Gamma'', \{\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \rightarrow C\}_i, \Gamma' \vdash E}\,Cut}{\Gamma'', I(\overrightarrow{p}) \rightarrow C, \Gamma' \vdash E}\,LI\!\rightarrow$$

- other cases are dealt with similarly to [6].

If A is principal on the left and not on the right, by cases on the last step of the right premise derivation :

- RI_i : We have

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\{\Gamma', A \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})\}_j}{\Gamma', A \vdash I(\overrightarrow{p})}\,RI_i}{\Gamma, \Gamma' \vdash I(\overrightarrow{p})}\,Cut$$

For each j we use the induction hypothesis :

$$\dfrac{\Gamma \vdash A \quad \Gamma', A \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})}{\Gamma, \Gamma' \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})}\,Cut$$

And we use the RI_i rule to get $\Gamma, \Gamma' \vdash I(\overrightarrow{p})$.
- LI

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\{\Gamma'', A, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash E\}_i}{\Gamma'', A, I(\overrightarrow{p}) \vdash E}\,LI}{\Gamma, \Gamma'', I(\overrightarrow{p}) \vdash E}\,Cut$$

For each i we use the induction hypothesis :

$$\dfrac{\Gamma \vdash A \quad \Gamma'', A, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash E}{\Gamma, \Gamma'', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash E}\,Cut$$

After renaming $\overrightarrow{y_i}$ if they occur free in Γ, we use LI to get $\Gamma, \Gamma'', I(\overrightarrow{p}) \vdash E$.

– $LI{\rightarrow}$

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Gamma'', A, \{\forall \overrightarrow{y_i}.\overrightarrow{H}_i(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}C\}_i \vdash E}{\Gamma'', A, I(\overrightarrow{p}){\rightarrow}C \vdash E}\; LI{\rightarrow}}{\Gamma, \Gamma'', I(\overrightarrow{p}){\rightarrow}C \vdash E}\; Cut$$

becomes :

$$\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma'', A, \{\forall \overrightarrow{y_i}.\overrightarrow{H}_i(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}C\}_i \vdash E}{\Gamma, \Gamma'', \{\forall \overrightarrow{y_i}.\overrightarrow{H}_i(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}C\}_i \vdash E}\; Cut}{\Gamma, \Gamma'', I(\overrightarrow{p}){\rightarrow}C \vdash E}\; LI{\rightarrow}$$

– Otherwise, see [6].

If A is principal in both premises, by cases on the shape of A :

– $A = I(\overrightarrow{p})$

$$\dfrac{\dfrac{\{\Gamma \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})\}_j}{\Gamma \vdash I(\overrightarrow{p})}\; RI_i \qquad \dfrac{\{\Gamma', \overrightarrow{H_k}(\overrightarrow{p}, \overrightarrow{y_k}) \vdash E\}_k}{\Gamma', I(\overrightarrow{p}) \vdash E}\; LI}{\Gamma, \Gamma' \vdash E}\; Cut$$

becomes :

$$\dfrac{\{\Gamma \vdash H_{i,j}(\overrightarrow{p}, \overrightarrow{t_i})\}_j \quad \begin{array}{c} \Gamma', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash E \\ \vdots\ \text{rule (1)} \\ \Gamma', \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{t_i}) \vdash E \end{array}}{\Gamma, \Gamma' \vdash E}\; \text{some } Cut$$

– $A = I(\overrightarrow{p}){\rightarrow}B$

$$\dfrac{\dfrac{\Gamma, I(\overrightarrow{p}) \vdash B}{\Gamma \vdash I(\overrightarrow{p}){\rightarrow}B}\; R{\rightarrow} \qquad \dfrac{\Gamma', \{\forall \overrightarrow{y_i}.\overrightarrow{H}_i(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}B\}_i \vdash E}{\Gamma', I(\overrightarrow{p}){\rightarrow}B \vdash E}\; LI{\rightarrow}}{\Gamma, \Gamma' \vdash E}\; Cut$$

For each constructor index i we can turn our proof of $\Gamma, I(\overrightarrow{p}) \vdash B$ into a proof of $\Gamma, \overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}) \vdash B$ by rule (7), choosing the $\overrightarrow{y_i}$ free in $B, \overrightarrow{p}$ and Γ. Thus we can use $R{\rightarrow}$ and $R\forall$ to obtain $\Gamma \vdash \forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}B$.
Then for each i, we do a cut on $\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}B$ with the second premise $\Gamma', \{\forall \overrightarrow{y_i}.\overrightarrow{H_i}(\overrightarrow{p}, \overrightarrow{y_i}){\rightarrow}B\}_i \vdash E$ and we get the sequent $\Gamma, \ldots, \Gamma, \Gamma' \vdash E$ which we reduce to $\Gamma, \Gamma' \vdash E$ by making contractions on Γ.

– The other cases are dealt with as in [6]

This closes our proof by induction.

This gives us the cut-elimination property for $LJTI$ by removing the topmost cuts first.

4 Embedding our calculus in a proof-search procedure

4.1 Basic strategy

To perform bottom-up proof-search using our calculus, we use bounded depth-first search, using our bound on non-decreasing rules.

We first notice that we can do without the atomicity condition in Ax and $La\!\rightarrow$ rules, since those generalized rules are admissible : for Ax see rule (9), and for $La\!\rightarrow$ use rule (10), cut and contraction. This can speed up proofs by avoiding the destruction of two opposite occurrences of the same compound formula followed by as many axiom rules as the number of its subformulae.

In order to refine our strategies we separated the inductive families in classes. First we distinguish between first-order inductive whose constructors may have first-order (quantified) variables, and propositional inductive families whose constructors are propositional formulae, and among them we have three classes :

- Those with no constructor are the absurd class (for instance $\bot$)
- Those with one constructor are the conjunctive class ($\wedge$,$\top$,...)
- Those with more than one constructors are the disjunctive class ($\vee$,Dec,...)

Of course the axiom and left-absurdity rules are to be used as soon as possible. Moreover, it is fundamental that we try to apply the generalized $La\!\rightarrow$ rule before trying any $LI\!\rightarrow$ rule in order to shortcut that part of parallel destruction.

This calculus also has a lot of invertible rules which must be used before the non-invertible ones, because there will be no need to backtrack if the proof fails next. Notice that for the conjunctive class, the right introduction rule is invertible. Moreover, some rules like $L\forall\!\rightarrow$ and $L\!\rightarrow\rightarrow$ are only partially invertible, so we first try to prove the non-invertible premise and if we succeed there will be no need to backtrack if the second premise fails.

The last point is that some rules generate more than one subgoal to be proved, so we try to delay them as much as possible.

4.2 Instantiation strategy

When all else fails we try to apply instantiating rules $L\forall$ and RI, with I a first-order inductive. To use those rules some terms t must replace the quantified variable(s). To find these terms, we use a well-known notion of polarity (see for instance [11]) to define the set of signed atomic subformulae $\mathcal{SF}(\Gamma \vdash G)$ of a sequent by induction on the structure of its formulae (see figure 2).

We remark that signed atomic subformulae in premises of rules are also in the conclusion, maybe in a more general form (with some terms replaced with metavariables). This can be seen as a kind of subformula property in our calculus, and in the end we only need pairs of matching subformulae of opposite signs used in axiom or $La\!\rightarrow$ rules, and inductive formulae with terminal rules : negative absurdity or positive tautology[2]. We call those particular subformulae trivial subformulae, and they are also necessary in a derivation.

[2] We call tautology any propositional inductive family with a constant constructor

$$
\begin{array}{ll}
\mathcal{SF}^+(A) = +A & \mathcal{SF}^-(A) = -A \text{ (}A\text{ atomic)} \\
\mathcal{SF}^+(A \to B) = \mathcal{SF}^-(A) \cup \mathcal{SF}^+(B) & \mathcal{SF}^-(A \to B) = \mathcal{SF}^+(A) \cup \mathcal{SF}^-(B) \\
\mathcal{SF}^+(\forall x.P\,[x]) = \mathcal{SF}^+(P\,[?_n]) & \mathcal{SF}^-(\forall x.P\,[x]) = \mathcal{SF}^-(P\,[?_n]) \\
\mathcal{SF}^+(I(\overrightarrow{p})) = \bigcup_{i,j} \mathcal{SF}^+(H_{i,j}(\overrightarrow{p}, \overrightarrow{?_{i,k}})) & \mathcal{SF}^-(I(\overrightarrow{p})) = \bigcup_{i,j} \mathcal{SF}^-(H_{i,j}(\overrightarrow{p}, \overrightarrow{?_{i,k}}))
\end{array}
$$

(where $?_n$ and $\overrightarrow{?_{i,k}}$ are fresh metavariables)

$$\mathcal{SF}(\Gamma \vdash G) = \mathcal{SF}^+(G) \cup \bigcup_{H \in \Gamma} \mathcal{SF}^-(H)$$

Fig. 2. signed atomic subformulae

When we want to use a trivial subformula under a quantifier or an inductive definition to prove our sequent, we just need any term t to instantiate our quantified variable, in order to bring that trivial subformula to the top and apply a terminating rule, so we create a goal stating we have a term to instantiate our variable and ask Coq to use `trivial` or `auto` to solve that non-logical goal. We have to use this trick because in Coq, unlike first-order logic, the quantification domain may be empty, and this emptiness is undecidable in general (type inhabitation is what Coq is all about).

Otherwise we try to build matching pairs of atomic subformulae, and that we do by using first-order unification between atomic subformulae of opposite sign, and by looking at the terms associated to the quantified variables in the unifiers, for example, if we have to prove that $\forall x.P\,[x] \vdash \exists y.P\,[f(y)]$, we have the signed atomic subformulae $-P\,[?_1]$ and $+P\,[f(?_2)]$. And we have $\{?_1 \mapsto f(?_2)\}$ as a unifier. So we will try to use a term of the form $f(?_2)$ to instantiate $?_1$.

Now, we can get three different kinds of terms to instantiate our variables: ground terms (without metavariables), open terms (containing metavariables but not outermost), or trivial terms (equal to a metavariable).

If we get ground terms we just use them so, turning $\forall x.P\,[x]$ into $P\,[t]$.

If we get open terms, we *specialize* our quantified formulae: in our example with $f(?_2)$, we turn $\forall x.P\,[x]$ in $\forall y.P\,[f(y)]$. For positive inductive formulae we do the same and we use $\exists$ to quantify over open positions in the term. For instance if we consider the following goal :

$$\Gamma, \forall x.\forall y.y = 2 \times f(y,x) + 1 \vdash \mathrm{Eucl\_div}(a, 2)$$

The unification algorithm will yield $f(a, ?_1)$ for q and 1 for r, and the specialization scheme will give the following goals to try to prove :

$$\Gamma, \forall x.\forall y.y = 2 \times f(y,x) + 1 \vdash \exists x.1 < 2 \wedge a = 2 \times f(a,x) + 1$$

If we get trivial terms, it means that there is a formula of opposite sign that unifies with this one and that this one doesn't need to be specialized, this is the case for example in $\forall x.P\,[x] \vdash \exists y.P\,[y]$. In that case, we proceed like we do with trivial subformulae and we get an additional Coq subgoal about domain inhabitation. Having destroyed our quantifier, we can hope the search procedure will finally bring the matching subformula in outermost position.

You can argue that our specialization scheme leads to non-termination, but in fact the calculus itself doesn't terminate so we just place a counter on the use of those rules plus the $L\forall\!\rightarrow$ rule, and we give a bound to our search procedure.

4.3 The `firstorder` tactic

As announced earlier, this proof-search procedure is available in Coq. Since our experience in maintaining the `tauto/intuition` tactic showed us that a lot of time was spend doing pattern matching on contexts (see [3]) we decided to avoid doing it too often.

So we decided to work at the ML level with a persistent data structure reflecting the logical content of the current subgoal. Since we are keeping track of the head-form of our formulae, we can work *modulo* constant unfolding and $\beta\iota$-reduction at a very low performance cost. The unification algorithm also does some reduction, but it is basically first-order unification since we are not supposed to have any quantified variable at the head of an application.

This implementation choice gave very encouraging results when compared to `tauto`. In some propositional examples `firstorder` solved the goal in less than 1 minute where `tauto` ran overnight without giving a result. For example try

$$(A_0\leftrightarrow A_1)\rightarrow(A_1\leftrightarrow A_2), (A_1\leftrightarrow A_2)\rightarrow(A_2\leftrightarrow A_0), (A_2\leftrightarrow A_0)\rightarrow(A_0\leftrightarrow A_1) \vdash A_0\leftrightarrow A_1$$

with a bigger odd number of variables.

The `firstorder` tactic is available in Coq `v8.0` and can be used like `tauto`. A global integer option may be set using the command (`Set Firstorder Depth` n). This option is the maximum number of non-terminating rules allowed in a branch of the proof, so increasing it may allow your goal to be solved at the cost of a longer search time.

However, in the current state, all propositional inductive definitions are supported but first-order ones are only supported when they have one constructor with only one first-order variable. We are planning to fully support first-order inductive families in the near future.

5 Conclusion & Future Work

We have presented a contraction-free sequent calculus to deal with first-order intuitionistic logic in the Coq proof assistant where most connectives are defined as inductive families. We have shown that this contraction-free calculus enjoys admissibility of contraction and cut-elimination, thus establishing a weak form of subformula property. We have shown how this calculus was implemented as a proof-search tactic in Coq.

Although our inductive formulae do not have more expressivity than standard first-order intuitionistic logic, they give a more uniform reasoning framework. From a more practical point of view they allow users to define their own connectives without having to consider if they would be supported by such or such automatic tactic.

Beside searching for smarter search strategies, there are two directions in which we should extend our work to be able to deal with inductive predicates such as `le` (less than or equal to). One is to try to consider recursive inductive definitions for which our theorems don't apply as such: the weight function is not well-defined and we may lose completeness. The other one is to have inductive predicates whose type in Coq would be an arity instead of a simple sort, but equality is one of those predicates so we couldn't avoid performing some equational reasoning. Indeed, we plan to handle such cases as part of a more general integration of equational reasoning and our proof-search procedure in Coq.

References

1. G. Bellin and J. Ketonen. A decision procedure revisited: Notes on direct logic, linear logic and its implementation. *Theoretical computer science*, 95(1):115–142, 1992.
2. The Coq Development Team (INRIA-LRI-CNRS). The Coq proof assistant reference manual. `http://coq.inria.fr`
3. D. Delahaye. A tactic language for the system Coq. In *Proceedings of Logic for Programming and Automated Reasoning*, volume 1955 of *LNCS/LNAI*, pages 85–95. Springer, 2000.
4. A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs.* American Mathematical Society, Providence, Rhode Island, 1987.
5. R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57(3):795–807, 1992.
6. R. Dyckhoff and S. Negri. Admissibility of structural rules for contraction-free systems of intuitionistic logic. *Journal of Symbolic Logic*, 65:1499–1518, 2000.
7. J.-C. Filliâtre. A decision procedure for direct predicate calculus: study and implementation in the Coq system. Technical Report 96–25, LIP, ENS Lyon, 1995.
8. J.Ketonen and R.Weyhrauch. A decidable fragment of predicate calculus. *Theoretical Computer Science*, 32:297–307, 1984.
9. S. C. Kleene. *Introduction to metamathematics*, volume I of *Bibliotheca Mathematica*. North-Holland, Amsterdam, 1952.
10. C. Kreitz. *The Nuprl Proof Development System, Version 5*, 2002. `http://www.cs.cornell.edu/Info/Projects/NuPrl/`
11. C. Kreitz and J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal of Universal Computer Science*, 5(3):88–112, 1999.
12. C. Munoz. Démonstration automatique dans la logique propositionnelle intuitionniste. Master's thesis, Université Paris 7, 1994.
13. C. Paulin-Mohring. Inductive definitions in the system Coq: Rules and properties. In M. Bezem and J. F. Groote, editors, *TLCA'93, Utrecht, The Netherlands, 16–18 March 1993*, volume 664 of *LNCS*, pages 328–345, Berlin, 1993. Springer-Verlag.
14. S. Schmitt, L. Lorigo, C. Kreitz, and A. Nogin. Integrating connection-based theorem proving into interactive proof assistants. In R. Gore, A. Leitsch, and T. Nipkow, editors, *IJCAR'01*, volume 2083 of *LNAI*, pages 421–426. Springer, 2001.