Sequent Calculus Viewed Modulo

Eric Deplagne

Sequent Calculus Viewed Modulo

Éric Deplagne
UHP - LORIA, B.P. 239, 54 506 Vandœuvre-lès-Nancy, France
Eric.Deplagne@loria.fr

Abstract. The first-order sequent calculus is generally considered as containing no **computation** but only pure **deduction**. But this is not completely true if we look at it carefully, using a *deduction modulo* framework. The origins of the computational part are first implicit behaviours of the calculus, then well known consequences that we do not want to prove any more. We end up with a calculus fully in the spirit of deduction modulo [DHK98].

1 Introduction

Dowek, Hardin and Kirchner propose in [DHK98] a formalism called *deduction modulo* that enables to precisely separe **deduction**, generally undecidable, from **computation**, clearly decidable, so as to be able to forget about easy computations and focus on key deductions.

The typical proof system in deduction modulo is the *sequent calculus modulo*, an extension of the classical first-order sequent calculus designed to take into account a congruence on propositions representing computations.

Viry [Vir98] had the idea that the classical first-order sequent calculus itself, turned into a rewrite system, can be viewed modulo and decomposed into computational and deductive parts using an *oriented rewrite theory* [Vir95].

In this paper we make precise what can be viewed as computation in the first-order sequent calculus. Doing that we obtain a system with a large computational part which is equivalent to the classical one.

2 Sequent calculus modulo

Let us first recall the notions from sequent calculus modulo that we use in this paper. This section intends to recall the most important notions from sequent calculus modulo and to give intuition on how deduction modulo works. Note however that in this paper we use a congruence allowing to identify not only formulas but also sets of sequents. The soundness of

Proceedings of the Fifth ESSLLI Student Session
Catherine Pilière (editor)

this congruence greatly depends on the properties of the first-order sequent calculus.

We work modulo some congruence $\equiv$ and when applying a deduction rule we use matching modulo this congruence. So if we have for instance $C \equiv D$ from the congruence, we are able to apply any rule using C on an input that actually contains D. For instance if we have $2 * 2 \equiv 4$ then a proof of

$$\underline{A}, 4 = 4 \vdash 2 * 2 = 4, \underline{B}$$

is simply

$$\frac{}{\underline{A}, 4 = 4 \vdash 2 * 2 = 4, \underline{B}}\ \text{Axiom}$$

The congruence is conveniently defined by a class rewrite system [JK86]. So we can have $C = D$ or $C \to D$, the two meaning that $C \equiv D$ but $C \to D$ meaning that operationally we replace C by D.

$$\frac{}{\underline{A}, C \vdash C, \underline{B}}\ \text{Axiom}$$

$$\frac{\underline{A}, D, C, \underline{A'} \vdash \underline{B}}{\underline{A}, C, D, \underline{A'} \vdash \underline{B}}\ \mathcal{L}X \qquad \frac{\underline{A} \vdash \underline{B}, D, C, \underline{B'}}{\underline{A} \vdash \underline{B}, C, D, \underline{B'}}\ \mathcal{R}X$$

$$\frac{\underline{A}, C, C \vdash \underline{B}}{\underline{A}, C \vdash \underline{B}}\ \mathcal{L}C \qquad \frac{\underline{A} \vdash C, C, \underline{B}}{\underline{A} \vdash C, \underline{B}}\ \mathcal{R}C$$

$$\frac{\underline{A} \vdash C, \underline{B}}{\underline{A}, \neg C \vdash \underline{B}}\ \mathcal{L}\neg \qquad \frac{\underline{A}, C \vdash \underline{B}}{\underline{A} \vdash \neg C, \underline{B}}\ \mathcal{R}\neg$$

$$\frac{\underline{A}, C, D \vdash \underline{B}}{\underline{A}, C \wedge D \vdash \underline{B}}\ \mathcal{L}\wedge \qquad \frac{\underline{A} \vdash C, \underline{B} \quad \underline{A} \vdash D, \underline{B}}{\underline{A} \vdash C \wedge D, \underline{B}}\ \mathcal{R}\wedge$$

$$\frac{\underline{A}, C \vdash \underline{B} \quad \underline{A}, D \vdash \underline{B}}{\underline{A}, C \vee D \vdash \underline{B}}\ \mathcal{L}\vee \qquad \frac{\underline{A} \vdash C, D, \underline{B}}{\underline{A} \vdash C \vee D, \underline{B}}\ \mathcal{R}\vee$$

$$\frac{\underline{A} \vdash C, \underline{B} \quad \underline{A}, D \vdash \underline{B}}{\underline{A}, C \Rightarrow D \vdash \underline{B}}\ \mathcal{L}\Rightarrow \qquad \frac{\underline{A}, C \vdash D, \underline{B}}{\underline{A} \vdash C \Rightarrow D, \underline{B}}\ \mathcal{R}\Rightarrow$$

$$\frac{\underline{A}, C\{a/x\}, \forall x.C \vdash \underline{B}}{\underline{A}, \forall x.C \vdash \underline{B}}\ \mathcal{L}\forall \qquad \frac{\underline{A} \vdash \forall x.C, C\{y/x\}, \underline{B}}{\underline{A} \vdash \forall x.C, \underline{B}}\ \mathcal{R}\forall$$

$$\frac{\underline{A}, C\{y/x\}, \exists x.C \vdash \underline{B}}{\underline{A}, \exists x.C \vdash \underline{B}}\ \mathcal{L}\exists \qquad \frac{\underline{A} \vdash \exists x.C, C\{a/x\}, \underline{B}}{\underline{A} \vdash \exists x.C, \underline{B}}\ \mathcal{R}\exists$$

In $\mathcal{R}\forall$ and $\mathcal{L}\exists$, y must be a fresh free variable.
In $\mathcal{L}\forall$ and $\mathcal{R}\exists$, a is any term.

Figure 7.1: The classical sequent calculus

3 Expliciting the sequent calculus

We give here (figure 7.1) a definition of the classical sequent calculus that is less general than the one in [GLT89] in the sense that it does not enable to easily extend it, to intuitionism for instance, and that it does not contain a Cut rule. However, it is sufficient for classical logic and will better serve our purpose. We have no weekening rule as weekenings do not nicely fit in our process. The rule $\mathcal{R}\vee$ is not well suited for intuitionism but replacing it by two rules makes theses rules contain implicit weekenings. Finally the rules for the quantifiers have implicit contractions in order to later avoid conflicts when we build contractions in the congruence.

The first-order sequent calculus is generally considered as containing no **computation** but only pure **deduction**. But if we look at it more precisely this is not really true, so we will modify the calculus by identifying as much computation as possible. Doing that our care will be in preserving the provability.

3.1 Formula sets

The $\underline{A}, \underline{B}, \ldots$ are sets of formulas, so the operator ',' :

- is commutative : this is implemented in rules $\mathcal{L}X$ and $\mathcal{R}X$.
- is associative : this is left implicit. Indeed we even have no parenthesis in $\mathcal{L}X$, $\mathcal{R}X$, $\mathcal{L}C$ and $\mathcal{R}C$.
- is idempotent : this is implemented in rules $\mathcal{L}C$ and $\mathcal{R}C$.
- admits a left and right unit, the empty set of formulas : this is implicit in the empty left or right hand side of the turnstyle in expressions like $\vdash \underline{B}$ or $\underline{A} \vdash$.

So we can add the symbol '$\bigtriangledown$' for the empty set of formulas and the axioms :

$$\begin{array}{rclcrcl} \underline{A}, (\underline{B}, \underline{C}) & = & (\underline{A}, \underline{B}), \underline{C} & & \underline{A}, \underline{B} & = & \underline{B}, \underline{A} \\ \underline{A}, \bigtriangledown & = & \underline{A} & & \underline{A}, \underline{A} & = & \underline{A} \end{array}$$

At this point we can eliminate $\mathcal{L}C$, $\mathcal{R}C$, $\mathcal{L}X$ and $\mathcal{R}X$ from the deduction system since they are replaced by the congruence.

3.2 Sequent sets

Two-premisse rules have a silent operator that means that we actually handle sets of sequents. So this operator that we denote '$\bullet$' like in

$$\frac{\underline{A} \vdash C, \underline{B} \quad \bullet \quad \underline{A}, D \vdash \underline{B}}{\underline{A}, C \Rightarrow D \vdash \underline{B}}\ \mathcal{L}\Rightarrow$$

has the same properties as ','. It's unit element will be denoted '$\Diamond$' and is nothing but the empty premisse of the Axiom rule

$$\frac{\Diamond}{\underline{A}, C \vdash C, \underline{B}}$$

We also add to the congruence the axioms for '$\bullet$' :

$$\begin{array}{rclcrcl} \underline{\Gamma} \bullet (\underline{\Delta} \bullet \underline{\Theta}) & = & (\underline{\Gamma} \bullet \underline{\Delta}) \bullet \underline{\Theta} & \quad & \underline{\Gamma} \bullet \underline{\Delta} & = & \underline{\Delta} \bullet \underline{\Theta} \\ \underline{\Gamma} \bullet \Diamond & = & \underline{\Gamma} & & \underline{\Gamma} \bullet \underline{\Gamma} & = & \underline{\Gamma} \end{array}$$

Here we need to point out that this changes the proof object from the proof tree usually used with sequent calculus into a sequence of sets of sequents.

$$\dfrac{\underline{A} \vdash C, \underline{B} \qquad \dfrac{\underline{A}, D \vdash \underline{B} \quad \underline{A}, E \vdash \underline{B}}{\underline{A}, D \vee E \vdash \underline{B}}\,\mathcal{L}\vee}{\underline{A}, C \Rightarrow (D \vee E) \vdash \underline{B}}\,\mathcal{L}\Rightarrow$$

becomes

$$\dfrac{\dfrac{\underline{A} \vdash C, \underline{B} \;\bullet\; \underline{A}, D \vdash \underline{B} \;\bullet\; \underline{A}, E \vdash \underline{B}}{\underline{A} \vdash C, \underline{B} \;\bullet\; \underline{A}, D \vee E \vdash \underline{B}}\,\mathcal{L}\vee}{\underline{A}, C \Rightarrow (D \vee E) \vdash \underline{B}}\,\mathcal{L}\Rightarrow$$

This change is more important than it seems to be, because it permits to put together some elements of the proof that would be far away in the proof tree and possibly use that to simplify the proof.

3.3 Explicit substitutions

Let us have a look at the rules dealing with the quantifiers. The $C\{a/x\}$ is a substitution mechanism that is left totally implicit. Making it explicit is not obvious as, due to the quantifiers, it is not grafting since it should avoid captures. So we will need an explicit substitution calculus [ACCL91] working with the quantifiers instead of the abstraction of λ-calculus. This calculus will have a better behaviour than the corresponding one for λ since its binders – the quantifiers – are formulas and cannot be introduced while normalizing a substitution. But the need not to capture variables does exist and we have the same solution, De Bruijn's indices [dB72]. The calculus also has to deal with the signature of our terms and formulas to find where substitutions have to take place.

We use $\lambda\sigma$ [ACCL91] as a basis for our calculus. Let us recall here the meaning of the notations. Substitutions become syntactic objects so they can be explicitly applied to a term or formula using the '[]' operator. A substitution is basically a list of terms to be substituted to the indices. The list constructor is denoted '$\cdot$', hence a substitution $a \cdot b$ means that the index 1 is to be replaced by a and the index 2 is to be replaced by b. The identity substitution is denoted 'id' and corresponds to the infinite list $1 \cdot 2 \cdot 3 \ldots$

The shift substitution is denoted '$\uparrow$' and corresponds to the infinite list $2 \cdot 3 \cdot 4 \ldots$, it is used to deal with the binder(s) and so that 2 is denoted $1[\uparrow]$, 3 is denoted $1[\uparrow][\uparrow]$, ... The last operator is '$\circ$' denoting the composition of two substitutions so that $1[\uparrow][\uparrow]$ is equivalent to $1[\uparrow \circ \uparrow]$.

We get axioms like

$$\begin{array}{rcl} f(a_1, \ldots, a_n)[s] & = & f(a_1[s], \ldots, a_n[s]) \\ P(a_1, \ldots, a_n)[s] & = & P(a_1[s], \ldots, a_n[s]) \end{array}$$

that deal with the signature,

$$\begin{array}{rcl} (\forall A)[s] & = & \forall(A[1 \cdot (s \circ \uparrow)]) \\ (\exists A)[s] & = & \exists(A[1 \cdot (s \circ \uparrow)]) \end{array}$$

that handle quantifier crossing, taking into account their binding power and avoiding capture,

$$(A \wedge B)[s] \quad = \quad A[s] \wedge B[s]$$

the axioms for connectors are obvious. We also add the axioms of σ except that Id and $Clos$ are duplicated for a substitution applied to a term and to a proposition:

$$\begin{array}{rcl@{\qquad}rcl} a[id] & = & a & A[id] & = & A \\ (a[s])[t] & = & a[s \circ t] & (A[s])[t] & = & A[s \circ t] \end{array}$$

4 Simplifying the calculus

Now that we have explicited the implicit computation in the classical sequent calculus, we seek for another source of computation. There are properties that are consequences of the logic and that we do not want to prove any more, because they are well known. We can build some of them in the calculus, but we will later see that we need to be careful about it in order to preserve the good properties of the resulting system.

$$\begin{array}{rcl} A \Rightarrow B & = & \neg A \vee B \\ \exists A & = & \neg\forall\neg A \end{array}$$

permit to drop the rules $\mathcal{L}\Rightarrow$, $\mathcal{R}\Rightarrow$, $\mathcal{L}\exists$ and $\mathcal{R}\exists$.

$$\begin{array}{rcl} \neg\neg A & = & A \\ A \wedge A & = & A \\ A \vee A & = & A \end{array}$$

permit to simplify proofs at an expense that we will see later in section 5.

We can also, by equating sequents and sets of sequents, directly move some rules into the congruence, including the *Axiom* rule:

$$\begin{array}{rcl@{\qquad}rcl} \underline{A}, \neg C \vdash \underline{B} & = & \underline{A} \vdash C, \underline{B} & \underline{A} \vdash \neg C, \underline{B} & = & \underline{A}, C \vdash \underline{B} \\ \underline{A}, C \wedge D \vdash \underline{B} & = & \underline{A}, C, D \vdash \underline{B} & \underline{A} \vdash C \vee D, \underline{B} & = & \underline{A} \vdash C, D, \underline{B} \end{array}$$

$$\underline{A} \vdash C \wedge D, \underline{B} \;=\; \underline{A} \vdash C, \underline{B} \bullet \underline{A} \vdash D, \underline{B}$$

$$\underline{A}, C \vee D \vdash \underline{B} \;=\; \underline{A}, C \vdash \underline{B} \bullet \underline{A}, D \vdash \underline{B}$$

$$\underline{A}, C \vdash C, \underline{B} \;=\; \Diamond$$

5 Orienting the equations

Now we want to orient the axioms in order to be operational and to actually prove that the congruence is decidable. The congruence is decidable if we can orient the axioms into a class rewrite system convergent modulo a set of axioms for which we can determine such a property. In practice, this means that only associativity-commutativity axioms can remain (figure 7.2). This also meets the requirements for operationality and ensures that we can implement it with a system using rewriting like for instance ELAN[1] [BKK+98].

$$\mathsf{E} = \left\{ \begin{array}{lrcl} Ac & \underline{A}, (\underline{B}, \underline{C}) & = & (\underline{A}, \underline{B}), \underline{C} \\ Cc & \underline{A}, \underline{B} & = & \underline{B}, \underline{A} \\ & & & \\ A\bullet & \underline{\Gamma} \bullet (\underline{\Delta} \bullet \underline{\Theta}) & = & (\underline{\Gamma} \bullet \underline{\Delta}) \bullet \underline{\Theta} \\ C\bullet & \underline{\Gamma} \bullet \underline{\Delta} & = & \underline{\Delta} \bullet \underline{\Gamma} \end{array} \right.$$

Figure 7.2: The remaining (associativity-commutativity) axioms

Doing that shows that we have to be careful about the properties that we build in, as there can be confluence problems. For instance, the axioms in [Vir98] introducing the boolean values *true* and *false* cannot be oriented successfully.

With the axioms of sections 3 and 4 there is a problem with expressions like

$$\underline{A} \vdash C \wedge D, C \wedge D, \underline{B}$$

since choosing to decompose the '$\wedge$' or to use the idempotency of ',' does not converge. This problem is solved by adding another axiom which was missing in [Vir98] and represents subsumption:

$$\underline{A} \vdash \underline{B} \bullet \underline{A}, \underline{A'} \vdash \underline{B'}, \underline{B} = \underline{A} \vdash \underline{B}$$

We should notice that this axiom is easy to express in our framework but cannot be applied while keeping the usual tree representation for proofs.

[1] `http://www.loria.fr/ELAN/`

$$
\mathsf{ER}_{\mathsf{Seq}} = \left\{ \begin{array}{lrcl}
Uc & \underline{A}, \bigtriangledown & \stackrel{=}{\longrightarrow} & \underline{A} \\
Ic & \underline{A}, \underline{A} & \stackrel{=}{\longrightarrow} & \underline{A} \\
U\bullet\text{-}1 & \underline{\Gamma} \bullet \Diamond & \stackrel{=}{\longrightarrow} & \underline{\Gamma} \\
U\bullet\text{-}2 & \underline{\Gamma} \bullet \Box & \stackrel{=}{\longrightarrow} & \underline{\Gamma} \\
I\bullet & \underline{\Gamma} \bullet \underline{\Gamma} & \stackrel{=}{\longrightarrow} & \underline{\Gamma} \\
 & & & \\
f\neg & \neg\neg A & \stackrel{=}{\longrightarrow} & A \\
fI\wedge & A \wedge A & \stackrel{=}{\longrightarrow} & A \\
fI\vee & A \vee A & \stackrel{=}{\longrightarrow} & A \\
f\Rightarrow & A \Rightarrow B & \stackrel{=}{\longrightarrow} & \neg A \vee B \\
f\exists & \exists A & \stackrel{=}{\longrightarrow} & \neg\forall\neg A \\
 & & & \\
s\neg l\text{-}1 & \underline{A}, \neg C \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash C, \underline{B} \\
s\neg l\text{-}2 & \neg C \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \bigtriangledown \vdash C, \underline{B} \\
s\neg r\text{-}1 & \underline{A} \vdash \neg C, \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A}, C \vdash \underline{B} \\
s\neg r\text{-}2 & \underline{A} \vdash \neg C & \stackrel{=}{\longrightarrow} & \underline{A}, C \vdash \bigtriangledown \\
 & & & \\
\mathcal{L}\wedge\text{-}1 & \underline{A}, C \wedge D \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A}, C, D \vdash \underline{B} \\
\mathcal{L}\wedge\text{-}2 & C \wedge D \vdash \underline{B} & \stackrel{=}{\longrightarrow} & C, D \vdash \underline{B} \\
\mathcal{R}\vee\text{-}1 & \underline{A} \vdash C \vee D, \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash C, D, \underline{B} \\
\mathcal{R}\vee\text{-}2 & \underline{A} \vdash C \vee D & \stackrel{=}{\longrightarrow} & \underline{A} \vdash C, D \\
\mathcal{R}\wedge\text{-}1 & \underline{A} \vdash C \wedge D, \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash C, \underline{B} \bullet \underline{A} \vdash D, \underline{B} \\
\mathcal{R}\wedge\text{-}2 & \underline{A} \vdash C \wedge D & \stackrel{=}{\longrightarrow} & \underline{A} \vdash C \bullet \underline{A} \vdash D \\
\mathcal{L}\vee\text{-}1 & \underline{A}, C \vee D \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A}, C \vdash \underline{B} \bullet \underline{A}, D \vdash \underline{B} \\
\mathcal{L}\vee\text{-}2 & C \vee D \vdash \underline{B} & \stackrel{=}{\longrightarrow} & C \vdash \underline{B} \bullet D \vdash \underline{B} \\
 & & & \\
Ax\text{-}1 & \underline{A}, C \vdash C, \underline{B} & \stackrel{=}{\longrightarrow} & \Diamond \\
Ax\text{-}2 & \underline{A}, C \vdash C & \stackrel{=}{\longrightarrow} & \Diamond \\
Ax\text{-}3 & C \vdash C, \underline{B} & \stackrel{=}{\longrightarrow} & \Diamond \\
Ax\text{-}4 & C \vdash C & \stackrel{=}{\longrightarrow} & \Diamond \\
 & & & \\
Sub\text{-}1 & \underline{A} \vdash \underline{B} \bullet \underline{A}, \underline{A}' \vdash \underline{B}', \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash \underline{B} \\
Sub\text{-}2 & \underline{A} \vdash \underline{B} \bullet \underline{A}, \underline{A}' \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash \underline{B} \\
Sub\text{-}3 & \underline{A} \vdash \underline{B} \bullet \underline{A} \vdash \underline{B}', \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash \underline{B} \\
Sub\text{-}4 & \underline{A} \vdash \bigtriangledown \bullet \underline{A}, \underline{A}' \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash \bigtriangledown \\
Sub\text{-}5 & \bigtriangledown \vdash \underline{B} \bullet \underline{A} \vdash \underline{B}', \underline{B} & \stackrel{=}{\longrightarrow} & \bigtriangledown \vdash \underline{B} \\
Sub\text{-}6 & \underline{A} \vdash \bigtriangledown \bullet \underline{A} \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \underline{A} \vdash \bigtriangledown \\
Sub\text{-}7 & \bigtriangledown \vdash \underline{B} \bullet \underline{A} \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \bigtriangledown \vdash \underline{B} \\
Sub\text{-}8 & \bigtriangledown \vdash \bigtriangledown \bullet \underline{A} \vdash \underline{B} & \stackrel{=}{\longrightarrow} & \bigtriangledown \vdash \bigtriangledown
\end{array} \right.
$$

$$\mathsf{ER} = \mathsf{ER}_{\forall\exists\sigma} \cup \mathsf{ER}_{\mathsf{Seq}}$$

Figure 7.3: The congruence (oriented)

$$
\mathsf{ER}_{\forall\exists\sigma} = \left\{
\begin{array}{lrcl}
App_f & f(a_1,\ldots,a_n)[s] & \stackrel{=}{\longrightarrow} & f(a_1[s],\ldots,a_n[s]) \\
App_P & P(a_1,\ldots,a_n)[s] & \stackrel{=}{\longrightarrow} & P(a_1[s],\ldots,a_n[s]) \\
App_\neg & (\neg A)[s] & \stackrel{=}{\longrightarrow} & \neg(A[s]) \\
App_\wedge & (A \wedge B)[s] & \stackrel{=}{\longrightarrow} & A[s] \wedge B[s] \\
App_\vee & (A \vee B)[s] & \stackrel{=}{\longrightarrow} & A[s] \vee B[s] \\
App_\Rightarrow & (A \Rightarrow B)[s] & \stackrel{=}{\longrightarrow} & A[s] \Rightarrow B[s] \\
Abs_\forall & (\forall A)[s] & \stackrel{=}{\longrightarrow} & \forall(A[1 \cdot (s \circ \uparrow)]) \\
Abs_\exists & (\exists A)[s] & \stackrel{=}{\longrightarrow} & \exists(A[1 \cdot (s \circ \uparrow)]) \\
Id_t & a[id] & \stackrel{=}{\longrightarrow} & a \\
Id_f & A[id] & \stackrel{=}{\longrightarrow} & A \\
Clos_t & (a[s])[t] & \stackrel{=}{\longrightarrow} & a[s \circ t] \\
Clos_f & (A[s])[t] & \stackrel{=}{\longrightarrow} & A[s \circ t] \\
VarCons & 1[a \cdot s] & \stackrel{=}{\longrightarrow} & a \\
IdL & id \circ s & \stackrel{=}{\longrightarrow} & s \\
ShiftCons & \uparrow \circ (a \cdot s) & \stackrel{=}{\longrightarrow} & s \\
AssEnv & (s_1 \circ s_2) \circ s_3 & \stackrel{=}{\longrightarrow} & s_1 \circ (s_2 \circ s_3) \\
MapEnv & (a \cdot s) \circ t & \stackrel{=}{\longrightarrow} & a[t] \cdot (s \circ t) \\
IdR & s \circ id & \stackrel{=}{\longrightarrow} & s \\
VarShift & 1 \cdot \uparrow & \stackrel{=}{\longrightarrow} & id \\
SCons & 1[s] \cdot (\uparrow \circ s) & \stackrel{=}{\longrightarrow} & s
\end{array}
\right.
$$

Figure 7.4: The substitution calculus for quantifiers

The subsumption axiom added, the orientation can be done and completion using CiME [CM96] adds rewrite rules (see figure 7.3) to handle the case of a set of formulas being empty like

$$\neg C \vdash \underline{B} \rightarrow \bigtriangledown \vdash C, \underline{B} \text{ and } C \vdash C, \underline{B} \rightarrow \Diamond$$

This ensures that ER is locally confluent modulo E. We conjecture the termination of the system.

6 Disconnecting deduction and congruence

Now we want to be able to focus on the application of the deduction rules, which has to be controlled undeterministically, and just do normalization

with the congruence. To do that we need to avoid that the congruence can interfere with the deduction. To ensure this, we use the techniques from [Vir95], so we turn the remaining deduction rules into rewrite rules to form an oriented rewrite theory. These rules have to be *coherent* with ER modulo E, which is achieved by *coherence completion* adding rules (see figure 7.5) to handle the case of a set of formulas being empty like

$$\forall C \vdash \underline{B} \to C[a \cdot id], \forall C \vdash \underline{B}$$

We also add the rule $\Diamond \to \Box$. This rule permits to check if the proof has been finished by the congruence, for instance in the propositional case where this rule is the only one to remain, traducing the decidability of the proposition calculus. The rule $\underline{\Gamma} \bullet \Box \xrightarrow{=} \underline{\Gamma}$ in ER is only needed to preserve coherence, but will never be used with a strategy using the congruence as a normalization. It does not affect the properties of ER.

We get that R is *strongly coherent* with ER modulo E, which ensures that any strategy can be safely applied in the use of the congruence with respect to the application of the rules of R. Notice that the congruence can lead to a major increase in the size of the object. This increase can actually be exponential like when reducing a formula to its conjunctive normal form.

$$\mathsf{R} = \begin{cases} \mathcal{L}\forall\text{-1} & \underline{A}, \forall C \vdash \underline{B} \longrightarrow \underline{A}, C[a \cdot id], \forall C \vdash \underline{B} \\ \mathcal{L}\forall\text{-2} & \forall C \vdash \underline{B} \longrightarrow C[a \cdot id], \forall C \vdash \underline{B} \\ \mathcal{R}\forall\text{-1} & \underline{A} \vdash \forall C, \underline{B} \longrightarrow \underline{A} \vdash \forall C, C[n \cdot id], \underline{B} \\ \mathcal{R}\forall\text{-2} & \underline{A} \vdash \forall C \longrightarrow \underline{A} \vdash \forall C, C[n \cdot id] \\ \mathit{congruent} & \Diamond \longrightarrow \Box \end{cases}$$

In $\mathcal{R}\forall$-1 and $\mathcal{R}\forall$-2, n must be a fresh free index.
In $\mathcal{L}\forall$-1 and $\mathcal{L}\forall$-2, a is any term.

Figure 7.5: The deduction rules

Theorem 1 *A sequent* $\underline{A} \vdash \underline{B}$ *has a proof in* R *modulo* ER ∪ E *if and only if it has one in the classical sequent calculus.*

A proof of $\underline{A} \vdash \underline{B}$ in R modulo ER ∪ E is a derivation from $\underline{A} \vdash \underline{B}$ to $\Box$. The results in sections 5 and 6 prove that using the strategy we have described is equivalent to using the relation R/ER ∪ E so it remains to be proved that $\underline{A} \vdash \underline{B} \overset{\mathsf{R/ER \cup E}}{\longrightarrow} \Box$ if and only if $\underline{A} \vdash \underline{B}$ has a proof in the classical sequent calculus. Notice that the relation R/ER ∪ E considers ER as a set of equations rather than as a set of rules.

The if part is easy, simulating each rule of the sequent calculus with rules of R or axioms of ER ∪ E and noticing that a proof is always finished by using the *congruent* rule.

The only if part must ensure that all the equations introduced in sections 3, 4 and 5 equate objects whose provability is the same in the classical sequent calculus. This is obvious for the equations introduced in section 3. This is still easy for the equations introduced in section 4 and 5 remembering that we want to preserve proofs and not their structure.

7 Conclusion

We have shown that the first-order sequent calculus can be viewed as a calculus modulo. First by expliciting its implicit computational elements, then by adding in the congruence several consequences of the calculus.

The only deduction rules remaining after that are the different versions of $\mathcal{R}\forall$ and $\mathcal{L}\forall$. We can see that these are the ones handling the quantifier and indeed it is well known that the undecidability of the first-order logic does live there. We have thus obtained a clear distinction between the undecidable deductive part and the decidable computational part placed in the congruence. This makes proof search easier since the computational part can be dealt with by simply normalizing the set of sequents we want to prove and thus we only have to look for the clever option when using a rule from R. This idea can be implemented in ELAN [BKK+98] by using unnamed rules for ER and named rules and a strategy on R.

Acknowledgements

I wish to thank Claude Kirchner for his useful comments and support throughout my work on this subject.

I also wish to thank Jürgen Stuber who pointed out the need to make implicit contractions in the rules for quantifiers to avoid conflict with the built-in rule.

I finally wish to thank the anonymous referees for their remarks that helped me to, I hope, make this paper easier to understand.

Bibliography

[ACCL91] Martin Abadi, Lucas Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[BKK+98] Peter Borovanský, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Christophe Ringeissen. An overview of ELAN. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the second International Workshop on Rewriting Logic and Applications*, volume 15, `http://www.elsevier.nl/locate/entcs/volume15.html`, Pont-à-Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science. `http://www.loria.fr/ELAN/`.

[CM96] Evelyne Contejean and Claude Marché. CiME: Completion modulo E. In Harald Ganzinger, editor, *7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 416–419, New Brunswick, NJ, USA, July 1996. Springer-Verlag. System Description available at `http://www.lri.fr/~{}demons/cime.html`.

[dB72] N. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 34(5):381–392, 1972.

[DHK98] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique, April 1998.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[JK86] J.-P. Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4):1155–1194, 1986. Preliminary version in Proceedings 11th ACM Symposium on Principles of Programming Languages, Salt Lake City (USA), 1984.

[Vir95] Patrick Viry. Rewriting modulo a rewrite system. Technical Report TR-20/95, University of Pise, December 1995.

[Vir98] Patrick Viry. Adventures in sequent calculus modulo equations. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 2nd International Workshop on Rewriting Logic and its Applications*, volume 15 of *Electronic Notes in Theoretical Computer Science*, pages 367–378, Pont-à-Mousson (France), September 1998. Elsevier Science.