Realisability semantics of abstract focussing, formalised

Stéphane Graham-Lengrand
CNRS, École Polytechnique, INRIA, SRI International

We present a sequent calculus for abstract focussing, equipped with proof-terms: in the tradition of Zeilberger's work, logical connectives and their introduction rules are left as a parameter of the system, which collapses the synchronous and asynchronous phases of focussing as macro rules. We go further by leaving as a parameter the operation that extends a context of hypotheses with new ones, which allows us to capture both classical and intuitionistic focussed sequent calculi.

We then define the realisability semantics of (the proofs of) the system, on the basis of Munch-Maccagnoni's orthogonality models for the classical focussed sequent calculus, but now operating at the higher level of abstraction mentioned above. We prove, at that level, the Adequacy Lemma, namely that if a term is of type A, then in the model its denotation is in the (set-theoretic) interpretation of A. This exhibits the fact that the universal quantification involved when taking the orthogonal of a set, reflects in the semantics Zeilberger's universal quantification in the macro rule for the asynchronous phase.

The system and its semantics are all formalised in Coq.

1 Introduction

The objective of this paper is to formalise a strong connection between *focussing* and *realisability*.

Focussing is a concept from proof theory that arose from the study of Linear Logic [Gir87, And92] with motivations in proof-search and logic programming, and was then used for studying the proof theory of classical and intuitionistic logics [Gir91, DJS95, DL07, LM09].

Realisability is a concept used since Kleene [Kle45] to witness provability of formulae and build models of their proofs. While originally introduced in the context of constructive logics, the methodology received a renewed attention with the concept of *orthogonality*, used by Girard to build models of Linear Logic proofs, and then used to define the realisability semantics of classical proofs [DK00].

Both focussing and realisability exploit the notion of *polarity* for formulae, with an asymmetric treatment of positive and negative formulae.

In realisability, a primitive interpretation of a positive formula such as $\exists xA$ (resp. $A_1 \vee A_2$) is given as a set of pairs (t,π), where t is a witness of existence and π is in the interpretation of $\{{}^{t}\!/\!{}_{x}\}A$ (resp. a set of injections $\mathsf{inj}_i(\pi)$, where π is in the interpretation of A_i). In other words, the primitive interpretation of positive formulae expresses a very contructive approach to provability. In contrast, a negative formula (such as $\forall xA$ or $A_1 \Rightarrow A_2$) is interpreted "by duality", as the set that is *orthogonal* to the interpretation of its (positive) negation. In classical realisability, a bi-orthogonal set completion provides denotations for all classical proofs of positive formulae.

In focussing, introduction rules for connectives of the same polarity can be chained without loss of generality: for instance if we decide to prove $(A_1 \vee A_2) \vee A_3$ by proving $A_1 \vee A_2$, then we can assume (without losing completeness) that a proof of this in turn consists in a proof of A_1 or a proof of A_2 (rather than uses a hypothesis, uses a lemma or reasons by contradiction).

Such a grouping of introduction steps for positives (resp. negatives) is called a *synchronous* phase (resp. *asynchronous* phase). In order to express those phases as (or collapse them into) single *macro steps*, some formalisms for *big-step focussing* (as in Zeilberger's work [Zei08a, Zei08b]) abstract away from logical connectives and simply take as a parameter the mechanism by which a positive formula can be decomposed into a collection of positive atoms and negative formulae. While the proof of a positive needs to exhibit such a decomposition, the proof of a negative needs to range over such decompositions and perform a case analysis.

This asymmetry, and this universal quantification over decompositions, are reminiscent of the orthogonal construction of realisability models. To make the connection precise, we formalise the construction of realisability models for (big-step) focussed systems in Zeilberger's style.

In [MM09], the classical realisability semantics was studied for the classical sequent calculus, with an extended notion of cut-elimination that produced focussed proofs. Here we want to avoid relying on the specificities of particular logical connectives, and therefore lift this realisability semantics to the more abstract level of Zeilberger's systems, whose big-step approach also reveals the universal quantification that we want to connect to the orthogonality construction. We even avoid relying on the specificities of a particular logic as follows:

- We do not assume that formulae are syntax, i.e. have an inductive structure, nor do we assume that "positive atoms" are particular kinds of formulae; positive atoms and formulae can now literally be two arbitrary sets, of elements called *atoms* and *molecules*, respectively.
- The operation that extends a context of hypotheses with new ones, is usually taken to be set or multiset union. We leave this extension operation as another parameter of the system, since tweaking it will allow the capture of different logics.

In Section 2 we present our abstract system called LAF, seen through the Curry-Howard correspondence as a typing system for (proof-)terms. Section 3 describes how to tune (i.e. instantiate) the notions of atoms, molecules, and context extensions, so as to capture the big-step versions of standard focussed sequent calculi, both classical and intuitionistic. Section 4 gives the definition of realisability models, where terms and types are interpreted, and proves the Adequacy Lemma. In very generic terms, if t is a proof(-term) for A then in the model the interpretation of t is in the interpretation of A. Finally, Section 5 exhibits a simple model from which we derive the consistency of LAF.

2 The abstract focussed sequent calculus LAF

An instance of LAF is given by a tuple of parameters $(\mathsf{Lab}_+, \mathsf{Lab}_-, \mathbb{A}, \mathbb{M}, \mathsf{Co}, \mathsf{Pat}, \Vdash)$ where each parameter is described below. We use $\rightarrow$ (resp. $\rightharpoonup$) for the total (resp. partial) function space constructor.

Since our abstract focussing system is a typing system, we use a notion of typing contexts, i.e. those structures denoted Γ in a typing judgement of the form $\Gamma \vdash \ldots$. Two kinds of "types" are used (namely atoms and molecules), and what is declared as having such types in a typing context, is two corresponding kinds of labels:[1] *positive labels* and *negative labels*, respectively ranging over Lab_+ and Lab_-. These two sets are the first two parameters of LAF.

[1]We choose to call them "labels", rather than "variables", because "variable" suggests an object identified by a name that "does not matter" and somewhere subject to α-conversion. Our labels form a deterministic way of indexing atoms and molecules in a context, and could accommodate De Bruijn's indices or De Bruijn's levels.

DEFINITION 1 (Generic contexts, generic decomposition algebras)

Given two sets $\mathscr{A}$ and $\mathscr{B}$, an $(\mathscr{A},\mathscr{B})$-*context algebra* is an algebra of the form

$$\left(\mathscr{G}, \left(\begin{array}{l} \mathscr{G}\times \mathsf{Lab}_+ \rightharpoonup \mathscr{A} \\ (\Gamma,x^+) \mapsto \Gamma[x^+] \end{array}\right), \left(\begin{array}{l} \mathscr{G}\times \mathsf{Lab}_- \rightharpoonup \mathscr{B} \\ (\Gamma,x^-) \mapsto \Gamma[x^-] \end{array}\right), \left(\begin{array}{l} \mathscr{G}\times \mathbb{D}_{\mathscr{A},\mathscr{B}} \to \mathscr{G} \\ (\Gamma,\Delta) \mapsto \Gamma;\Delta \end{array}\right)\right)$$

whose elements are called $(\mathscr{A},\mathscr{B})$-*contexts*, and where $\mathbb{D}_{\mathscr{A},\mathscr{B}}$, which we call the $(\mathscr{A},\mathscr{B})$-*decomposition algebra* and whose elements are called $(\mathscr{A},\mathscr{B})$-*decompositions*, is the free algebra defined by the following grammar:

$$\Delta, \Delta_1, \ldots ::= \ a \mid {\sim}b \mid \bullet \mid \Delta_1, \Delta_2$$

where a (resp. b) ranges over $\mathscr{A}$ (resp. $\mathscr{B}$).

Let $\mathbb{D}_{\mathsf{st}}$ abbreviate $\mathbb{D}_{\mathtt{unit},\mathtt{unit}}$, whose elements we call *decomposition structures*.

The *(decomposition) structure* of an $(\mathscr{A},\mathscr{B})$-decomposition Δ, denoted $|\Delta|$, is its obvious homomorphic projection in $\mathbb{D}_{\mathsf{st}}$.

We denote by $\mathsf{dom}^+(\Gamma)$ (resp. $\mathsf{dom}^-(\Gamma)$) the subset of Lab_+ (resp. Lab_-) where $\Gamma[x^+]$ (resp. $\Gamma[x^-]$) is defined, and say that Γ is *empty* if both $\mathsf{dom}^+(\Gamma)$ and $\mathsf{dom}^-(\Gamma)$ are.

Intuitively, an $(\mathscr{A},\mathscr{B})$-decomposition Δ is simply the packaging of elements of $\mathscr{A}$ and elements of $\mathscr{B}$; we could flatten this packaging by seeing $\bullet$ as the empty set (or multiset), and Δ_1, Δ_2 as the union of the two sets (or multisets) Δ_1 and Δ_2. Note that the coercion from $\mathscr{B}$ into $\mathbb{D}_{\mathscr{A},\mathscr{B}}$ is denoted with $\sim$. It helps distinguishing it from the coercion from $\mathscr{A}$ (e.g. when $\mathscr{A}$ and $\mathscr{B}$ intersect each other), and in many instances of LAF it will remind us of the presence of an otherwise implicit negation. But so far it has no logical meaning, and in particular $\mathscr{B}$ is not equipped with an operator $\sim$ of syntactical or semantical nature.

Now we can specify the nature of the LAF parameters:

DEFINITION 2 (LAF parameters) Besides Lab_+ and Lab_-, LAF is parameterised by

- two sets $\mathbb{A}$ and $\mathbb{M}$, whose elements are respectively called *atoms* (denoted a, a',...), and *molecules* (denoted M, M',...);
- an $(\mathbb{A},\mathbb{M})$-context algebra Co, whose elements are called *typing contexts*;
- a *pattern algebra*, an algebra of the form

  $$\left(\mathsf{Pat}, \left(\begin{array}{l} \mathsf{Pat} \to \mathbb{D}_{\mathsf{st}} \\ p \mapsto |p| \end{array}\right)\right)$$

  whose elements are called *patterns*,
- a *decomposition relation*, i.e. a set of elements

  $$(\_ \Vdash \_ : \_) : (\mathbb{D} \times \mathsf{Pat} \times \mathbb{M})$$

  such that if $\Delta \Vdash p : M$ then the structure of Δ is $|p|$.

The $(\mathbb{A},\mathbb{M})$-decomposition algebra, whose elements are called *typing decompositions*, is called the *typing decomposition algebra* and is denoted $\mathbb{D}$.

The group of parameters $(\mathbb{A},\mathbb{M})$ specifies what the instance of LAF, as a logical system, talks about. A typical example is when $\mathbb{A}$ and $\mathbb{M}$ are respectively the sets of (positive) atoms and the set of (positive) formulae from a polarised logic. Section 3 shows how our level of abstraction allows for some interesting variants. In the Curry-Howard view, $\mathbb{A}$ and $\mathbb{M}$ are our sets of types.

The last group of parameters (Pat, $\Vdash$) specifies the structure of molecules. If $\mathbb{M}$ is a set of formulae featuring logical connectives, those parameters specify the introduction rules for the connectives. The intuition behind the terminology is that the decomposition relation $\Vdash$ decomposes a molecule, according to a pattern, into a typing decomposition which, as a first approximation, can be seen as a "collection of atoms and (hopefully smaller) molecules".

DEFINITION 3 (LAF system) Proof-terms are defined by the following syntax:

Positive terms	Terms^+	$t^+ ::= pd$
Decomposition terms	$\mathsf{Terms}^{\mathsf{d}}$	$d ::= x^+ \mid f \mid \bullet \mid d_1, d_2$
Commands	Terms	$c ::= \langle x^- \mid t^+ \rangle \mid \langle f \mid t^+ \rangle$

where p ranges over Pat, x^+ ranges over Lab_+, x^- ranges over Lab_-, and f ranges over the partial function space $\mathsf{Pat} \rightharpoonup \mathsf{Terms}$.

LAF is the inference system of Fig. 1 defining the derivability of three kinds of sequents

$$\begin{array}{lcl} (\_ \vdash [\_ : \_]) & : & (\mathsf{Co} \times \mathsf{Terms}^+ \times \mathbb{M}) \\ (\_ \vdash \_ : \_) & : & (\mathsf{Co} \times \mathsf{Terms}^{\mathsf{d}} \times \mathbb{D}) \\ (\_ \vdash \_) & : & (\mathsf{Co} \times \mathsf{Terms}) \end{array}$$

We further impose in rule async that the domain of function f be exactly those patterns that can decompose M ($p \in \mathsf{Dom}(f)$ if and only if there exists Δ such that $\Delta \Vdash p : M$).

$\mathsf{LAF}^{\mathsf{cf}}$ is the inference system LAF without the cut-rule.

$$\frac{\Delta \Vdash p : M \quad \Gamma \vdash d : \Delta}{\Gamma \vdash [pd : M]}\ \mathsf{sync}$$

$$\frac{}{\Gamma \vdash \bullet : \bullet} \qquad \frac{\Gamma \vdash d_1 : \Delta_1 \quad \Gamma \vdash d_2 : \Delta_2}{\Gamma \vdash d_1, d_2 : \Delta_1, \Delta_2}$$

$$\frac{\Gamma\left[x^+\right] = a}{\Gamma \vdash x^+ : a}\ \mathsf{init} \qquad \frac{\forall p, \forall \Delta, \quad \Delta \Vdash p : M \ \Rightarrow\ \Gamma; \Delta \vdash f(p)}{\Gamma \vdash f : {\sim} M}\ \mathsf{async}$$

$$\frac{\Gamma \vdash [t^+ : \Gamma\left[x^-\right]]}{\Gamma \vdash \left\langle x^- \mid t^+ \right\rangle}\ \mathsf{select} \qquad \frac{\Gamma \vdash f : {\sim} M \qquad \Gamma \vdash [t^+ : M]}{\Gamma \vdash \left\langle f \mid t^+ \right\rangle}\ \mathsf{cut}$$

Figure 1: LAF

An intuition of LAF can be given in terms of proof-search:

When we want to "prove" a molecule, we first need to decompose it into a collection of atoms and (refutations of) molecules (rule sync). Each of those atoms must be found in the current typing context (rule init). Each of those molecules must be refuted, and the way to do this is to consider all the possible ways that this molecule could be decomposed, and for each of those decompositions, prove the inconsistency of the current typing context extended with the decomposition (rule async). This can be done by proving one of the molecules refuted in the typing context (rule select) or refuted by a complex proof (rule cut). Then a new cycle starts.

Now it will be useful to formalise the idea that, when a molecule M is decomposed into a collection of atoms and (refutations of) molecules, the latter are "smaller" than M:

DEFINITION 4 (Well-founded LAF instance)
We write $M' \triangleleft M$ if there are Δ and p such that $\Delta \Vdash p{:}M$ and $\sim M'$ is a leaf of Δ.
The LAF instance is *well-founded* if (the smallest order containing) $\triangleleft$ is well-founded.

Well-foundedness is a property that a LAF instance may or may not have, and which we will require to construct its realisability semantics. A typical situation where it holds is when $M' \triangleleft M$ implies that M' is a sub-formula of M.

The above intuitions may become clearer when we instantiate the parameters of LAF with actual literals, formulae, etc in order to capture existing systems: we shall therefore we illustrate system LAF by specifying different instances, providing each time the long list of parameters, that capture different focussed sequent calculus systems.

While LAF is defined as a typing system (in other words with proof-terms decorating proofs in the view of the Curry-Howard correspondence), the traditional systems that we capture below are purely logical, with no proof-term decorations. When encoding the former into the latter, we therefore need to erase proof-term annotation, and for this it is useful to project the notion of typing context as follows:

DEFINITION 5 (Referable atoms and molecules) Given a typing context Γ, let $\mathsf{Im}^+(\Gamma)$ (resp. $\mathsf{Im}^-(\Gamma)$) be the image of function $x^+ \mapsto \Gamma[x^+]$ (resp. $x^- \mapsto \Gamma[x^-]$), i.e. the set of atoms (resp. molecules) that can be refered to, in Γ, by the use of a positive (resp. negative) label.

3 Examples in propositional logic

The parameters of LAF will be specified so as to capture: the classical focussed sequent calculus LKF and the intuitionistic one LJF [LM09].

3.1 Polarised classical logic - one-sided

In this sub-section we define the instance LAF_{K1} corresponding to the (one-sided) calculus LKF:

DEFINITION 6 (Literals, formulae, patterns, decomposition)
Let $\mathbb{A}$ be a set of elements called *atoms* and ranged over by $a, a', \ldots$.
Negations of atoms $a^\perp, a'^\perp, \ldots$ range over a set isomorphic to, but disjoint from, $\mathbb{A}$.
Let $\mathbb{M}$ be the set defined by the first line of the following grammar for (polarised) formulae of classical logic:

Positive formulae	$P, \ldots$	$::= a \mid \top^+ \mid \bot^+ \mid A \wedge^+ B \mid A \vee^+ B$
Negative formulae	$N, \ldots$	$::= a^\perp \mid \top^- \mid \bot^- \mid A \wedge^- B \mid A \vee^- B$
Unspecified formulae	A	$::= P \mid N$

Negation is extended to formulae as usual by De Morgan's laws (see e.g. [LM09]).
The set Pat of *pattern* is defined by the following grammar:

$$p, p_1, p_2, \ldots ::= \_^+ \mid \_^- \mid \bullet \mid (p_1, p_2) \mid \mathsf{inj}_i(p)$$

The decomposition relation $(\_ \Vdash \_ : \_) : (\mathbb{D} \times \mathsf{Pat} \times \mathbb{M})$ is the restriction to molecules of the relation defined inductively for all formulae by the inference system of Fig. 2.
The map $p \mapsto |p|$ can be inferred from the decomposition relation.

$$\frac{}{\bullet \Vdash \bullet : \top^+} \qquad \frac{}{\sim N^\perp \Vdash \_^- : N} \qquad \frac{}{a \Vdash \_^+ : a}$$

$$\frac{\Delta_1 \Vdash p_1 : A_1 \quad \Delta_2 \Vdash p_2 : A_2}{\Delta_1, \Delta_2 \Vdash (p_1, p_2) : A_1 \wedge^+ A_2} \qquad \frac{\Delta \Vdash p : A_i}{\Delta \Vdash \mathsf{inj}_i(p) : A_1 \vee^+ A_2}$$

Figure 2: Decomposition relation for LAF_{K1}

Keeping the sync rule of LAF_{K1} in mind, we can already see in Fig. 2 the traditional introduction rules of positive connectives in polarised classical logic. Note that these rules make LAF_{K1} a well-founded LAF instance, since $M' \triangleleft M$ implies that M' is a sub-formula of M. The rest of this sub-section formalises that intuition and explains how LAF_{K1} manages the introduction of negative connectives, etc. But in order to finish the instantiation of LAF capturing LKF, we need to define typing contexts, i.e. give Lab_+, Lab_-, and Co. In particular, we have to decide how to refer to elements of the typing context. To avoid getting into aspects that may be considered as implementation details (in [GL14a] we present two implementations based on De Bruijn's indices and De Bruijn's levels), we will stay rather generic and only assume the following property:

DEFINITION 7 (Typing contexts) We assume that context extensions satisfy:

$$\begin{array}{llll} \mathsf{Im}^+(\Gamma; a) & = \mathsf{Im}^+(\Gamma) \cup \{a\} & \mathsf{Im}^-(\Gamma; a) & = \mathsf{Im}^-(\Gamma) \\ \mathsf{Im}^+(\Gamma; \sim M) & = \mathsf{Im}^+(\Gamma) & \mathsf{Im}^-(\Gamma; \sim M) & = \mathsf{Im}^-(\Gamma) \cup \{M\} \\ \mathsf{Im}^\pm(\Gamma; \bullet) & = \mathsf{Im}^\pm(\Gamma) & \mathsf{Im}^\pm(\Gamma; (\Delta_1, \Delta_2)) & = \mathsf{Im}^\pm(\Gamma; \Delta_1; \Delta_2) \end{array}$$

where $\pm$ stands for either $+$ or $-$.

We now relate (cut-free) $\mathsf{LAF}^{\mathsf{cf}}_{K1}$ and the LKF system of [LM09] by mapping sequents:

DEFINITION 8 (Mapping sequents)

We encode the sequents of LAF_{K1} (regardless of derivability) to those of LKF as follows:

$$\begin{array}{lcl} \phi(\Gamma \vdash c) & := & \vdash \mathsf{Im}^+(\Gamma)^\perp, \mathsf{Im}^-(\Gamma) \Uparrow \\ \phi(\Gamma \vdash x^+ : a) & := & \vdash \mathsf{Im}^+(\Gamma)^\perp, \mathsf{Im}^-(\Gamma) \Downarrow a \\ \phi(\Gamma \vdash f : \sim P) & := & \vdash \mathsf{Im}^+(\Gamma)^\perp, \mathsf{Im}^-(\Gamma) \Downarrow P^\perp \\ \phi(\Gamma \vdash [t^+ : P]) & := & \vdash \mathsf{Im}^+(\Gamma)^\perp, \mathsf{Im}^-(\Gamma) \Downarrow P \end{array}$$

THEOREM 1 ϕ satisfies structural adequacy between $\mathsf{LAF}^{\mathsf{cf}}_{K1}$ and LKF.

The precise notion of adequacy used here is formalised in [GL14a]; let us just say here that it preserves the derivability of sequents in a compositional way (a derivation π in one system is mapped to a derivation π' in the other system, and its subderivations are mapped to subderivations of π').

3.2 Polarised intuitionistic logic

In this sub-section we define the instance LAF_J corresponding to the (two-sided) calculus LJF. Because it it two-sided, and the LAF framework itself does not formalise the notion of side (it is not incorrect to see LAF as being intrinsically one-sided), we shall embed a *side information* in the notions of atoms and molecules:

DEFINITION 9 (Literals, formulae, patterns, decomposition)

Let $\mathbb{L}^+$ (resp. $\mathbb{L}^-$) be a set of elements called positive (resp. negative) literals, and ranged over by $l^+, l_1^+, l_2^+, \ldots$ (resp. $l^-, l_1^-, l_2^-, \ldots$). Formulae are defined by the following grammar:

$$\begin{array}{lll}
\text{Positive formulae} & P, \ldots & ::= l^+ \,|\, \top^+ \,|\, \bot^+ \,|\, A{\wedge^+}B \,|\, A{\vee}B \\
\text{Negative formulae} & N, \ldots & ::= l^- \,|\, \top^- \,|\, \bot^- \,|\, A{\wedge^-}B \,|\, A{\Rightarrow}B \,|\, \neg A \\
\text{Unspecified formulae} & A & ::= P \,|\, N
\end{array}$$

We *position* a literal or a formula on the left-hand side or the right-hand side of a sequent by combining it with an element, called *side information*, of the set $\{\mathsf{l}, \mathsf{r}\}$: we define

$$\begin{array}{lll}
\mathbb{A} & := & \{(l^+, \mathsf{r}) \,|\, l^+ \text{ positive literal}\} \cup \{(l^-, \mathsf{l}) \,|\, l^- \text{ negative literal}\} \cup \{(\bot^-, \mathsf{l})\} \\
\mathbb{M} & := & \{(P, \mathsf{r}) \,|\, P \text{ positive formula}\} \cup \{(N, \mathsf{l}) \,|\, N \text{ negative formula}\}
\end{array}$$

In the rest of this sub-section v stands for either a negative literal l^- or $\bot^-$.

The set Pat of *pattern* is defined by the following grammar:

$$\begin{array}{lll}
p, p_1, p_2, \ldots & ::= & \_^+_{\mathsf{r}} \,|\, \_^-_{\mathsf{r}} \,|\, \bullet_{\mathsf{r}} \,|\, (p_1, p_2) \,|\, \mathsf{inj}_i(p) \\
 & | & \_^+_{\mathsf{l}} \,|\, \_^-_{\mathsf{l}} \,|\, \bullet_{\mathsf{l}} \,|\, p_1 :: p_2 \,|\, \pi_i(p) \,|\, \curvearrowright(p)
\end{array}$$

The decomposition relation $(\_ \Vdash \_ : \_) : (\mathbb{D} \times \mathsf{Pat} \times \mathbb{M})$ is the restriction to molecules of the relation defined inductively for all positioned formulae by the inference system of Fig. 3.

$$\frac{}{\sim(N, \mathsf{l}) \Vdash \_^-_{\mathsf{r}} : (N, \mathsf{r})} \qquad \frac{}{(l^+, \mathsf{r}) \Vdash \_^+_{\mathsf{r}} : (l^+, \mathsf{r})}$$

$$\frac{}{\bullet \Vdash \bullet_{\mathsf{r}} : (\top^+, \mathsf{r})}$$

$$\frac{\Delta_1 \Vdash p_1 : (A_1, \mathsf{r}) \quad \Delta_2 \Vdash p_2 : (A_2, \mathsf{r})}{\Delta_1, \Delta_2 \Vdash (p_1, p_2) : (A_1{\wedge^+}A_2, \mathsf{r})} \qquad \frac{\Delta \Vdash p : (A_i, \mathsf{r})}{\Delta \Vdash \mathsf{inj}_i(p) : (A_1{\vee}A_2, \mathsf{r})}$$

$$\frac{}{\sim(P, \mathsf{r}) \Vdash \_^-_{\mathsf{l}} : (P, \mathsf{l})} \qquad \frac{}{(l^-, \mathsf{l}) \Vdash \_^+_{\mathsf{l}} : (l^-, \mathsf{l})}$$

$$\frac{}{(\bot^-, \mathsf{l}) \Vdash \bullet_{\mathsf{l}} : (\bot^-, \mathsf{l})} \qquad \frac{\Delta \Vdash p : (A, \mathsf{r})}{\Delta, (\bot^-, \mathsf{l}) \Vdash \curvearrowright(p) : (\neg A, \mathsf{l})}$$

$$\frac{\Delta_1 \Vdash p_1 : (A_1, \mathsf{r}) \quad \Delta_2 \Vdash p_2 : (A_2, \mathsf{l})}{\Delta_1, \Delta_2 \Vdash p_1 :: p_2 : (A_1{\Rightarrow}A_2, \mathsf{l})} \qquad \frac{\Delta \Vdash p : (A_i, \mathsf{l})}{\Delta \Vdash \pi_i(p) : (A_1{\wedge^-}A_2, \mathsf{l})}$$

Figure 3: Decomposition relation for LAF_J

Again, we can already see in Fig. 3 the traditional right-introduction rules of positive connectives and left-introduction rules of negative connectives, and again, it is clear from these rules that LAF_J is well-founded.

We now interpret LAF_J sequents as intuitionistic sequents (from e.g. LJF [LM09]):

Definition 10 (LAF_J sequents as two-sided LJF sequents)

First, when $\pm$ is either $+$ or $-$, we define

$$\begin{aligned} \mathsf{Im}^{\pm\mathsf{r}}(\Gamma) &:= \{A \mid (A,\mathsf{r}) \in \mathsf{Im}^{\pm}(\Gamma)\} \\ \mathsf{Im}^{+\mathsf{l}}(\Gamma) &:= \{l^- \mid (l^-,\mathsf{l}) \in \mathsf{Im}^{+}(\Gamma)\} \\ \mathsf{Im}^{-\mathsf{l}}(\Gamma) &:= \{N \mid (N,\mathsf{l}) \in \mathsf{Im}^{-}(\Gamma)\} \end{aligned}$$

Then we define the encoding:

$$\begin{array}{lcl} \phi(\Gamma \vdash c) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] \longrightarrow [\mathsf{Im}^{+\mathsf{l}}(\Gamma), \mathsf{Im}^{-\mathsf{r}}(\Gamma)] \\ \phi(\Gamma \vdash x^+ : (l^-,\mathsf{l})) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] \xrightarrow{l^-} [\mathsf{Im}^{+\mathsf{l}}(\Gamma), \mathsf{Im}^{-\mathsf{r}}(\Gamma)] \\ \phi(\Gamma \vdash f : \sim(P,\mathsf{r})) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] \xrightarrow{P} [\mathsf{Im}^{+\mathsf{l}}(\Gamma), \mathsf{Im}^{-\mathsf{r}}(\Gamma)] \\ \phi(\Gamma \vdash [t^+ : (N,\mathsf{l})]) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] \xrightarrow{N} [\mathsf{Im}^{+\mathsf{l}}(\Gamma), \mathsf{Im}^{-\mathsf{r}}(\Gamma)] \\ \phi(\Gamma \vdash x^+ : (l^+,\mathsf{r})) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] -_{l^+}\!\!\rightarrow \\ \phi(\Gamma \vdash f : \sim(N,\mathsf{l})) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] -_{N}\!\!\rightarrow \\ \phi(\Gamma \vdash [t^+ : (P,\mathsf{r})]) & := & [\mathsf{Im}^{+\mathsf{r}}(\Gamma), \mathsf{Im}^{-\mathsf{l}}(\Gamma)] -_{P}\!\!\rightarrow \end{array}$$

In the first four cases, we require $\mathsf{Im}^{+\mathsf{l}}(\Gamma), \mathsf{Im}^{-\mathsf{r}}(\Gamma)$ to be a singleton (or be empty).

Definition 11 (Typing contexts) We assume that we always have $(\bot^-,\mathsf{l}) \in \mathsf{Im}^+(\Gamma)$ and that

$$\begin{array}{llll} \mathsf{Im}^+(\Gamma;(l^+,\mathsf{r})) & = \mathsf{Im}^+(\Gamma) \cup \{(l^+,\mathsf{r})\} & \mathsf{Im}^-(\Gamma;a) & = \mathsf{Im}^-(\Gamma) \\ \mathsf{Im}^+(\Gamma;\sim M) & = \mathsf{Im}^+(\Gamma) & \mathsf{Im}^-(\Gamma;\sim(N,\mathsf{l})) & = \mathsf{Im}^-(\Gamma) \cup \{(N,\mathsf{l})\} \\ \mathsf{Im}^{\pm}(\Gamma;\bullet) & = \mathsf{Im}^{\pm}(\Gamma) & \mathsf{Im}^{\pm}(\Gamma;(\Delta_1,\Delta_2)) & = \mathsf{Im}^{\pm}(\Gamma;\Delta_1;\Delta_2) \end{array}$$

$$\begin{array}{ll} \mathsf{Im}^+(\Gamma;(v,\mathsf{l})) & = \{(l^+,\mathsf{r}) \mid (l^+,\mathsf{r}) \in \mathsf{Im}^+(\Gamma)\} \cup \{(v,\mathsf{l}),(\bot^-,\mathsf{l})\} \\ \mathsf{Im}^-(\Gamma;\sim(P,\mathsf{r})) & = \{(N,\mathsf{l}) \mid (N,\mathsf{l}) \in \mathsf{Im}^-(\Gamma)\} \cup \{(P,\mathsf{r})\} \end{array}$$

where again $\pm$ stands for either $+$ or $-$ and v stands for either a negative literal l^- or $\bot^-$.

The first three lines are the same as those assumed for $K1$, except they are restricted to those cases where we do not try to add to Γ an atom or a molecule that is interpreted as going to the right-hand side of a sequent. When we want to do that, this atom or molecule should overwrite the previous atom(s) or molecule(s) that was (were) interpreted as being on the right-hand side; this is done in the last two lines, where $\mathsf{Im}^{+\mathsf{l}}(\Gamma), \mathsf{Im}^{-\mathsf{r}}(\Gamma)$ is completely erased.

Theorem 2 ϕ satisfies structural adequacy between $\mathsf{LAF}_J^{\mathsf{cf}}$ and LJF.

The details are similar to those of Theorem 1, relying on the LJF properties expressed in [LM09].

4 Realisability semantics of LAF

We now look at the semantics of LAF systems, setting as an objective the definition of models and the proof of their correctness at the same generic level as that of LAF.

In this section, $\mathbb{P}(\mathscr{A})$ stands for the power set of a given set $\mathscr{A}$.

Given a LAF instance, we define the following notion of realisability algebra:

Definition 12 (Realisability algebra) A *realisability algebra* is an algebra of the form

$$\left(\mathscr{L}, \mathscr{P}, \mathscr{N}, \bot, \tilde{\mathsf{Co}}, \left(\begin{array}{l} \mathsf{Pat} \rightarrow (\mathbb{D}_{\mathscr{L},\mathscr{N}} \rightarrow \mathscr{P}) \\ p \mapsto \tilde{p} \end{array}\right), \left(\begin{array}{cc} (\mathsf{Pat} \rightharpoonup \mathsf{Terms}) \times \tilde{\mathsf{Co}} & \rightharpoonup \mathscr{N} \\ (f,\rho) & \mapsto [\![f]\!]_\rho \end{array}\right), \left(\begin{array}{l} \mathbb{A} \rightarrow \mathbb{P}(\mathscr{L}) \\ a \mapsto [\![a]\!] \end{array}\right)\right)$$

where

- $\mathscr{L},\mathscr{P},\mathscr{N}$ are three arbitrary sets of elements called *label denotations*, *positive denotations*, *negative denotations*, respectively;
- $\perp$ is a relation between negative and positive denotations ($\perp\ \subseteq \mathscr{N}\times\mathscr{P}$), called the *orthogonality relation*;
- $\tilde{\mathsf{Co}}$ is a $(\mathscr{L},\mathscr{N})$-context algebra, whose elements, denoted $\rho,\rho',\ldots$, are called *semantic contexts*.

The $(\mathscr{L},\mathscr{N})$-decomposition algebra $\mathbb{D}_{\mathscr{L},\mathscr{N}}$ is abbreviated $\tilde{\mathbb{D}}$; its elements, denoted $\mathfrak{d}$, $\mathfrak{d}'\ldots$, are called *semantic decompositions*.

Given a model structure, we can define the interpretation of proof-terms. The model structure already gives an interpretation for the partial functions f from patterns to commands. We extend it to all proof-terms as follows:

DEFINITION 13 (Interpretation of proof-terms)

Positive terms (in Terms^+) are interpreted as positive denotations (in $\mathscr{P}$), decomposition terms (in $\mathsf{Terms}^\mathsf{d}$) are interpreted as semantic decompositions (in $\tilde{\mathbb{D}}$), and commands (in Terms) are interpreted as pairs in $\mathscr{N}\times\mathscr{P}$ (that may or may not be orthogonal), as follows:

$$\begin{array}{lll|lll|lll}
[\![pd]\!]_\rho & := & \tilde{p}([\![d]\!]_\rho) & [\![\bullet]\!]_\rho & := & \bullet & [\![\langle x^-\mid t^+\rangle]\!]_\rho & := & (\rho\,[x^-]\,,[\![t^+]\!]_\rho)\\
 & & & [\![d_1,d_2]\!]_\rho & := & [\![d_1]\!]_\rho,[\![d_2]\!]_\rho & [\![\langle f\mid t^+\rangle]\!]_\rho & := & ([\![f]\!]_\rho,[\![t^+]\!]_\rho)\\
 & & & [\![x^+]\!]_\rho & := & \rho\,[x^+] & & & \\
 & & & [\![f]\!]_\rho & := & [\![f]\!]_\rho \text{ as given by the realisability algebra} & & &
\end{array}$$

Our objective is now the Adequacy Lemma whereby, if t is of type A then the interpretation of t is in the interpretation of A. We have already defined the interpretation of proof-terms in a model structure. We now proceed to define the interpretation of types.

In system LAF, there are four concepts of "type inhabitation":

1. "proving" an atom by finding a suitable positive label in the typing context;
2. "proving" a molecule by choosing a way to decompose it into a typing decomposition;
3. "refuting" a molecule by case analysing all the possible ways of decomposing it into a typing decomposition;
4. "proving" a typing decomposition by inhabiting it with a decomposition term.

Correspondingly, we have the four following interpretations, with the interpretations of atoms (1.) in $\mathbb{P}(\mathscr{L})$ being arbitrary and provided as a parameter of a realisability algebra:

DEFINITION 14 (Interpretation of types and typing contexts) Assume the instance of LAF is well-founded. We define (by induction on the well-founded ordering between molecules):

2. the positive interpretation of a molecule in $\mathbb{P}(\mathscr{P})$;
3. the negative interpretation of a molecule in $\mathbb{P}(\mathscr{N})$;
4. the interpretation of a typing decomposition in $\mathbb{P}(\mathbb{D}_{\mathscr{L},\mathscr{N}})$:

$$\begin{array}{lll}
[\![M]\!]^+ & := \{\tilde{p}(\mathfrak{d}) \in \mathscr{P} & \mid \mathfrak{d} \in [\![\Delta]\!],\text{ and } \Delta \Vdash p{:}M\} \\
[\![M]\!]^- & := \{\mathfrak{n} \in \mathscr{N} & \mid \forall \mathfrak{p} \in [\![M]\!]^+, \mathfrak{n} \perp \mathfrak{p}\} \\
[\![\bullet]\!] & := \{\bullet\} & \\
[\![\Delta_1,\Delta_2]\!] & := \{\mathfrak{d}_1,\mathfrak{d}_2 & \mid \mathfrak{d}_1 \in [\![\Delta_1]\!] \text{ and } \mathfrak{d}_2 \in [\![\Delta_2]\!]\} \\
[\![a]\!] & := [\![a]\!] & \text{as given by the realisability algebra} \\
[\![\sim M]\!] & := \{\sim\mathfrak{n} & \mid \mathfrak{n} \in [\![M]\!]^-\}
\end{array}$$

We then define the interpretation of a typing context:

$$\begin{array}{ll}
[\![\Gamma]\!] := \{\rho \in \tilde{\mathsf{Co}} \mid & \forall x^+ \in \mathsf{dom}^+(\rho),\ \rho\,[x^+] \in [\![\Gamma\,[x^+]]\!] \\
 & \forall x^- \in \mathsf{dom}^-(\rho),\ \rho\,[x^-] \in [\![\Gamma\,[x^-]]\!]^- \quad \}
\end{array}$$

Now that we have defined the interpretation of terms and the interpretation of types, we get to the Adequacy Lemma.

Lemma 3 (Adequacy for LAF) We assume the following hypotheses:

Well-foundedness: The LAF instance is well-founded.

Typing correlation: If $\rho \in [\![\Gamma]\!]$ and $\mathfrak{d} \in [\![\Delta]\!]$ then $(\rho;\mathfrak{d}) \in [\![\Gamma;\Delta]\!]$.

Stability: If $\mathfrak{d} \in [\![\Delta]\!]$ for some Δ and $[\![f(p)]\!]_{\rho;\mathfrak{d}} \in \perp$, then $[\![f]\!]_\rho \perp \tilde{p}(\mathfrak{d})$.

We conclude that, for all $\rho \in [\![\Gamma]\!]$,

1. if $\Gamma \vdash [t^+{:}M]$ then $[\![t^+]\!]_\rho \in [\![M]\!]^+$;
2. if $\Gamma \vdash d{:}\Delta$ then $[\![d]\!]_\rho \in [\![\Delta]\!]$;
3. if $\Gamma \vdash t$ then $[\![t]\!]_\rho \in \perp$.

Proof: See the proof in Coq [GL14b]. □

Looking at the Adequacy Lemma, the stability condition is traditional: it is the generalisation, to that level of abstraction, of the usual condition on the orthogonality relation in orthogonality models (those realisability models that are defined in terms of orthogonality, usually to model classical proofs [Gir87, DK00, Kri01, MM09]): orthogonality is "closed under anti-reduction". Here, we have not defined a notion of reduction on LAF proof-terms, but intuitively, we would expect to rewrite $\langle f \mid pd\rangle$ to $f(p)$ "substituted by d".

On the other hand, the typing correlation property is new, and is due to the level of abstraction we operate at: there is no reason why our data structure for typing contexts would relate to our data structure for semantic contexts, and the extension operation, in both of them, has so far been completely unspecified. Hence, we clearly need such an assumption to relate the two.

However, one may wonder when and why the typing correlation property should be satisfied. One may anticipate how typing correlation could hold for the instance LAF_{K1} of LAF. Definition 7 suggests that, in the definition of a typing context algebra, the extension operation does not depend on the nature of the atom a or molecule M that is being added to the context. So we could *parametrically* define $(\mathscr{A},\mathscr{B})$-contexts for any sets $\mathscr{A}$ and $\mathscr{B}$ (in the sense of relational parametricity [Rey83]). The typing context algebra would be the instance where $\mathscr{A} = \mathbb{A}$ and $\mathscr{B} = \mathbb{M}$ and the semantic context algebra would be the instance where $\mathscr{A} = \mathscr{L}$ and $\mathscr{B} = \mathscr{N}$. Parametricity of context extension would then provide the typing correlation property.

5 Example: boolean models to prove Consistency

We now exhibit models to prove the consistency of LAF systems.

We call *boolean realisability algebra* a realisability algebra where $\perp\!\!\!\perp = \emptyset$. The terminology comes from the fact that in such a realisability algebra, $[\![M]\!]^-$ can only take one of two values: $\emptyset$ or $\mathscr{N}$, depending on whether $[\![M]\!]^+$ is empty. A boolean realisability algebra satisfies Stability.

Theorem 4 (Consistency of LAF instances) Assume we have a well-founded LAF instance, and a boolean realisability algebra for it, where typing correlation holds and there is an empty semantic context $\rho_\emptyset$. There is no empty typing context $\Gamma_\emptyset$ and command t such that $\Gamma_\emptyset \vdash t$.

Proof: The previous remark provides Stability. If there was such a $\Gamma_\emptyset$ and t, then we would have $\rho_\emptyset \in [\![\Gamma_\emptyset]\!]$, and the Adequacy Lemma (Lemma 3) would conclude $[\![t]\!]_{\rho_\emptyset} \in \emptyset$. □

We provide such a realisability model that works with all "parametric" LAF instances:

Definition 15 (Trivial model for parametric LAF instances)

Assume we have a parametric LAF instance, i.e. an instance where the typing context algebra Co is the instance $\mathscr{G}_{\mathbb{A},\mathbb{M}}$ of a family of context algebras $(\mathscr{G}_{\mathscr{A},\mathscr{B}})_{\mathscr{A},\mathscr{B}}$ whose notion of extension is defined parametrically in $\mathscr{A},\mathscr{B}$. The *trivial boolean model* for it is:

$$\begin{array}{rrl}
\mathscr{L} := \quad \mathscr{P} := & \mathscr{N} & := \mathtt{unit} \\
 & \perp\!\!\!\perp & := \emptyset \\
\forall \mathfrak{d} \in \tilde{\mathbb{D}}, & \tilde{p}(\mathfrak{d}) & := () \\
\forall f : \mathsf{Pat} \rightharpoonup \mathsf{Terms}, \forall \rho \in \tilde{\mathsf{Co}}, & [\![f]\!]_\rho & := () \\
\forall a \in \mathbb{A}, & [\![a]\!] & := \mathtt{unit}
\end{array}
\qquad
\begin{array}{lrl}
\tilde{\mathsf{Co}} := \mathscr{G}_{\mathtt{unit},\mathtt{unit}} & & \\
\text{and therefore} & & \\
\forall \rho \in \tilde{\mathsf{Co}}, \forall x^+ \in \mathsf{dom}^+(\rho), & \rho\,[x^+] & := () \\
\forall \rho \in \tilde{\mathsf{Co}}, \forall x^- \in \mathsf{dom}^-(\rho), & \rho\,[x^-] & := ()
\end{array}$$

Note that, not only can $[\![M]\!]^-$ only take one of the two values $\emptyset$ or $\mathtt{unit}$, but $[\![M]\!]^+$ can also only take one of the two values $\emptyset$ or $\mathtt{unit}$.

We can now use such a structure to derive consistency of parametric LAF instances:

Corollary 5 (Consistency for parametric LAF instances) Assume we have a parametric LAF instance that is well-founded and assume there is an empty $(\mathtt{unit},\mathtt{unit})$-context in $\mathscr{G}_{\mathtt{unit},\mathtt{unit}}$. Then there is no empty typing context $\Gamma_\emptyset$ and command t such that $\Gamma_\emptyset \vdash t$.
In particular, this is the case for LAF_{K1}.

The system LAF_J does not fall in the above category since the operation of context extension is not parametric enough: when computing $\Gamma; a$ (resp. $\Gamma; \sim M$), we have to make a case analysis on whether a is of the form (l^+, r) or (v, l) (resp. whether M is of the form (N, l) or (P, r)).

But we can easily adapt the above trivial model into a not-as-trivial-but-almost model for LAF_J, as is shown in [GL14a], Ch. 6.

6 Conclusion and Further Work

6.1 Contributions

In this paper we have used, and slightly tweaked, a system with proof-terms proposed by Zeilberger for *big-step focussing* [Zei08a], which abstracts away from the idiosyncrasies of logical connectives and (to some extent) logical systems: In particular we have shown how two focussed sequent calculi of the literature, namely LKF and LJF [LM09], are captured as instances of this abstract focussing system LAF.

Building on Munch-Maccagnoni's description [MM09] of classical realisability in the context of polarised classical logic and focussing, we have then presented the realisability models for LAF, thus providing a generic approach to the denotational semantics of focussed proofs. Central to this is the Adequacy Lemma 3, which connects typing and realisability, and our approach is generic in that the Adequacy Lemma is proved once and for all, holding for all focussed systems that can be captured as LAF instances.

Incidently, a by-product of this paper is that we provided proof-term assigments for LKF and LJF, and provided realisability semantics for their proofs. We believe this to be new.

But showing the Adequacy Lemma at this level of abstraction was also motivated by the will to exhibit how close typing and realisability are while differing in an essential way:

6.2 Typing vs. realisability

Concerning the *positives*, typing and realisability simply mimic each other: Ignoring contexts,

- in typing, $\vdash [t^+ : M]$ means t^+ is of the form pd with $\Delta \Vdash p : M$ and $\vdash d : \Delta$ for some Δ;
- in realisability, $[\![t^+]\!] \in [\![M]\!]^+$ means $[\![t^+]\!]$ is of the form $\tilde{p}(\mathfrak{d})$ with $\Delta \Vdash p : M$ and $\mathfrak{d} \in \Delta$ for some Δ.

Concerning the *negatives*, it is appealing to relate the quantification in rule async with the quantification in the orthogonality construction:

- in typing, $\vdash f : \sim M$ means that for all p and Δ such that $\Delta \Vdash p : M$, we have $;\Delta \vdash f(p)$ (Δ extending the empty typing context);
- in realisability, $[\![f]\!] \in [\![M]\!]^-$ means that for all p and Δ such that $\Delta \Vdash p : M$, for all $\mathfrak{d} \in [\![\Delta]\!]$ we have $[\![f]\!] \perp \tilde{p}(\mathfrak{d})$, usually obtained from $[\![f(p)]\!]_{;\mathfrak{d}} \in \perp$ ($\mathfrak{d}$ extending the empty semantic context).

In both cases, a "contradiction" needs to be reached for all p and Δ decomposing M. But in typing, the proof-term $f(p)$ witnessing the contradiction can only depend on the pattern p and must treat its holes (whose types are agregated in Δ) parametrically, while in realisability, the reason why $[\![f]\!] \perp \tilde{p}(\mathfrak{d})$ holds, though it may rely on the computation of $f(p)$, can differ for every $\mathfrak{d} \in [\![\Delta]\!]$. It is the same difference that lies between the usual rule for $\forall$ in a Natural Deduction system for arithmetic, and the ω-rule [Hil31, Sch50]:

$$\frac{A}{\forall n A}\ \forall\text{-intro} \qquad \frac{\{^0/_n\}A \quad \{^1/_n\}A \quad \{^2/_n\}A \quad \ldots}{\forall n A}\ \omega$$

The difference explains why typing is (usually) decidable, while realisability is not, and contributes to the idea that "typing is but a decidable approximation of realisability".

We believe that we have brought typing and realisability as close as they could get, emphasising where they coincide and where they differ, in a framework stripped from the idiosyncrasies of logics, of their connectives, and of the implementation of those functions we use for witnessing refutations, i.e. inhabiting negatives (e.g. the λ of λ-calculus).

6.3 Coq formalisation and additional results

In this paper we have also exhibited simple realisability models to prove the consistency of LAF instances.

The parameterised LAF system has been formalised in Coq [GL14b], together with realisability algebras. The Adequacy Lemma 3 and the Consistency result (Corollary 5) are proved there as well. Because of the abstract level of the formalism, very few concrete structures are used, only one for $(\mathscr{A},\mathscr{B})$-decompositions and one for proof-terms; rather, Coq's *records* are used to formalise the algebras used throughout the paper, declaring type fields for the algebras' support sets, term fields for operations, and proof fields for specifications. Coercions between records (and a few structures declared to be canonical) are used to lighten the proof scripts.

Besides this, the Coq formalisation presents no major surprises. It contributes to the corpus and promotion of machine-checked theories and proofs. However, formalising this in Coq was a particularly enlightening process, directly driving the design and definitions of the concepts. In getting to the essence of focussing and stripping the systems from the idiosyncrasies of logics and of their connectives, Coq was particularly useful: Developing the proofs led to identifying the concepts (e.g. what a typing context is), with their basic operations and their minimal specifications. Definitions and hypotheses (e.g. the three hypotheses of the Adequacy Lemma) were systematically weakened to the minimal form that would let proofs go through. Lemma statements were identified so as to exactly fill-in the gaps of inductive proofs, and no more.

The formalisation was actually done for a *first-order* version of LAF, that is fully described in [GL14a]. That in itself forms a proper extension of Zeilberger's systems [Zei08a]. In this paper though we chose to stick to the propositional fragment to simplify the presentation.

Regarding realisability models, more interesting examples than those given here to prove consistency can obviously be built to illustrate this kind of semantics. In particular, realisability models built from the term syntax can be used to prove normalisation properties of LAF, as shown in [GL14a]. Indeed, one of the appeals of big-step focussing systems is an elegant notion of cut-reduction, based on rewrite rules on proof-terms and with a functional programming interpretation in terms of *pattern-matching*. A cut-reduction system at the abstraction level of LAF is given in [GL14a], in terms of an abstract machine (performing head-reduction). A command t is evaluated in an evaluation context ρ; denoting such a pair as $\langle\!\langle\, t \mid \rho \,\rangle\!\rangle$, we have the main reduction rule (where d' stands for the evaluation of d in the evaluation context ρ):

$$\langle\!\langle\, \langle f \mid pd \rangle \mid \rho \,\rangle\!\rangle \longrightarrow \langle\!\langle\, f(p) \mid \rho; d' \,\rangle\!\rangle$$

Normalisation of this system (for a well-founded LAF instance), is proved by building a syntactic realisability model, in which orthogonality holds when the interaction between a negative denotation and a positive one is normalising. This model, together with the head normalisation result, are also formalised in Coq [GL14b]. It forms a formal connection, via the orthogonality techniques, between proofs of normalisation *à la* Tait-Girard and realisability. From this termination result, an informal argument is proposed [GL14a] to infer cut-elimination, but the argument still needs to be formalised in Coq. This is tricky since, cut-elimination needing to be performed arbitrarily deeply in a proof-tree ("under lambdas"), we need to formalise a notion of reduction on those functions we use for witnessing refutations, for which we have no syntax.

Finally, more work needs to be done on formalising the connections between LAF and other related systems: firstly, LAF is very strongly related to *ludics* [Gir01], a system for big-step focussing for linear logic, and which is also related to game semantics. LAF can be seen as a non-linear variant of ludics, our proof-term-syntax more-or-less corresponding to ludics' *designs*. But in order to get linearity, LAF would need to force it in the typing rules for the decomposition terms x^+, $\bullet$, and d_1, d_2. It would also be interesting to investigate whether or how LAF could be adapted to modal logics.

References

[And92] J. M. Andreoli. Logic programming with focusing proofs in linear logic. *J. Logic Comput.*, 2(3):297–347, 1992. DOI:`10.1093/logcom/2.3.297`

[DJS95] V. Danos, J.-B. Joinet, and H. Schellinx. LKQ and LKT: sequent calculi for second order logic based upon dual linear decompositions of classical implication. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Proc. of the Work. on Advances in Linear Logic*, volume 222 of *London Math. Soc. Lecture Note Ser.*, pages 211–224. Cambridge University Press, 1995.

[DK00] V. Danos and J.-L. Krivine. Disjunctive tautologies as synchronisation schemes. In P. Clote and H. Schwichtenberg, editors, *Proc. of the 9th Annual Conf. of the European Association for Computer Science Logic (CSL'00)*, volume 1862 of *LNCS*, pages 292–301. Springer-Verlag, 2000. DOI:`10.1007/3-540-44622-2_19`

[DL07] R. Dyckhoff and S. Lengrand. Call-by-value λ-calculus and LJQ. *J. Logic Comput.*, 17:1109–1134, 2007. DOI:`10.1093/logcom/exm037`

[Gir87] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987. DOI:`10.1016/0304-3975(87)90045-4`

[Gir91] J.-Y. Girard. A new constructive logic: Classical logic. *Math. Structures in Comput. Sci.*, 1(3):255–296, 1991. DOI:`10.1017/S0960129500001328`

[Gir01] J. Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001. DOI:`10.1017/S096012950100336X`

[GL14a] S. Graham-Lengrand. *Polarities & Focussing: a journey from Realisability to Automated Reasoning*. Habilitation thesis, Université Paris-Sud, 2014. Available at `http://hal.archives-ouvertes.fr/tel-01094980`

[GL14b] S. Graham-Lengrand. Polarities & focussing: a journey from realisability to automated reasoning – Coq proofs of Part II, 2014. `http://www.lix.polytechnique.fr/~lengrand/Work/HDR/`

[Hil31] D. Hilbert. Die Grundlegung der elementaren Zahlenlehre. *Mathematische Annalen*, 104:485–494, 1931.

[Kle45] S. Kleene. On the interpretation of intuitionistic number theory. *J. of Symbolic Logic*, 10:109–124, 1945. DOI:`10.2307/2269016`

[Kri01] J.-L. Krivine. Typed lambda-calculus in classical Zermelo-Frænkel set theory. *Arch. Math. Log.*, 40(3):189–205, 2001. DOI:`10.1007/s001530000057`

[LM09] C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoret. Comput. Sci.*, 410(46):4747–4768, 2009. DOI:`10.1016/j.tcs.2009.07.041`

[MM09] G. Munch-Maccagnoni. Focalisation and classical realisability. In E. Grädel and R. Kahle, editors, *Proc. of the 18th Annual Conf. of the European Association for Computer Science Logic (CSL'09)*, volume 5771 of *LNCS*, pages 409–423. Springer-Verlag, 2009. DOI:`10.1007/978-3-642-04027-6_30`

[Rey83] J. C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Proc. of the IFIP 9th World Computer Congress - Information Processing*, pages 513–523. North-Holland, 1983.

[Sch50] K. Schütte. Beweistheoretische Erfassung der unendlichen Induktion in der Zahlentheorie. *Mathematische Annalen*, 122:369–389, 1950.

[Zei08a] N. Zeilberger. Focusing and higher-order abstract syntax. In G. C. Necula and P. Wadler, editors, *Proc. of the 35th Annual ACM Symp. on Principles of Programming Languages (POPL'08)*, pages 359–369. ACM Press, 2008. DOI:`10.1145/1328438.1328482`

[Zei08b] N. Zeilberger. On the unity of duality. *Ann. Pure Appl. Logic*, 153(1-3):66–96, 2008. DOI:`10.1016/j.apal.2008.01.001`