Ideas in Logic and Computer-Science related to Ludics

Misao Nagayama (永山 操)
Tokyo Woman's Christian Univ. (東京女子大学文理学部)
misao@twcu.ac.jp

Abstract

This note reports author's current thoughts and observations in logic and computer-science related to ludics.

1 Motivations.

When we work in proof theory, our basic objects are proofs and formulas. A proof consists of a series of inference rules: In order to successfully accumulate inference rules up to the axioms, which is supposed to the goal, we normally develop a heuristics. It is to become a proof-search algorithm. Such algorithms are often embedded into meta-mathematical proofs for the completeness theorems. We can name some proof-search algorithms, for example, resolution and unification for propositional and first-order logic, Skolem's theorem and Herbrand's theorem for first-order logic, and focalization for linear logic.

We would like to investigate such proof-search algorithms in terms of conputational relevance. We like to develop some formal system in which these algorithms themselves are to be investigated meta-mathematically. We expect such a formal system to provide

- a more general and abstract notion extending traditional notions in logic, and
- a concretion of proof-search algorithms.

We could see λ-calculus as such a formal system. Emphasized in a phrase "terms as programs", λ-terms are not only an abstraction of mathematical functions, but also concrete programs we know how to evaluate them. In summary, the λ-terms are both

- a more general and abstract notion of mathematical functions, and
- a concretion of programs.

We hope for a similar formal system not for mathematical functions but for proof-search algorithms. Based on this idea we study ludics introduced by Girard ([Gi]). Our topics are organized as follows:

- Section 2: Inference rules.
- Section 3: Axioms.
- Section 4: Variables.
- Section 5: Proofs.
- Section 6: Formulas.
- Section 7: Normalization.

2 Inference rules.

We started our discussion without mentioning any particular formal system. In order to analyze proofs further, we choose one specific formal system, namely Sequent calculus. Sequent calculus is equipped with an axiom scheme of simple form $A \vdash A$, and inference rules: the structural rules, and the left introduction and the right introduction rules for the logical connectives. We simply refer the right introduction rule for a logical connective as its inference rule.

Sequent calculus is preferred because of its internal completeness, namely the cut-elimination theorem. Thanks to the theorem, the search space for proofs becomes much smaller. An application of the last inference rule introduces the outermost logical connective. A bottom-up proof-search in a cut-free system boils down to find the last rule. One step to find the last rule can be thought as a partial function which provides an inference rule for a given ordered sequence of inference rules up to the current point. In fact, this idea leads us to the notion of innocent functions in game semantics ([HO],[FH]). We discuss their work in the last section.

In linear logic, the logical connectives $\&$, $\mathcal{P}$ and $\forall$ are negative, while $\oplus$, $\otimes$ and $\exists$ are positive. Let us call an inference rule for a negative connective and for a positive one by a negative rule and by positive one, respectively. The notion of the polarity arises in the following aspects:

- The law of distributivity,
- the reversibility and determinism,
- focalization and the priority in proof-search algorithms.

The law of distributivity holds:

$$\begin{gathered}A \otimes (B \oplus C) \sim (A \otimes B) \oplus (A \otimes C)\\ \exists x(A \otimes B(x)) \sim A \otimes \exists x B(x)\\ A \mathcal{P} (B \& C) \sim (A \mathcal{P} B) \& (A \mathcal{P} C)\\ \forall x(A \mathcal{P} B(x)) \sim A \mathcal{P} \forall B(x).\end{gathered}$$

In linear logic, the negative rules are deterministic and reversible: For case of &, the inference rule is:

$$\frac{\vdash A, \Gamma \quad \vdash B, \Gamma}{\vdash A \& B, \Gamma}\ (\&).$$

Conversely, we can prove each premise from the conclusion. In the following, premise $\vdash A, \Gamma$ is derived.

$$\frac{\vdash A \& B, \Gamma \quad \dfrac{\vdash A, A^{\perp}}{\vdash A, A^{\perp} \oplus B^{\perp}}\ (\oplus)}{\vdash A, \Gamma}\ (Cut).$$

For case of $\mathcal{P}$, the inference rule is:

$$\frac{\vdash A, B, \Gamma}{\vdash A \mathcal{P} B, \Gamma}\ (\mathcal{P}).$$

Similarly, we can prove the premise from the conclusion:

$$\frac{\vdash A \mathcal{P} B, \Gamma \quad \dfrac{\vdash A^{\perp}, A \quad \vdash B^{\perp}, B}{\vdash A, B, A^{\perp} \otimes B^{\perp}}\ (\otimes)}{\vdash A, B, \Gamma}\ (Cut).$$

For case of $\forall$,

$$\frac{\vdash A[y/x], \Gamma}{\vdash \forall x A, \Gamma}\ (\forall),$$

where variable y is not free in Γ. We can prove the premise from the conclusion.

$$\frac{\vdash \forall x A, \Gamma \quad \dfrac{\vdash A[y/x]^{\perp}, A[y/x]}{\vdash \exists x A^{\perp}, A[y/x]}\ (\exists)}{\vdash A[y/x], \Gamma}\ (\&).$$

On the other hand, the positive rules are non-deterministic and irreversible.

The focalization allows us to treat a cluster of negative rules, and a cluster of positive ones, as *synthetic* connectives. We give a higher priority to the negative rules than to the positive ones. Thus we have the following proof search algorithm:

- Choose one of the formulas in the conclusion and a negative rule: Apply the rule and decompose the formula. Notice that this is deterministic, and the other formulas in the context remain the same.
- Continue choosing a negative rule, until it is no longer available in the conclusion.
- Choose one of the formulas in the conclusion and a positive rule: Apply the rule and decompose the formula. Notice that this is non-deterministic, the context may split in a non-recoverable manner (in the case of $\otimes$), or one of the subformulas may be lost (in the case of $\oplus$).

As for the negativity of the universal quantifier $\forall$, let us consider a provable formula in first-order classical logic

$$\exists x \forall y (P(x) \to P(y)).$$

We build its cut-free proof in classical sequent calculus in a bottom-up way. Then we soon realize that we need to defer our irreversible choice to avoid making a crucial error, instead, to apply the Contraction rule. Thus we eliminate $\exists$ in one of the two formulas in the conclusion. Then we are left with two formulas, whose outermost quantifiers are $\forall$ and $\exists$, respectively:

$$\dfrac{\dfrac{\vdash \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}{\vdash \exists x \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\exists)}{\vdash \exists x \forall y(P(x) \to P(y))}\,(Cont.)$$

Which quantifier should we eliminate first, $\forall$ or $\exists$? Our proof-search algorithm above tells us that we should choose a negative connective when possible, which is $\forall$, in this case. Thus we further proceed the derivation:

$$\dfrac{\dfrac{\dfrac{\vdash P(x) \to P(z), \exists x \forall y(P(x) \to P(y))}{\vdash \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\forall)}{\vdash \exists x \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\exists)}{\vdash \exists x \forall y(P(x) \to P(y))}\,(Cont.)$$

Introducing a fresh variable z in the top inference rule is very tricky; there seems no particular reason to do this at this point. We think that this is an example of variables used as an anonymous representative. We think that Sequent calculus does not have a capacity to describe how to introduce a fresh variable and to use it effectively.

After this crucial step, we need to eliminate $\exists$ in the left formula using the variable

z, which is again a little tricky:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash P(x) \to P(z), \forall y(P(z) \to P(y))}{\vdash P(x) \to P(z), \exists x \forall y(P(x) \to P(y))}\,(R\exists)}{\vdash \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\forall)}{\vdash \exists x \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\exists)}{\vdash \exists x \forall y(P(x) \to P(y))}\,(Cont.)$$

The last step is matching so that a pair of $P(z)$ should later be used to supply an axiom. Matching is completed by the positive rule for $\exists$. This is an irreversible and non-deterministic step, leading us to the successful end of the proof. Moreover, we can interpret this step in a game semantic manner: The initial negative rule (Opponent) offers a function $P(z)$, or options $P(z)$ (in the sense that different z's provide various options), and the next positive rule (Player) responds to it by supplying $P(z)$ by a suitable variable. Here the variable z is chosen. Finally, we conclude our proof as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(z) \vdash P(z)}{P(x), P(z) \vdash P(z), P(y)}\,(Weak.)}{P(x) \vdash P(z), P(z) \to P(y)}\,(R \to)}{\vdash P(x) \to P(z), P(z) \to P(y)}\,(R \to)}{\vdash P(x) \to P(z), \forall y(P(z) \to P(y))}\,(R\forall)}{\vdash P(x) \to P(z), \exists x \forall y(P(x) \to P(y))}\,(R\exists)}{\vdash \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\forall)}{\vdash \exists x \forall y(P(x) \to P(y)), \exists x \forall y(P(x) \to P(y))}\,(R\exists)}{\vdash \exists x \forall y(P(x) \to P(y))}\,(Cont.)$$

It will be interesting to investigate further a computational aspect of the polarity in proof-search algorithms. In ludics, the inference rules are introduced not based on the left or right, but based on the polarity.

- Daimon:

$$\dfrac{}{\vdash \Lambda}\,\dagger$$

- Positive rule (one premise for each $i \in I$, Λ_i's pairwise disjoint and included in Λ):

$$\dfrac{\cdots \xi i \vdash \Lambda_i \cdots}{\vdash \xi, \Lambda}\,(+, \xi, I)$$

- Negative rule (one premise for each $J \in \mathcal{N}$, all Λ_J's included in Λ):

$$\dfrac{\cdots \vdash \xi * J, \Lambda_J, \cdots}{\xi \vdash \Lambda}\,(-, \xi, \mathcal{N})$$

The basic objects in ludics are *designs*. Syntactically, designs are trees built from the rules above.

3 Axioms.

We observe the following two distinctive roles, syntactical and semantical, played by axioms normally in any proof system:

- Syntactical: To denote the terminals or initials of accumulated sequences of inference rules, and
- Semantical: To denote concepts taken for granted, such as valid formulas, typically the identity $A \vdash A$.

The first property leads us to two positive designs interpreted as terminals in ludics:

- *Daimon* $\mathfrak{Dai}$, consisting of † itself,

  $$\overline{\vdash \Lambda}\ \dagger,$$

  which means convergence, while

- the *partial design* Ω, the empty set,

  $$\overline{\vdash \Lambda}\ \Omega,$$

  which means divergence.

Besides being designs themselves, each daimon and Ω can be used, just as an axiom, as a rule to end a tree of inference rules at top of the tree.

The daimon † and the partial design Ω are unique maximal and minimal designs with respect to the observational order for the designs [Gi].(See also [C](p.15) for its formal definition.)

Besides such an algebraic treatment, Curien emphasizes a similarity between recoverable errors in computer science and daimon in ludics: Errors and daimon help us to terminate computation and to interactively explore the behavior of programs or proofs. Moreover, computation is stream-like, or demand-driven. This view helps us to better understand the observational order $\preceq$ in Girard's notation, in such a way that for designs $\mathfrak{D}_1$ and $\mathfrak{D}_2$, $\mathfrak{D}_1 \preceq \mathfrak{D}_2$ holds iff $\mathfrak{D}_2$ is more likely to terminate or converge in computation than $\mathfrak{D}_1$ is. Furthermore, the observational order allows us to analyze a structural difference between the two designs it orders. For example, a greater design $\mathfrak{D}_2$ can be obtained from $\mathfrak{D}_1$ by as follows; we replace either

- an Ω in $\mathfrak{D}_1$ with a tree $\neq \Omega$, or
- a sub-tree in $\mathfrak{D}_1$ by a $\dagger$,

to obtain $\mathfrak{D}_2$.

However, replacing "axiom-like" objects freely by something else seems against the second role of the axioms; namely, axioms denote something valid.

As for the gap we just mentioned, let us examine an example in semantics of Intutionistic predicate logic, which is taken from a text by Ono ([O]).

Let $(M, \leq, U, \models)$ be a Kripke model satisfying the following conditions:

1. $(M, \leq)$ is an ordered set,
2. $U : M \longrightarrow \mathcal{P}(W)$ is a map from M to the power set of some set W, satisfying the two conditions;
   (a) for any $a \in M$, $U(a) \neq \emptyset$ holds,
   (b) for any $a, b \in M$, $a \leq b$ implies $U(a) \subseteq U(b)$.
3. $\models$ is a binary relation between an element $a \in M$ and a closed formula A such that
   (a) $a \models P(\underline{u}_1, \cdots, \underline{u}_m) \Longleftrightarrow (\underline{u}_1, \cdots, \underline{u}_m) \in P^{I(a)}$,
   (b) $a \models A \wedge B \Longleftrightarrow a \models A$ and $a \models B$,
   (c) $a \models A \vee B \Longleftrightarrow a \models A$ or $a \models B$,
   (d) $a \models A \rightarrow B \Longleftrightarrow \forall b (a \leq b \Rightarrow (b \not\models A$ or $b \models B))$,
   (e) $a \models \neg A \Longleftrightarrow \forall b (a \leq b \Rightarrow b \not\models A)$,
   (f) $a \models \forall x A \Longleftrightarrow \forall b \forall u (a \leq b \Rightarrow (u \in U(b) \Rightarrow b \models A[\underline{u}/x]))$,
   (g) $a \models \exists x A \Longleftrightarrow \exists u (u \in U(b) \wedge a \models A[\underline{u}/x])$,

   where $I(a)$, for each $a \in M$, of the interpretation of the predicate symbols I is defined so that (1) $P^{I(a)} \subseteq U(a)^m$ and (2) $a \leq b$ implies $P^{I(a)} \subseteq P^{I(b)}$; and $\underline{u}$ stands for the name of u.

A formula A is valid in the frame $(M, \leq, U)$ iff it is true in the Kripke model $(M, \leq, U, \models)$ for any valuation $\models$.

Let M be a set satisfying the condition that for any $a \in M$, there exists a maximal element $a^*(\geq a)$ in M. Then formula

$$\forall x \neg\neg P(x) \rightarrow \neg\neg \forall x P(x)$$

is valid in the frame $(M, \leq, U)$. Let us demonstrate how to prove the validity of this formula. Firstly,

$$(1)\ \ b \models \forall x \neg\neg P(x)$$

is equivalent to

$$(1)'\ \ \forall c \forall u \forall d \exists e (b \leq c \Rightarrow (u \in U(c) \Rightarrow (c \leq d \Rightarrow (d \leq e \wedge e \models P[\underline{u}/x])))).$$

And

$$(2)\ \ b \models \neg\neg\forall P(x)$$

is equivalent to

$$(2)'\ \ \forall c \exists d \forall e \forall u (b \leq c \Rightarrow (c \leq d \wedge (d \leq e \Rightarrow (u \in U(c) \Rightarrow e \models P[\underline{u}/x])))).$$

We assume condition (1) and imply condition (2). The condition (2)' is of prenex form whose quantifiers are $\forall c \exists d \forall e \forall u$, and for any c instantiating the first quantifier $\forall c$, the second existential quantifier is ought to be instantiated by c^*. Thanks to the maximality of c^*, the next $\forall$ is trivially instantiated only by c^*. Thus the condition (2)' boils down to

$$(2)''\ \forall u (u \in U(c^*) \Rightarrow c^* \models P[\underline{u}/x]).$$

On the other hand, the condition (1)' is of prenex form whose quantifiers are $\forall c \forall u \forall d \exists e$, and the first quantifier $\forall c$ is ought to be instantiated by c^*. Again, thanks to the maximality of c^*, the rest of the quantifiers are all trivially instantiated only by c^*. Therefore the condition (1)' boils down to

$$(1)''\ \forall u (u \in U(c^*) \Rightarrow c^* \models P[\underline{u}/x]).$$

Clearly (1)" and (2)" are identical, and we have shown the claim.

Notice that the role played by the maximal element c^* reminds us the role of daimon. It is maximal, and it stops further inquiries for instantiating quantifiers in a non-trivial way. Moreover, instantiating a universal quantifier in (1) and an existential quantifier in (2) syntactically corresponds to (L $\forall$) and (R $\exists$), respectively in Sequent calculus: The polarities of these rule are both positive, as same as the polarity of daimon. It will be interesting to study further proof-search algorithms to check the validity of formulas in the Kripke frame, and to investigate its computational relevance.

Finally, the second role of the axiom leads us to a design called *Fax*, intended as the identity, or rather its infinite η-expansion in ludics. It plays a role of a function mapping one formula to the other isomorphic one. Fax $\mathfrak{Fax}_{\xi,\xi'}$ is the following design:

$$\dfrac{\dfrac{\cdots \xi' * i \vdash \xi * i \cdots \quad \vdots\, \mathfrak{Fax}_{\xi' * i, \xi * i}}{\cdots \vdash \xi', \xi * I \cdots}\,(\xi', I)}{\xi \vdash \xi'}\,(\xi, \mathcal{P}_f(\mathcal{N}))$$

See Faggian et als. [FFDQ] for further discussions on Fax.

4 Variables.

In [AC] by Amadio and Curien, it is explained that the categorical interpretation of λ-calculus in CCC can be seen as a way of compiling a language with variables into a languages without them. The slogan there is that variables are replaced by projections. In other words, rather than giving a symbolic reference in the form of variable, one provides a path for accessing a certain information in the context. This is the starting point to define an abstract machine, which provides a high-level description of data-structures and algorithms used to efficiently reduce λ-terms.

In this section, we discuss a well-known partial algorithm to check the validity of a formula based on Skolem's and Herbrand's theorems. We think that the combination of two theorems provides us a usage of variables in logic, which seems to share the same perspective mentioned in the slogan above. As for the formulations of the theorems, we shall again count on the text by Ono ([O]).

In first-order predicate logic, Skolem's theorem provides us, for a given formula, a prenex existential formula called *Skolem normal form*: It is valid iff so is the original one, provided that we consider the former validity in the models for an extended language with new constants and function symbols. A formula

$$\forall x \exists y \exists z \forall w P(x, y, z, w)$$

has a Skolem normal form

$$\exists y \exists z P(c, y, z, f(y, z)),$$

In the Skolem normal form, the universal quantifiers in the original formula disappear, and the quantified variables x, w are substituted for the terms made of newly introduced a 0-ary constant c and a binary function symbol f, respectively. The parameters of each c and f are determined by the variables existentially quantified, between the one universally quantified at work and the another one, universally quantified on the left, closest to the one at work.

Let $\mathfrak{L}$ a fixed language containing at least one constant symbols. Let *Herbrand universe*, denoted by $H_{\mathfrak{L}}$, be the collection of variable-free terms of $\mathfrak{L}$. *Herbrand structure* $< H_{\mathfrak{L}}, J >$ for language $\mathfrak{L}$ is a structure such that for each variable-free term t in $\mathfrak{L}$ is, via J, interpreted by term t in $H_{\mathfrak{L}}$ itself. Notice that the former t is to be interpreted is a syntactical object, while the latter t in $H_{\mathfrak{L}}$ is a semantical one. Thus the notion of Herbrand structures introduces objects both syntactical and semantical.

Let A be a quantifier-free predicate formula

$$P(s) \rightarrow (Q(s, t) \vee P(t)),$$

where P, Q are predicate symbols unary and binary, respectively, and s, t are terms without variables.

Atomic predicates $P(s)$ and $Q(s,t)$, containing different predicate symbols P an Q respectively, are evaluated independently. In other words, each P and Q refers, just as a pointer, independently to a value in the Herbrand structure. On the other hand, if s,t are different terms, say, containing distinct function symbols, then they are interpreted by distinct elements in the Herbrand universe. Therefore, predicates $P(s)$ and $P(t)$ are evaluated independently. In other words, each s,t refers via its function symbols playing their role as a pointer, independently to a value in the Herbrand structure.

The observation above reminds us the view presented earlier in this section, that is, replacing a symbolic reference in the form of variable, one provides a path for accessing a certain information in the context. In our discussions, we replaced symbolic references in the form of universally quantified variable by a path or a pointer played by function symbols, thanks to the Skolem normalization, referring mutually independently to the values in the Herbrand universe, which is the context in our setting.

The proposition below introduces a decidable algorithm to check the validity of any quantifier-free formula A with no free variables.

Proposition 4.1 *Let A be a quantifier-free formula with no free variable in $\mathfrak{L}$. Then A is valid iff A is true in any Herbrand structure for $\mathfrak{L}$. Moreover, A is valid iff the propositional formula $\pi(A)$ is tautology; where $\pi(A)$ is obtained from A by replacing mutually distinct atomic predicates by corresponding mutually distinct propositional variables.*

5 Proofs.

A proof is built as a tree of inference rules. Normally, a proof-search algorithm provides us how to accumulate inference rules successfully up to the axioms, which is the goal.

Faggian and Hyland describe in [FH] that designs, the basic objects in ludics, are both:

- an abstraction of formal proofs, and
- a concretion of their semantical interpretation.

These two aspects of designs agree with our motivations presented in the first section: That is, we are looking for:

- a more general and abstract notion of formal logic, and
- a concretion of proof-search algorithms.

Let us remember Herbrand's theorem: The theorem allows us to check the validity of closed existential prenex formulas by means of a partial algorithm.

Theorem 5.1 (Herbrand) *Let $\exists x_0 \cdots x_n B$ be a closed existential prenex formula in language $\mathcal{L}$, where formula B contains no quantifier symbol. Then $\exists x_0 \cdots x_n B$ is valid iff there exists a natural number m (≥ 1) and terms $t_{i1}, \cdots, t_{in}$ $(i = 1, \cdots m)$ of Herbrand universe such that*

$$B[t_{11}/x_1, \cdots, t_{1n}/x_n] \vee \cdots \vee B[t_{m1}/x_1, \cdots, t_{mn}/x_n]$$

is true in any Herbrand structure for $\mathcal{L}$.

The notions of the structures, the universe and the validity in the theorem definitely suggest that the theorem be semantical. However, based on the theorem, we obtain a partial algorithm to check the validity of a closed formula: There is where, we believe, semantics and proof-search algorithms meet. A good algorithm is coupled with a good data structure. In our view, a good data structure in the partial algorithm here is a growing list of tuples of variable-free terms in the Herbrand universe: The list starts as the empty set; and the list is added tuples of terms mentioned in the theorem in the course of the partial algorithm up to the list:

x_1	$\cdots$	x_n
t_{11}	$\cdots$	t_{1n}
t_{21}	$\cdots$	t_{2n}
$\cdots$	$\cdots$	$\cdots$
t_{m1}	$\cdots$	t_{mn}

When the list arrives at this stage, the given formula $\exists x_0 \cdots x_n B$ is shown to be valid. It will be interesting to investigate how the above data structure is maintained by the partial algorithm, and if and how it is related to Sequential algorithms due to Berry and Curien ([AC]).

Thus we have started with a proof as a tree of inference rules, and proposed a "proof-search algorithms as semantics" view. In the next section, we shall discuss another aspect of a proof, namely its syntax in Sequent calculus.

6 Formulas.

A proof in Sequent calculus is a tree of sequents; and a sequent contains formulas. One role of formulas in a proof is to denote the location. However, a formula does not simply denote a location, but also denote a proof description, by which we can tell, how to merge two disjoint proofs into one, preserving its correctness. For example, a pair of cut-formulas indicate us how to merge two disjoint proofs via normalization.

The designs are supposed to be an abstraction of formal proofs. and the role of formulas as a proof description in fact is played by designs, which itself is the counterpart of proofs in ludics. In other words, the conceptual distinction between proofs and formulas are less clear in ludics than in the typed λ-calculus, where emphasized are "Formulas as types" and "Proofs as terms" in the Curry-Howard isomorphism.

We shall study the notion of types in ludics by means of filter models, that is, the syntax of intersection types. (Chapter 3.3, p. 54. [AC]).

Definition 6.1 (ets, p. 55, [AC]) *Let $(D, \bullet)$ be an applicative structure. Consider the following operation on subsets of D:*

$$A \to B = \{d \in D : \forall e \in A\ (d \bullet e \in B)\}.$$

A subset of $\mathcal{P}(D)$ is called an extended type structure (ets for short) if it is closed under finite set theoretical intersections and under the operation $\to$ just defined.

An *extended abstract type structure (eats for short)* is given by a preorder $(S, \leq)$, called the carrier, whose elements are often called types: For the definition, refer to Definition 3.3.1 [AC]. The following lemma provides us a means to obtain an eats from an ets defined above.

Lemma 6.2 (Lemma 3.3.7, p. 55, [AC]) *An ets, ordered by inclusion is an eats.*

A filter of an inf-semi-lattice S can be defined in a standard way. (See Definition 3.3.8, p. 56, [AC], for instance.) The filter domain of an eats S is the set $\mathcal{F}(S)$ of filters of S, ordered by inclusion.

Lemma 6.3 (Lemma 3.3.10, p. 56, [AC]) *If S is an eats, then $\mathcal{F}(S)$ is a complete algebraic lattice.*

In ludics, we have an applicative structure defined by the collection of designs denoted by T, equipped with binary operation $Plays(\mathfrak{D}; \mathfrak{E})$. As a subset of $\mathcal{P}(T)$, we take *behaviors*, where a behavior is a set of designs equal to its biorthogonal. As for discussions on the behaviors, we follow [Gi].

The behaviors of the same base is closed under the set theoretical intersection.

Definition 6.4 (Inter, [Gi]) *Let $\mathbf{G}_k$ be a family of behaviors of the same base. Then we define $\bigcap_k \mathbf{G}_k$ as the intersection of the $\mathbf{G}_k$.*

The connective $\bigcap$ is strictly commutative and associative.

Theorem 6.5 (Theorem 14, [Gi]) *Let $\mathfrak{F}$ and $\mathfrak{A}$ be negative and positive designs, respectively. There there exists a unique design $(\mathfrak{F})\mathfrak{A}$, such that for any positive design $\mathfrak{B}$, the following holds.*

$$\ll \mathfrak{F} \mid \mathfrak{A} \odot \mathfrak{B} \gg = \ll (\mathfrak{F})\mathfrak{A} \mid \mathfrak{B} \gg$$

In the theorem above, $\mathfrak{A} \odot \mathfrak{B}$ is the commutative tensor product only defined for positive designs (refer to Definition 31). The binary operation $\multimap$ for ets in ludics is given by a negative behavior $\mathbf{G} \multimap \mathbf{H}$, which is defined as $\mathbf{G} \bowtie \mathbf{H}$ (Definition 34), for two alien behaviors $\mathbf{G}$ positive and $\mathbf{H}$ negative. (As for alienation, see Definition 39.)

$$\mathbf{G} \multimap \mathbf{H} = \{\mathfrak{A} \odot \mathfrak{B}; \mathfrak{A} \in \mathbf{G}, \mathfrak{B} \in \mathbf{H}^{\perp}\}^{\perp}$$

Proposition 6.6 (Proposition 12, [Gi]) *Let behaviors* $\mathbf{G}$ *and* $\mathbf{H}$ *be positive and negative, respectively; then*

$$\mathfrak{F} \in \mathbf{G} \multimap \mathbf{H} \Leftrightarrow \forall \mathfrak{A}(\mathfrak{A} \in \mathbf{G} \Rightarrow (\mathfrak{F})\mathfrak{A} \in \mathbf{H}).$$

Furthermore,

$$\mathfrak{F} \in \mathbf{G} \multimap \mathbf{H} \Leftrightarrow \forall \mathfrak{A}(\mathfrak{A} \in \mathbf{H}^{\perp} \Rightarrow (\mathfrak{F})\mathfrak{A} \in \mathbf{G}^{\perp}).$$

It will be interesting to investigate further an eats structure of the behaviors $\mathbf{BV}$. We discuss a connection between filters of the behaviors and *incarnations*.

Let $\mathfrak{D}$ a fixed design in $\mathbf{G}$. Then the set of designs in $\mathbf{G}$ included in $\mathfrak{D}$ is a non-empty family: The intersection of the family, called the *incarnation*, also belongs to $\mathbf{G}$, due to Closure theorem (Theorem 7).

Definition 6.7 (Incarnation, [Gi]) *Let* $\mathbf{G}$ *and* $\mathfrak{D}$ *be a behavior and its design, respectively. The incarnation* $\mid \mathfrak{D} \mid_{\mathbf{G}}$ *is defined as follows:*

$$\mid \mathfrak{D} \mid_{\mathbf{G}} = \bigcap \{\mathfrak{D}'; \mathfrak{D}' \subset \mathfrak{D} \; and \, \mathfrak{D}' \in \mathbf{G}\}.$$

The incarnation belongs to $\mathbf{G}$.

Now the following proposition is immediate.

Proposition 6.8 (A filter by incarnation.) *A collection* $x_{\mathfrak{D}}$ *of behaviours*

$$x_{\mathfrak{D}} = \{\mathbf{G}; \mid \mathfrak{D} \mid_{\mathbf{G}} \subset \mathfrak{D}\}$$

forms a filter.

Proof. In order to show that $x_{\mathfrak{D}}$ is a filter, we check the following two conditions:

- $\mathbf{G}, \mathbf{H} \in x_{\mathfrak{D}} \Rightarrow \mathbf{G} \cap \mathbf{H} \in x_{\mathfrak{D}}$,
- $\mathbf{G} \in x_{\mathfrak{D}}$ and $\mathbf{G} \subset \mathbf{H} \Rightarrow \mathbf{H} \in x_{\mathfrak{D}}$

The first condition is due to Theorem 9 on the intersection and incarnation: We obtain

$$| \mathfrak{D} |_{\mathbf{G}\cap\mathbf{H}}=| \mathfrak{D} |_{\mathbf{G}} \cup | \mathfrak{D} |_{\mathbf{H}} .$$

The second condition is due to the contravariance easily shown from the definition of the incarnation, i.e.,

$$\mathbf{G} \subset \mathbf{H} \Rightarrow | \mathfrak{D} |_{\mathbf{H}} \subset | \mathfrak{D} |_{\mathbf{G}} .$$

∎

We conjecture that the filters $\mathfrak{F}(\mathbf{BV})$ on behaviors is a complete algebraic lattice. We note that there are maps between $\mathfrak{F}(\mathbf{BV})$ and the collection $\mathbf{T}$ of designs,

$$x : \mathbf{T} \longrightarrow \mathfrak{F}(\mathbf{BV})$$

defined as $x(\mathfrak{D}) = x_{\mathfrak{D}}$, and

$$\mathfrak{D} : \mathfrak{F}(\mathbf{BV}) \longrightarrow \mathbf{T}$$

defined by

$$\mathfrak{D}(x) = \bigcup_{\mathbf{G}\in x} | \mathfrak{D} |_{\mathbf{G}} .$$

It will be interesting to further investigate these maps, a complete algebraic lattice structure in $\mathfrak{F}(\mathbf{BV})$, and its connection to one of the main results, stated in Theorem 10, of the additives in ludics:

$$| \mathbf{G}\&\mathbf{H} |=| \mathbf{G} | \times | \mathbf{H} |,$$

where $| \mathbf{G} |$ is the collection of designs $\mathfrak{D}$ in $\mathbf{G}$, satisfying $\mathfrak{D} =| \mathfrak{D} |_{\mathbf{G}}$.

7 Normalization.

Faggian and Hyland investigate ludics by means of HON game semantics in [FH], which we follow closely in this section. Here we develop syntax of designs as an abstraction of formal proofs. In particular, we pay attention to the following three roles of the formulas.

- A location, relative and absolute.
- An initial sequent.
- A class of proofs characterized by normalization.

We have corresponding notions in ludics, respectively:

- An address.

- A base.
- A behavior.

An *action* is a pair (ξ, I) of an address ξ, called a *focus* and I a finite set of natural numbers. An action corresponds to the application of an inference rule. A *base* is a sequent of addresses corresponding to the initial sequent of the derivation, or the conclusion of the proof.

Thanks to the focalization, it suffices to consider only sequents of form $\Xi \vdash \Lambda$, Ξ consisting at most one element and finite Λ.

A design is given by a base and a tree of actions. A tree of actions can be thought as a set of Sequent calculus branches [FH]. The precise definition is given by a set of mutually *coherent chronicles*, where our intentions are

- Chronicles: A formal branch in a focalized sequent calculus derivation.
- Coherence: A condition to let a set of chronicles all belong to the same proof.

Definition 7.1 (Chronicles, [FH]) *A chronicle of base $\Xi \vdash \Lambda$ is a sequence of actions $< \kappa_0, \kappa_1, \cdots, \kappa_n >$ such that:*

- *Alternation.*
- *Positive and Negative focuses.*
- *Destruction of focuses.*
- *Daimon. Daimon can only appear as the last action.*

Definition 7.2 (Coherence, [FH]) *The chronicles $\mathfrak{c}_1, \mathfrak{c}_2$ are coherent when*

- *Comparability. Either one extends the other, or they first differ on negative actions,*
- *Propagation. If $\mathfrak{c}_1, \mathfrak{c}_2$ first differ on κ_1, κ_2 with distinct focuses, then all ulterior focuses are distinct.*

Definition 7.3 (Designs, [Gi])) *A design $\mathcal{D}$ of base $\Xi \vdash \Lambda$ is a set of chronicles of base $\Xi \vdash \Lambda$ such that:*

- *Arborescence. $\mathfrak{D}$ is closed under restriction.*
- *Coherence. The chronicles of $\mathfrak{D}$ are pairwise coherent.*
- *Positivity. If $\mathfrak{c} \in \mathfrak{D}$ has no extension in $\mathfrak{D}$, then its last action is positive.*

- *Totality. If the base is positive, then $\mathfrak{D}$ is non-empty.*

The actions and interactions of designs correspond to moves and plays in the game semantic setting. For the simplicity, the bases of designs $\mathfrak{D}$ and $\mathfrak{E}$ are $\vdash<>$ and $<>\vdash$, respectively.

Definition 7.4 (Linear positions, [FH]) *A sequence of actions s is a linear position or a play, it it satisfies the following conditions:*

- *Alternation. Parity alternates.*
- *Justification. Each action is either initial or is justified by an earlier action.*
- *Linearity. Any address appears at most once.*
- *Daimon. Daimon can only appear as the last action.*

The notions of chronicles and linear positions are very close.

Definition 7.5 (Plays (binary), [FH]) *We define a play $\mathcal{P}$ denoted as $Plays(\mathfrak{D};\mathfrak{E})$ as follows:*

$$\begin{array}{l} \epsilon \in \mathcal{P} \\ p \in \mathcal{P} \text{ is a } P \text{ to play position and} \lceil p \rceil^P \kappa \in \mathfrak{D}, \text{ then } p\kappa \in \mathcal{P} \\ p \in \mathcal{P} \text{ is an } O \text{ to play position and} \lceil p \rceil^O \kappa \in \mathfrak{E}, \text{ then } p\kappa \in \mathcal{P} \end{array}$$

We note that $Plays(\mathfrak{D};\mathfrak{E})$ is totally ordered by the initial segment. Thus we denote the sup of $Plays(\mathfrak{D};\mathfrak{E})$ by $[\mathfrak{D} \rightleftharpoons \mathfrak{E}]$, which is possibly infinite.

Definition 7.6 (Dispute, Convergence, [FH]) *A sequence of actions $[\mathfrak{D} \rightleftharpoons \mathfrak{E}]$ is called a* dispute*, if it is finite and terminated with a Daimon. The normalization between designs $\mathfrak{D}$ and $\mathfrak{E}$ converges, if $[\mathfrak{D} \rightleftharpoons \mathfrak{E}]$ is a dispute, and diverges, otherwise.*

Definition 7.7 (Orthogonality, [FH]) *A design $\mathfrak{D}$ is called* orthogonal *to $\mathfrak{E}$, when the normalization between them converges. The design $\mathfrak{E}$ is called a counter-design of $\mathfrak{D}$.*

A *legal position* is a linear position satisfying the *visibility* condition (Def.7, [FH]).

Proposition 7.8 (Chronicles, Fact 2, Prop.1 [FH]) *Let p be a linear position in $Plays(\mathfrak{D};\mathfrak{E})$. Then,*

- *for any $q \sqsubseteq r^P \sqsubseteq p$, $\lceil q \rceil^P$ is a chronicle of $\mathfrak{D}$, and similarly,*
- *for any $q \sqsubseteq r^O \sqsubseteq p$, $\lceil q \rceil^O$ is a chronicle of $\mathfrak{E}$.*

- *Moreover p is a legal position.*

Due the last statement in the proposition above, any dispute is a legal position. Thus the chronicles in ludics correspond to the linear positions in game semantics; and the plays or disputes correspond to the legal positions.

Thus from a sequence of actions p in $Plays(\mathfrak{D};\mathfrak{E})$, the chronicles of $\mathfrak{D}$ and $\mathfrak{E}$ are obtained by view operations $\ulcorner\urcorner^P$, $\ulcorner\urcorner^O$, respectively. Conversely, for a given finite legal position p on the universal arena, we can extract a design and a counter-design such that the dispute between them is p and minimal among such pairs of designs. (Proposition 2, [FH]).

Definition 7.9 (Plays (unary), [FH]) *For a given design $\mathfrak{D}$, we define a unary Plays as:*

$$Plays(\mathfrak{D}) = \bigcup\{Plays(\mathfrak{D};\mathfrak{E}); \mathfrak{E} \textit{is a counter-design}\}$$

The following proposition explains $Plays(\mathfrak{D})$ in terms of $Plays(\mathfrak{D};\mathfrak{E})$ and $\mathfrak{D}$.

Proposition 7.10 (Designs, Facts 3,4,5, [FH]) *Let $\mathfrak{D},\mathfrak{E}$ be a pair of a design and a counter-design. Then,*

- $Plays(\mathfrak{D}) \cap Plays(\mathfrak{E}) = Plays(\mathfrak{D};\mathfrak{E})$,
- $\mathfrak{D} \subseteq Plays(\mathfrak{D})$,
- $Plays(\mathfrak{D}) = \{p : p$ *is a legal position; and for all* $q(\sqsubseteq r^+ \sqsubseteq p)$ *satisfies* $\ulcorner q \urcorner \in \mathfrak{D}\}$.

An abstract notion characterizing $Plays(\mathfrak{D})$ is the notion of *strategies* (Def. 9, [FH]). In particular, $Plays(\mathfrak{D})$ is an *innocent* strategy (Def. 10, [FH]).

Definition 7.11 (Views, Def. 11, [FH]) *Let S be an X-strategy. We define*

$$Views(S) = \{\ulcorner q \urcorner^X : q \sqsubseteq r^+, r \in S\}.$$

The following proposition summarizes the relationship between $Views$ and $Plays$.

Proposition 7.12 (Fact 11, [FH]) *Let $\mathfrak{D}$ be a design, and S be an innocent strategy satisfying propagation. Then,*

- *$View(S)$ is a design, and*

$$Plays(Views(S)) = S,$$

- *$Plays(\mathfrak{D})$ is the smallest innocent strategy containing $\mathfrak{D}$, and*

$$Views(Plays(\mathfrak{D})) = \mathfrak{D}.$$

As we mentioned earlier, $Plays(\mathfrak{D})$ is an innocent strategy. Hence, we apply construction $Views$ to $Plays(\mathfrak{D})$, and we obtain

$$Views(Plays(\mathfrak{D})) = \{\lceil q \rceil : q \sqsubseteq r^+, r \in Plays(\mathfrak{D})\}.$$

The proposition above implies

$$Views(Plays(\mathfrak{D})) = \mathfrak{D}.$$

Due to the proposition above, we have thus obtained the following characterization of designs in terms of plays.

Proposition 7.13 (Designs, Prop. 3, [FH])

$$\mathfrak{D} = \{\lceil q \rceil : q \sqsubseteq r^+, r \in Plays(\mathfrak{D})\}.$$

Let us remind the idea presented by Amadio and Curien (p.43, [AC]), that the meaning of a term should be the collection of properties it satisfies in a suitable logic. In fact, the filter models of λ-calculus is based on this idea.

In this note, we have presented two characterizations of designs as collections, the one based on chronicles, and the other based on plays. A design thought as a term in ludics, its characterization as a collection, can be taken as a meaning defined by the properties it satisfies in a suitable logic, in the above sense.

- Which characterization is better, the one based on chronicles, or the one on plays.

Faggian and Hyland prefer the characterization based on plays. The plays are an interactive notion: So are the behaviors.

- What is the relationship between the chronicles in the plays and the behaviors?

- Does the notion of chronicles correspond to the notion of types in the filter models in some way?

These questions remind us the discussions in the previous section.

- What is the relationship between the designs and the filters of behaviors?

According to the theory of HON game semantics, the collection of innocent strategies ordered by inclusion is a dI-domain; a consistently complete, algebraic CPO which is coprime algebraic and satisfies axiom: Every compact element dominates only finitely many elements. (See [AC], [HO]). It will be interesting to investigate the duality for algebraic dcpo's in the collection of designs.

References

[AC] Amadio, R.M. and Curien, P.-L. *Domains and Lambda-Calculi*, Cambridge Tracts in Theoretical Computer Science 46.

[C] Curien, P.-L. *Introduction to ludics*, available from http://www.pps.jussieu.fr/~curien/.

[FH] Faggian, C. and Hyland, M. *Designs, disputes, and strategies*, CSL'02. Lecture Notes in Computer Science, Springer, Berlin, available from http://www.dpmms.cam.ac.uk/~cf245/pub.html.

[FFDQ] Faggian, C., Fleury-Donnadieu, M.-R. and Quatrini, M. *An introduction to uniformity in Ludics*, available from http://www.dpmms.cam.ac.uk/~cf245/pub.html.

[HO] Hyland, J.M.E. and Ong, C.-H.L. *On Full Abstraction for PCF: I, II, and III*, Information and Computation 163 pp.285-408 (2000).

[Gi] Girard, J.-Y. *Locus Solum*, Mathematical Structure in Computer Science 11 pp.301-506 (2001).

[O] 小野 寛晰. 情報科学における論理, 情報数学セミナー, 日本評論社 (1994).