SuperPower Lorenzen Games

Valeria de Paiva and Harley Eades III

May 2014

Abstract

This note discusses game semantics, in the style of Lorenzen, for Full Intuitionistic Linear Logic, following Blass and Rahman.

1 Blass on Linear Logic Games

Blass is responsible, to a large extent, for the introduction of game-theoretic ideas related to Linear Logic[3]. The project was enthusiastically embraced by Hyland, Abramsky and many others and there are now several conferences dedicated to the theme, especially to the application of game-theoretical ideas to programming language semantics.

The essence of the idea of a game theoretic semantics of a logic is that a game (or an argument, dialogue or protocol) consists of two players, one (the Proponent or Player) seeking to establish the truth of a formula under consideration (trying to prove it) while the other (the Opponent) disputing it, trying to prove it false. The two players alternate, attacking and defending their positions. As Blass puts it, the heart of the semantics consists of rules of the debate between the players.

The idea of using these dialogues goes back to, at least, Lorenzen in the late 50s. Lorenzen wanted to show that his dialogues were an alternative semantics for *Intuitionistic* Logic, but somehow the games looked much more like Classical Logic. Felscher came up with conditions that guaranteed that only constructive formulas were provable, but the conditions are not very transparent. Similarly Blass first introduced games for Linear Logic, but the composition of strategies in his games was not associative, a problem fixed by Abramsky and Jaghadeesan. But their semantics has problems of its own, as described in [2].

We recall that Blass presented two slightly different semantics for Linear Logic using games in[4, 5]. The semantics in [3] is for affine logic, that is the rule of weakening is validated. Both semantics say that to play the game $A\&B$, the opponent has to choose one of A or B and then the chosen game is played. And to play the game $A \oplus B$ the proponent chooses one of A or B and then the chosen game is played. A play of $A \otimes B$ consists of interleaved runs of the two games A and B, whenever it is the proponent's turn to play in either or both constituents, he must play there. When it's the opponent's turn to move

in both components, he must choose one and move in it. Blass also provided an interpretation of the modality "of course" , denoted by "!", similar to a repeated tensor product. Since linear negation is interpreted simply as exchange of players, "par", the multiplicative disjunction, and linear implication were *not* described, as in Classical Linear Logic they can be defined in terms of tensor and linear negation.

2 Lorenzen Games

Lorenzen games were developed by Lorenz, Felscher and Rahman, who together with co-workers, established a collection of games for specific (non-classical) logics, including Linear Logic. This general framework was named Dialogic. We recap the basics of Lorenzen games, as expounded by the Rahman school ([10]).

The language L is composed of the standard components of first-order logic, with four connectives $\wedge, \vee, \rightarrow, \neg$, with small letters $(a, b, c, \ldots)$ for prime formulas and capital italic letters $(A, B, C, \ldots)$ for complex formulas. We need two labels, **O** and **P** (standing for the players, Opponent and Proponent, respectively). We also need some special force symbols: ?... and !... , where ? stands for attack (or question) and ! stands for defense. The set of rules in dialogic is divided into *particle* rules and *structural* rules. Particle rules describe the way a formula can be attacked and defended, according to its main connective. Structural rules, on the other hand, specify the general organization of the game. We restrict ourselves to propositional logic. In the following we describe three systems, intuitionistic logic, classical logic and classical linear logic, before we embark on the novelty of this note, *full intuitionistic* linear logic.

3 Games for Intuitionistic and Classical Logic

The table with particle rules for intuitionistic and classical logic is as follows:

Assertion	Attack	Response
$A \wedge B$	$\wedge_L$	A
$A \wedge B$	$\wedge_R$	B
$A \vee B$	?	A or B
$A \rightarrow B$	A	B
$\neg A$	A	$-$

Disjunction and conjunction differ only by the player who has the choice of the immediate subformula with which the game will proceed: in a conjunction, the challenger may choose, confident that either conjunct can be refuted; in a disjunction the choice lies with the defender. Thus, to defend a conjunction, a player must be able to defend any of the conjuncts, while in the case of a disjunction, it is sufficient to be able to defend one of the disjuncts. Most importantly to challenge an implication essentially amounts to providing a proof

of the antecedent and claiming that the other player will fail to build a proof of the consequent from it. The defence against such an attack then consists of a proof of the consequent. And for negation, the only way to attack the assertion $\neg A$ is to assert A, and be prepared to defend this assertion. Thus there is no proper defence against such an attack, but it may be possible to counterattack the assertion of A. Lorenzen games were born in the context of constructivism, so this is not a surprise, the constructive negation of A is $A \to \bot$.

The difference between classical and intuitionistic logic resides on the structural rules. For classical logic the rules are:

S1 **P** may assert an atomic formula only after it has been asserted by **O**.

S4 A **P**-assertion may be attacked at most once.

S5 **O** can react only upon the immediately preceding **P**-statement.

while for intuitionistic logic the structural rules are the ones above plus:

S2 If p is an **P**-position, and if at round $n-1$ there are several open attacks made by **O**, then only the latest of them may be answered at n (and the same with **P** and **O** reversed).

S3 An attack may be answered at most once.

A *dialogue* can be thought of as a set of dialogical games, structured as a tree, the root of which is the thesis of the dialogue. Splits in the dialogue tree are generated by the propositional choices of the Opponent. Any possible attack by the Opponent against a conjunction, any possible defence of a disjunction, and either possible reaction in the case of an attack by the Proponent against an implication he defends (counterattack or defence) will generate a new branch in the dialogue tree. No move of the Proponent will open a new branch. A completed dialogue tree will thus contain all the Opponent's possible choices.

Starting with the thesis produced by the Proponent, the dialogue unravels connectives, until we get to atomic propositions. A dialogical game is said to be closed iff there is some atomic formula which has been played by both players. A dialogical game is finished iff it is closed or the rules do not allow any further move by the player who has to move.

For a set S of dialogue rules governing how the game is to proceed, a formula A is S-valid iff Proponent **P** has a *winning strategy* for A. A strategy for a player in a dialogical game is a function telling him what to do, according to what has previously happened in the game, to win it.

The problem with the structural rules is that it is not clear which modifications can be made to them without changing the set of provable formulas.

4 Lorenzen Games for Linear Logic

We now recall Rahman's[9] definitions of Lorenzen games for Linear logic, see [8]. In Linear Logic each occurrence of one formula in a proof must be taken as

a distinct resource for the inference process. In the dialogical framework, each move is an action. It is therefore natural to consider that two distinct moves are different actions, consuming different resources, even when the two moves have the same propositional content. But we have to introduce some book-keeping to deal with usage of resources.

To keep linearity and show validity of a formula we must use all and each formula that has been asserted throughout the dialogue. And we cannot use one round more than once.

To adapt the Lorenzen games to Linear Logic, we have to provide a new table of particle rules (for the new linear connectives) and we have to adopt some new principles, like:

1. An atomic **O**-formula has been used iff this formula is the propositional content of a **P**-move. Atomic **P**-formulas are used by the very move in which they appear.

2. A complex formula A has been used iff all the possible aggressive and defensive moves related to A have been stated.

Linear dialogues are contextual, the flow of information within the proof is constrained by an explicit structure of *contexts*, ordered by a relation of subordination. The introduction of a new context (which corresponds to a splitting of contexts in the sequent calculus) will always be a consequence of the Opponent's choices. The Proponent will stay in the same context as long as he can.

A sequent $\Gamma \vdash A$ is the statement that from assumptions Γ one can infer conclusion A. From the dialogical point of view, assumptions are the Opponent's conc1essions, while conclusions are the Proponent's claims. Splitting contexts for tensor occurs when it is asserted by the Proponent, so the dialogical particle rule will let the challenger choose the context where the dialogue will proceed. Dually, the multiplicative disjunction par will generate a splitting of contexts when asserted by the Opponent, thus the particle rule will let the defender choose the context. The interpretation of linear implication $\multimap$ remains constant, to attack it, we one must assert the antecedent, hoping that the opponent cannot use it to prove the consequent. Also the interpretation of linear negation stays the same.

Assertion	Attack	Response
$A \otimes B$ in context μ	$\otimes_L$ in subcontext ν of μ	A in ν
$A \otimes B$ in context μ	$\otimes_R$ in subcontext ν' of μ	B in ν'
$A \parr B$ in context μ	? in μ	A in subcontext ν of μ
$A \parr B$ in context μ	? in μ	B in subcontext ν' of μ
$A \multimap B$ in context μ	A in subcontext ν of μ	B in subcontext ν of μ
$\neg A$ in context μ	A in subcontext ν of μ	$-$ in subcontext ν of μ
$A \,\&\, B$ $A \,\&\, B$	$?\&_L$ $?\&_R$	A B
$A \oplus B$ $A \oplus B$	? ?	A B

Rahman also introduces an interpretation for the linear logic exponentials in terms of games, but says that the proof that the games are sound and complete for the exponentials is future work.

It is worth noticing that while Rahman deals exclusively with classical linear logic, he does mention that the intuitionistic structural rule could be used instead. Using the intuitionistic structural rule we should arrive at a semantics for Full Intuitionistic Linear Logic.

5 Full Intuitionistic Linear Logic

Full Intuitionistic Linear Logic (FILL) is a variant of Linear Logic introduced by de Paiva and Hyland in [7]. FILL is an interesting system, as it has independent multiplicative and additive connectives, just like Intuitionistic Logic has three independent connectives, conjunction, disjunction and implication.

FILL came into existence from work on Dialectica categories, which can be considered a different kind of game, a superpower game where players, instead of playing a proper game, simply face-off their collections of strategies. In one round the superpower game is decided, either Proponent has a winning strategy or Opponent has one or neither has one.

There are two main formalizations of FILL that enjoy cut-elimination. In the next two sections we give detailed definitions of both of these formalizations.

5.1 FILL Using Dependency Relations

One of the major obstacles of defining a formalization of FILL is obtaining cut-elimination. An initial formalization of FILL constrained the implication right rule to at most one formula on the right, but it was pointed out by Schellinx that

this breaks cut-elimination [11]. A second formalization of FILL used a term-assignment to constrain Classical Linear Logic to FILL [7], but this formalization did not enjoy cut-elimination either as pointed out Biereman [1]. Later, Braüer and de Paiva gave an alternate formalization of FILL based on dependency-relations that starts with Classical Linear Logic, and then constrains derivations using a dependency tracking system based on a dependency relation [6]. Then a FILL derivation is defined as a Classical Linear Logic derivation where only a single conclusion can depend on the hypothesis begin discharged.

Definition 1. *The syntax of formulas and contexts of Classical Linear Logic are defined as follows:*

$$\begin{array}{ll} \textit{(Formulas)} & A, B, C ::= A \otimes B \mid I \mid A \parr B \mid \bot \mid A \multimap B \\ \textit{(Contexts)} & \Gamma, \Delta ::= \cdot \mid A \mid \Gamma_1, \Gamma_2 \end{array}$$

We will consider contexts to be lists of formulas where $\cdot$ is the empty list, and Γ_1, Γ_2 denotes appending the list Γ_1 with the list Γ_2.

Definition 2. *The inference rules for Classical Linear Logic are defined as follows:*

$$\frac{}{A' \vdash A''}\ \textsc{Dax} \qquad \frac{\Gamma_1 \vdash B', \Delta_1 \quad \Gamma_2, B'' \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_2, \Delta_1}\ \textsc{Dcut}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \textsc{Dtl} \qquad \frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2 \vdash B, \Delta_2}{\Gamma_1, \Gamma_2 \vdash A \otimes B, \Delta_1, \Delta_2}\ \textsc{Dtr} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, I \vdash \Delta}\ \textsc{Dil}$$

$$\frac{}{\cdot \vdash I}\ \textsc{Dir} \qquad \frac{\Gamma_1, A \vdash \Delta_1 \quad \Gamma_3, B \vdash \Delta_2}{\Gamma_1, \Gamma_3, A \parr B \vdash \Delta_1, \Delta_2}\ \textsc{Dparl}$$

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \textsc{Dparr} \qquad \frac{}{\bot \vdash \cdot}\ \textsc{Dpl} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\ \textsc{Dpr}$$

$$\frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \multimap B \vdash \Delta_1, \Delta_2}\ \textsc{Dimpl} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \textsc{Dimpr}$$

Now we must define when a classical derivation is a FILL derivation, but this requires the notion of a dependency relation. To define the latter we will need a generalization of the usual notion of composition of relations.

Definition 3. *Suppose A_1, A_2, B_1, and B_2 are sets, and $r_1 \subseteq A_1 \times B_1$ and $r_2 \subseteq A_2 \times B_2$ are relations. Then we define the relation $r_1 \star r_2 \subseteq (A_1 \cup (A_2 - B_1)) \times (B_2 \cup (B_1 - A_2))$ as*

$$\begin{array}{l} [(r_1 \cap (A_1 \times (B_1 \cap A_2))) * (r_2 \cap ((A_2 \cap B_1) \times B_2))] \cup \\ [r_1 \cap (A_1 \times (B_1 - A_2))] \cup \\ [r_2 \cap ((A_2 - B_1) \times B_2)] \end{array}$$

where $*$ is the usual composition of relations.

The previous definition is due to Braüner and de Paiva in [6]. In that paper they give further explanations of the definition that the reader might find helpful. At this point we define a relation that calculates the set of dependencies of a classical derivation.

Definition 4. *Suppose τ is a classical derivation whose end sequent is $\Gamma \vdash \Delta$. Now we define the relation $Dep(\tau) \subseteq \Gamma \times \Delta$ where Γ and Δ are considered as sets of formula occurrences by induction on the structure of τ:*

Case. *If τ consists solely of an application of either of the rules* DIR *or* DPL*, then $Dep(\tau) = \emptyset$.*

Case. *If τ consists of only an application of the axiom rule where $\Gamma \vdash \Delta \equiv A \vdash A$, then $Dep(\tau) = \{(A, A)\}$.*

Case. *If the last rule applied in τ was* DCUT *whose two immediate subderivations are τ_1 and τ_2 with final sequents $\Gamma_1 \vdash B', \Delta_1$ and $\Gamma_2, B'' \vdash \Delta_2$, then $Dep(\tau) = Dep(\tau_1) \star \{B'\} \star \{B''\} \star Dep(\tau_2)$.*

Case. *If the last rule applied in τ was* DTL *whose immediate subderivation is τ_1 with final sequent $\Gamma_1, A, B \vdash \Delta_1$, then $Dep(\tau) = \{A \otimes B\} \star \{A, B\} \star Dep(\tau_1)$.*

Case. *If the last rule applied in τ was* DTR *whose two immediate subderivations are τ_1 and τ_2 with final sequents $\Gamma_1 \vdash A, \Delta_1$ and $\Gamma_2 \vdash B, \Delta_2$ respectively, then $Dep(\tau) = (Dep(\tau_1) \cup Dep(\tau_2)) \star \{A, B\} \times \{A \otimes B\}$.*

Case. *If the last rule applied in τ was* DIL *whose immediate subderivation is τ_1 with final sequent $\Gamma_1 \vdash \Delta_1$, and $\Gamma \vdash \Delta \equiv \Gamma_1, I \vdash \Delta_1$, then $Dep(\tau) = \{I\} \star \emptyset \star Dep(\tau_1)$.*

Case. *If the last rule applied in τ was* DPARL *whose two immediate subderivations are τ_1 and τ_2 with final sequents $\Gamma_1, A \vdash \Delta_1$ and $\Gamma_3, B \vdash \Delta_2$ respectively, then $Dep(\tau) = \{A \parr B\} \times \{A, B\} \star (Dep(\tau_1) \cup Dep(\tau_2))$.*

Case. *If the last rule applied in τ was* DPARR *whose immediate subderivation is τ_1 with final sequent $\Gamma_1 \vdash A, B, \Delta_1$, then $Dep(\tau) = Dep(\tau_1) \star \{A, B\} \times \{A \parr B\}$.*

Case. *If the last rule applied in τ was* DPR *whose immediate subderivation is τ_1 with final sequent $\Gamma_1 \vdash A, B, \Delta_1$, and $\Gamma \vdash \Delta \equiv \Gamma_1 \vdash \bot, \Delta_1$, then $Dep(\tau) = Dep(\tau_1) \star \emptyset \times \{\bot\}$.*

Case. *If the last rule applied in τ was* DIMPL *whose two immediate subderivations are τ_1 and τ_2 with final sequents $\Gamma_1 \vdash A, \Delta_1$ and $\Gamma_2, B \vdash \Delta_2$ respectively, then $Dep(\tau) = Dep(\tau_1) \star \{A, A \multimap B\} \times \{B\} \star Dep(\tau_2)$.*

Case. *If the last rule applied in τ was* DIMPR *whose immediate subderivation is τ_1 with final sequent $\Gamma_1, A \vdash B, \Delta_1$, then $Dep(\tau) = \emptyset \times \{A\} \star Dep(\tau_1) \star \{B\} \times \{A \multimap B\}$.*

If τ is a derivation of $\Gamma \vdash \Delta$ where A and B are formulas in Γ and Δ respectively, then we say that B depends on A if and only if $(A, B) \in Dep(\tau)$.

Finally, we arrive at this sections main definition.

Definition 5. *A FILL derivation is a classical derivation where whenever the implication right rule (*DIMPR*) is applied to a derivation τ of the sequent $\Gamma, A \vdash B, \Delta$ to obtain $\Gamma \vdash A \multimap B, \Delta$ none of the formulas in Δ depend on A in τ. That is, for all $C \in \Delta$, $(A, C) \notin Dep(\tau)$.*

For a more detailed explanation of this logic see [6]. We will freely use the cut-elimination theorem for FILL whose proof can also be found in [6].

5.2 FILL Using a Term Assignment

TODO

6 Lorenzen Games for FILL

In this section we formally define Lorenzen games for FILL. Then using this interpretation we prove soundness with respect to the dependency relation formalization of FILL (Section 5.1) and the term assignment formalization of FILL (Section 5.2).

We first enrich the language of FILL (Definition 1) to obtain the language of the games:

(Force)	$f ::= ? \mid !$
(Players)	$\mathbf{X}, \mathbf{Y} ::= \mathbf{O} \mid \mathbf{P}$
(Expressions)	$e ::= \otimes_L \mid \otimes_R \mid \parr \mid A$

So the language of Lorenzen games contains force symbols for attack (?) and defend (!), two players called the opponent (**O**) and the proponent (**P**), and finally a number of expressions representing moves. Note that the expressions contain every well-defined FILL formula.

Definition 6. *A* dialogical context *is a sequence of integers (called labels) denoted $\nu = x_0.x_1. \ldots .x_n$ where each part of the sequence is separated by dots. There is one distinguished context called the original context denoted by* 1*. Finally, the set of labels is semi-ordered by a relation of succession, forming a tree. That is, if ν denoted a dialogical context, then $\nu.i$ (for $1 \leq i \leq n$) denotes the n successors of ν.*

We call the depth of a context its rank where the original context has rank 1. We denote a player making a move by $\nu \vdash_{\mathbf{X}} f\, e$ which should read as the player **X** played move e in the dialogical context ν, and it should be considered either an attack or a defend depending on f.

Definition 7. *A move $\nu \vdash_{\boldsymbol{X}} f\, e$ is said to have* propositional content *iff e is a FILL formula.*

At this point we begin to specify how the evolution of the game is formalized. This requires the definition of two sets of rules: the particle rules, and the structural rules. The former describe the state of the dialogue with respect to the current formula under debate. This state is mainly composed of an assignment of players to roles, and players to subformulas of the formula under debate. Thus, the particle rules describe a local meaning of the formula under debate. The structural rules, however, determine the global meaning of the formula under debate. These rules describe the general organization of the dialogue, and can be thought of as orchestrating the particle rules as the dialogue progresses. It is up to the structural rules to guide the dialogue from start to finish.

We begin by defining the particle rules. The particle rules can be viewed as a state transition system.

Definition 8. *Suppose A is a FILL formula, and let $C = \{c_1, \ldots\}$ be a set of constants. Then a* state *of the dialogue for A is a triple $\langle \psi, \phi, \gamma \rangle$ where:*

- $\psi : \{\mathbf{O}, \mathbf{P}\} \to \{?, !\}$ *is a one-to-one correspondence of roles to players. We will write ψ_{rev} for the assignment that reverses the roles of the players. Note that $(\psi_{\mathsf{rev}})_{\mathsf{rev}} = \psi$.*
- *ϕ is a player-signed formula denoted X-B for some FILL formula B where $B \in \mathsf{Sub}(A)$.*
- *γ is an assignment of dialogical contexts to player-signed formulas.*

The particle rules now describe how to move from one state to the next based on the main connective of the player-signed formula.

Definition 9. *Suppose $S_1 = \langle \psi, \phi = \psi^{-1}(!)\text{-}B, \gamma \rangle$. Then we define which states are reachable from S_1 based on the main connective of B:*

> ***Particle Rule for Tensor**. Suppose $B = A_1 \otimes A_2$ and $\gamma = \gamma_{\phi/\nu}$, where the latter denotes γ assigning the player-signed formula ϕ to the context ν. Then the following characterizes the reachable states from S_1 after an attack α.*
>
> - *If $\alpha = \psi^{-1}(?)\text{-}\otimes_R$ in the dialogical subcontext μ of ν chosen by $\psi^{-1}(?)$, then $S_2 = \langle \psi, \psi^{-1}(!)\text{-}A_1, \gamma_{A_1/\mu} \rangle$ is reachable from S_1.*
> - *If $\alpha = \psi^{-1}(?)\text{-}\otimes_L$ in the dialogical subcontext μ of ν chosen by $\psi^{-1}(?)$, then $S_3 = \langle \psi, \psi^{-1}(!)\text{-}A_2, \gamma_{A_1/\mu} \rangle$ is reachable from S_1.*
>
> *The formula $A_1 \otimes A_2$ is called used and is bracketed if and only if the states S_2 and S_3 have been reached.*
>
> ***Particle Rule for Par**. Suppose $B = A_1 \parr A_2$ and $\gamma = \gamma_{\phi/\nu}$. Then $\psi^{-1}(!)$ can choose one of the following possible transitions from S_1.*
>
> - *If $\alpha = \psi^{-1}(?)\text{-}\parr$ in the dialogical subcontext μ of ν chosen by $\psi^{-1}(!)$, then $S_2 = \langle \psi, \psi^{-1}(!)\text{-}A_1, \gamma_{A_1/\mu} \rangle$ is reachable from S_1.*

– *If* $\alpha = \psi^{-1}(?)\text{-}\parr$ *in the dialogical subcontext* μ *of* ν *chosen by* $\psi^{-1}(!)$, *then* $S_3 = \langle \psi, \psi^{-1}(!)\text{-}A_2, \gamma_{A_1/\mu} \rangle$ *is reachable from* S_1.

The formula $A_1 \parr A_2$ *is called used and is bracketed if and only if the states* S_2 *and* S_3 *have been reached.*

Particle Rule for Implication*. Suppose* $B = A_1 \multimap A_2$ *and* $\gamma = \gamma_{\phi/\nu}$. *The following is reachable from* S_1*:*

– $S_2 = \langle \psi_{\mathsf{rev}}, \psi_{\mathsf{rev}}^{-1}(!)\text{-}A_1, \gamma_{A_1/\mu} \rangle$ *where* μ *is a dialogical subcontext of* ν *choose by* $\psi^{-1}(!)$.

The state following S_2 *is now chosen by* $\psi^{-1}(!)$ *and can proceed by applying the appropriate particle rule to* S_2*, or can proceed to the state:*

– $S_3 = \langle \psi, \psi^{-1}(!)\text{-}A_2, \gamma_{A_1/\mu'} \rangle$ *where* μ' *is a dialogical subcontext of* ν *choose by* $\psi^{-1}(!)$.

The formula $A_1 \multimap A_2$ *is called used and is bracketed if and only if the states* S_2 *and* S_3 *have been reached.*

The structural rules:

7 Conclusions

We recalled games for Linear Logic in two traditions, Blass-style games (as discussed by Hyland, Abramsky and Jagadhesan and many others) and Lorenzen-style games (as introduced by Rahman, Keiff and others). It seems clear that Lorenzen games have the ability to model full intuitionistic linear logic easily. Emphasis is given to the interpretation of linear implication, instead of negation.

This note is just a first step in the programme to develop *compositional* versions of Lorenzen Linear logic games. We need still to prove our soundness and completeness result and more importantly we need to make sure that strategies are compositional, to make sure we can produce categories of Lorenzen games.

References

[1] G.M. Bierman. A note on full intuitionistic linear logic. *Annals of Pure and Applied Logic*, 79(3):281 – 287, 1996.

[2] A Blass. Some semantical aspects of linear logic. *Logic Journal of IGPL*, 5(4):487–503, 1997.

[3] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56:183–200, 1992.

[4] Andreas Blass. Is game semantics necessary? In Egon Börger, Yuri Gurevich, and Karl Meinke, editors, *Computer Science Logic*, volume 832 of *Lecture Notes in Computer Science*, pages 66–77. Springer Berlin Heidelberg, 1994.

[5] Andreas Blass. Questions and answers: A category arising in linear logic complexity theory, and set theory. In *Proceedings of the Workshop on Advances in Linear Logic*, pages 61–81, New York, NY, USA, 1995. Cambridge University Press.

[6] Torben Brauner and Valeria Paiva. A formulation of linear logic based on dependency-relations. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, pages 129–148. Springer Berlin Heidelberg, 1998.

[7] Martin Hyland and Valeria de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, 64(3):273 – 291, 1993.

[8] Laurent Keiff. Dialogical logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2011 edition, 2011.

[9] S. Rahman. Un desafio para las teorias cognitivas de la competencia log- ica: los fundamentos pragmaticos de la semantica de la logica linear. *Manuscrito*, XXV(2):381–432, October 2002.

[10] Shahid Rahman and Laurent Keiff. On how to be a dialogician. In Daniel Vanderveken, editor, *Logic, Thought and Action*, volume 2 of *Logic, Epistemology, and the Unity of Science*, pages 359–408. Springer Netherlands, 2005.

[11] Harold Schellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 1991.

A The full specification of FILL

$$
\begin{array}{lll}
\mathit{term\_var},\ w,\ x,\ y,\ z,\ v & & \\
\mathit{index\_var},\ i,\ j,\ k & & \\
\mathit{form},\ A,\ B,\ C,\ D,\ E & ::= & \\
 & \mid & I \\
 & \mid & \bot \\
 & \mid & A \multimap B \\
 & \mid & A \otimes B \\
 & \mid & A \parr B \\
 & \mid & (A) \qquad \mathsf{S} \\
 & & \\
\mathit{patterns},\ p & ::= & \\
 & \mid & * \\
\end{array}
$$

	x	
	$p_1 \otimes p_2$	
	$p_1 \parr p_2$	
	$-$	
	(p)	S

$term,\ t,\ e,\ d,\ f,\ g,\ u$	::=	
	x	
	$*$	
	$\circ$	
	$e_1 \otimes e_2$	
	$e_1 \parr e_2$	
	$\lambda x.t$	
	let t be p in e	
	$f\ e$	
	let-pat $z\ p\ e$	M
	$[t/x]t'$	M
	$[t/x, e/y]t'$	M
	(t)	S
	t	M
	t	M

Γ	::=	
	$x : A$	
	$\cdot$	
	Γ, Γ'	
	$x : A$	
	A	

Δ	::=	
	$t : A$	
	$\cdot$	
	$\Delta \mid \Delta'$	
	Δ	
	A	
	Δ, Δ'	

$games$	::=	
	$\otimes_L$	
	$\otimes_R$	

$$
\begin{array}{lll}
 & | & \parr \\
 & | & \mathbf{O} \\
 & | & \mathbf{P} \\
 & | & ? \\
 & | & ! \\
\end{array}
$$

$$
\begin{array}{lll}
\textit{formula} & ::= & \\
 & | & \textit{judgement} \\
 & | & \textit{formula}_1 \quad \textit{formula}_2 \\
 & | & (\textit{formula}) \\
 & | & x \notin \mathsf{FV}(\Delta) \\
\end{array}
$$

$$
\begin{array}{lll}
\textit{InferRules} & ::= & \\
 & | & \Gamma \vdash \Delta \\
 & | & f = e \\
\end{array}
$$

$$
\begin{array}{lll}
\textit{judgement} & ::= & \\
 & | & \textit{InferRules} \\
\end{array}
$$

$$
\begin{array}{lll}
\textit{user\_syntax} & ::= & \\
 & | & \textit{term\_var} \\
 & | & \textit{index\_var} \\
 & | & \textit{form} \\
 & | & \textit{patterns} \\
 & | & \textit{term} \\
 & | & \Gamma \\
 & | & \Delta \\
 & | & \textit{games} \\
 & | & \textit{formula} \\
\end{array}
$$

$\boxed{\Gamma \vdash \Delta}$

$$\frac{}{x : A \vdash x : A}\ \textsc{Ax}$$

$$\frac{\Gamma \vdash t : A \mid \Delta \quad y : A, \Gamma' \vdash f_i : B_i}{\Gamma, \Gamma' \vdash \Delta \mid [t/y]f_i : B_i}\ \textsc{Cut}$$

$$\frac{\Gamma \vdash e_i : A_i}{\Gamma, x : I \vdash \mathsf{let}\, x\, \mathsf{be}\, *\, \mathsf{in}\, e_i : A_i}\ \textsc{Il}$$

$$\frac{}{\cdot \vdash * : I}\ \textsc{Ir}$$

$$\frac{\Gamma, x : A, y : B \vdash f_i : C_i}{\Gamma, z : A \otimes B \vdash \mathsf{let}\, x\, \mathsf{be}\, x \otimes y\, \mathsf{in}\, f_i : C_i}\ \textsc{Tl}$$

$$\frac{\Gamma \vdash e : A \mid \Delta \quad \Gamma' \vdash f : B \mid \Delta'}{\Gamma, \Gamma' \vdash e \otimes f : A \otimes B \mid \Delta \mid \Delta'}\ \textsc{Tr}$$

$$\frac{}{x :\bot \vdash \cdot}\ \textsc{Pl}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \circ :\bot \mid \Delta}\ \textsc{Pr}$$

$$\frac{\Gamma, x : A \vdash d_i : C_i \quad \Gamma', y : B \vdash f_j : D_j}{\Gamma, \Gamma', z : A \parr B \vdash \mathsf{let}\ z\ \mathsf{be}\ x \parr - \ \mathsf{in}\ d_i : C_i \mid \mathsf{let}\ z\ \mathsf{be}\ - \parr y\ \mathsf{in}\ f_j : D_j}\ \textsc{Parl}$$

$$\frac{\Gamma, x : A \vdash d_i : C_i \quad \Gamma', y : B \vdash f_j : D_j}{\Gamma, \Gamma', z : A \parr B \vdash \text{let-pat}\ z\ (x \parr -)\ d_i : C_i \mid \text{let-pat}\ z\ (- \parr y)\ f_j : D_j}\ \textsc{NParl}$$

$$\frac{\Gamma \vdash \Delta \mid e : A \mid f : B \mid \Delta'}{\Gamma \vdash \Delta \mid e \parr f : A \parr B \mid \Delta'}\ \textsc{Parr}$$

$$\frac{\Gamma \vdash e : A \mid \Delta \quad \Gamma', x : B \vdash f_i : C_i}{\Gamma, y : A \multimap B, \Gamma' \vdash [y\ e/x] f_i : C_i \mid \Delta}\ \textsc{Impl}$$

$$\frac{\Gamma, x : A \vdash e : B \mid \Delta \quad x \notin \mathsf{FV}(\Delta)}{\Gamma \vdash \lambda x.e : A \multimap B \mid \Delta}\ \textsc{Impr}$$

$$\frac{}{A' \vdash A''}\ \textsc{Dax}$$

$$\frac{\Gamma_1 \vdash B', \Delta_1 \quad \Gamma_2, B'' \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_2, \Delta_1}\ \textsc{Dcut}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \textsc{Dtl}$$

$$\frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2 \vdash B, \Delta_2}{\Gamma_1, \Gamma_2 \vdash A \otimes B, \Delta_1, \Delta_2}\ \textsc{Dtr}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, I \vdash \Delta}\ \textsc{Dil}$$

$$\frac{}{\cdot \vdash I}\ \textsc{Dir}$$

$$\frac{\Gamma_1, A \vdash \Delta_1 \quad \Gamma_3, B \vdash \Delta_2}{\Gamma_1, \Gamma_3, A \parr B \vdash \Delta_1, \Delta_2}\ \textsc{Dparl}$$

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \textsc{Dparr}$$

$$\frac{}{\bot \vdash \cdot}\ \textsc{Dpl}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\ \textsc{Dpr}$$

$$\frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \multimap B \vdash \Delta_1, \Delta_2}\ \textsc{Dimpl}$$

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \textsc{Dimpr}$$

$\boxed{f = e}$

$$\frac{}{(\lambda x.e)\ e' = [e'/x]e}\ \textsc{Eq\_Beta}$$

$$\frac{}{\lambda x.f\ x = f}\ \textsc{Eq\_Eta}$$

$$\frac{}{\mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ e = e}\ \textsc{Eq\_I}$$

$$\frac{}{\mathsf{let}\ u\ \mathsf{be}\ *\ \mathsf{in}\ [*/z]f = [u/z]f}\ \textsc{Eq\_Stp}$$

$$\frac{}{\mathsf{let}\ e \otimes t\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ u = [e/x, t/y]u}\ \textsc{Eq\_T1}$$

$$\frac{}{\mathsf{let}\ u\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [x \otimes y/z]f = [u/z]f}\ \textsc{Eq\_T2}$$

$$\frac{}{\mathsf{let}\ u\ ⅋\ t\ \mathsf{be}\ x\ ⅋ -\ \mathsf{in}\ e = [u/x]e}\ \textsc{Eq\_P1}$$

$$\frac{}{\mathsf{let}\ u\ ⅋\ t\ \mathsf{be}\ -\ ⅋ y\ \mathsf{in}\ e = [t/y]e}\ \textsc{Eq\_P2}$$

$$\frac{}{(\mathsf{let}\ x\ \mathsf{be}\ x\ ⅋ -\ \mathsf{in}\ x)\ ⅋\ (\mathsf{let}\ u\ \mathsf{be}\ -\ ⅋ y\ \mathsf{in}\ y) = u}\ \textsc{Eq\_P3}$$