A DEGREE STRUCTURE ON EFFECTIVE OPEN OR CLOSED GAMES AND GAME SEMANTICS OF LINEAR LOGIC

ZHENHAO LI

1. Introduction

Not to find game semantics for "proof systems", but study games for their own right.

1.1. **Playable games.** In 2011 Andreas Blass in a workshop at Tübingen introduced the concept of "playable games" which are turn based 2-player games with perfect information having the following additional features. The rules of a playable game are computable, or putting it another way, a computer can perform the role of a referee. A referee of a game determines which player to start, whether a move is legal, whether the game is finished after each move, if yes which player has won, and if no which is to move next. Blass also required that each play of a playable game ends after finitely many moves.

If a $\mathbf{player_i}$ in a game A can make sure that he or she can win every play of A, we say i has a winning strategy in A. A wining strategy is not necessarily computable. In fact in all games that we consider interesting, no player has a computable winning strategy. The reason is that if some player has a computable winning strategy in game A then a computer can win each play of A and we don't think A is an interesting game. Our research program is to investigate those games in which no player has a computable winning strategy.

1.2. **An example of effectively open games.** It follows from a well known result of Zermelo that each 2-player finite-choice finite game with perfect information is determined. In fact each winning strategy in that kind of games is computable since there is a natural number that bounds the number of total moves of every play. As a result, if in chess we forbidden any configure of board to occur more than once during the same play, then a computer can win every play of chess playing white or can do so playing black. But there is an intuition against forbidden multiple occurrence of configures of board: a play may learn some thing between two occurrence of the same configure of board. Although it is natural to call a infinite play a draw, we prefer to take another way, namely to fix one player in each game in advance and to specify that every infinite play of that game is won by him or her.

Since all interesting games in real world can be formalized or modeled as 2-player 2-choice infinite games with perfect information. Without loss of generality we may assume that the choices are number 0 and number 1. Or expressing it in the language of set theory, we study games on set 2. We can see each finished finite play as the set of all infinite plays that extends the finite play which are all won by the same winner as in the finite play. So our games all become infinite games and the play space of each game is the whole Cantor space. It is a tradition to call the set of all infinite plays of each game won by the first player the payoff class of that game, and identify a game with its payoff class. It is easy to see that the payoff class of each game under our consideration is either effective open or effective close.

We focus on games of the this type in this paper and study their degrees of non-determinacy. From now on, we always mean games of that type by the word "game". We will give precise definition of a game in the next section. Before doing so we would like to point out that all games under our consideration are relaxedly determined, according to a well known result of Gale and Stewart in [GS53] which requires the axiom of choice. Because the payoff class of each game considered here is open for one player and closed for the other in Cantor space. Well shall see that there are non-determined ones among these games. Moreover we can compare the difficulties of winning among them and we shall see that there is a nontrivial structures on the family of game under our restriction.

2. Degrees of non-determinacy

2.1. **Meta-games.** In this paper we reserve the term game to the special kind of game in which two players move alternately and $\mathbf{player_0}$ moves first. By *meta-game* we mean games that are build up using the special kind of games which we call games.

Formally we define meta-games recursively. Firstly games are meta-games. From any meta-game, we can get another meta-game. The *dual* of a meta-game $\mathcal{G}$ is obtained by exchanging the roles of $\mathbf{player_0}$ and $\mathbf{player_1}$ in $\mathcal{G}$, and is denoted by $\overline{\mathcal{G}}$. So the goal of $\mathbf{player_0}$ ($\mathbf{player_1}$) in $\overline{\mathcal{G}}$ is to play and win as $\mathbf{player_1}$ ($\mathbf{player_0}$) in $\mathcal{G}$.

Equivalently $\overline{\mathcal{G}}$ is obtained by exchanging the names of $\mathbf{player_0}$ and $\mathbf{player_1}$ in $\overline{\mathcal{G}}$. It is easy to see that $\mathbf{player_0}$ ($\mathbf{player_1}$) in $\overline{\mathcal{G}}$ can be identified with $\mathbf{player_1}$ ($\mathbf{player_0}$) in $\overline{\mathcal{G}}$.

It is easy to see that $\mathbf{player_0}$ ($\mathbf{player_1}$) has a winning strategy in $\overline{\mathcal{G}}$ if and only if $\mathbf{player_1}$ ($\mathbf{player_0}$) has a winning strategy in $\mathcal{G}$.

We will define one more operation, the *tensor product* $\bigotimes$. It will be used in the definition of the orderings of games. The tensor product $\bigotimes_{i\in I}\mathcal{G}_i$ of an effectively indexed family of games $\{\mathcal{G}_i|\ i\in I\}$ is played as follows. At the beginning of each move of the game, $\mathbf{player_0}$ chooses a game $\mathcal{G}_i$ and then a move is made in that chosen $\mathcal{G}_i$; this is considered one move in the

game. The game is won by $\mathbf{player_0}$ if and only if at least one of the $\mathcal{G}_i$'s has been won by her.

We define the dual of the tensor product $\bigotimes$, the *dual tensor product* $\overline{\bigotimes}$. At the beginning of each move of the game, $\mathbf{player_1}$ chooses a game $\mathcal{G}_i$ and then a move is made in that chosen $\mathcal{G}_i$; this is considered one move in the game. The game is won by $\mathbf{player_1}$ if and only if at least one of the $\mathcal{G}_i$'s has been won by him.

2.2. **Reducibility.** A game $\mathcal{G}_0$ is reducible to a game $\mathcal{G}_1$, denoted by $\mathcal{G}_0 \leq \mathcal{G}_1$ if and only $\mathbf{player_1}$ has a winning strategy in the game $\mathcal{G}_0 \otimes \overline{\mathcal{G}}_1$.

A game $\mathcal{G}_0$ is equivalent to a game $\mathcal{G}_1$, denoted by $\mathcal{G}_0 \equiv \mathcal{G}_1$ if and only $\mathcal{G}_0 \leq \mathcal{G}_1$ and $\mathcal{G}_1 \leq \mathcal{G}_0$.

A game $\mathcal{G}_0$ is incomparable with a game $\mathcal{G}_1$, denoted by $\mathcal{G}_0 \| \mathcal{G}_1$ if and only $\mathcal{G}_0 \not\leq \mathcal{G}_1$ and $\mathcal{G}_1 \not\leq \mathcal{G}_0$.

Intuitively, $\mathcal{G}_0 \leq \mathcal{G}_1$ when $\mathbf{player_0}$ can make sure that if she plays $\mathcal{G}_0$ together with a version of $\mathcal{G}_1$ where the roles of the two players are exchanged, and she can switch between the two games, then whenever her opponent wins $\mathcal{G}_1$ as $\mathbf{player_0}$ she wins $\mathcal{G}_0$. In a sense $\mathbf{player_0}$ know how to win $\mathcal{G}_0$ if someone can show her how to win $\mathcal{G}_1$ as $\mathbf{player_0}$, and so $\mathcal{G}_0$ is easier for her to win than $\mathcal{G}_1$.

Proposition 1. *If two games are incomparable, then both of them are non-determined.*

Proof. Suppose $\mathcal{A}$ is a determined game. If $\mathcal{A}$ is a win for $\mathbf{player_1}$ then $\overline{\mathcal{B}}\tilde{\otimes}\mathcal{A}$ is a win for $\mathbf{player_1}$ and hence $\mathcal{B} \leq \mathcal{A}$ for any $\mathcal{B}$. Otherwise $\mathcal{A}$ is a win for $\mathbf{player_0}$ and $\mathcal{A} \otimes \overline{\mathcal{B}}$ is a win for $\mathbf{player_0}$ and hence $\mathcal{A} \leq \mathcal{B}$ for any $\mathcal{B}$. □

Lemma 2. *For any two game $\mathcal{A}$ and $\mathcal{B}$, $\mathbf{player_1}$ has a winning strategy in $\mathcal{A}\otimes\mathcal{B}$ if and only if he has a winning strategy in $\mathcal{A}$ or has a winning strategy in $\mathcal{B}$; and similarly $\mathbf{player_0}$ has a winning strategy in $\mathcal{A}\tilde{\otimes}\mathcal{B}$ if and only if she has a winning strategy in $\mathcal{A}$ or has a winning strategy in $\mathcal{B}$.*

Theorem 3. *The reducibility relation $\leq$ is a preordering on $\mathfrak{G}$, the set of all games.*

Proof. We prove the case of $n = 1$ in our informal style and leave other cases to the reader.

Reflexivity: Consider the following strategy φ_e for $\mathbf{player_1}$ in $G \multimap G$. When playing an actual play of $G \multimap G$, φ_e essentially starts by copying $\mathbf{player_0}$'s first move in $\mathcal{G}$ as his first move in $\overline{G}$. Then $\mathbf{player_0}$ has to reply within the same game. Whenever it's $\mathbf{player_1}$'s move, switch to the other game, and copy the move $\mathbf{player_0}$ just made. The result of this strategy is that the "game board" of $\mathcal{G}$ is identical identical to that of $\overline{G}$. As $\mathbf{player_1}$ plays opposite roles in the two games, he wins $G \multimap G$ and hence φ_e is a winning strategy for $\mathbf{player_1}$.

Transitivity: Let φ_e and ψ be winning strategies for $\mathbf{player_1}$ in $G \rightharpoonup F$ and $F \rightharpoonup E$, respectively. Consider the following strategy χ for $\mathbf{player_1}$ in $G \rightharpoonup E$. On its work tape χ simulate all three games $G \rightharpoonup F$, $F \rightharpoonup E$ and $G \rightharpoonup F$.

The strategy χ works as follows. It begins by leting $\mathbf{player_0}$ starts the play of E. Then χ start simulating ψ and $F \rightharpoonup E$ by taking the move just made by 0 in E as if made in playing $F \rightharpoonup E$. Now if ψ tells $\mathbf{player_1}$ to reply in E, χ copies $\mathbf{player_1}$'s moves in E produced by ψ back as its moves in $G \rightharpoonup E$. Now $\mathbf{player_0}$ has to make her next move in E again and χ copies that move back to the simulation of $F \rightharpoonup E$ and run ψ again. It continues like this until ψ dictates $\mathbf{player_1}$ switch to $\overline{F}$. After simulating that move by ψ in $\overline{F}$, $F \rightharpoonup E$ requires $\mathbf{player_0}$ to replay in $\overline{F}$. At this point χ suspends the simulation of ψ and $F \rightharpoonup E$ and start the simulation of φ_e and $G \rightharpoonup F$. Notice that the first move of $\mathbf{player_0}$ in F has just been made as $\mathbf{player_1}$'s in $\overline{F}$. If φ_e requires $\mathbf{player_1}$ to make moves in $\overline{G}$, then χ switch to $\overline{G}$ in $G \rightharpoonup E$ and copies φ_e's move in $\overline{G}$ back to the play of $G \rightharpoonup E$, and it continues simulating φ_e and $G \rightharpoonup F$ as long as φ_e does not switch to F. If φ_e chooses to make a move within F, χ extend the same "board" of F and $\overline{F}$ by φ_e's move. Now it is $\mathbf{player_0}$'s turn to move in F, or equally $\mathbf{player_1}$'s turn to move in $\overline{F}$, and χ suspends the simulation of φ_e and $G \rightharpoonup F$ and resume the simulation of ψ and $F \rightharpoonup E$. Now it is clear how χ works.

It is easy to check that it is impossible that $\mathbf{player_0}$ wins both $\overline{G}$ and E in $G \rightharpoonup E$ if $\mathbf{player_1}$ follows χ.

□

3. Games and degrees of non-determinacy

In this paper we assume that the alphabet of our Turing machines contains 0, 1 and B(for blank). and each Turing machine can read binary strings as input. We identify $\sigma \in 2^{<\omega}$ with natural number n such that the binary representation of $n+1$ is $1^\frown\sigma$; see [Neis]. It is easy to see that each natural number n uniquely codes a binary string and the number 0 codes the empty string ϵ. As a consequence a function $x \in 2^\omega$ uniquely codes a function in $2^{2^{<\omega}}$ and in addition we use f to name that function.

Fix an enumeration of all Turing machines and identity Turing machines with their indices. Let φ_e denote the (partial) function that computed by the e-th Turing machine. We say the machine e, or the function φ_e, is *total* if and only if $(\forall \sigma \in 2^{<\omega})\ [\varphi_e(\sigma) = 0 \vee \varphi_e(\sigma) = 1]$. A function $f \in 2^\omega$ is computable if and only $f = \varphi_e$ for some total e.

Let W_e be $\{\sigma \in 2^{<\omega} : \varphi_e(\sigma)\downarrow\}$, and χ_A to be the characteristic function of set A. It is easy to see that there is a computable function f s.t. for each e such that $\varphi_e = \chi_A$ for some set A, $W_{f(e)} = A$.

As mentioned in the introduction, each infinite game is played by 2 players in an alternate pattern. We call the player who moves first $\mathbf{player_0}$ and the

other $\mathbf{player_1}$, and the set of all infinite plays won by $\mathbf{player_0}$ is called the payoff class of that game. We sometimes use the same notation for a game and its payoff class. For example, we identify a game $\mathcal{G}$ with its payoff class $\mathcal{G}$. So if $x \in 2^\omega$ then $\mathbf{player_0}$ wins the play x of $\mathcal{G}$ if and only if $x \in \mathcal{G}$.

The *negation* of a game $\mathcal{G}$ is the game with payoff class $\overline{\mathcal{G}}$.

Define $[\![\sigma]\!] = \{x \in 2^\omega : \sigma \prec x\}$ for each $\sigma \in 2^{<\omega}$ and $[\![A]\!] = \bigcup_{\sigma\in A}[\![\sigma]\!]$ for each $A \subseteq 2^{<\omega}$. We associate each Turing machine φ_e an open set $[\![W_e]\!]$, and a closed set $2^\omega - [\![W_e]\!]$. We say $\mathcal{A} \subset 2^\omega$ is *effectively open* if $\mathcal{A} = [\![W_e]\!]$ for some e, and in this case it has an *open index* e; and $\mathcal{A}$ is *effectively closed* if $\mathcal{A} = 2^\omega - [\![W_{e'}]\!]$ for some e', in this case it has a *closed index* e'.

We say a set $A \subseteq 2^{<\omega}$ is closed upwards if for $(\forall\sigma \in A)\ (\forall\tau)\ [\sigma \prec \tau \rightarrow \tau \in A]$, and it is closed downwards if $(\forall\sigma \in A)\ (\forall\tau)\ [\tau \prec \sigma \rightarrow \tau \in A]$. We say A is a tree if A is closed downwards, and is a computable tree if $A = W_e^1$ for some e. When A is a tree, define $[A] = \{x \in 2^\omega : (\forall n)\ x\restriction n \in A\}$.

In this paper we consider games whose payoff classs are effectively open or closed sets. A game is an *effectively open game* if its payoff class is an effectively open class and is an *effectively closed game* if its payoff class is a effectively closed class. An effectively open (closed) game has an open index e if its payoff set has an open (closed) index e.

It is easy to see that a game $\mathcal{G}$ is effectively open (effectively closed) if and only if $\overline{\mathcal{G}}$ is effectively closed (effectively open).

The following lemma is well known.

Lemma 4. *(1) There is a computable function f such that $\varphi_{f(e)} = \chi_A$ for a set A that is closed upwards and such that $[\![A]\!] = [\![W_e]\!]$ for every e.*

(2) There is a computable function g such that $\varphi_{g(e)} = \chi_A$ for a set A that is a tree and such that $[A] = 2^\omega - [\![W_e]\!]$ for every e.

Proof. (1) Consider the following procedure ϕ with parameter e. It decides memberships of all members of $2^{<\omega}$ in a lexical order. The n-th string σ is in (i.e., having value 1) if and only if for some $\tau \prec \sigma$, τ has already been decided in, or $\tau \in W_{e,n}$. Obviously ϕ is total. It is easy to check that $[\![\phi]\!] = [\![W_e]\!]$. Applying the $s - m - n$ theorem we get the computable function f required.

(2) The the procedure $\lambda\sigma.1 - \phi(\sigma)$ gives g.

□

By Lemma 4, effectively open games can be seen as games that have computable referees. A referee for a effectively open game decide at each move whether the game has been won by $\mathbf{player_0}$, if yes then the game is over and won by $\mathbf{player_0}$ and if not the game requires another move. If $\mathbf{player_0}$ wins a play of an effectively open game, she has won after finitely many moves. $\mathbf{player_1}$ can only win by keeping the play infinitely long.

A *strategy* for $\mathbf{player_0}$ is a total computable function φ_e, and e is an index of that strategy. If e is an index of a strategy for $\mathbf{player_0}$, for each $x \in 2^{\leq\omega}$,

$\varphi_e * x$ denotes the unique maximal $y \in 2^{\leq\omega}$ such that $(\forall n < |x|)y(2n+1) = x(n))$ and $(\forall n \leq |x|)y(2n) = \varphi_e(x\restriction n)$. Note that $x\restriction 0$ is ϵ since 0 is the empty set $\emptyset$. Intuitively $\varphi_e * x$ is the (partial) play which is the result of $\mathbf{player_0}$ following strategy φ_e and $\mathbf{player_1}$ play x bit after bit as his moves. We call the set $\varphi_e * 2^\omega = \{\varphi_e * x : x \in 2^\omega\}$ the strategy tree of φ_e for $\mathbf{player_0}$.

A *strategy* for $\mathbf{player_1}$ is a partial computable function that is defined on every $\sigma \in 2^{<\omega} - \{\epsilon\}$. If φ_e is a strategy for 1, for each $x \in 2^{\leq\omega}$, $x * \varphi_e$ denotes the unique maximal $y \in 2^{\leq\omega}$ such that $(\forall n < |x|)y(2n) = x(n)$ and $(\forall n < |x|)y(2n+1) = \varphi_e(x\restriction n+1)$. Note that $x\restriction 0$ is ϵ since 0 is the empty set $\emptyset$. Intuitively $x * \varphi_e$ is the infinite play which is the result of $\mathbf{player_1}$ following strategy φ_e and $\mathbf{player_0}$ play x bit after bit as her moves. We call the set $2^\omega * \varphi_e = \{x * \varphi_e : x \in 2^\omega\}$ the strategy tree of φ_e for $\mathbf{player_1}$.

A strategy for $\mathbf{player_i}$ in a game $\mathcal{G}$ is a *winning strategy* for him/her in $\mathcal{G}$ if $\varphi_e * 2^\omega \subset G$ when $i = 0$, or $2^\omega * \varphi_e \subset \overline{\mathcal{G}}$ when $i = 1$. A game is called *determined* if one of the players has a winning strategy. Otherwise it is called *non-determined*.

For any $x \in 2^{\leq\omega}$, $(x)_0$ denote the the maximal string s such that $s(i) = x(2i)$ for $i < |s|$ and $(x)_1$ denote the the maximal string t such that $t(i) = x(2i+1)$ for $i < |t|$. So $(x)_0$ is the sequence of even bits in x and $(x)_1$ is the sequence of odd bits in x. For $S, T \subseteq {}^{<\omega}2$, let $S * T$ be the set $\{b \in {}^{<\omega}2 : (b)_0 \in S \wedge (b)_1 \in T\}$.

For any two Turing machines ψ and φ_e, $\psi * \phi$ denotes the unique maximal $y \in 2^{\leq\omega}$ such that $(\forall 2n < |y|)\ y(2n) = \psi((y\restriction 2n)_1)$ and $(\forall 2n+1 < |y|)\ y(2n+1) = \phi((y\restriction 2n+1)_0)$.

If ψ is a partial computable function, we say it is a *partial strategy* for $\mathbf{player_0}$ if

$$(\forall i < \omega)\ [|\psi * \varphi_i| \neq \omega \to \exists n \in \omega\ |\psi * \varphi_i| = 2n],$$

and we say it is a *weak winning strategy* for $\mathbf{player_0}$ in game $\mathcal{G}$ if satisfies the furthermore condition

$$(\forall i < \omega)\ |\psi * \varphi_i| = \omega \to \psi * \varphi_i \in \mathcal{G}.$$

Similarly, If ψ is a partial computable function, we say it is a *partial strategy* for $\mathbf{player_1}$ if

$$(\forall i < \omega)\ [|\psi * \varphi_i| \neq \omega \to \exists n \in \omega\ |\psi * \varphi_i| = 2n+1],$$

and we say it is a *weak winning strategy* for $\mathbf{player_1}$ in game $\mathcal{G}$ if satisfies the furthermore condition

$$(\forall i < \omega)\ |\psi * \varphi_i| = \omega \to \psi * \varphi_i \in \overline{\mathcal{G}}.$$

A game is called *weakly determined* if one of the players has a weak winning strategy. Otherwise it is called *weakly non-determined*.

3.1. **Non-determinacy.** Determinate games can be easily constructed. Note that the empty set is an effective open set and $\mathbf{player_1}$ has a trivial winning strategy in the effective open game with empty payoff class. He can blindly

play 1 at each of his move. We are interested in finding out non-determined ones and relations between them.

Theorem 5. *There is a non-determined effectively closed game.*

Proof. It is well known that there is a Π^0_1 class that does not contain any computable member (set). Let T be a computable tree such that $[T]$ is such a Π^0_1 class. Consider another computable true $2^{<\omega} * T$. Then effectively closed game $[2^{<\omega} * T]$ is non-determined. It is easy to see that $\mathbf{player_0}$ does not have a winning strategy. Now suppose $\mathbf{player_1}$ has a winning strategy φ_e, and let x be the infinite sequence of 0's. Clearly $(x * \phi)_1$ is a computable path in T which contradicts the assumption about T. □

3.2. **Incomparable games.**

Theorem 6. *There are two incomparable non-determined effectively closed game.*

Proof. It is known that there is an infinite Π^0_1 class such that any two members of it are Turing incomparable. Let T be a computable tree such that $[T]$ is such a Π^0_1 class, and let σ and τ be two incompatible finite strings in T such that $[\![\sigma]\!] \cap [T] \neq \emptyset$ and $[\![\tau]\!] \cap [T] \neq \emptyset$.

Let T_1 be the computable true such that $[T_1] = [\![\sigma]\!] \cap [T]$ and let T_2 be the computable true such that $[T_2] = [\![\tau]\!] \cap [T]$. Now the effectively closed games $\mathcal{G}_1 = [2^{<\omega} * T_1]$ and $\mathcal{G}_2 = [T_2 * 2^{<\omega}]$ are incomparable. Clearly $\mathbf{player_1}$ does not have a winning strategy in $\mathcal{G}_1 \otimes \overline{\mathcal{G}_2}$ by Lemma 2.

Suppose φ is a winning strategy for $\mathbf{player_0}$ in the game $\mathcal{G}_1 \otimes \overline{\mathcal{G}_2}$. It is easy to see that there is a play of $\mathcal{G}_1 \otimes \overline{\mathcal{G}_2}$ such that $\overline{\mathcal{G}_2}$ is won by $\mathbf{player_1}$, and $\mathbf{player_1}$ always plays 0 in $\mathcal{G}_1$. Because $\overline{\mathcal{G}_2}$ is won by $\mathbf{player_1}$, the sequence X of his moves in $\overline{\mathcal{G}_2}$ is a member of $[T_2]$. Since $\mathbf{player_0}$ uses the winning strategy φ, $\mathcal{G}_1$ is won by her and hence the sequence Y of her moves in $\mathcal{G}_1$ must be a member of $[T_1]$. But then Y is computable from X, contradiction. □

Corollary 7. *There is a countable anti-chain in the game lattice.*

Proof. The same proof. Fix a Π^0_1 class that has continuum many mutually Turing incomparable infinite paths; see Jockusch-Soare [76]. □

Theorem 8. *For each two Π^0_1 classes $[T]$ and $[S]$, $[T]$ is Medvedev reducible to $[S]$ if and only if the effectively closed game $[T * 2^{<\omega}]$ is reducible to the effectively closed game $[S * 2^{<\omega}]$.*

Proof. The left to right direction is trivial. Now suppose $[T * 2^{<\omega}] \leq [S * 2^{<\omega}]$, which means $\mathbf{player_0}$ has a winning strategy φ in the game $[T * 2^{<\omega}] \otimes \overline{[S * 2^{<\omega}]}$. For each $X \in [S]$, consider the play of $[T * 2^{<\omega}] \otimes \overline{[S * 2^{<\omega}]}$ where $\mathbf{player_1}$ always plays 0 in $[T * 2^{<\omega}]$ and plays the sequence X in $\overline{[S * 2^{<\omega}]}$, and $\mathbf{player_0}$ uses the winning strategy φ. By the choice of X we know that $\overline{[S * 2^{<\omega}]}$ is won by $\mathbf{player_1}$ or unfinished. Since φ is a winning strategy,

$[T * 2^{<\omega}]$ must has been won by $\mathbf{player_0}$, which means the sequence of her moves is in $[T]$. Now it is not hard to see that we can get a Turing functional Φ based on φ such that $\Phi(X) \in [T]$ for any $X \in [S]$.

□

Corollary 9. *The map that takes the Medvedev degree of each Π^0_1 class $[T]$ to the degree of non-determinacy of the effectively closed game $[T * 2^{<\omega}]$ is an order preserving embedding of the Medvedev degrees of Π^0_1 classes to the degrees of non-determinacy of effectively closed games. Moreover the embedding preserves the meet but not the join.*

Proof. The first part follows directly from the theorem. It is not hard to show that the embedding preserve the meet. To see that it does not preserve the join, let T_1 and T_2 be the same as in Theorem 6 and we show that the game $[(T_1 * T_2) * 2^{<\omega}]$ is not below the game $[T_1 * 2^{<\omega}] \vee [T_2 * 2^{<\omega}]$. Note that in the game $[T_1 * 2^{<\omega}] \vee [T_2 * 2^{<\omega}]$, each play is essentially either a member of $[T_1]$ or that of $[T_2]$, say $[T_1$ without loss of generality. It is impossible to compute a member of T_2 from a member of $[T_1]$ and hence it is also impossible to compute a member of $[T_1 * T_2]$ from a member of $[T_1]$. □

Proposition 10. *Let A, B and C three arbitrary games. Then $A \vee (B \wedge C) \leq (A \vee B) \wedge (A \vee C)$.*

Proof. Suppose we are $\mathbf{player_1}$. In $A \vee (B \wedge C)$, $\mathbf{player_0}$ must decide at the beginning whether to play A or $B \wedge C$. At the mean time in $\overline{(A \vee B) \wedge (A \vee C)}$ $\mathbf{player_0}$ must decide at the beginning whether to play $\overline{A \vee B}$ or $\overline{A \vee C}$. No matter how $\mathbf{player_0}$ makes her decision, we can choose properly in the next round to make sure that we end up playing $X \overline{\otimes} \overline{X}$, X being A, or B, or C. □

But the other direction does not hold in general:

Proposition 11. *Let A, B and C be pairwise incomparable. Then $A \vee (B \wedge C) < (A \vee B) \wedge (A \vee C)$.*

Proof. We only need to show that $(A \vee B) \wedge (A \vee C) \not\leq A \vee (B \wedge C)$. Suppose $A \vee (B \wedge C) \leq (A \vee B) \wedge (A \vee C)$. Then the winning strategy for $\mathbf{player_1}$ in the game $(A \vee B) \wedge (A \vee C) \overline{\otimes} \overline{A \vee (B \wedge C)}$ chooses $A \vee B$ or $A \vee C$ on the left hand and chooses $\overline{A}$ or $\overline{B \wedge C}$ on the right hand. Suppose the choices are $A \vee B$ and $\overline{B \wedge C}$. Then $\mathbf{player_0}$ can choose to play A on the left hand and choose to play $\overline{B}$ on the right hand. Now the winning strategy for $\mathbf{player_1}$ in the game $(A \vee B) \wedge (A \vee C) \overline{\otimes} \overline{A \vee (B \wedge C)}$ gives another winning strategy for $\mathbf{player_1}$ in the game $A \overline{\otimes} \overline{B}$ which proves $B \leq A$. Contradiction. □

Hence the game lattice is not distributive. We can even show that it is not modular.

Proposition 12. *Let A, B and C be arbitrary. Then $(A \wedge C) \vee (B \wedge C) \leq [(A \wedge C) \vee B] \wedge C$.*

Proof. In the game $(A \wedge C) \vee (B \wedge C) \overline{\otimes} \overline{[(A \wedge C) \vee B] \wedge C}$ $\mathbf{player_0}$ must choose between $A \wedge C$ and $B \wedge C$ on the left board and between $\overline{C}$ and $\overline{(A \wedge C) \vee B}$ on the right board. If the choice of her on the right board is $\overline{C}$, then $\mathbf{player_1}$ can always choose C on the left board and win the game by copycat strategy.

If her choice on the right is $\overline{(A \wedge C) \vee B}$, then depending on the choice of $\mathbf{player_0}$'s choice on the left, $\mathbf{player_1}$ can make sure that they end up playing $X \overline{\otimes} \overline{X}$, X being A, or B, or C. If $\mathbf{player_0}$ choses $A \wedge C$ on the left board, then $\mathbf{player_1}$ choses to play $\overline{A \wedge C}$ on the right board, and after $\mathbf{player_0}$ making her choice between $\overline{A}$ and $\overline{C}$ he choses A or C accordingly; if $\mathbf{player_0}$ choses $B \wedge C$ on the left board, $\mathbf{player_1}$ can choose B on the left board and $\overline{B}$ on the right board.

□

Proposition 13. *Let A, B and C be three pairwise incomparable games. Then $(A \wedge C) \vee (B \wedge C) < [(A \wedge C) \vee B] \wedge C$.*

Proof. We only need to show that $[(A \wedge C) \vee B] \wedge C \not\leq (A \wedge C) \vee (B \wedge C)$. Suppose $\mathbf{player_1}$ has computable winning strategy in the game $[(A \wedge C) \vee B] \wedge C \overline{\otimes} \overline{(A \wedge C) \vee (B \wedge C)}$. That strategy tells him which to choose between C and $(A \wedge C) \vee B$ on the right board and which to choose between $\overline{A \wedge C}$ and $\overline{B \wedge C}$ on the left board. The choice on the right cannot be C because $\mathbf{player_0}$ can make sure they end up playing $\overline{X} \overline{\otimes} C$ where X is not C. So the strategy tells $\mathbf{player_1}$ to play $(A \wedge C) \vee B$ on the right board. But $\mathbf{player_0}$ can then choose B on the right board and $\overline{X}$ for some $X \neq B$. Contradiction. □

Corollary 14. *The game lattice is not modular and hence not distributive.*

Corollary 15. *Not every finite partial order can be imbedded into the game lattice.*

Question 16 (open)**.** *Is the first order theory of the game lattice(preordering) decidable?*

4. A general framework for game semantics

Blass gave very good introductions and backgrounds on the works of game semantics for linear logic in [Bla94] and [Bla97].The author of this paper introduced the concept of game algebras of cardinal numbers in his master thesis [Li11]. We generalise that idea to introduce a framework that is general enough to cover a large scale of works on game semantics for various logic systems.

A propositional logic $\mathfrak{L}$ has two parts: an effective syntax and an effective derivation (proof) system. The effective syntax $\mathfrak{L}$ is an algorithm that decide whether a sequence of symbols taken from a set of propositional variables and a set of logic symbols (connectives) which is denoted by $\mathfrak{O}$ is a well formed formula of the logic. We use $\mathfrak{L}(\mathfrak{O})$ to denote the logic $\mathfrak{L}$ whose set

of logic symbols is $\mathfrak{O}$. The effective derivation (proof) system of $\mathfrak{L}$ contains a computable set of formulas which are called the axioms and an algorithm that decide if a finite sequence of formulas forms a legal derivation (proof) of the last formula in the sequence. The set of formulas which have legal derivations are called theorems of $\mathfrak{L}$. Note that this framework works well with sequent calculus, since a comma can be treated like a logic symbol.

A game algebra is $(\mathfrak{G}, \mathfrak{O}_{\mathfrak{G}})$ where $\mathfrak{G}$ is a class of games and $\mathfrak{O}_{\mathfrak{G}}$ is the collection of operations defined on $\mathfrak{G}$ for each symbol in $\mathfrak{O}$ viewed as operators on games. A semantics is $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ where $(\mathfrak{G}, \mathfrak{O}_{\mathfrak{G}})$ is a game algebra, $\mathfrak{A}$ is a subclass of $\mathfrak{G}$, and $\mathfrak{S}$ is a class of functions. We call the games in $\mathfrak{A}$ atomic games. Note that if we are given a formula in $\mathfrak{L}(\mathfrak{O})$ and we replace propositional variables in the formula by arbitrary games in $\mathfrak{A}$ and view the symbols in $\mathfrak{O}$ as game operators, then that formula now represents a game in $\mathfrak{G}$.

Definition 17. *Let $\mathfrak{L}(\mathfrak{O})$ be a propositional logic and $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ be a game semantics.*

A formula ϕ of $\mathfrak{L}(\mathfrak{O})$ is valid in $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ if for each assignment of atomic games in $\mathfrak{A}$ to the propositional variables in ϕ, $\mathbf{player_1}$ *has a winning strartegy that is in $\mathfrak{S}$ in the game that interpretates ϕ under that assignment.*

The game semantics $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ is sound for the logic $\mathfrak{L}(\mathfrak{O})$ if every theorem of $\mathfrak{L}(\mathfrak{O})$ is valid in $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$; $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ is complete for $\mathfrak{L}(\mathfrak{O})$ if every formula of $\mathfrak{L}(\mathfrak{O})$ that is valid in $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ is a theorem of $\mathfrak{L}(\mathfrak{O})$.

In many known works on game semantics, there are no distinguishments between atomic games and non-atomic games. That is a special case in our new framework, namely the case when $\mathfrak{A} = \mathfrak{G}$. For this special case, readers can find motivations and intuitive explanations of game validity in chapter 3 of the author's master thesis. We want to work in this new framework for two reasons. Firstly it makes a lot of sense to assign only atomic games to propositional variables since the latter are the atoms in logic and some times we want to keep atomic things simple. Secondly, in the new framework the game operators $\mathfrak{O}_{\mathfrak{G}}$ need not to be closed on the $\mathfrak{A}$ and so in general we can start with any class of games that we are interested in without worrying about closures of game operators.

The author bridged the study of degrees of non-determinacy of games and game semantics in his master thesis, and gave a sufficient condition that is purely game focused for a complete semantics for the additive fragment of affine logic. For more details about linear logic and its additive and multiplicative fragments see [Bla92]. We state a meta-theorem below which is a generalisation of Theorem 3.3.1 in Zhenhao's master thesis. The proof is essentially the same as in Theorem 3.3.1 in Zhenhao's master thesis. We put the proof of this metatheorem in the last section.

Definition 18. *An additive Blass game semantics (for linear logic) is a game semantics* $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ *where* $\mathfrak{O}_{\mathfrak{G}}$ *is the set of logic symbols of the additive fragment of linear logic and the corresponding game operators in* $\mathfrak{O}_{\mathfrak{G}}$ *are the game operators defined in Section 2, in the way shown in table 1.*

TABLE 1. The interpretation of $\mathfrak{O}$ in $\mathfrak{O}_{\mathfrak{G}}$

Symbols in $\mathfrak{O}$	*Corresponding game operators*
$\perp$*(linear negation)*	$\sim$
	$\wedge$
	$\vee$
	$\Cap$
	$\Cup$

[1]

Theorem 19. *Let* $\mathfrak{L}(\mathfrak{O})$ *be the additive fragment of linear logic, and* $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ *a Blass game semantics. If there is a countable subset* $\mathcal{C}$ *of* $\mathfrak{A}$ *such that for each finite sequence* $A_1, ..., A_k, B_1, ..., B_n$ *of distinct games in* $\mathcal{C}$*,* **player$_1$** *does not have a winning strategy that is in* $\mathfrak{S}$ *in the game* $A_1 \overline{\otimes} ... \overline{\otimes} A_k \overline{\otimes} \overline{B_1} \overline{\otimes} ... \overline{\otimes} \overline{B_n}$*, then* $(\mathfrak{A}, \mathfrak{G}, \mathfrak{O}_{\mathfrak{G}}, \mathfrak{S})$ *is complete for* $\mathfrak{L}(\mathfrak{O})$*.*

A direct consequence of the metatheorem is that the effective games provide a complete semantics if we require strategies to be computable.

Corollary 20. *The effective games gives a complete semantics.*

Proof. There is a Π^0_1 class such that the paths are independent, i.e., no path is computable from the total information of any finite many other paths. □

5. Proof of the metatheorem

Our proof is based on the proof of the Additive Completeness Theorem in [Bla92]. We omit some details here to avoid repitition. Interested reader can read our proof here along with [Bla92].

Let $\Gamma = C_1, C_2, ..., C_m$ where C_k are additive formulas. Consider an arbitrary game interpretation and the corresponding game Γ which is the game $C_1 \overline{\otimes} ... \overline{\otimes} C_m$. The moves in any component game C_k come in two phases. In phase 1, the players are choosing conjuncts or disjuncts in subformulas for game C_k. For example, if C_k is $(p \wedge \overline{q}) \vee r$, where p, q, r are propositional variables, then phase 1 contains P's opening move, choosing $p \wedge \overline{q}$ or r, and, if he chooses the former, then phase 1 also contains O's reply, choosing p or $\overline{q}$. Each phase 1 move replaces the kth component of Γ by one of its conjuncts or disjuncts, and phase 1 continues in the kth

[1] if we use the sequent calculus, $\overline{\otimes}$ is a symbol in the additive fragment because commas on the right hand of a sequent is interpreted by $\overline{\otimes}$.

component until it is reduced to a literal, i.e., to a propositional variable or the negation of one one or $\top$ or $\bot$. Then comes phase 2, in which the players play (the game associated to) that literal. In any component, the phase 1 moves precede the phase 2 moves, but it is possible for phase 2 to begin in one component before phase 1 is finished in another component. It is also possible for a play of Γ to have only finitely many moves in some component, and then phase 1 may not be finished there.

At any stage of the play, we write Γ' for the current list of component games. Initially, Γ' is Γ, but every phase 1 move replaces some formula in Γ' with one of its conjuncts or disjuncts.

The proceeding discussion concerned an arbitrary play of Γ. We now focus our attention on particular plays of Γ in which P follows a strategy σ while O plays phase 1 so that Γ' is never provable. O can do this, as shown below. Initially, Γ' is Γ, which is unprovable, by assumption. If Γ' is unprovable at some point during the play, and if P then makes a phase 1 move, then Γ' will still be unprovable after this move. Indeed a phase 1 move of P replaces a component of the form $A \vee B$ with A or with B, so, up to order of components, Γ' before the move was $\Delta, A \vee B$ and Γ' after the move is either Δ, A or Δ, B. But if either $\vdash \Delta, A$ or $\vdash \Delta, B$, then, by rule $(\vee)$, we have $\vdash \Delta, A \vee B$, a contradiction. Thus, phase 1 moves of P cannot make Γ' provable. A phase 1 move of O changes Γ' form $\Delta, A \wedge B$, to Δ, A or Δ, B. By rule $(\wedge)$, if $\Delta, A \wedge B$ is unprovable, then so is at least one of Δ, A or Δ, B. So O can make his phase 1 moves in accordance with instruction we just gave.

Now consider a interpretation $\mathcal{I}$ that maps each propositional variable v_i to $[G_i]$. We show that any strategy σ of P cannot be a winning strategy. Consider those plays of Γ satisfying the description in the last paragraph. By the preceding discussion, Γ' never contains 1, otherwise Γ' would be provable by rule (1). Similarly, Γ' can never contain both p and $\overline{p}$ for any propositional variable p, by the logical axiom and the weakening rule. So eventually Γ' is $(\overline{\bigotimes}_{i\in I} G_i) \mathbin{\overline{\otimes}} (\overline{\bigotimes}_{j\in J} \overline{G}_j)$ where I and J are finite and $I \cap J = \emptyset$. By our assumption, σ cannot be a winning strategy for P. This completes the proof.

6. THE MULTIPLICATIVE FRAGMENT

Theorem 21 (effective)**.** *If a multiplicative formula (normal form) is game valid, then it is an instance of a binary tautology.*

Proof. ?????????? Property: we can use a single game for all the literals. We can give a name to this property.

Fix a Π^0_1 class whose members are independent and let T be its tree. Consider the game G_T where player0 plays a finite string $\sigma \in T$ (otherwise she loses) and then player1 plays an infinite sequence x and wins the game if $\sigma x \in [T]$.

Let C be a multiplicative formula that is not an instance of a binary tautology. Fix a prefix-free set of strings σ_l for each occurrence of literal

l. For each occurrence of literal l fix an infinite sequence $x_{l,\tau}$ such that $\tau x_{l,\tau} \in [T]$ and that $\tau x_{l,\tau}$ is unique for each occurrence of literal l.

Suppose P uses σ. O chooses, at each of his moves, a subgame (=occurrence of literal) l in which (1) he can legally move (i.e., the current labels between l and the root of the parse tree are all True), and (2) the current position in l contains as few moves as possible, subject to (1). In l, O uses either σ_l or $x_{l,\tau}$ where τ is P's move as his sequence of moves, depending which role she plays.

Freezing stages, looking back

If l and l' are two occurrences of the same literal, and if the play x that we have just produced has infinite subsequences $(x)_l$, and $(x)_{l'}$ of moves in these two subgames, then $(x)_l \neq (x)_{l'}$, because O's moves in these two plays are different.

On the other hand, it is possible that l and l' are occurrences of p and $\overline{p}$, respectively, and that $(x)_l = (x)_{l'}$ and these subsequences are infinite. Indeed, O's moves z_l in $(x)_l$ might match P's moves in $(x)_{l'}$, (since l' is the negation of l, the players have reversed roles) and vice versa; for example, P's strategy σ might involve copying O's moves between l and l'. If this occurs, we say that l and l' are matched. Notice that, by the preceding paragraph, any l is matched with at most one l'.

Consider the formula C^* obtained from C by changing all occurrences of variables to distinct variables except that matched occurrences of literals p and $\overline{p}$ retain the same variable. Clearly, C is an instance of C^* and C^* is binary. But we assumed that C is not an instance of a binary tautology. So C^* is not a tautology. Fix a truth assignment making C^* false.

We regard this truth assignment as assigning truth values as labels to the leaves of the parse tree of C. This labeling, which we extend in the usual way to the whole parse tree and call the **preferred** labeling, need not be a real truth assignment for C, since different occurrences of the same variable in C became different variables in C^*, and may thus have received different truth values. However, if l and l' are matched literals in C, then one remained the negation of the other in C^*, so they received opposite truth values. Summarizing the properties of the preferred labeling that we will need later, we have

(1) matched literals have opposite truth values, and

(2) the root is labeled False.

For each literal occurrence l such that $(x)_l$ is infinite in the play x described above, we note that $(x)_l$ is a member of ${}^{\omega}2$ that was not decided at any previous stage of the definition of the G_p's. Indeed, z_l, the subsequence of Os moves in $(x)_l$, was chosen to differ from the subsequence of either players moves in any previously decided sequence. We can therefore freely define $G_p((x)_l)$ for p's. We use this freedom to try to make the labeling of the parse tree associated to x match the preferred labeling. Thus, if l is an occurrence of p (respectively, $\overline{p}$) and is labeled True (respectively, False) in the preferred labeling, then we define $G_p((x)_l) = 0$. On the other hand, if l is

an occurrence of p (respectively, $\overline{p}$) and is labeled False (respectively, True) in the preferred labeling, then we define $G_p((x)_l) = 1$. (Other decisions, about $G_p((x)_l)$ when l is neither p nor $\overline{p}$, can be made arbitrarily.) The decisions just described do not conflict with one another. Indeed, the only possibility for conflict would be if $(x)_l = (x)_{l'}$ for two distinct occurrences of literals, l and l'. But then l and l' are matched and therefore get opposite truth values in the preferred labeling. Since one of l and l' is an occurrence of some p and the other of $\overline{p}$, opposite labels ensure that $G_p((x)_l) = G_p((x)_{l'})$.

This completes the description of stage σ of the construction of the G_p's. It remains to verify that this stage ensures that σ is not a winning strategy for P in the game C. For this purpose, we consider the play x used for stage σ. It was defined as a play where P uses strategy σ, so we need only check that O wins this play.

If the sequence (x), of moves in (the game corresponding to) l were infinite for every occurrence l of a literal in C, then our task would be trivial. The decisions made at stage σ would ensure that the labeling of the parse tree of C associated to the play x agrees, at all leaves and therefore at all other nodes as well, with the preferred labeling. Since the latter makes the root false, it follows (by the truth-table descriptions of $\otimes$ and $\overline{\otimes}$ games) that O wins the play x. Unfortunately, there is no reason to expect each $(x)_l$ to be infinite, and a finite $(x)_l$ may give l a label (as always, True if O is to move, False if P is to move) different from the preferred label. So a subtler argument is needed. This argument is quite similar to one already used in the proof of Theorem 22, so we omit some details.

In the play of the game C, at all sufficiently late stages, we have, for each literal occurrence l,

(1) if $(x)_l$ is finite, then all moves that will ever be made in subgame l have already been made, and

(2) if $(x)_l$ is infinite, then the number of moves already made in subgame l exceeds the length of every finite $(x)_{l'}$.

At moves of O this late in the game, she does not move in any subgame l for which $(x)_l$ is finite (by (1)), but he would move in such a subgame if he legally could (by the second clause in the description of how O chooses her moves in x, and by (2)). So, when O is to move this late in the game, the path from each such l to the root must contain a label False. A move of O only decreases labels, so such a False is still present afterward, when P is to move next. So, at all sufficiently late stages

(3) if $(x)_l$ is finite, then the path from l to the root contains at least one label False.

If we temporarily fix an I such that $(x)_l$ is finite and if we consider, on the path from l to the root, the False nearest l, we see that its location is unaffected by moves of O and can move only toward the root at moves of P. So this False is always at the same location X from some stage on. At such late stages, O will never move in subgames l beyond X in the parse tree (i.e., occurrences of literals within the subformula X), and therefore P

will move there only finitely often. Waiting until all these moves have been made, we see that, at all sufficiently late stages in the play, nothing happens beyond X, so the labeling of the subtree with root X remains unchanged. In particular, as X was chosen to have label False at all sufficiently late stages, it also has label False in the final labeling associated to the play x.

We have shown that every l for which $(x)_l$, is finite is within a subformula $X(I)$ (meaning a subformula X containing l) whose final label is False. We complete the proof by considering the following three labelings of the parse tree of C.

(a): The preferred labeling.
(b): The final labeling associated to the play x.
(c): The labeling that agrees with **(a)** and **(b)** at all l for which $(x)_l$ is infinite but assigns False to all l for which $(x)_l$ is finite.

Notice that (c) makes sense, because we already know that **(a)** and **(b)** agree at l when $(x)_l$ is infinite. We also know that (a) labels the root C with False; by monotonicity of $\otimes$ and $\overline{\otimes}$, (c) also labels the root with False. Now consider what happens if we change labeling (b) to **(c)**. The only changes at leaves of the parse tree are decreases (from True to False) at some l's for which $(x)_l$ is finite. The only changes at interior nodes are decreases (by monotonicity again) along the paths from such l's to the root. But every such path contains a node $X(I)$ that was already labeled False in **(b)** and that is therefore unaffected by the decreases in going from **(b)** to **(c)**. But if the change at l does not affect the label at $X(I)$, it cannot affect labels nearer the root. In particular, C has the same label in (b) as in **(c)**, and we already know that the latter is False. So C is False in the labeling associated to x, i.e., O wins the play x.

This shows the stage σ of our construction prevents σ from being a winning strategy for P in C. The whole construction therefore ensures that P has no winning strategy in (this interpretation of) C, and so C is not valid.

We construct (the degrees of) reversely strict games on 2 to interpret the game variables in C so that P has no winning strategy in the interpretation of C, or just game C for easier reading. The game G_p associated to a game variable p will be reversely strict and on 2 so that G_p has been defined on finite sequences. We have yet to specify the G_p on infinite sequences, but G_p being reversely strict is enough to determine the set of positions in the game C and the set of strategies for P in C. As there are only countably many positions (finite plays), the number of strategies is the cardinality $|^\omega 2| = 2^{\aleph_0}$, or the cardinality $\mathfrak{c}$ of the continuum. By **AC**, fix a well-ordering of the set of all strategies for P, having order-type $2^{\aleph_0}$: thus, each strategy σ has fewer than $2^{\aleph_0}$ predecessors in this well-ordering.

We will define G_p on infinite sequences by transfinite recursion over this well-ordering. At the recursion step associated to a strategy σ, we will decide, for finitely many $x \in {}^\omega 2$, the value of $G_p(x)$. We say that these x's are decided at stage σ. These decisions will be made in a way that ensures

that σ is not a winning strategy for P in C. As every possible strategy for P in C occurs in our well-ordering, the whole construction will ensure that P has no winning strategy for C. The rest of the proof consists of showing how to carry out one step in the induction, say the step associated to σ.

There have been fewer than $2^{\aleph_0}$ previous steps, each deciding only finitely many $x \in {}^{\omega}2$. We split each of these xs into the two subsequences of moves attributable to the two players, i.e., $x|_0$ and $x|_1$. Thus, for each decided x, one subsequence consists of the even-numbered moves in x, the other of the odd-numbered moves, because we are dealing with strict games. There are fewer than $2^{\aleph_0}$ subsequences so obtained – two from each of fewer than $2^{\aleph_0}$ decided x's – so we can do the following.

For each occurrence l of a literal (i.e., a positive occurrence of a variable or negated variable) in C, choose a different sequence $z_l \in {}^{\omega}2$ that does not occur as the sequence of moves of either player in any previously decided x. Note that the same variable or negated variable may have several occurrences, corresponding to several subgames of C; these count as different literals l and have different z_l's assigned to them.

Construct a play of C as follows. P uses σ. O chooses, at each of his moves, a subgame (=occurrence of literal) l in which (1) he can legally move (i.e., the current labels between l and the root of the parse tree are all True), and (2) the current position in l contains as few moves as possible, subject to (1). In l, O uses z_l as his sequence of moves.

If l and l' are two occurrences of the same literal, and if the play x that we have just produced has infinite subsequences $(x)_l$, and $(x)_{l'}$ of moves in these two subgames, then $(x)_l \neq (x)_{l'}$, because O's moves in these two plays are $z_l \neq z_{l'}$.

On the other hand, it is possible that l and l' are occurrences of p and $\overline{p}$, respectively, and that $(x)_l = (x)_{l'}$ and these subsequences are infinite. Indeed, O's moves z_l in $(x)_l$ might match P's moves in $(x)_{l'}$, (since l' is the negation of l, the players have reversed roles) and vice versa; for example, P's strategy σ might involve copying O's moves between l and l'. If this occurs, we say that l and l' are matched. Notice that, by the preceding paragraph, any l is matched with at most one l'.

Consider the formula C^* obtained from C by changing all occurrences of variables to distinct variables except that matched occurrences of literals p and $\overline{p}$ retain the same variable. Clearly, C is an instance of C^* and C^* is binary. But we assumed that C is not an instance of a binary tautology. So C^* is not a tautology. Fix a truth assignment making C^* false.

We regard this truth assignment as assigning truth values as labels to the leaves of the parse tree of C. This labeling, which we extend in the usual way to the whole parse tree and call the **preferred** labeling, need not be a real truth assignment for C, since different occurrences of the same variable in C became different variables in C^*, and may thus have received different truth values. However, if l and l' are matched literals in C, then one remained the negation of the other in C^*, so they received opposite

truth values. Summarizing the properties of the preferred labeling that we will need later, we have

(1) matched literals have opposite truth values, and

(2) the root is labeled False.

For each literal occurrence l such that $(x)_l$ is infinite in the play x described above, we note that $(x)_l$ is a member of ${}^{\omega}2$ that was not decided at any previous stage of the definition of the G_p's. Indeed, z_l, the subsequence of Os moves in $(x)_l$, was chosen to differ from the subsequence of either players moves in any previously decided sequence. We can therefore freely define $G_p((x)_l)$ for p's. We use this freedom to try to make the labeling of the parse tree associated to x match the preferred labeling. Thus, if l is an occurrence of p (respectively, $\overline{p}$) and is labeled True (respectively, False) in the preferred labeling, then we define $G_p((x)_l) = 0$. On the other hand, if l is an occurrence of p (respectively, $\overline{p}$) and is labeled False (respectively, True) in the preferred labeling, then we define $G_p((x)_l) = 1$. (Other decisions, about $G_p((x)_l)$ when l is neither p nor $\overline{p}$, can be made arbitrarily.) The decisions just described do not conflict with one another. Indeed, the only possibility for conflict would be if $(x)_l = (x)_{l'}$ for two distinct occurrences of literals, l and l'. But then l and l' are matched and therefore get opposite truth values in the preferred labeling. Since one of l and l' is an occurrence of some p and the other of $\overline{p}$, opposite labels ensure that $G_p((x)_l) = G_p((x)_{l'})$.

This completes the description of stage σ of the construction of the G_p's. It remains to verify that this stage ensures that σ is not a winning strategy for P in the game C. For this purpose, we consider the play x used for stage σ. It was defined as a play where P uses strategy σ, so we need only check that O wins this play.

If the sequence (x), of moves in (the game corresponding to) l were infinite for every occurrence l of a literal in C, then our task would be trivial. The decisions made at stage σ would ensure that the labeling of the parse tree of C associated to the play x agrees, at all leaves and therefore at all other nodes as well, with the preferred labeling. Since the latter makes the root false, it follows (by the truth-table descriptions of $\otimes$ and $\overline{\otimes}$ games) that O wins the play x. Unfortunately, there is no reason to expect each $(x)_l$ to be infinite, and a finite $(x)_l$ may give l a label (as always, True if O is to move, False if P is to move) different from the preferred label. So a subtler argument is needed. This argument is quite similar to one already used in the proof of Theorem 22, so we omit some details.

In the play of the game C, at all sufficiently late stages, we have, for each literal occurrence l,

(1) if $(x)_l$ is finite, then all moves that will ever be made in subgame l have already been made, and

(2) if $(x)_l$ is infinite, then the number of moves already made in subgame l exceeds the length of every finite $(x)_{l'}$.

At moves of O this late in the game, she does not move in any subgame l for which $(x)_l$ is finite (by (l)), but he would move in such a subgame if he

legally could (by the second clause in the description of how O chooses her moves in x, and by (2)). So, when O is to move this late in the game, the path from each such l to the root must contain a label False. A move of O only decreases labels, so such a False is still present afterward, when P is to move next. So, at all sufficiently late stages

(3) if $(x)_l$ is finite, then the path from l to the root contains at least one label False.

If we temporarily fix an I such that $(x)_l$ is finite and if we consider, on the path from l to the root, the False nearest l, we see that its location is unaffected by moves of O and can move only toward the root at moves of P. So this False is always at the same location X from some stage on. At such late stages, O will never move in subgames l beyond X in the parse tree (i.e., occurrences of literals within the subformula X), and therefore P will move there only finitely often. Waiting until all these moves have been made, we see that, at all sufficiently late stages in the play, nothing happens beyond X, so the labeling of the subtree with root X remains unchanged. In particular, as X was chosen to have label False at all sufficiently late stages, it also has label False in the final labeling associated to the play x.

We have shown that every l for which $(x)_l$, is finite is within a subformula $X(I)$ (meaning a subformula X containing l) whose final label is False. We complete the proof by considering the following three labelings of the parse tree of C.

(a): The preferred labeling.
(b): The final labeling associated to the play x.
(c): The labeling that agrees with **(a)** and **(b)** at all l for which $(x)_l$ is infinite but assigns False to all l for which $(x)_l$ is finite.

Notice that (c) makes sense, because we already know that **(a)** and **(b)** agree at l when $(x)_l$ is infinite. We also know that (a) labels the root C with False; by monotonicity of $\otimes$ and $\overline{\otimes}$, (c) also labels the root with False. Now consider what happens if we change labeling (b) to **(c)**. The only changes at leaves of the parse tree are decreases (from True to False) at some l's for which $(x)_l$ is finite. The only changes at interior nodes are decreases (by monotonicity again) along the paths from such l's to the root. But every such path contains a node $X(I)$ that was already labeled False in **(b)** and that is therefore unaffected by the decreases in going from **(b)** to **(c)**. But if the change at l does not affect the label at $X(I)$, it cannot affect labels nearer the root. In particular, C has the same label in (b) as in **(c)**, and we already know that the latter is False. So C is False in the labeling associated to x, i.e., O wins the play x.

This shows the stage σ of our construction prevents σ from being a winning strategy for P in C. The whole construction therefore ensures that P has no winning strategy in (this interpretation of) C, and so C is not valid. □

7. PROOF THE MULTIPLICATIVE FRAGMENT

The essential ideas are the following. The (countable many) games are fixed first. Then we show that player 0 can enforce the preferred so that any fixed strategy for player 1 cannot be winning.

All results in this subsection were proved by Blass [**?**]. We will sometimes omit long and tedious recursive definition and inductive proofs and replace them with informal discussion which can be easily translated into precise mathematics by a careful reader.

By Theorem **??** if we have a syntactic characterization of normal forms of valid multiplicative formulas, we thereby also have a syntactic characterization of valid multiplicative formulas themselves. So in the rest of this section by multiplicative formulas, we mean norm forms of them.

Multiplicative formulas can be read as formulas in classical propositional logic, with $\otimes$ and $\overline{\otimes}$ read as conjunction and disjunction, respectively (and $\overline{\ }$, 1 and 0 read as a negation, truth and falsity, respectively). We do not speak of translating multiplicative formulas into the standard symbolism of classical logic (replacing $\otimes$ with A, etc.), as this might lead to confusion with the additive connectives. Instead, we pretend that classical logic is formulated with $\overline{\ }$, $\otimes$ and $\overline{\otimes}$ as its connectives, so that multiplicative formulas of game logic are also formulas of classical logic. So it makes sense to speak of a multiplicative formula being a tautology, or of a positive or negative occurrence of a variable in a multiplicative formula, or of any other concept familiar from classical propositional logic. In particular, by an instance or (substitution instance) of a multiplicative formula A, we mean a formula obtained by replacing the variables in A uniformly by some multiplicative formulas. By the literals in a multiplicative formula C, we will mean the occurrences of variables and negated variables from which C is built by $\otimes$ and $\overline{\otimes}$. We call a multiplicative formula binary if each variable has at most one positive and one negative occurrence.

Note that being a tautology is a syntactic feature of a multiplicative formula, since there are consistent and complete proof systems for classical propositional logic.

Theorem 22. *A multiplicative formula (normal form) in $\mathcal{L}$ is κ-valid if it is an instance of a binary tautology.*

Proof. Since any instance of a valid formula is clearly also valid, it suffices to prove that all binary tautologies are valid. In fact, it will suffice to consider binary tautologies in which every variable occurs exactly twice (once positively and once negatively) and 1 and 0 do not occur. For brevity, we call such tautologies **special**.

To see that we may confine attention to special tautologies, suppose these were known to be valid, and consider an arbitrary binary tautology C. Starting with C, we repeatedly replace subformulas according to the following rules as long as any of the rules apply.

(1) Any literal whose variable having only one occurrence is replaced by 0.
(2) A subformula of the form $1 \otimes A$ or $0 \overline{\otimes} A$ is replaced by A.
(3) A subformula of the form $1 \overline{\otimes} A$ (respectively, $0 \otimes A$) is replaced by 1 (respectively 0).

It is clear that all the formulas produced are binary tautologies, that the process terminates (because each replacement reduces the total number of occurrences of propositional variables, $\otimes$, and $\overline{\otimes}$), and that the final result C' is either a special tautology or simply 1. So by our assumption (and the obvious validity of 1), C' is valid. We intend to infer from this that C is valid, as desired. But this is easy: if a valid formula B' is obtained from a formula B by a single replacement of the form (1), (2) or (3), then B is also valid. Thus, we have shown that we can safely confine our attention to special tautologies.

Before continuing our proof, we introduce a notational simplification for the ease of reading. We do not distinguish games and degrees in the following sense. We often use games rather than degrees to interpret variables, and interpret formulas by games built up from games interpreting variables and game operators the way stated by the formulas literally and we use the formulas themselves to name games interpreting them, and then show P has a (no) winning strategy in the game interpreting a given formula ϕ. But by doing this, we have shown that $\bar{\mathcal{I}}(\phi) = 1$ ($\bar{\mathcal{I}}(\phi) \neq 1$) where $\mathcal{I}$ is such that $\mathcal{I}(g) = [g]$ for each variable g. (Note that the g on the right of $=$ is the game interpreting variable g, according to our naming method.)

Let us show that each special tautology is κ-valid. Fix a special tautology C and fix a game interpretation. We assume that the games assigned to variables are strict games. We will complete the proof of the 'if' half of the theorem by describing a winning strategy for P in the game C.

Recall that the literals in C are the occurrences of variables and negated variables from which C is built by $\otimes$ and $\overline{\otimes}$. Thus, a negative occurrence of a variable p does not count as a literal; rather its context $\overline{p}$ is a literal. As C is special, the literals come in pairs, each containing p and $\overline{p}$ for some variable p. The game C consists of subgames, one for each literal, and who is to move in a given position (finite play) of C and who has lost a given play of C can be obtained by a truth-table method from the same information about those literal games. More precisely, we think in terms of the parse tree of C, with C at the root, literals at the leaves, and $\otimes$ or $\overline{\otimes}$ labeling the internal nodes. A position or play gives a labeling of the leaves as we give truth values to formulas in classical logic (True if P has won or O is to move, False if O has won or P is to move), and the truth values propagate from the leaves to the other nodes according to the truth tables for conjunction ($\otimes$) and disjunction ($\overline{\otimes}$). This can be easily justified from the definitions of $\otimes$ and $\overline{\otimes}$ and we omit the details here.

Since C is a tautology, the root will have label True provided each pair of literals, p and $\overline{p}$, have opposite truth values, so that the labeling is really a

truth assignment (in the sense of classical logic). In fact, since only positive connectives are used in the tree, the root will also have label True if some p and $\overline{p}$ are both labeled True. But it is, of course, entirely possible that p and $\overline{p}$ are both labeled False in a particular position or play (e.g., if P is to move in both of these subgames), and then the root C may well be labeled False.

Whenever P is to move at a certain position in game C, i.e., when C is labeled False, his move consists of choosing a path through the parse tree from the root to a leaf l, such that all the nodes along the path are labeled False, and then making a move in l. (The choice of path will involve a real choice at $\overline{\otimes}$ nodes, where a False label means that both successors are also labeled False. At $\otimes$ nodes, usually (i.e., expect at the first visit to this node) only one successor will be labeled False; see the discussion following the definition of $\otimes$.)

The essential idea for P's winning strategy is to make sure that the plays in paired subgames p and $\overline{p}$ are identical. Since he plays opposite roles in these two subgames, he will (if infinitely many moves are made in each of them) win one and lose the other, so the final labeling of the tree will be a real truth assignment, C will be labeled True, and so P will win C. We must still show that P can carry out the proposed strategy and that he will win even if some of the subgames are unfinished (i.e., have only finitely many moves made in them). For this purpose, we must describe the strategy in somewhat more detail.

P is to ensure that, at each moment during the play of the game, for each pair p, $\overline{p}$ of literals, either the positions (=sequences of moves already played) in p and $\overline{p}$ are identical or else one of them equals the other plus one subsequent move made by O. This condition is certainly satisfied initially, as all positions are initially the empty sequence.

Furthermore, this condition cannot be destroyed by a move of O. To see this, suppose the condition is satisfied at a certain moment, which we call before, and that O then moves, say in subgame p . If the positions in p and $\overline{p}$ were identical before, then afterward the position in p is that in $\overline{p}$ plus the single move just made by O, so the condition remains satisfied. If the position in p before were that in $\overline{p}$ plus a move by O, then O could not have moved in p as it would be P's turn there.

Finally, if the position in $\overline{p}$ before were that in p plus a move of O, then the presence of that move of O in $\overline{p}$ means that the position without the move, the before position in p, is a position with O to move in $\overline{p}$, hence is a position with P to move in p; so again O could not move in p. This shows that a move of O in p (or, for symmetrical reasons, in $\overline{p}$) cannot destroy the condition that P is trying to maintain.

The preceding discussion shows furthermore that, as long as the condition is satisfied, whenever the positions in p and $\overline{p}$ are different, it is P's turn to move in both of them, whereas of course if the positions in p and $\overline{p}$ are equal, then P is to move in one and O in the other.

We show next that if the condition holds at a certain position and if P is to move, then he can move so as to maintain the condition. More precisely, we show that there is a literal l such that (1) the path from l to the root in the parse tree of C is labeled entirely with False, so that P can legally move in the subgame l, and (2) the position in $\bar{l}$ is one move longer than that in l. Here (2) means that O has made a move in $\bar{l}$ which P can simply copy in l, thereby maintaining the desired condition. We call a literal l **good** at a given position if (1) and (2) hold.

Lemma 23. *Let the parse tree of C be labeled, using the appropriate truth tables at the interior nodes but arbitrary labels at t he leaves. If C is labeled False, then there is a pair p, $\overline{p}$ of literals such that the paths joining them to the root are both labeled entirely with False.*

Proof of Lemma 23. Suppose we had a labeling that is a counterexample to the lemma. We saw earlier that, because C is a tautology yet labeled False and because the connectives at interior nodes are monotone, there must be a pair of literals p, $\overline{p}$ both labeled False. As the labeling is a counter-example to the lemma, the path from one of $p, \overline{p}$, say p, to the root contains a label True. Alter the labeling of the leaves by changing p from False to True, and consider the resulting new labeling of the parse tree (in accordance with the connectives, as always). The change at the leaf p can affect only the labels along the path from p to the root and indeed can only increase these labels (i.e., change False to True) because the connectives are monotone. There was already a True label somewhere on this path. That label will therefore be unchanged. But then all labels between that True and the root are also unaffected by our change at p. In particular, the root C retains its previous label, False. This fact, and the fact that no True has been changed to False, means that our modified labeling is still a counterexample to the lemma. It has strictly fewer leaves labeled False than the original counterexample. So, by repeating the process, we have a contradiction. □

Lemma 23 shows that whenever P is to move, there is a pair of subgames p, $\overline{p}$ such that P can legally move in either of them. In particular, the positions in p and $\overline{p}$ cannot be identical, for then it would be P's move in only one of them. If the condition that P wants to maintain holds, then the position in one of p, $\overline{p}$ is one move longer than in the other. But then the latter is a good subgame.

We have seen that if P is to move and the condition holds, then there is a good subgame and P can move in any good subgame so as to maintain the condition. Once the good subgame is chosen, the appropriate move for P is unique; it consists of copying O's last move in the paired subgame. We specify P's strategy more completely by requiring that if there are several good subgames, then he should move in one where the sequence of previous moves is as short as possible. If several are equally short, choose the leftmost one in the parse tree.

Having described P's strategy and verified its feasibility, we show that it is a winning strategy. Suppose x were a play in which P used this strategy but lost. For each literal l, we let $(x)_l$ be the subsequence of moves in x in subgame l. We indicated a proof earlier that P wins if each $(x)_l$ is infinite; the possibility of some $(x)_l$'s being finite necessitates a subtler argument.

Label the nodes of the parse tree of C in the usual way for the play x. As P lost, C is labeled False. Apply Lemma 23 to obtain (the leftmost) p and $\overline{p}$ such that the paths joining these literals to the root are both labeled entirely with False. Then $(x)_p$, and $(x)_{\overline{p}}$, cannot both be infinite, for then they would be identical, thanks to P's strategy, and would have opposite labels (by definition of $\overline{\ }$). Nor can one be finite and the other infinite, for P's strategy ensures that their lengths never differ by more than one. So both are finite, and one, say $(x)_p$, without loss of generality, is one move shorter than the other. Fix such a p.

While playing the game C, leading to the play x, the players arrive after finitely many moves at a position with the following properties for every literal l.

(1) If $(x)_l$ is finite, then all moves that will ever be made in subgame l have already been made. (Subgames that will remain unfinished have been permanently abandoned.)
(2) If $(x)_l$ is infinite, then the number of moves already made in subgame l exceeds the number that have been (or ever will be) made in any finite $(x)_{l'}$.

Of course, once (1) and (2) hold, they continue to hold at all later positions. Call a position 1,2-late if conditions (1) and (2) hold.

Consider any 1,2-late position with P to move. By (1), P's move is in some l such that $(x)_l$ is infinite. But, by his strategy, P moves in a good literal where the current move sequence is as short as possible. By (2), the current move sequence in p is shorter than in I, since $(x)_p$ is finite and $(x)_l$ infinite. So if p were good, P would not have moved in l. Therefore, p is not good. But the position in $\overline{p}$, i.e., $(x)_{\overline{p}}$, is one move longer than the position in p which is $(x)_p$, because of our choice of p. So the only way for p not to be good is that, on the path from p to the root, there is a node labeled True.

We claim that, from some moment on, the positions leading to the play x satisfy

(3) Some node between the root and p is labeled True.

We have just shown this for positions where P is to move. When P moves, however, labels only increase. (One leaf goes from False (with P to move) to True (with O to move), the other leaves are unchanged, and internal nodes are given by monotone connectives.) So P's move cannot destroy (3). Thus, all 1,2-late positions, except possibly the first, are in fact 1,2,3-late (in the obvious sense).

At any 1,2,3-late position, consider the location of the True label nearest p on the path from p to the root. Consider how this location changes as the play proceeds. A move of P is always at a good literal, is therefore never at a leaf beyond this (or any) True label, and therefore never affects either this True label or the False labels between it and p. So the location is unchanged when P moves. A move of O can only decrease labels (from True to False), by the dual of the argument in the preceding paragraph. So the False labels between p and the location being studied are not changed; the True label at this location may change to False, and in this case the new location of the True nearest p (which still exists by (3)) is nearer the root. In summary, the location of the True nearest p moves, at 1,2,3-late stages of the play, only toward the root. As the path on which it moves is finite, it must eventually stop moving. Let X be its final location. Thus, at all sufficiently late stages of the play, we have

(4) X is labeled True.

At such stages, P will never move in literals that are beyond X in the parse tree (i.e., are subformulas of X), because he only moves in good literals and these have only False labels between them and the root. Therefore, at 1,2,3,4-late stages, O moves at most once in any literal beyond X, for once he moves in such a literal, it is P's turn there, and it remains P's turn there forever since P does not move there any more. Therefore, from some stage on, we have

(5) No moves are made in literals beyond X.

But this means that the labeling of leaves beyond X does not change any more. This labeling is therefore the same for any 1,2,3,4,5-late stage as for the final (infinite) play x. The same therefore holds for the label of X. But X is labeled True at a 1,2,3,4,5-late stage, by (4), and is labeled False for x, by our choice of p. This contradiction shows that, when he uses the strategy we described, P cannot lose C. This completes the proof. □

With the help of the Axiom of Choice, we can show that the syntactic characterization we just gave for multiplicative formulas is complete.

Theorem 24 (effective)**.** *If a multiplicative formula (normal form) is game valid, then it is an instance of a binary tautology.*

Theorem 25 (**AC**)**.** *If a multiplicative formula (normal form) in $\mathcal{L}$ is κ-valid, it is an instance of a binary tautology.*

Proof. Let C be a multiplicative formula that is not an instance of a binary tautology. We construct (the degrees of) reversely strict games on 2 to interpret the game variables in C so that P has no winning strategy in the interpretation of C, or just game C for easier reading. The game G_p associated to a game variable p will be reversely strict and on 2 so that G_p has been defined on finite sequences. We have yet to specify the G_p on infinite sequences, but G_p being reversely strict is enough to determine the

set of positions in the game C and the set of strategies for P in C. As there are only countably many positions (finite plays), the number of strategies is the cardinality $|^{\omega}2| = 2^{\aleph_0}$, or the cardinality $\mathfrak{c}$ of the continuum. By **AC**, fix a well-ordering of the set of all strategies for P, having order-type $2^{\aleph_0}$: thus, each strategy σ has fewer than $2^{\aleph_0}$ predecessors in this well-ordering.

We will define G_p on infinite sequences by transfinite recursion over this well-ordering. At the recursion step associated to a strategy σ, we will decide, for finitely many $x \in {}^{\omega}2$, the value of $G_p(x)$. We say that these x's are decided at stage σ. These decisions will be made in a way that ensures that σ is not a winning strategy for P in C. As every possible strategy for P in C occurs in our well-ordering, the whole construction will ensure that P has no winning strategy for C. The rest of the proof consists of showing how to carry out one step in the induction, say the step associated to σ.

There have been fewer than $2^{\aleph_0}$ previous steps, each deciding only finitely many $x \in {}^{\omega}2$. We split each of these xs into the two subsequences of moves attributable to the two players, i.e., $x|_0$ and $x|_1$. Thus, for each decided x, one subsequence consists of the even-numbered moves in x, the other of the odd-numbered moves, because we are dealing with strict games. There are fewer than $2^{\aleph_0}$ subsequences so obtained – two from each of fewer than $2^{\aleph_0}$ decided x's – so we can do the following.

For each occurrence l of a literal (i.e., a positive occurrence of a variable or negated variable) in C, choose a different sequence $z_l \in {}^{\omega}2$ that does not occur as the sequence of moves of either player in any previously decided x. Note that the same variable or negated variable may have several occurrences, corresponding to several subgames of C; these count as different literals l and have different z_l's assigned to them.

Construct a play of C as follows. P uses σ. O chooses, at each of his moves, a subgame (=occurrence of literal) l in which (1) he can legally move (i.e., the current labels between l and the root of the parse tree are all True), and (2) the current position in l contains as few moves as possible, subject to (1). In l, O uses z_l as his sequence of moves.

If l and l' are two occurrences of the same literal, and if the play x that we have just produced has infinite subsequences $(x)_l$, and $(x)_{l'}$ of moves in these two subgames, then $(x)_l \neq (x)_{l'}$, because O's moves in these two plays are $z_l \neq z_{l'}$.

On the other hand, it is possible that l and l' are occurrences of p and $\overline{p}$, respectively, and that $(x)_l = (x)_{l'}$ and these subsequences are infinite. Indeed, O's moves z_l in $(x)_l$ might match P's moves in $(x)_{l'}$, (since l' is the negation of l, the players have reversed roles) and vice versa; for example, P's strategy σ might involve copying O's moves between l and l'. If this occurs, we say that l and l' are matched. Notice that, by the preceding paragraph, any l is matched with at most one l'.

Consider the formula C^* obtained from C by changing all occurrences of variables to distinct variables except that matched occurrences of literals p and $\overline{p}$ retain the same variable. Clearly, C is an instance of C^* and C^* is

binary. But we assumed that C is not an instance of a binary tautology. So C^* is not a tautology. Fix a truth assignment making C^* false.

We regard this truth assignment as assigning truth values as labels to the leaves of the parse tree of C. This labeling, which we extend in the usual way to the whole parse tree and call the **preferred** labeling, need not be a real truth assignment for C, since different occurrences of the same variable in C became different variables in C^*, and may thus have received different truth values. However, if l and l' are matched literals in C, then one remained the negation of the other in C^*, so they received opposite truth values. Summarizing the properties of the preferred labeling that we will need later, we have

(1) matched literals have opposite truth values, and

(2) the root is labeled False.

For each literal occurrence l such that $(x)_l$ is infinite in the play x described above, we note that $(x)_l$ is a member of ${}^{\omega}2$ that was not decided at any previous stage of the definition of the G_p's. Indeed, z_l, the subsequence of Os moves in $(x)_l$, was chosen to differ from the subsequence of either players moves in any previously decided sequence. We can therefore freely define $G_p((x)_l)$ for p's. We use this freedom to try to make the labeling of the parse tree associated to x match the preferred labeling. Thus, if l is an occurrence of p (respectively, $\overline{p}$) and is labeled True (respectively, False) in the preferred labeling, then we define $G_p((x)_l) = 0$. On the other hand, if l is an occurrence of p (respectively, $\overline{p}$) and is labeled False (respectively, True) in the preferred labeling, then we define $G_p((x)_l) = 1$. (Other decisions, about $G_p((x)_l)$ when l is neither p nor $\overline{p}$, can be made arbitrarily.) The decisions just described do not conflict with one another. Indeed, the only possibility for conflict would be if $(x)_l = (x)_{l'}$ for two distinct occurrences of literals, l and l'. But then l and l' are matched and therefore get opposite truth values in the preferred labeling. Since one of l and l' is an occurrence of some p and the other of $\overline{p}$, opposite labels ensure that $G_p((x)_l) = G_p((x)_{l'})$.

This completes the description of stage σ of the construction of the G_p's. It remains to verify that this stage ensures that σ is not a winning strategy for P in the game C. For this purpose, we consider the play x used for stage σ. It was defined as a play where P uses strategy σ, so we need only check that O wins this play.

If the sequence (x), of moves in (the game corresponding to) l were infinite for every occurrence l of a literal in C, then our task would be trivial. The decisions made at stage σ would ensure that the labeling of the parse tree of C associated to the play x agrees, at all leaves and therefore at all other nodes as well, with the preferred labeling. Since the latter makes the root false, it follows (by the truth-table descriptions of $\otimes$ and $\overline{\otimes}$ games) that O wins the play x. Unfortunately, there is no reason to expect each $(x)_l$ to be infinite, and a finite $(x)_l$ may give l a label (as always, True if O is to move, False if P is to move) different from the preferred label. So a subtler

argument is needed. This argument is quite similar to one already used in the proof of Theorem 22, so we omit some details.

In the play of the game C, at all sufficiently late stages, we have, for each literal occurrence l,

(1) if $(x)_l$ is finite, then all moves that will ever be made in subgame l have already been made, and

(2) if $(x)_l$ is infinite, then the number of moves already made in subgame l exceeds the length of every finite $(x)_{l'}$.

At moves of O this late in the game, she does not move in any subgame l for which $(x)_l$ is finite (by (l)), but he would move in such a subgame if he legally could (by the second clause in the description of how O chooses her moves in x, and by (2)). So, when O is to move this late in the game, the path from each such l to the root must contain a label False. A move of O only decreases labels, so such a False is still present afterward, when P is to move next. So, at all sufficiently late stages

(3) if $(x)_l$ is finite, then the path from l to the root contains at least one label False.

If we temporarily fix an I such that $(x)_l$ is finite and if we consider, on the path from l to the root, the False nearest l, we see that its location is unaffected by moves of O and can move only toward the root at moves of P. So this False is always at the same location X from some stage on. At such late stages, O will never move in subgames l beyond X in the parse tree (i.e., occurrences of literals within the subformula X), and therefore P will move there only finitely often. Waiting until all these moves have been made, we see that, at all sufficiently late stages in the play, nothing happens beyond X, so the labeling of the subtree with root X remains unchanged. In particular, as X was chosen to have label False at all sufficiently late stages, it also has label False in the final labeling associated to the play x.

We have shown that every l for which $(x)_l$, is finite is within a subformula $X(I)$ (meaning a subformula X containing l) whose final label is False. We complete the proof by considering the following three labelings of the parse tree of C.

(a): The preferred labeling.
(b): The final labeling associated to the play x.
(c): The labeling that agrees with **(a)** and **(b)** at all l for which $(x)_l$ is infinite but assigns False to all l for which $(x)_l$ is finite.

Notice that (c) makes sense, because we already know that **(a)** and **(b)** agree at l when $(x)_l$ is infinite. We also know that (a) labels the root C with False; by monotonicity of $\otimes$ and $\overline{\otimes}$, (c) also labels the root with False. Now consider what happens if we change labeling (b) to **(c)**. The only changes at leaves of the parse tree are decreases (from True to False) at some l's for which $(x)_l$ is finite. The only changes at interior nodes are decreases (by monotonicity again) along the paths from such l's to the root. But every such path contains a node $X(I)$ that was already labeled False in **(b)** and

that is therefore unaffected by the decreases in going from **(b)** to **(c)**. But if the change at l does not affect the label at $X(I)$, it cannot affect labels nearer the root. In particular, C has the same label in (b) as in **(c)**, and we already know that the latter is False. So C is False in the labeling associated to x, i.e., O wins the play x.

This shows the stage σ of our construction prevents σ from being a winning strategy for P in C. The whole construction therefore ensures that P has no winning strategy in (this interpretation of) C, and so C is not valid. □

References

[Bla92] Andreas Blass. A game semantics for linear logic. *Ann. Pure Appl. Logic*, 56(1-3):183–220, 1992.

[Bla94] Andreas Blass. Is game semantics necessary? In *Computer science logic (Swansea, 1993)*, volume 832 of *Lecture Notes in Comput. Sci.*, pages 66–77. Springer, Berlin, 1994.

[Bla97] Andreas Blass. Some semantical aspects of linear logic. *Log. J. IGPL*, 5(4):487–503, 1997.

[GS53] David Gale and Frank M. Stewart. Infinite games with perfect information. *Annals of Mathematical Studies*, 2(28):245–266, 1953.

[Li11] Zhenhao Li. Degrees of non-determinacy and game logics on cardinals under the axiom of determinacy. Master's thesis, University of Amsterdam, 2011.