On abstract resource semantics and computabilty logic

Ilya Mezhirov
The German Research Center for Artificial Intelligence, TU Kaiserslautern
Email: ilya.mezhirov@dfki.uni-kl.de

Nikolay Vereshchagin*
Lomonosov Moscow State University,
Leninskie Gory, Moscow 119991,
Email: ver@mccme.ru

Abstract

We show that the uniform validity is equivalent to the non-uniform validity for Blass' semantics of [A game semantics for linear logic. Annals of Pure and Applied Logic 56 (1992) 183–220]. We present a shorter proof (than that of [G. Japaridze. The intuitionistic fragment of computability logic at the propositional level. Annals of Pure and Applied Logic 147 (2007), No.3, pp.187-227]) of the completeness of the positive fragment of intuitionistic logic for this semantics, computability logic semantics, and the abstract resource semantics.

1 Japaridze's abstract resource semantics and intuitionistic logic

Sub-structural logics are often understood as logics of resources. Propositional variables mean certain types of abstract resources (rather than assertions, as in classical logic). Each occurrence of a variable p identifies one unit of a resource p. The connectives $\wedge$ (multiplicative AND), $\vee$ (multiplicative OR), $\sqcap$ (additive AND), $\sqcup$ (additive OR), $\neg$ (negation), $!, ?$ (exponential AND and OR) and $\mathbf{0}, \mathbf{1}$ (constants) have the following meaning. The expression $x \wedge y$ means one unit of x and one unit of y (for example, $x \wedge x$ is two units of the resource x). The expression $x \sqcap y$ means an obligation to provide one unit of the resource x or one unit of the resource y where the consumer of resources (the user) makes the choice between x and y (for example, $x \sqcap x$ is the same as x). The formula $x \sqcup y$

*The work was supported in part by the RFBR grant 09-01-00709

means also an obligation to provide one unit of the resource x or one unit of the resource y. However, this time the choice between x and y is made by the provider (again $x \sqcup x$ is the same as x).

What is the interpretation of $\vee$ (multiplicative OR)? Let us use the following metaphor. Assume that resources are coins of different types. Each occurrence of a variable x is regarded as a coin of type x. The coins can be genuine or fake, and the consumer cannot distinguish between genuine and fake ones. The expression $x \vee y$ is understood as a pair of coins, a coin of type x and a coin of type y, such that at least one of them is genuine (however, the user has no idea, which one).

The expression $\neg x$ means the obligation of the user to pass to the provider one coin of type x (we can understand x also as the obligation of the provider to pass to the user one coin of type x). The formula $!\,x$ is understood as an infinite stock of genuine coins of type x. The expression $?\,x$ means an infinite stock of coins of type x such that at least one coin in the stock is genuine (and the user does not know which one). In other words $!\,x$ is a countable version of the multiplicative AND, $x \wedge x \wedge x \wedge \ldots$, and $?\,x$ is a countable version of the multiplicative OR, $x \vee x \vee x \vee \ldots$.

The constants $\mathbf{0}, \mathbf{1}$ are understood as non-accomplishable obligations (tasks): $\mathbf{0}$ is the obligation of the provider and $\mathbf{1}$ is the obligation of the user.

In the next two sections we give a formal definition of a semantics which clarifies this intuition. From both the technical and philosophical viewpoints, this semantics coincides with Japaridze's "The logic of tasks" semantics defined in [5] and later extended to what has been termed *abstract resource semantics* in [8, 12]. What we call "a coin" is called "a task" in [5] and an "abstract resource" in [8, 12]. A genuine coin is a task accomplished by the provider. A false coin is a task that the provider failed to accomplish.

1.1 Syntax

Fix a countable set of *variables* $x_1, x_2, \ldots$. A *literal* is a variable x_i or a negated variable $\neg x_i$. Formulae are obtained from literals, constants $\mathbf{1}, \mathbf{0}$ and connectives $\wedge, \vee, \sqcap, \sqcup, ?, !$ in the usual way. We consider formulae where negations appear only in front of variables. When we write $\neg A$, we always mean the formula dual to A, that is, the formula which is obtained from A by changing each connective, each variable and each constant to its dual: $\vee \leftrightarrow \wedge$, $\sqcup \leftrightarrow \sqcap$, $! \leftrightarrow ?$, $\neg x_i \leftrightarrow x_i$ and $\mathbf{1} \leftrightarrow \mathbf{0}$.

The notion of a sub-formula of a formula A is defined in a usual way. Occurrences of a sub-formula B in A will be called *oformulae*. We define *oliterals* and *ovariables* in a similar way. If an ovariable is preceded by negation then it is called *negative*, otherwise *positive*.

1.2 Semantics

Each formula is assigned a two player game $[A]$ of perfect information. The players are called *Alice* and *Bob*. A formula A will be called valid if Alice has a

winning strategy in the game $[A]$.

First we replace in A each oformula of the type $!\,B$ by the infinite formula $B \wedge B \wedge B \ldots$, and each oformula of the type $?\,B$ by the formula $B \vee B \vee B \ldots$. It is easy to see that the order of replacements does not affect the result, which is an infinite formula of a finite depth.

In the game $[A]$, the players make moves in turn. It does not matter who moves first. In his turn, Bob may perform any finite sequence $a_1, \ldots, a_k$ of *actions* (the sequence may be empty). Each action a_i has the form "choose the left formula in $B \sqcap C$" or "choose the right formula in $B \sqcap C$", where $B \sqcap C$ is an oformula of A. For each oformula $B \sqcap C$ Bob may only once make an action "choose $\ldots$ in $B \sqcap C$", that is, he is not allowed to change the choices he has made.

In her turn, Alice may also perform any finite (possibly empty) sequence of *actions*. Each her action has one of the following two forms. (1) "Choose the left (right) formula in $B \sqcup C$" where $B \sqcup C$ is an oformula of A. (2) "Allocate U to V", where U is a negative occurrence of a variable x_i in A and V is a positive occurrence of the same variable x_i in A. The informal meaning of this action is that Alice wants to accomplish her obligation V using the coin U provided by Bob.

For each positive ovariable V Alice may perform at most one action of the form $U \mapsto V$, and for each negative ovariable U she may perform at most one action of the form $U \mapsto V$. In other words, actions of type (2) establish a matching between positive and negative ovariables that respects variables' names. Note that Bob is not allowed to make allocations. For each oformula $B \sqcup C$ Alice may only once make an action of the type "choose $\ldots$ in $B \sqcup C$".

Each play consists of infinite (countably many) number of moves. When the play is finished, we define who has won as follows. Call a *coin evaluation* any mapping e from the set of all negative ovariables to the set $\{0, 1\}$. The informal meaning of a coin evaluation is the following: If $e(U) = 0$ then the coin U provided by Bob is fake, otherwise (if $e(U) = 1$) it is genuine. Given a coin evaluation e, we recursively extend e to all oformulae of A. We say that B is won by Bob, if $e(B) = 0$, and otherwise by Alice.

(1) If V is a positive ovariable, then $e(V) = 1$, if for some negative ovariable U such that $e(U) = 1$, in the course of the play, Alice has performed an action "allocate U to V". The informal meaning: Alice has accomplished her obligation V if she has allocated a genuine resource U to it.

(2) If B is an occurrence of a negative literal $\neg U$, then Alice has won B iff $e(U) = 0$. Thus $\neg$ acts as a classical negation. Rules (1) and (2) imply the following. If Alice has performed an action "allocate U to V" then exactly one of oformulae $\neg U, V$ is won by Alice.

(3) Alice has won an oformula $B \wedge C$ iff she has won both B and C. She has won an oformula $B \vee C$ iff she has won B or C.

(4) In a similar way we define who has won occurrences of $!\,B$ and $?\,B$: an oformula $B_1 \wedge B_2 \wedge B_3 \ldots$ (we use subscripts to distinguish between different occurrences of B) is won by Alice iff she has won all oformulae $B_1, B_2, B_3 \ldots$. An oformula $B_1 \vee B_2 \vee B_3 \ldots$ is won by Alice iff she has won at least one of

oformulae $B_1, B_2, B_3 \ldots$.

(5) Alice has won an oformula $B \sqcup C$ iff, in the course of the play, she has decided between B and C in $B \sqcup C$ and has won the chosen oformula. That is, she has performed the action "choose the left formula $B \sqcup C$" and she has won B, or she has performed the action "choose the right formula in $B \sqcup C$" and she has won C. If she has not decided between B and C then she has lost $B \sqcup C$.

(6) For an oformula of the form $B \sqcap C$ the definition is similar (we swap Alice and Bob). Bob has won an oformula $A \sqcap B$ iff in the course of the play, he has decided between B and C and has won the chosen oformula.

(7) Every occurrence of $\mathbf{0}$ is won by Bob and every occurrence of $\mathbf{1}$ is won by Alice.

Finally, we say that Alice has won the play iff *for every* coin evaluation e she has won the entire formula A. The informal meaning: Alice has accomplished the obligation A whatever coins Bob has used.

There is an equivalent way to define the result of a play. Call ovariables U and V *matching* if Alice has allocated U to V. Define an *ovariable evaluation*, as any mapping from the set of *all* ovariables to $\{0, 1\}$. Call e *correct* if $e(U) = e(V)$ whenever U matches V. Alice has won a play if for every ovariable evaluation she has won the formula A according to the above rules (2)–(7). Indeed, the worst (for Alice) correct ovariable evaluation satisfies item (1) above anyway and thus can be identified by a coin evaluation.

In other words, to find who has won a play make the following steps. Replace by $\mathbf{0}$ all the oformulae of type $B \sqcup C$ such that Alice has not chosen B or C. Replace by $\mathbf{1}$ all the oformulae $B \sqcap C$ where Bob has not decided between B and C. Replace each remaining occurrence of a formula of the form $B \sqcup C$ or $B \sqcap C$ by the chosen subformula. (The order of replacements does not affect the result.) Then replace each ovariable by a new variable in such a way that matching ovariables are replaced by the same variables and non-matching ones by different ones. The resulting formula is an infinite formula of finite depth with connectives $\neg, \vee, \wedge$ and constants $\mathbf{1}, \mathbf{0}$. Alice has won the play iff this formula is a classical tautology (where $\wedge$ is understood as AND, $\vee$ as OR, and $\neg x$ as the negation of x). The above described transformation is what is called *elementarization* in [5, 6, 14].

Here is an example of play.

1. Initial position: $((\neg x \wedge x) \sqcup y) \vee ((x \vee \neg x) \sqcap \neg y)$
2. Bob chooses the left formula in $(\neg x \wedge x) \sqcup y$. The resulting position is $(\neg x \wedge x) \vee ((x \vee \neg x) \sqcap \neg y)$ (we have deleted the unchosen formula).
3. Alice chooses the left formula in $(\neg x \wedge x) \sqcup y$. The resulting position is $(\neg x \wedge x) \vee (x \vee \neg x)$ (we have deleted the unchosen formula).
4. Alice allocates the first negative occurrence of x to the second positive occurrence of x. The resulting position is $(\neg x_1 \wedge x) \vee (x_1 \vee \neg x)$ (we have used subscripts to identify allocated ovariables).

5. Alice allocates the second negative occurrence of x to the first positive occurrence of x. The resulting position is $(\neg x_1 \wedge x_2) \vee (x_1 \vee \neg x_2)$

6. Then the players make infinitely many passes.

Alice has won, as after elementarization we obtain a classical tautology $(\neg x_1 \wedge x_2) \vee (x_1 \vee \neg x_2)$

The definition of the game $[A]$ is completed. We call a formula A *accomplishable*[1] if Alice has a winning strategy in the game $[A]$. We call a formula A *computably accomplishable* if Alice has a computable winning strategy in the game $[A]$. The simplest accomplishable formula is $x \vee \neg x$: in order to win Alice just allocates the second x to the first x.

If A has no exponential connectives then the game $[A]$ is essentially finite (every player can perform only finitely many actions) and thus the game $[A]$ is accomplishable iff it is computably accomplishable. In this case at least one of the players has a (computable) winning strategy. In the general case it is unknown whether accomplishability is equivalent to computable accomplishability.

Remark. It is important that Alice cannot distinguish fake and genuine coins. (Formally, that means that we choose a coin evaluation after the play is finished.) For example, the formula $A = (\neg x \wedge \neg x) \vee x$ is not accomplishable. In the game $[A]$ Alice has essentially two strategies: (1) she uses the first of Bob's coins (allocates the first x to the third x) (2) she uses the second one. Both strategies do not win. Indeed, if only one coin is genuine, namely, the coin that has not been allocated, Alice has lost the formula A.

The definition of an accomplishable formula is very robust: we can change the definition of the game in many ways so that the class of (computably) accomplishable formulae does not change. Below we present several such modifications.

1. We may forbid Alice (or Bob, or both) to perform several actions in one move. Indeed, postponing actions never hurts the player. By the same reason it does not matter who starts the play.[2]

2. We may ban all actions inside oformulae B and C belonging to an oformula $B \sqcup C$ (respectively, $B \sqcap C$) such that a choice action has not yet been applied to $B \sqcup C$ (respectively, $B \sqcap C$).

3. We may assume that Bob must decide for every negative ovariable at the start of the game, whether it is genuine or false, but Alice does not know that decision. In this case the game $[A]$ becomes a game of imperfect information. This change decreases Bob's power and there are formulae A for which Bob has a winning strategy in the game $[A]$ before the change and has no winning strategy in $[A]$ after the change. This happens, say, for the formula $(\neg x \wedge \neg x) \vee x$. The notion of accomplishability however does not change, as it is defined through Alice's strategies and not Bob's ones.

[1] We use here the terminology of [5].

[2] The games having that property are called static, see e.g. [6]. We will consider other static games in the next section.

4. We can define the game $[A]$ recursively. To this end we need a notion of a "game with coins" and operations on such games that correspond to connectives.

Games of the form $[A]$ can be regarded as vending machines: negative ovariables can be identified with slots for coins and positive ovariables with compartments for releasing the products. For instance, a vending machine that accepts 1 Euro coin and 50 cents coins and sells coffee and tea, for 1 Euro each, can be represented by the formula

$$!(\neg(1\text{ Euro} \sqcup (50\text{ cents} \wedge 50\text{ cents})) \vee (\text{tea} \sqcap \text{coffee})).$$

Informally, the game $[A]$ is accomplishable iff the vending machine can work without having any resources in advance.

Remark. The calculus CL2 found in [8] provides a sound and complete axiomatisation for the set of all accomplishable formulae that have no exponential connectives. It is unknown whether the entire set of accomplishable formulae is computably enumerable. The similar question is open for computably accomplishable formulae, too.

1.3 Accomplishable formulae and affine logic

Girard's affine logic [4] is a useful tool to prove accomplishability of many formulae. Consider the following its variant. A *sequent* is a finite list of formulae. A sequent consisting of formulae $A_1, \ldots, A_n$ is denoted by $\vdash A_1, \ldots, A_n$. The order of formulae in a sequent does not matter, but the multiplicity of each formula matters. That is, $\vdash p, p$ and $\vdash p$ are different sequents. A sequent $\vdash A$ consisting of a single formula A is identified with the formula A.

Axioms: $\vdash A, \neg A$, $\quad \vdash \mathbf{1}$.

Derivation rules:

$\frac{\vdash \Gamma, A \quad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}$ (Cut). $\qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, (A \wedge B)}$ (Introducing $\wedge$).

$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, (A \vee B)}$ (Introducing $\vee$). $\qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, (A \sqcap B)}$ (Introducing $\sqcap$).

$\frac{\vdash \Gamma, A}{\vdash \Gamma, (A \sqcup B)}$, $\quad \frac{\vdash \Gamma, B}{\vdash \Gamma, (A \sqcup B)}$ (Introducing $\sqcup$). $\qquad \frac{\vdash \Gamma}{\vdash \Gamma, A}$ (Weakening).

$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?\, A}$ (Dereliction). $\qquad \frac{\vdash \Gamma, ?\, A, ?\, A}{\vdash \Gamma, ?\, A}$ (Contraction). $\qquad \frac{\vdash ?\, \Gamma, A}{\vdash ?\, \Gamma, !\, A}$ (R).

Here $?\,\Gamma$ denotes the list obtained from Γ by prefixing by $?$ all formulae in the list.

Call the formula $A_1 \vee \ldots \vee A_n$ the *formula image* of the sequent $\vdash A_1, \ldots, A_n$. If the list is empty (that is, $n = 0$), its formula image is equal to $\mathbf{0}$. A sequent is called *(computably) accomplishable* if so is its formula image.

Theorem 1. *The set of accomplishable sequents contains all axioms of the affine logic and is closed under all its derivation rules. The same applies to computable accomplishable sequents. (Hence all derivable formulae are computably accomplishable.)*

Proof. Axioms: the second axiom is accomplishable by definition. We need to show that the first one is accomplishable as well. To this end we have to construct Alice's computable winning strategy in the game $[A \vee \neg A]$. Alice can win the game $[A \vee \neg A]$ by merely mimicking Bob's moves. More specifically, there is a natural one-to-one correspondence between ovariables of A and $\neg A$. We call corresponding ovariables *dual*. If U is a positive ovariable then its dual is negative, and vice versa. Alice allocates every negative ovariable in $A \vee \neg A$ to its dual ovariable. This is done in such a way that at the end of the game all the allocations have been made (say one allocation per move). Every occurrence of $\sqcup$ (respectively, $\sqcap$) in $A \vee \neg A$ is naturally assigned the "dual" occurrence of $\sqcap$ (respectively, $\sqcup$) in $A \vee \neg A$. If Bob makes a choice in an oformula $B \sqcap C$ then Alice (say, on the next move) makes the same choice in the dual oformula $\neg B \sqcup \neg C$. This strategy guarantees that for every ovariable evaluation Alice wins exactly one of the formulae $A, \neg A$ and hence the entire formula $A \vee \neg A$.

Cut: Assume that both formulae $A \vee B$ and $\neg A \vee C$ are accomplishable. Fix Alice's strategies S, T that win the games $[A \vee B]$ and $[\neg A \vee C]$, respectively. We will define Alice's strategy winning the game $[B \vee C]$.

Roughly speaking, Alice plays the games $[B \vee A]$ and $[\neg A \vee C]$ simultaneously, using strategies S and T. The moves made by S, T inside B, C are made in the real game $[B \vee C]$, and the moves inside A and $\neg A$ are made on the imaginary play-board A, where the strategies S, T play against each other. That is, the strategy S considers the moves made by T in the game $[\neg A]$, as Bob's moves in the game A, and conversely the strategy T considers the moves made by S in the game $[A]$, as Bob's moves in the game $[\neg A]$.

Now we will describe Alice's strategy more accurately. It is easier to do this assuming different rules of the game. It is not hard to see that the notion of accomplishability does not change, if we change the rules of the game as follows. Bob is allowed in his turn to make an action "deposit a coin in U" where U is a positive ovariable, and Alice is allowed to make an allocation $U \to V$ only after Bob has deposited a coin in U (that is, U is "non-empty"). Bob defines $e(U)$ immediately, when he deposits a coin in U (but Alice does not know $e(U)$). We will say that Bob has deposited a genuine coin in U if $e(U) = 1$. If Bob has never deposited a coin into U then $e(U) = 0$. The result of the play is defined as earlier using the coin evaluation defined in this way. In this proof we will use these rules. We will imagine that Bob makes a move "deposit a coin in U" by putting a real coin onto U and Alice allocates U to V by moving this coin to V. The position in the play is specified by the choices made by the players and locations of the coins, where for each coin sitting on a positive ovariable its previous location is specified.

Alice's strategy. Alice splits the current position $b \vee c$ in the game $[B \vee C]$ into positions b and c. She also keeps an imaginary position a in the game A. In her turn, she first applies the strategy S to the position $b \vee a$ in the game $[B \vee A]$ and obtains a new position $b' \vee a'$. She then applies the strategy T to the position $\neg a' \vee c$ and obtains a position $\neg a'' \vee c'$. Here $\neg a'$ represents the position in the game $\neg A$ obtained from a' by changing all connectives to their duals, changing p into $\neg p$ and $\neg p$ into p for all oliterals. The resulting position

in the game $[B \vee C]$ is $b' \vee c'$ and the resulting imaginary position is a''.

This description has an important drawback. What happens when S moves a coin from b to a, that is, from a negative ovariable U in B to a positive ovariable V in A, or in the other direction, from a negative ovariable V in A to a positive ovariable U in B? In the first case Alice moves the coin into the "temporary store" (later she will probably decide on which ovariable to put it). The locations in the temporary store correspond to ovariables in A. In the second case Alice takes the coin from the location U in the temporary store and puts it on V. Alice acts similarly when T moves coins from C to $\neg A$ or in the other direction. When S (respectively, T) moves a coin within A (respectively, $\neg A$), Alice moves that coin within the temporary store.

Thus Alice uses the "temporary store" while playing $[B \vee C]$. It is clear that this does not increase her power. Indeed, instead of putting a coin into the temporary store she can wait until the coin will be taken back from the store. If the coin is never taken back, the coin is not moved at all from its original location.

We have to prove that this strategy wins. Let $b \vee c$ be the final position in the game $[B \vee C]$ and a the final position in the imaginary play. As S is a winning strategy, Alice has won b or a. In the first case we are done. Otherwise the strategy S has won the imaginary play, hence the strategy T has lost it. As T is a winning strategy, it has won c.

Introducing $\wedge$*:* We (that is, Alice) are given strategies winning the games $[A \vee C]$ and $[B \vee D]$ and have to design a strategy winning the game $[A \vee B \vee (C \wedge D)]$. We just apply the strategies independently. If the first strategy has won the formula A or the second strategy has won B, we are done. Otherwise the first strategy has won C and the second strategy has won D hence we have won the formula $C \wedge D$.

Introducing $\vee$: for this rule the formula image of the upper sequent coincides with the formula image of the lower sequent. Thus we have nothing to prove.

Introducing $\sqcap$*:* We (Alice) are given strategies to win the games $[A \vee B]$ and $[A \vee C]$. We have to design a strategy to win the game $[A \vee (B \sqcap C)]$. We do not do anything until Bob makes a choice in the formula $B \sqcap C$. If that never happens, we have won. Otherwise we apply the first strategy, if Bob has chosen B and we apply the second strategy otherwise.

The first rule of Introducing $\sqcup$*:* We (Alice) are given strategy that wins $[A \vee B]$. To win $[A \vee (B \sqcup C)]$ we first choose B and then apply the given strategy. For the second rule the arguments are similar.

Weakening, Dereliction and Contraction: straightforward.

R: We (Alice) are given a strategy S that wins the game $[? A_1 \vee \ldots \vee ? A_n \vee B]$. Consider the game $[!(? A_1 \vee \ldots \vee ? A_n \vee B)]$. We can win this game by applying the strategy S' that consists of countable number of independent copies of S. A slight modification S'' of S' wins the game $[? A_1 \vee \ldots \vee ? A_n \vee ! B]$. Let us define S''. For each i, both formulas $!(? A_1 \vee \ldots \vee ? A_n \vee B)$ and $? A_1 \vee \ldots \vee ? A_n \vee ! B$ have countably many occurrences of A_i. Put them into a one-to-one correspondence, and let $(A_i^j)'$ denote the occurrence that corresponds to A_i^j. Put also occurrences

of B into a one-to-one correspondence. If S' makes a move that involves A_i^j or B^j then S'' makes a similar move using $(A_i^j)'$ or $(B^j)'$ instead. Let us show that S'' wins. Run S'' and S' in parallel against the same adversary (that is, we mimic in the game $[!(?\,A_1 \vee \ldots \vee ?\,A_n \vee B)]$ all the moves made by a real adversary playing the game $[?\,A_1 \vee \ldots \vee ?\,A_n \vee !\,B]$). Then for every j, i, the strategy S' has won A_i^j if and only if S'' has won $(A_i^j)'$, and the same applies to B^j. If S' has won B^j for all j, then S'' has won $(B^j)'$ for all j and hence has won the entire formula. Otherwise, S' has won at least one of A_i^j (even infinitely many of them), thus S'' has won $?\,A_1 \vee \ldots \vee ?\,A_n$. □

The affine logic is not complete with respect to the abstract resource semantics. An example of an accomplishable non-provable formula is

$$[(\neg a \vee \neg b) \wedge (\neg c \vee \neg d)] \vee [(a \vee c) \wedge (b \vee d)].$$

(This formula was used by Blass in [2] to show that the affine logic is incomplete with respect to his semantics.) We refer to [16], where the reader can find a complete game semantics for the linear logic and references to other game semantics that are better tailored for the affine or linear logic.

We conclude this section by showing that the set of accomplishable formulae is closed under the substitution. We will need this later.

Theorem 2. *Assume that a formula A' is obtained from an accomplishable formula A by substituting a formula B for all occurrences of a variable p. Then A' is accomplishable.*

Proof. Fix Alice's winning strategy S in the game $[A]$. We will play $[A']$ (in Alice's part) as follows. While playing $[A']$ we simultaneously play an imaginary game $[A]$, where we apply the strategy S. We copy to the real game all the choices and allocations made by S in the imaginary game except allocations involving the variable p. We also copy to the imaginary game $[A]$ all Bob's choices made in the real game $[A']$ outside occurrences of B. If in the imaginary play the strategy S allocates an occurrence U of p to an occurrence V of p, we declare the corresponding occurrences of B symmetrical. All pairs of symmetrical occurrences are *synchronised*: we mimic all Bob's choices made inside one of these occurrences in the other one. Notice that U is positive and V is negative, therefore we are able to do that. Besides, we allocate every negative ovariable in either of these occurrences to the dual positive ovariable in the other occurrence.

Let us prove that this strategy wins. Let e' be an ovariable evaluation in the game $[A']$. Drop the values of e' on all the ovariables from the occurrences of B. The resulting mapping e evaluates all the ovariables of A except for p. Extend e to occurrences of p as follows: $e(U) = 1$ iff, in the play $[A']$, Alice[3] has won the oformula U' (the occurrence of B obtained by substituting B for U). The defined evaluation respects matchings. Indeed, if U has been allocated to V then we have synchronised U' and V' and hence the results of U' and V' coincide. It is easy to show by induction that, for this pair of evaluations, the

[3]more precisely, the player who has played U' in Alice's part

result of the imaginary game coincides with the result of the real play. Since S is a winning strategy, we have won A and hence A'. □

1.4 Accomplishable formulae and the intuitionistic propositional calculus IPC

Consider the Girard's translation from the language of propositional formulae with connectives $\wedge, \vee, \to, \bot$ into the language of affine logic. Each formula A is assigned a formula A^* defined recursively: $(x_i)^* = x_i$ and

$$(A \vee B)^* = !\,A^* \sqcup !\,B^*, \quad (A \wedge B)^* = A^* \sqcap B^*,$$
$$(A \to B)^* = \neg(!\,A^*) \vee B^* = ?(\neg A^*) \vee B^*, \quad \bot^* = \mathbf{0}.$$

This translation preserves provability. The next lemma shows that the combination of this translation and abstract resource semantics yields a sound semantics for the intuitionistic calculus. Call an intuitionistic formula *(computably) accomplishable* if its translation is (computably) accomplishable.

Lemma 3. *The set L of accomplishable intuitionistic formulae is a superintuitionistic logic (that is, L contains all axioms of IPC and is closed under Modus Ponens and substitution). The same holds for the set of all computably accomplishable intuitionistic formulae.*

Proof. 1. The translations of all axioms of IPC are derivable in the affine logic [4] and all derivable formulae are accomplishable by Theorem 1. Hence L contains all axioms of IPC.

2. Let A and B be intuitionistic formulae such that A^* and $(A \to B)^*$ are accomplishable. We have to prove that B^* is accomplishable, too. As A^* is accomplishable, the formula $!\,A^*$ is accomplishable as well. The formula B^* can be obtained from $\neg !\,A^* \vee B^*$ and $!\,A^*$ by applying Cut. Hence B^* is accomplishable by Theorem 1.

3. L is closed under substitution by Theorem 2. □

Remark. If we defined the translation of $A \vee B$ as just $A^* \sqcup B^*$ then the translation of the axiom

$$(x \to z) \to ((y \to z) \to (x \vee y \to z))$$

would not be accomplishable.

It turns out that this semantics of IPC is complete for the positive fragment of IPC.

Theorem 4. *If A is a positive intuitionistic formula (that is, A does not contain $\bot$) and A^* is accomplishable then A is derivable in IPC.*

One of the technical tools in the proof of this theorem is the following theorem, which was stated by Medvedev in [15] (without a complete proof) and

proved in [3]. A *critical implication* is a (positive) formula of intuitionistic language of the form $A_1 \wedge \cdots \wedge A_m \to R$, where R is an OR of variables and each A_i has the form $(P_i \to Q_i) \to Q_i$. Here every P_i is an AND of variables, every Q_i is an OR of variables, and, for all i, the formulae P_i and Q_i have disjoint sets of variables. The number of variables in R, P_i and Q_i is positive and may be equal to 1.

Theorem 5 ([15, 3])**.** *If a super-intuitionistic logic contains no critical implication then its positive fragment coincides with the positive fragment of IPC.*

Proof of Theorem 4. We have seen (Lemma 3) that the set of accomplishable intuitionistic formulae is a super-intuitionistic logic. Thus by Theorem 5 it suffices to prove that every critical implication is not accomplishable. Let us show this first for a very simple critical implication

$$A = ((x \to y) \to y) \wedge ((y \to x) \to x) \to x \vee y$$

whose translation is

$$A^* = ?[(!(?\,\neg x \vee y) \wedge \neg y) \sqcup (!(?\,\neg y \vee x) \wedge \neg x)] \vee (!\,x \sqcup !\,y)$$

From Bob's viewpoint the game $[A^*]$ is simpler than the game $[B]$, where B is obtained from A^* by dropping all exponential ANDs:

$$B = ?[((?\,\neg x \vee y) \wedge \neg y) \sqcup ((?\,\neg y \vee x) \wedge \neg x)] \vee (x \sqcup y).$$

We will present Bob's winning strategy in the game $[B]$. In this game he cannot make any actions, thus we need to prove that Bob wins the play whatever actions performs Alice. We distinguish two cases.

Case 1: Alice has not made any choice in the formula $x \sqcup y$. Then she has lost the entire formula, if all x's and all y's are true. Indeed, Bob has won the formula $x \sqcup y$. Moreover, he has won all oformulae $(?\,\neg x \vee y) \wedge \neg y$ and $(?\,\neg y \vee x) \wedge \neg x$, since he has won all occurrences of $\neg x$ and $\neg y$.

Case 2: Alice has made a choice in $x \sqcup y$. Assume that she has chosen x (the other case is similar). Then she has lost the entire formula, if all x's are false and all y's are true. Indeed, Bob has won the formula $x \sqcup y$ and all oformulae $(?\,\neg x \vee y) \wedge \neg y$, as he has won $\neg y$. Besides, Bob has won all oformulae $(?\,\neg y \vee x) \wedge \neg x$, as he has won $?\,\neg y \vee x$.

Consider now the general case. Let $A = A_1 \wedge \cdots \wedge A_m \to R$ be a critical implication where A_i has the form $A_i = (P_i \to Q_i) \to Q_i$ and $R = x_1 \vee \cdots \vee x_n$ where $n \geq 2$ (w.l.o.g. we may assume that R is an OR of all variables). The translation of A is $A^* = ?(\neg A_1^* \sqcup \ldots \sqcup \neg A_m^*) \vee R^*$ where

$$\neg A_i^* = !(?\,\neg P_i^* \vee Q_i^*) \wedge \neg Q_i^*.$$

and $R^* = !\,x_1 \sqcup \ldots \sqcup !\,x_n$. Drop the operation ! in all the formulae $!(?\,\neg P_i^* \vee Q_i^*)$ and in the formula $!\,x_1 \sqcup \ldots \sqcup !\,x_n$. The resulting formulae $?\,\neg P_i^* \vee Q_i^*$ and

$x_1 \sqcup \ldots \sqcup x_n$ are harder for Bob. Thus it suffices to construct Bob's winning strategy for the game $[B]$ where

$$B = ((?\,\neg P_1^* \vee Q_1^*) \wedge \neg Q_1^*) \vee \ldots \vee ((?\,\neg P_m^* \vee Q_m^*) \wedge \neg Q_m^*) \vee (x_1 \sqcup \ldots \sqcup x_n).$$

Recall that P_i is an AND of variables and Q_i is an OR of variables. Thus while playing $[B]$ Bob is allowed to make choices in oformulae $\neg Q_i^*$ for all i such that Q_i has at least two variables. Fix any such oformula $(?\,\neg P_i^* \vee Q_i^*) \wedge \neg Q_i^*$. Bob makes a choice in $\neg Q_i^*$ in either of the following two cases: (1) Alice has made a choice in the oformula Q_i^*. In this case, on his next move, Bob mimics Alice's choice in $\neg Q_i^*$ (unless he has already made the choice). (2) Alice has made a choice, say x_j, in the formula $x_1 \sqcup \ldots \sqcup x_n$. Then, on his next move, Bob chooses in $\neg Q_i^*$ any variable different from x_j (unless he has made the choice earlier).

We will show now that Bob wins. To this end define a correct ovariable evaluation as follows. If Alice has not made a choice in $x_1 \sqcup \ldots \sqcup x_n$ then declare all ovariables true. Otherwise let x_j be the chosen ovariable; declare all occurrences of x_j false and all other ovariables true.

We have to prove that Bob has won the entire formula B. Consider the first case: Alice has not made a choice in $x_1 \sqcup \ldots \sqcup x_n$ and all ovariables are true. In this case Bob has won the formula $x_1 \sqcup \ldots \sqcup x_n$. We have to prove that for all i he has won every occurrence of $(?\,\neg P_i^* \vee Q_i^*) \wedge \neg Q_i^*$. Fix i and an occurrence of this formula. Bob has won all occurrences of $\neg P_i^*$ thus he has won $?\,\neg P_i^*$. As Bob has not made any choice according to item (2), he has mimicked Alice's choice in Q_i^* (provided there was any). Therefore he has won exactly one of the oformulae Q_i^* and $\neg Q_i^*$ and hence the entire oformula $(?\,\neg P_i^* \vee Q_i^*) \wedge \neg Q_i^*$.

In the second case Alice has chosen x_j in $x_1 \sqcup \ldots \sqcup x_n$, all ovariables x_j are false, and all other ovariables are true. Again Bob has won $x_1 \sqcup \ldots \sqcup x_n$. We have to show that he has won all occurrences of $(?\,\neg P_i^* \vee Q_i^*) \wedge \neg Q_i^*$. Fix any of them and distinguish the following two cases. (1) Alice has chosen the variable x_j in an oformula $\neg P_i^*$. In this case she has won $?\,\neg P_i^*$ and we need to prove that Bob has won $\neg Q_i^*$. Bob has chosen some ovariable in $\neg Q_i^*$ using either option (1), or option (2). As P_i and Q_i have disjoint sets of variables, x_j does not occur in Q_i and thus he has chosen an ovariable different from x_j any way. Hence he has won $\neg Q_i^*$. (2) For every oformula $\neg P_i^*$ Alice has not chosen x_j in it. In this case Bob has won $?\,\neg P_i^*$. If he has won Q_i^*, we are done. Otherwise Q_i has a variable different from x_j and Alice has chosen such variable in Q_i^*. Bob either has mimicked Alice's choice, or has chosen in $\neg Q_i^*$ a variable different from x_j by himself. Anyway he has won $\neg Q_i^*$. □

Theorem 4 does not generalise to negative formulae. We know two examples of a formula A such that A^* is accomplishable but A is not provable in IPC: Rose formula from [18] and Japaridze's formula from [7]. The latter one is simple enough to present it here:

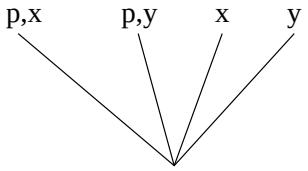
$$(\neg p \to x \vee y) \wedge (\neg\neg p \to x \vee y) \to (\neg p \to x) \vee (\neg p \to y) \vee (\neg\neg p \to x) \vee (\neg\neg p \to y). \tag{1}$$

Here $\neg B$ is an abbreviation for $B \to \bot$.

Theorem 6. *The formula* (1) *is accomplishable but not derivable in IPC.*

Proof. The counter Kripke model is shown on Fig. 1.

Figure 1: The counter model: nodes of the tree represent worlds labelled by variables that are valid in them.



It remains to prove that (1) is accomplishable. We will prove that a more general formula $A = B \to C$ where

$$B = (\neg p \to x \vee y) \wedge (\neg\neg p \to u \vee v),$$

$$C = (\neg p \to x) \vee (\neg p \to y) \vee (\neg\neg p \to u) \vee (\neg\neg p \to v),$$

is accomplishable.

The translation of A is equal to $A^* = ?\,\neg B^* \vee C^*$ where

$$\neg B^* = (!(\neg p)^* \wedge (\neg\,!\,x \sqcap \neg\,!\,y)) \sqcup (!(\neg\neg p)^* \wedge (\neg\,!\,u \sqcap \neg\,!\,v)).$$

We will construct Alice's winning strategy in the game $[?\,D \vee C^*]$ which is even harder for her, where

$$D = (!(\neg p)^* \wedge (\neg x \sqcap \neg y)) \sqcup (!(\neg\neg p)^* \wedge (\neg u \sqcap \neg v))$$

is obtained from $\neg B^*$ by dropping ! in front of x, y, u, v. In the course of the play we call an occurrence of $\neg x \sqcap \neg y$ in A^* *active* if it is located inside an occurrence of D such that Alice has chosen the left formula $!(\neg p)^* \wedge (\neg x \sqcap \neg y)$ in that occurrence. In a similar way we define active occurrences of $\neg u \sqcap \neg v$. Alice does not make any choice in the formula C^* until Bob has made a choice in some of active occurrences of $\neg x \sqcap \neg y$, $\neg u \sqcap \neg v$. This means that, until that moment, Alice and Bob are playing the game

$$[?((!(\neg p)^* \wedge \mathbf{1}) \sqcup (!(\neg\neg p)^* \wedge \mathbf{1})) \vee \mathbf{0}] = [((\neg p \to \bot) \wedge (\neg\neg p \to \bot) \to \bot)^*].$$

As the formula defining this game is derivable in IPC, Theorem 3 implies that Alice has a winning strategy in this game. Alice plays according to this strategy until Bob has made a choice in some of active occurrences of $\neg x \sqcap \neg y$, $\neg u \sqcap \neg v$.

Assume that Bob has made choice in an active occurrence of $\neg x \sqcap \neg y$ or an occurrence where he had made the choice earlier has become active (the case

of $\neg u \sqcap \neg v$ is similar). Once that has happened, Alice changes her behaviour. Assume that Bob has chosen $\neg x$ (the other case is similar). Alice then "forgets" all the other occurrences of $!(\neg p)^* \wedge (\neg x \sqcap \neg y)$ and all the occurrences of $!(\neg\neg p)^* \wedge (\neg u \sqcap \neg v)$ and chooses $(\neg p \rightarrow x)^*$ in C^*. That is, starting from that moment Alice and Bob play the game $[\neg(\neg p \rightarrow x)^* \vee (\neg p \rightarrow x)^*]$. The formula defining this game is accomplishable being an axiom of affine logic. Unfortunately Alice cannot immediately use the strategy winning this game, as some moves might have been already made inside occurrences of $\neg p$ in

$$\neg(\neg p \rightarrow x)^* = !(\neg p)^* \wedge \neg x = !\,?\,\neg p \wedge \neg x.$$

However it easy to overcome this problem. An infinite number of occurrences of $\neg p$ in every occurrence of $?\,\neg p$ have not been yet allocated by Alice. Therefore the current position in the game $\neg(\neg p \rightarrow x)^*$ is not worse for Alice than the initial position. □

2 The semantics of Japaridze's computability logic and accomplishability

2.1 Static games

We will use a rather general notion of games from [6, 14] between two players, called Environment and Machine.

A *move* is a string over the keyboard alphabet. A *labelled move (labmove)* is a move prefixed by `E` or `M` (the prefix indicates who has done the move, Environment or Machine). A *run* is a finite or infinite sequence of labmoves. A *position* is a finite run.

A *game*[4] is specified by a set of runs L and a function W mapping runs to the set $\{\mathtt{E}, \mathtt{M}\}$. All runs in L are called *legal*, all other runs are called *illegal*. If $W(\Gamma) = P$, we say that the run Γ is *won* by P. Otherwise it is *lost* by P.

The set L must have the following properties: (1) the empty sequence (called the *initial position*) belongs to L and (2) if a (finite or infinite) run Γ is in L then all its finite prefixes are in L too.

Let $\Gamma = \alpha_1, \alpha_2, \ldots$ be a run and α_i a labmove in that run. We say that α_i is the first *first illegal* labmove in Γ if $\alpha_1, \ldots, \alpha_{i-1}$ is legal run but $\alpha_1, \ldots, \alpha_i$ is not. Each illegal run Γ has exactly one first illegal labmove. The function W must have the following property: every illegal run is lost by that Player who has made the first illegal move in it.

We have not yet defined how to play a game. It is not obvious since in some positions both players can make a legal move and the rules do not specify who has the turn to play in such a position.

There are eight ways to play a game. We have to make three choices: who starts the game, how many moves (at most one or several) is Environment allowed to make in its turn, and how many moves (at most one or several) is

[4] called a *constant game* in [6]

Machine allowed to make in its turn. For example, the game can be played as follows: Environment starts the play; in its turn, each player either makes a move or passes. Another way: Machine starts the play; in its turn, Environment can make any finite sequence of moves (including the empty sequence); in its turn, Machine can make a move or pass. For all eight ways to play the game, we assume that the play lasts infinitely long and the turn to play alternates.

For certain games it is crucial which of the eight modes to play is chosen (for example, if $W(\Pi) = \mathtt{E}$ if Π starts with a move of $\mathtt{E}$ and $W(\Pi) = \mathtt{M}$ otherwise). There are however two important classes of games, *strict* games and more general *static* games, for which it does not matter.

A game is called *strict* if for every legal position Δ at most one player can make a move α so that the resulting position (Δ, α) is legal. Most games considered in the literature are strict ones. However, the operations on games we are going to define do not look natural when applied to strict games. They look natural when applied to more general static games defined in the next two paragraphs. Informally, static games are those games in which it never hurts the player to postpone moves.

Let Γ, Δ be (finite or infinite) runs. We say that Δ is a *Machine-delay* of Γ if Δ is obtained from Γ by postponing certain Machine's moves (may be infinitely many). Formally, the following conditions should hold. (1) Erasing all moves of Environment in Γ and Δ results in the same run; the same holds for erasing Machine's moves. (2) For all k and l if kth move of Machine is made later than lth move of Environment in Γ then so is in Δ.

We define a notion of *Environment-delay* in a similar way. We say that the game is *static* if the following holds for every player P, every run Γ and every P-delay Δ of Γ. (1) If P has not made the first illegal move in Γ then P has not made the first illegal move in Δ either. (2) If Γ is won by P then so is Δ.

We call a static game *winnable* if Machine has a winning strategy in the game. It does not matter which of the above eight ways to play the game to choose: the class of winnable static games is robust under switching between the eight playing modes. However, it is important that we do not force any player to make a move in its turn. We call a static game *computably winnable* if Machine has a computable winning strategy in the game (that is, there is a Turing machine that wins the game).

Now we will define operations $\neg, \wedge, \vee, \sqcup, \sqcap, \curlywedge, \curlyvee$ on games. Those operation will preserve static property. We will then call a formula with connectives $\neg, \wedge, \vee, \sqcup, \sqcap, \curlywedge, \curlyvee$ winnable if every substitution of static games for variables results in a winnable game. It happens that that a formula is winnable iff it is accomplishable.

2.2 Operations on games

The operation of *negation* $\neg$ just swaps the roles of players: Machine plays in Environment's part and vice versa. The set of legal runs of $\neg A$ is obtained from that of A by replacing each run Γ by its dual run $\neg\Gamma$, which is is obtained from Γ by exchanging labels $\mathtt{E}$ and $\mathtt{M}$ in all labmoves. Machine wins a run Γ in $\neg A$

iff the dual run $\neg\Gamma$ is won by Environment in A: $W_{\neg A}(\Gamma) = \neg W_A(\neg\Gamma)$ (where $\neg\mathtt{M} = \mathtt{E}$ and $\neg\mathtt{E} = \mathtt{M}$).

The *choice conjunction* applied to games A, B produces the following game $A \sqcap B$. First, Environment decides between A and B. Then the chosen game is played. If Environment has not decided, it loses. Formally a non-empty run is legal iff it starts with Environment's move "choose left" or "choose right" and the rest of the run is a legal run of A if the first move is "choose left" and is a legal run of B if the first move is "choose right". A legal run is won by Machine in the following three cases: (1) it is empty, (2) the first move is "choose left" and the rest of the run is won by Machine in the game A, and (3) the first move is "choose right" and the rest of the run is won by Machine in the game B.

The *choice disjunction* $A \sqcup B$ of A, B is dual to $A \sqcap B$. This time Machine has to decide between A and B (and it loses if it has not decided). In other words, $A \sqcup B = \neg(\neg A \sqcap \neg B)$.

Parallel disjunction $\vee$. In the game $A \vee B$ the players play two games A and B simultaneously. In order to win, Machine has to win at least one game. Formally, a run Γ is legal if the following holds. Let Γ^i denote the result of removing from Γ all the labmoves that do not have the form $Pi.\alpha$ (where $P = \mathtt{E}, \mathtt{M}$) and replacing the prefix $Pi.\alpha$ by $P\alpha$ in all the remaining labmoves. A run Γ is legal if every its labmove has the form $Pi.\alpha$ (where $i = 1, 2$) and the runs Γ^1, Γ^2 are legal runs of A, B, respectively. Such a run is won by Machine if either Γ^1 is won by Machine in A or Γ^2 is won by Machine in B (or both).

Parallel conjunction $\wedge$ is dual to parallel disjunction. The difference is that this time Machine has to win both games. In other words, $A \wedge B = \neg(\neg A \vee \neg B)$.

Parallel recurrence $\curlywedge$. The game $\curlywedge A$ is essentially an infinite parallel conjunction $A \wedge A \wedge A \wedge \ldots$. Players play simultaneously an infinite number of plays and in order to win Machine has to win all plays. Formally, a run is legal if all its labmoves have the form $Pi.\alpha$ where $i = 1, 2, 3, \ldots$ (we spell natural numbers in decimal notation, say) and all $\Gamma^1, \Gamma^2, \Gamma^3, \ldots$ are legal runs of A. Such a run is won by Machine if all Γ^i are won by Machine in the game A.

Parallel corecurrence $\curlyvee$ is dual to parallel recurrence $\curlywedge$. Again players play infinitely many plays in the game A. However, this time in order to win, Machine has to win at least one play. That is, $\curlyvee A = \neg(\curlywedge \neg A)$.

All these operations preserve the static property. However not all of them preserve strictness property. Consider, for example, the multiplicative conjunction. Assume that the games A and B are strict, the first move in A is made by Environment and the first move in B is made by Machine. Then the first move in $A \wedge B$ can be made by either player. Nevertheless all eight ways to play $A \wedge B$ are equivalent (for all strict games A, B). So we could fix any of the eight ways to play and thus convert $A \wedge B$ into an equivalent strict game. However such stipulating would be quite unnatural (bureaucratic, using Japaridze's word). As a result, we would not have the property $A \wedge B = \neg(\neg A \vee \neg B)$. The games $A \wedge B$ and $\neg(\neg A \vee \neg B)$ would be only equivalent in a sense.

Another reason to prefer static games is that all the computational problems can be quite naturally expressed as static games and not as strict games (see [14] for many examples). That is why Japaridze has chosen static games as a basis

for his Game semantics.

2.3 Winnable = accomplishable

There is a similarity between operations on static games and abstract resource semantics. Consider, for instance, a formula $A \sqcup B$ and a static game $G \sqcup H$. Machine has won $G \sqcup H$ if it has made a choice between G and H and has won the chosen game. Similarly, Alice has won $A \sqcup B$ if she has chosen between A and B and has won the chosen formula. The same holds for all connectives. The following lemma expresses this similarity in a rigorous form.

Let $A(x_1, \ldots, x_n)$ be a formula and $G_1, \ldots, G_n$ static games. Substituting G_i for x_i in A and performing all operations spelled in A, we obtain a static game $A(G_1, \ldots, G_n)$. Exponential AND and OR are interpreted as $\curlywedge, \curlyvee$ respectively. Let R be a run in the game $[A]$ and S a run in the game $A(G_1, \ldots, G_n)$. Let $P_1.(\text{choose } B_1)$, $P_2.(\text{choose } B_2)$, ... be the sequence of all the choice moves made in R. Call R and S *similar* if the sequence of all the choice moves in S is equal to $P_1'.(\text{choose } B_1(G_1, \ldots, G_n))$, $P_2'.(\text{choose } B_2(G_1, \ldots, G_n))$, ... where $B_i(G_1, \ldots, G_n)$ is the sub-game of $A(G_1, \ldots, G_n)$ corresponding to the oformula B_i and Alice$'$ = Machine, Bob$'$ = Environment. The following lemma is straightforward.

Lemma 7. *Let R and S be similar runs. Consider the following ovariable evaluation: an occurrence of p_i is true iff Machine has won the corresponding occurrence of G_i.*[5] *Then Machine has won S iff Alice has won R for this evaluation.*

We say that a formula A is *winnable*, if for all static games $G_1, \ldots, G_n$ the resulting game $A(G_1, \ldots, G_n)$ is winnable. We say that A is *computably winnable*, if for all static games $G_1, \ldots, G_n$ the game $A(G_1, \ldots, G_n)$ is computably winnable.

Consider also the uniform version of winnability. Call a formula A is *uniformly* (computably) winnable, if there is a (computable) strategy winning the game $A(G_1, \ldots, G_n)$ for all static games $G_1, \ldots, G_n$.

Every static game is equivalent to a strict game. Thus, in the definition of winnability we may restrict ourselves to strict games. On the other hand, if we consider only determined games[6] we obtain a weaker notion. Why? The class of determined games is closed under all operations considered. And who wins the game is determined just classically: Machine wins (i.e., it has a winning strategy in) $A \wedge B$ iff it wins A and wins B, Machine wins $A \vee B$ if it wins A or wins B etc. Thus, if we restrict the class of static games to determined ones, a formula is winnable iff after dropping exponentials and identifying multiplicative and additive connectives it becomes a classical tautology.

For uniform winnability and computable winnability this is not the case. For example, it might be that Machine has a computable winning strategy in

[5] More precisely, the player who has played that copy of G_i in Machine's part has won it.

[6] The game is called *determined* in either Machine, or Environment has a winning strategy in the game.

the game $A \vee B$ but does not have computable winning strategy in either A or B. Therefore for these versions of winnability it seems quite natural to restrict the class of games to determined ones. However, it turns out that this restriction does not affect the classes of uniformly winnable and uniformly computably winnable formulae. We do not know whether this is true for computably winnable formulae.

Theorem 8. *The following three properties of a formula A are equivalent:*
(1) A is computably accomplishable,
(2) A is uniformly computably winnable,
(3) there is computable strategy that wins every game $A(G_1, \ldots, G_n)$ where $G_1, \ldots, G_n$ are 2-move[7] strict games.

Proof. [8] It is straightforward that (2) $\Rightarrow$ (3).

Proof of (1) $\Rightarrow$ (2). Assume that Alice has a computable winning strategy S in the game $[A]$. We have to show that Machine has a computable winning strategy that wins every game of the form $A(G_1, \ldots, G_n)$.

For every sub-formula B of A consider the "sub-game" $B(G_1, \ldots, G_n)$ of $A(G_1, \ldots, G_n)$. If B has the form $C \sqcup D$ or $C \sqcap D$ then every choice between C and D corresponds to a choice move between $C(G_1, \ldots, G_n)$ and $D(G_1, \ldots, G_n)$ in the game $A(G_1, \ldots, G_n)$. Thus we have 1-1 correspondence between choice moves in the game $[A]$ and choice moves in the game $A(G_1, \ldots, G_n)$. To every occurrence p_k^j of the variable p_k in A corresponds a sub-game G_k of $A(G_1, \ldots, G_n)$, which will be called G_k^j.

We will now construct Machine's winning strategy. While playing $A(G_1, \ldots, G_n)$ Machine plays in its mind an imaginary game $[A]$, where it applies the strategy S for Alice and makes certain moves for Bob. More specifically, all the choices made by Environment in the game $A(G_1, \ldots, G_n)$ are mimicked immediately in the game $[A]$, as Bob's moves, and, conversely, all choices that S has made in $[A]$ are mimicked immediately in $A(G_1, \ldots, G_n)$. Thus, after every Machine's move, all the choices made in the real and imaginary games are identical.

Besides, if S on a certain move has allocated $\neg p_k^j$ to p_k^m, Machine starts from that time the "synchronisation" of the games G_k^j and G_k^m. The latter means that it mimics in G_k^j all Environment's moves made in G_k^m, and vice versa. This is indeed possible, as Machine plays G_k^j in Environment's part. Thus, starting from the time of that allocation, after every Machine's move, the current position Γ_k^m in G_k^m is a Machine-delay of the current position Γ_k^j in G_k^j.[9] Hence, when the game is played, the run Γ_k^m is a Machine-delay of the run Γ_k^j.

[7] We call a game a k-move game, if every legal run has at most k labmoves.

[8] Morally, this proof is similar to soundness/completeness proof for system CL2 in [9]. Although the syntax of CL2 does not include parallel recurrences, those are long $\vee, \wedge$ after all.

[9] Formally, Γ_k^m and Γ_k^j are defined as follows. It is easy to extract from the current position in the game $A(G_1, \ldots, G_n)$ the sequence of labmoves "made in G_k^m". That sequence is Γ_k^m. To obtain Γ_k^i we first extract from the current position the sequence of labmoves Δ made in $\neg G_k^j$ and then let $\Gamma_k^i = \neg\Delta$.

We have to prove that the constructed Machine's strategy S' wins the game $A(G_1, \ldots, G_n)$. Fix Environment's moves against S'. We obtain a run Γ' played against S' and the corresponding imaginary run Γ played against S. As S is a winning strategy, the imaginary run Γ is won by Alice. Therefore for every correct ovariable evaluation Alice has won the formula A.

Let us choose the following ovariable evaluation. Let Γ_i^l stand for the run in the sub-game G_i^l. If the run Γ_i^l is Machine won (that is, $W_{G_i}(\Gamma_i^l) = \mathtt{M}$) then declare p_i^l true, and otherwise false. We need to show that this evaluation is correct. Assume that a negative ovariable p_k^j has been allocated to a positive ovariable p_k^m, and thus the games G_k^j and G_k^m were synchronised. If $W_{G_k}(\Gamma_k^j) = \mathtt{M}$ then, as G_k is a static game, $W_{G_k}(\Gamma_k^m) = \mathtt{M}$, as well. It may happen however that $W_{G_k}(\Gamma_k^j) = \mathtt{E}$ and $W_{G_k}(\Gamma_k^m) = \mathtt{M}$ (that is, Machine has won both games $\neg G_k^j$ and G_k^m). In this case we have declared p_k^j false and p_k^m true and thus the ovariable evaluation is incorrect. But the incorrectness is in favour of Alice (as all connectives are monotone) and hence Alice has won A for this ovariable evaluation. By Lemma 7 Machine has won the run Γ', and we are done.

We will show now that, conversely, the negation of (1) implies the negation of (3). Let us fix a formula $A(p_1, \ldots, p_n)$ which is not computably accomplishable and a computable Machine's strategy S. We have to define strict 2-move games $G_1, \ldots, G_n$ such that S loses the game $A(G_1, \ldots, G_n)$. The first move x in all $G_1, \ldots, G_n$ must be made by Machine and the second move y must be made by Environment. Machine wins G_i iff either it has made the first move but Environment has not made the second move, or both moves x, y has been played and the pair (x, y) belongs to a certain set W_i. Thus we have to define $W_1, \ldots, W_n$.

To every copy p_k^j of p_k in A corresponds a game G_k inside $A(G_1, \ldots, G_n)$, which will be called G_k^j. We will fix now how Environment plays the games G_k^j against S. If p_k^j is negative then Environment has to make the first move in G_k^j and otherwise the second move. Anyway, once it should move it plays j in G_k^j (we identify strings over the keyboard alphabet and natural numbers). If there are infinitely many negative occurrences of p_k, then Environment makes the moves in them in some order so that by the end of the play all these moves have been made (say at time j it makes the move in $\neg G_k^j$).

Call a pair of sub-games G_k^j, G_k^m a *collision*, if for some x, y, in both games, the first move is x and the second move is y. Environment's strategy guarantees that in this case p_k^j is negative and p_k^m is positive and $x = j$, $y = m$ (or vice versa). In particular, for every j there is at most one m such that G_k^j, G_k^m is a collision and, conversely, for every m there is at most one such j.

Now we have to define Environment's choices in the game $A(G_1, \ldots, G_n)$ and the sets $W_1, \ldots, W_k$. To this end consider the following Alice's strategy S' to play the game $[A]$. The strategy keeps an imaginary position in the game $A(G_1, \ldots, G_n)$ (at the start it keeps the initial position). In her turn Alice: (a) mimics in the imaginary play $A(G_1, \ldots, G_n)$ all choices made by Bob on the last move in $[A]$, (b) makes moves in copies of $G_1, \ldots, G_n$, as described above, (c)

applies Machine's strategy S to the resulting position (items (a)–(c) are done in Alice's mind, they do not affect immediately the real game); (d) mimics in $[A]$ all choices made by S in $A(G_1, \ldots, G_n)$, (e) makes allocation as follows: Alice allocates p^j_k to p^m_k, if the pair G^j_k, G^m_k has become a collision. As we have noticed, these allocations obey the rules of the game.

This strategy S' is computable. Therefore, Bob can play so that Alice has lost the game $[A]$. Fix such Bob's moves and make these moves against S' in the imaginary play. Mimicking them in the real play, we will obtain Environment's moves in $A(G_1, \ldots, G_n)$.

It remains to define $W_1, \ldots, W_n$. Fix a correct ovariable evaluation e such that Alice has lost the formula $[A]$. Let Γ^j_k be the sequence of moves played in G^j_k. Let $W_k(\Gamma^j_k) = \mathtt{M}$ iff p^j_k is true. Environment's strategy guarantees the following. If $\Gamma^j_k = \Gamma^m_k$ (for $j \neq m$) then G^j_k, G^m_k is a collision. Thus Alice has allocated p^j_k to p^m_k and hence $e(p^j_k) = e(p^m_k)$. This shows that $W_1, \ldots, W_n$ are well defined. By Lemma 7 Machine has won $A(G_1, \ldots, G_n)$ iff Alice has won A for this evaluation, and we are done. □

In [6] Japaridze conjectures that computable winnability coincides with uniform computable winnability (Conjecture 26.1). In the preliminary version of this paper [17] we claimed, without a proof, that this conjecture is true. We have to admit that we have no valid proof for this. However, if we drop the computability requirement, then uniform winnability coincides with the non-uniform one.

Theorem 9. *The following four properties of a formula A are equivalent:*
(1) A is accomplishable.
(2) A is uniformly winnable.
(3) A is winnable.
(4) There is a Machine's strategy winning every game of the form $A(G_1, \ldots, G_n)$, where $G_1, \ldots, G_n$ are 2-move strict games.

Proof. It is straightforward that (2) ⇒ (3) and (2) ⇒ (4). To prove the implication (1) ⇒ (2) recall the following. In the proof of the implication (1) ⇒ (2) from Theorem 8, we transformed every Alice's strategy S that wins $[A]$ to a strategy S' that wins $A(G_1, \ldots, G_n)$ for all static games $G_1, \ldots, G_n$. (That transformation preserves computability, which does not matter any more.)

The proof of ¬(1) ⇒ ¬(4). In the proof of the implication ¬(1) ⇒ ¬(3) in Theorem 8 we have constructed a transformation $S \mapsto S'$ from Machine's strategies in the game $A(G_1, \ldots, G_n)$ to Alice's strategies in $[A]$ that has the following property. If S' does not win A then there are 2-move strict games $G_1, \ldots, G_n$ such that S does not win $A(G_1, \ldots, G_n)$. This shows ¬(1) ⇒ ¬(4).

The proof of the implication ¬(1) ⇒ ¬(3) is similar to the proof of ¬(1) ⇒ ¬(3) in Theorem 8. However, this time we have to beat uncountably many strategies (and not one strategy) and we need indetermined games $G_1, \ldots, G_n$.

Assume that A is not accomplishable. We will define strict games $G_1, \ldots, G_n$ so that Machine has no winning strategy in the game $A(G_1, \ldots, G_n)$. The turn

of move in the games $G_1, \ldots, G_n$ will alternate, starting with Environment, say, and all moves made in turn will be legal. If the run is finite then it is lost by that player who has turn to move. Thus we have to define the value of winning functions $W_1, \ldots, W_n$ on infinite runs.

We will use diagonal arguments. There are continuum strategies in static games. Indeed, a strategy is a mapping from the set of all positions into to the set of all moves joint with "pass" (we assume the following playing mode: in its turn, each player either makes a move or passes). The set of all positions is countable, as a countable union of countable sets (for every fixed k, the set of all positions consisting of k moves is countable, as a product of countable sets). Finally, there are continuum mappings from $\mathbb{N}$ to $\mathbb{N}$, as

$$|\mathbb{N}^{\mathbb{N}}| \leq (2^{\mathbb{N}})^{\mathbb{N}} = 2^{|\mathbb{N}\times\mathbb{N}|} = 2^{|\mathbb{N}|} = \mathfrak{c}.$$

Let us fix a 1-1 mapping from the ordinal $\mathfrak{c}$ to the set of all strategies. The strategy assigned to $\alpha \in \mathfrak{c}$ will be called S_α. We will use transfinite recursion on ordinals in $\mathfrak{c}$. On step α we will define Environment's moves against S_α and we will define $W_1, \ldots, W_n$ on countably many runs. Thus, before step α, $W_1, \ldots, W_n$ will be defined on less than $\mathfrak{c}$ runs.

Step α. As $W_1, \ldots, W_n$ have been defined so far on less than $\mathfrak{c}$ runs, there is a sequence of moves $a_1, a_2, \ldots$ that has the following property. For all $i \leq n$, W_i has not been defined on *any* run of the form

$$*, *, \mathtt{E}.a_1, *, \mathtt{E}.a_2, *, \mathtt{E}.a_3, *, \ldots$$

and on *any* run of the form

$$*, *, *, \mathtt{M}.a_1, *, \mathtt{M}.a_2, *, \mathtt{M}.a_3, \ldots.$$

We fix now Environment's moves in the game $A(G_1, \ldots, G_n)$. In the copy G_k^j of G_k Environment plays j and then $a_1, a_2, \ldots$. This applies for both negative and positive occurrences of p_k. By the choice of $a_1, a_2, \ldots$, whatever moves Machine makes in G_k^j, the resulting run Γ_k^j either is finite (and hence lost by Machine), or it is infinite and W_k has not been defined on Γ_k^j on previous steps. So we are free to define the value of W_k on every infinite resulting run. The only constraint is that there might be collisions: a collision is a pair G_k^j, G_k^m such that the runs Γ_k^j, Γ_k^m coincide. This may happen only if p_k^j is negative and p_k^m is positive (or vice versa). In this case both Γ_k^j, Γ_k^m start with j, m (or m, j) and Environment can see when it happens after Machine has made its first moves in both G_k^j, G_k^m.

The rest of the proof is similar to the proof of $\neg(1)\Rightarrow\neg(3)$ in Theorem 8. We first define Alice's strategy S'_α in $[A]$. In her turn Alice: (a) mimics in the imaginary play $A(G_1, \ldots, G_n)$ all choices made by Bob on the last move in $[A]$, (b) makes moves in copies of $G_1, \ldots, G_n$, as described above, (c) applies Machine's strategy S_α to the resulting position; (d) mimics in $[A]$ all choices made by S_α in $A(G_1, \ldots, G_n)$, (e) allocates p_k^j to p_k^m, if the first two moves in G_k^j, G_k^m have been made and they coincide.

As S'_α does not win $[A]$, there are Bob's moves such that S'_α has lost $[A]$. Fix such moves. Mimicking them in the game $A(G_1, \ldots, G_n)$ we obtain Environment choice moves against S_α. Thus all Environment's moves against S_α are fixed. As the formula A has been lost by Alice, we can define $W_1, \ldots, W_n$ on all resulting runs so that Machine has lost $A(G_1, \ldots, G_n)$. Notice that we have defined $W_1, \ldots, W_n$ on at most countably many runs. □

A sequent is called *(uniformly) (computably) winnable* if so is its formula image. Theorems 8 and 9 show that winnability is a sound semantics for affine logic.

Theorem 10. *The set of winnable sequents contains all axioms of the affine logic and is closed under all its derivation rules and under the substitution. The same applies to uniformly winnable sequents, computably winnable sequents, and uniformly computably winnable sequents. (Hence all derivable formulae are uniformly computably winnable.)*

Proof. For winnability the statement is true, since winnability coincides with accomplishability. The same applies to uniform winnability and uniform computable winnability. For computable winnability this is proved similar to Theorems 1 and 2. □

Remark. The soundness of affine logic with respect to computable uniform winnability (and hence all other sorts of winnabilities) was shown in [14].

This theorem together with Theorem 4 and Lemma 3 show that winnability is a sound game theoretic semantics for intuitionistic propositional calculus, which is complete for its positive fragment.

Corollary 11. *Let A be a formula in the intutionistic language. If A is provable in IPC then the formula A^* is uniformly computable winnable. Conversely, if A is positive and non-provable in IPC then A^* is not winnable and A^* is not uniformly winnable for 2-move strict games (corollary of Theorem 9).*

What about non-uniform computable winnability for k-move and determined games? The second item of Corollary 11 does not imply that there are *determined* games $G_1, \ldots, G_n$ such that the game $A^*(G_1, \ldots, G_n)$ is not computably winnable.

Theorem 12. *If a positive formula A is not provable in IPC then there are determined static games $G_1, \ldots, G_n$ such that the game $A^*(G_1, \ldots, G_n)$ is not computably winnable.*

Proof. Let L stand for the set of all formulae such that for all determined static games $G_1, \ldots, G_n$ the game $A^*(G_1, \ldots, G_n)$ is computably winnable. All operations on static games used in Girard's translation preserve determinacy. Thus L is a super-intuitionistic logic (the proof is similar to the proof of Lemma 3). By Theorem 5, it suffices to prove that L contains no critical implications.

Let A be a critical implication. We have to find determined static games $G_1, \ldots, G_n$ such that $A(G_1, \ldots, G_n)$ is not computably winnable. All the games

$G_1, \ldots, G_n$ will have the form "produce a binary sequence f", where $f : \mathbb{N} \to \{0,1\}$. More specifically, we will define sequences $f_1, \ldots, f_n$ and G_i will be the following strict game: a run is legal in G_i if it consists only of Machine's moves, a run $\mathtt{M}x_1, \mathtt{M}x_2, \ldots$ is won by Machine (that is, $W_i(x_1, x_2, \ldots) = \mathtt{M}$) if it is infinite and $x_t = f_i(t)$ for all $t = 0, 1, 2, \ldots$. Obviously, whatever functions $f_1, \ldots, f_n$ we choose later, the games $G_1, \ldots, G_n$ will be determined (won by Machine).

It is not hard to prove that there are functions $f_1, \ldots, f_n$ such that every f_i is not Turing reducible to the tuple consisting of $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n$. That is, there is no Turing machine that computes f_i given an oracle that is able to compute all functions $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n$. Fix such tuple of functions $f_1, \ldots, f_n$.

The rest of the proof is similar to the proof of Theorem 4. That is, Environment applies Bob's strategy, as defined there. Besides it plays in the copies of games $G_1, \ldots, G_n$ as follows. In every sub-game of the form $\neg G_i$ (where Environment has to make moves) it plays $f_i(0), f_i(1), \ldots$. It plays $\neg G_1, \ldots, \neg G_n$ in this way until Machine has made a choice in $G_1 \sqcup \ldots \sqcup G_n$. Starting from that time, it stops making any moves in all the sub-games $\neg G_j$, where G_j is the game chosen by Machine (and keeps playing in G_i for $i \neq j$).

We will show now that Environment wins. Assume first that Machine has not made a choice in the game $G_1 \sqcup \ldots \sqcup G_n$. Then $W_1, \ldots, W_n$ evaluate to $\mathtt{M}$ on the runs in all copies of $G_1, \ldots, G_n$ that correspond to negative ocurrences of $p_1, \ldots, p_n$. Even if $W_1, \ldots, W_n$ evaluate to $\mathtt{M}$ on all runs corresponding to positive ocurrences of $p_1, \ldots, p_n$, Environment wins. Indeed, by Lemma 7 Machine has won the game $A(G_1, \ldots, G_n)$ iff Alice has won the formula A for the similar run in the game $[A]$ and for the evaluation that makes true all ovariables. As shown in the proof of Theorem 4, Alice has lost the similar run in the game $[A]$ for this ovariable evaluation.

Assume that Machine has made a choice in the game $G_1 \sqcup \ldots \sqcup G_n$ and has chosen, say, G_j. Again $W_1, \ldots, W_n$ evaluate to $\mathtt{M}$ on the runs in all copies of $G_1, \ldots, G_{j-1}, G_{j+1}, \ldots, G_n$ that correspond to negative ocurrences of $p_1, \ldots, p_n$. Let us prove that W_j evaluates to $\mathtt{E}$ on all the runs in copies of G_j corresponding to positive copies of p_j. Indeed, Environment has ceased to provide any information about f_j. Thus Machine's playing G_j can be viewed as a computation of f_j with oracles for $f_1, \ldots, f_{j-1}, f_{j+1}, \ldots, f_n$. As any such computation makes an error, Machine has lost every positive copy of G_j. Again by Lemma 7 and arguments similar to those from Theorem 4, Machine has lost the game $A(G_1, \ldots, G_n)$. □

2.4 Countable branching recurrence

There is another way to interpret exponential connectives as operations on games, called *branching recurrence* and *corecurrence* in [14]. There are two versions of them, countable and uncountable. Countable branching recurrence and corecurrence were essentially introduced by Blass in [1]. In the present form, they were defined in [13].

The countable branching recurrence $\mathbin{⫰}^{\aleph_0}$ is the operation on games defined as follows. In the game $\mathbin{⫰}^{\aleph_0}A$ players play countably many plays in the game A and Machine has to win all plays (like in the game $\mathbin{⅄}A$). However this time its job is even more difficult, as Environment may "copy" positions. Informally, we can imagine that a position in the game $\mathbin{⫰}^{\aleph_0}A$ is a tuple $\langle p_1, \ldots, p_n\rangle$ of positions of A. The initial position is the tuple $\langle p_1\rangle$ where p_1 is the initial position in A. If the current position is $\langle p_1, \ldots, p_n\rangle$, then each player is allowed to make a legal move in any of the positions $p_1, \ldots, p_n$. Environment is also allowed to copy any p_i, in which case the new position is equal to $\langle p_1, \ldots, p_n, p_i\rangle$.

This definition is very informal because we have defined moves in terms of positions and not the other way around, as our framework prescribes.

Moreover, the described game may be not static.[10] To obtain an equivalent static game we need to understand a copying operation as splitting one position (the parent) into two new positions (the children). If a move made in a position P at time t is postponed to time $t' > t$ and by the time t' the position P has been split, then that move is automatically played in all descendants of P. More specifically, we assign to each position an address, which is a binary string rather than a natural number. When a position with address w is split, the children positions get addresses $w0$ and $w1$. When a play is finished we obtain a finite or infinite tree consisting of all addresses used. All leaves in that tree are addresses of the played games. If the tree is infinite then some infinite paths are also addresses of played games: those infinite paths which have finite number of 1s. We will call such infinite binary sequences *bounded*.

Formally, a run Γ is legal if it is a sequence of labmoves of the form $Pw.\alpha$ and $\mathtt{E}$(split w) having the following two properties. (1) For every w and every proper prefix u of w every ocurrence of a labmove of the form $Pw.\alpha$ or $\mathtt{E}$(split w) in Γ is preceded by exactly one ocurrence of the labmove $\mathtt{E}$(split u). (2) For every finite or infinite binary sequence w consider the sequence $\Gamma(w)$ of all labmoves in Γ of the form $Pu.\alpha$—with "u." removed—where u is a prefix of w. For each finite or infinite w the run $\Gamma(w)$ must be a legal run of A. A legal run Γ is won by Machine if $\Gamma(w)$ is won by Machine for all leaves w and all bounded infinite branches w.

In the definition of *uncountable branching recurrence* $⫰$, we stipulate that Γ is won by Machine if $\Gamma(w)$ is won for all leaves w and for all infinite branches w (and not only bounded). Thus the difference between countable and uncountable versions is due to different understandings which plays in G have been played in the course of a play in $⫰G$.

Countable branching corecurrence $⫯^{\aleph_0}$ is defined in the dual way so that $⫯^{\aleph_0}A = \neg(⫰^{\aleph_0}\neg A)$. Let us change the interpretation of ! and ? to $⫰^{\aleph_0}$ and $⫯^{\aleph_0}$, respectively. We obtain new notions of winnability and computable winnability, which does not coincide with the old ones.

[10] Indeed, assume that the initial position is lost by Machine and every run of length 1 is won by Machine. Then the position $\mathtt{M1}.\alpha, \mathtt{E}.$(copy position 1) is won by Machine but the position $\mathtt{E}.$(copy position 1), $\mathtt{M1}.\alpha$ is lost.

Lemma 13. *The formula*

$$?(\neg x \sqcap \neg y) \vee (!\, x \sqcup !\, y),$$

is (1) not winnable if exponentials are interpreted as parallel recurrence and corecurrence but is (2) uniformly computably winnable provided exponentials are interpreted as countable branching recurrence and corecurrence.

Proof. (1) As winnability for parallel recurrence and corecurrence coincides with accomplishability, it suffices to prove that the formula is not accomplishable. Here is Bob's winning strategy. Bob starts choosing $\neg x$ in all occurrences of $\neg x \sqcap \neg y$ (one choice per move). He goes on until Alice has made a choice in $!\, x \sqcup !\, y$. If Alice has chosen $!\, y$, then she will lose the game, as Bob can declare all y's false and all x's true. Otherwise (if Alice has chosen $!\, x$), Bob stops choosing $\neg x$ in remaining oformulae $\neg x \sqcap \neg y$ and starts choosing $\neg y$ instead. Again Bob has won, as he can declare true all ovariables he has chosen and declare all others false.

(2) Let G, H be any static games. Here is Machine's computable winning strategy in the game

$$?(\neg G \sqcap \neg H) \vee (!\, G \sqcup !\, H),$$

Machine waits until Environment makes a choice between $\neg G$ and $\neg H$ in the initial position of $\neg G \sqcap \neg H$. W.l.o.g. assume that it has chosen $\neg G$. Then Machine chooses $!\, G$ in $!\, G \sqcup !\, H$. Thus the game has basically become $?\, \neg G \vee !\, G$ starting in its initial position. Indeed, when Machine splits a position in the game $?(\neg G \sqcap \neg H)$ in two positions, the choice made by Bob automatically applies to both children-positions. As the formula $?\, \neg x \vee !\, x$ is derivable in the affine logic, Machine has a computable winning strategy in the game $?\, \neg G \vee !\, G$. □

Theorem 14. *The set of winnable sequents (exponentials are interpreted as countable branching recurrence and corecurrence) contains all axioms of the affine logic and is closed under all its derivation rules and under the substitution. The same applies to uniformly winnable sequents, computably winnable sequents, and uniformly computably winnable sequents. (Hence all derivable formulae are uniformly computably winnable.)*

Proof. The proof is similar to that of Theorem 1. □

Theorems 8 and 9 remain valid for the new notions of winnability:

Theorem 15. *The following two properties of a formula A are equivalent (if exponentials are interpreted as countable branching recurrence and corecurrence):*
(1) A is uniformly computably winnable.
(2) There is a computable strategy winning every game $A(G_1, \ldots, G_n)$ where $G_1, \ldots, G_n$ are 2-move strict games.

Theorem 16. *The following three properties of a formula A are equivalent (if exponentials are interpreted as countable branching recurrence and corecurrence):*

(1) A is uniformly winnable.
(2) A is winnable.
(3) There is a Machine's strategy winning every game of the form $A(G_1, \ldots, G_n)$, where $G_1, \ldots, G_n$ are 2-move strict games.

Proof of Theorem 15. Obviously (1) implies (2).

One can prove the reverse implication along the same lines as in Theorem 8: we can change the definition of the game $[A]$ so that both (1) and (2) are equivalent to existence of a computable winning strategy in $[A]$. However the new notion of accomplishability has no independent interest so we will prove the implication (2)⇒(1) directly.

Assume that (1) is false and fix a computable Machine's strategy S in $A(G_1, \ldots, G_n)$. We have to define 2-move strict games $G_1, \ldots, G_n$ such that S loses $A(G_1, \ldots, G_n)$. The first move in all $G_1, \ldots, G_n$ is played by Machine and the second one by Environment. Let Environment play j in G_k^j once it has turn to move in G_k^j. This ensures that for every j there is at most one m such that (G_k^j, G_k^m) is a collision (the runs in G_k^j and G_k^m coincide). We need to define choice moves of Environment and the winning conditions in the games $G_1, \ldots, G_n$ so that Environment wins.

To this end consider another strategy S' to play $A(H_1, \ldots, H_n)$. The strategy S' plays the real game $A(H_1, \ldots, H_n)$ simultaneously with an imaginary game $A(G_1, \ldots, G_n)$. In its turn S': (a) mimics in the imaginary play $A(G_1, \ldots, G_n)$ all choices and copy moves made by Environment on the last move in $A(H_1, \ldots, H_n)$, (b) makes moves in copies of $G_1, \ldots, G_n$, as described above, (c) applies S to the resulting position; (d) mimics in $A(H_1, \ldots, H_n)$ all choices and copy moves made by S in $A(G_1, \ldots, G_n)$, (e) starts synchronising H_k^j and H_k^m, if G_k^m, G_k^j has become a collision.

As S' is computable, there are static games $H_1, \ldots, H_n$ and Environment's moves in $A(H_1, \ldots, H_n)$ such that S' has lost $A(H_1, \ldots, H_n)$. Mimicking choices and copy moves in $A(G_1, \ldots, G_n)$ we obtain Environment's choices and copy moves in $A(G_1, \ldots, G_n)$. The winning condition in G_k is defined as follows: Machine has won G_k^j iff Machine has won H_k^j. The game G_k is well defined. Indeed, if the run in G_k^j and G_k^m coincide then H_k^j and H_k^m were synchronized and hence the runs in H_k^j and H_k^m coincide as well. □

Proof of Theorem 16. It is straightforward that (1)⇒(2) and (1)⇒(3). The implication (3)⇒(1) was basically proved in the proof of Theorem 15. Indeed, we have defined a transformation of every strategy S to a strategy S' and a transformation of every tuple of games $H_1, \ldots, H_n$ to 2-move strict games $G_1, \ldots, G_n$ such that the following holds. If S' does not win $A(H_1, \ldots, H_n)$ then S does not win $A(G_1, \ldots, G_n)$.

The proof of implication $\neg(1) \Rightarrow \neg(2)$ is entirely similar to the proof of $\neg(1) \Rightarrow \neg(3)$ in Theorem 9 and therefore we omit it. □

Corollary 11 remains true for countable branching recurrence and corecurrence and, moreover, in the translation of $A \vee B$ we may omit !. More specifically,

let $(x_i)^\dagger = x_i$ and

$$(A \vee B)^\dagger = A^\dagger \sqcup B^\dagger, \quad (A \wedge B)^\dagger = A^\dagger \sqcap B^\dagger,$$
$$(A \to B)^\dagger = \neg(!\, A^\dagger) \vee B^\dagger = ?(\neg A^\dagger) \vee B^\dagger, \quad \bot^\dagger = \mathbf{0}.$$

Theorem 17. *The set of all formulae A such that $A^\dagger$ is winnable is a superintuitionistic logic (if exponentials are interpreted as countable branching recurrence and corecurrence). The same holds for computable winnability, and uniform versions of winnability and computable winnability. On the other hand, if a positive formula is not provable in IPC then the formula $A^\dagger$ is not winnable and there are determined static games such that the game $A^\dagger(G_1, \dots, G_n)$ is not computably winnable.*

Proof. It is easy to verify that translations of all axioms of IPC are uniformly computably winnable, except for the axiom

$$A = (x \to z) \to ((y \to z) \to (x \vee y \to z)).$$

The translation of the latter is equal to

$$A^\dagger = ?(!\, x \wedge \neg z) \vee ?(!\, y \wedge \neg z) \vee ?(\neg x \sqcap \neg y) \vee z.$$

The formula $A^\dagger$ is not derivable in the affine logic, as it is not accomplishable. However it is uniformly computably winnable (if exponentials are interpreted as countable branching recurrence and corecurrence). Indeed, by Lemma 13 the formula

$$B = ?(\neg x \sqcap \neg y) \vee (!\, x \sqcup !\, y)$$

is uniformly computably winnable. It is easy to verify that the sequent $\vdash \neg B \vee A^\dagger$ is derivable in the affine logic. Hence it is uniformly computably winnable. The sequent $\vdash A^\dagger$ follows from $\vdash \neg B \vee A^\dagger$ and $\vdash B$ by Cut, and hence is uniformly computably winnable as well.

Thus translations of all axioms of IPC are uniformly computably winnable. The closure under Substitution and Modus ponens is proved similar to Theorem 1. The second part of the theorem is proved similar to Theorems 4 and 12. □

Theorem 17 is true for the Girard's translation as well, and the proof is similar.

2.5 Historical and terminological remarks

The notions of a computably winnable and uniformly computably winnable formula were defined in [6] under the name a *(uniformly) valid* formula. The uncomputable versions of these were also considered in [6], as a property of games rather than a property of formulae. The notion of a winnable formula (only for countable branching recurrence) was first considered by Blass in [2]. Although Blass has considered only strict games and his definition of a countable branching recurrence differs in some technical details from the above definition, the class of winnable formulae is the same.

2.6 Uncountable branching recurrence

Theorem 17 is true for uncountable branching recurrence as well, which was shown in [11]. Our proof of Theorem 17 (based on Theorem 5) also works for uncountable branching recurrence and corecurrence and provides a shorter proof than that of [11].

Acknowledgements

We are sincerely grateful to Giorgi Japaridze for explaining many subtle things about game operations, pointing to the relevant papers and many useful comments on previous versions of the paper. We are grateful to Alexander Shen and other participants of Kolmogorov seminar at Moscow State University for bringing our interest to game semantics of the intuitionistic logic. Many thanks to anonymous referees for useful comments.

References

[1] A. Blass. Degrees of indeterminacy of games. Fundamenta Mathematicae 77 (1972) 151–166.

[2] A. Blass. A game semantics for linear logic. Annals of Pure and Applied Logic 56 (1992) 183–220.

[3] A. V. Chernov, D.P. Skvortsov, E.Z. Skvortsova, N.K. Vereshchagin. Variants of Realisability for Propositional Formulas and the Logic of the Weak Law of Excluded Middle. In: Proc. of Computer Science Logic'02, LNCS, vol. 2471, Springer 2002, pp. 74–88.

[4] G.Y. Girard. Linear logic. Theoretical Computer Science 50 (1) (1987) 1–102.

[5] G. Japaridze. The logic of tasks. Annals of Pure and Applied Logic 117 (2002) 263–295.

[6] G. Japaridze, Introduction to computability logic. Annals of Pure and Applied Logic 123 (2003) 1–99.

[7] Giorgi Japaridze, A letter to N.V. of 04.11.2005.

[8] G. Japaridze. Introduction to Cirquent Calculus and Abstract Resource Semantics. Journal of Logic and Computation 16(4) (2006) 489–532

[9] G. Japaridze. Propositional computability logic II. ACM Transactions on Computability Logic 7 (2006) 331–362.

[10] G. Japaridze. Intuitionistic computability logic. Acta Cybernetica 18(1) (2007) 77–113.

[11] G. Japaridze. The intuitionistic fragment of computability logic at the propositional level. Annals of Pure and Applied Logic 147(3) (2007) 187–227.

[12] G. Japaridze. Sequential operations in computability logic. Information and Computation 206 (2008) 1143–1475.

[13] G. Japaridze. Many concepts and two logics of algorithmic reduction. Studia Logica 91 (2009) 1–24.

[14] G. Japaridze. In the beginning was game semantics. In: Games: Unifying logic, Language and Philosophy. O. Majer, A.-V. Pietarinen and T. Tulenheimo, eds. Springer 2009, pp. 249–350.

[15] Yu. T. Medvedev. Finite problems. Doklady Akad. Nauk SSSR, 3 (1962) 227–230.

[16] P.-A. Melliés. Asynchronous games 4: A fully complete model of propositional linear logic. Proc. Twentieth Annual IEEE Symposium on Logic in Computer Science, June 26th-29th, 2005, Chicago, USA, pp. 386-395.

[17] I. Mezhirov, N. Vereshchagin. On Game Semantics of the Affine and Intuitionistic Logics. In: W. Hodges, R.J.G.B. de Queiroz, eds., Proc. 15th International Workshop on Logic, Language, Information and Computation (WoLLIC 2008), Edinburgh, UK, July 1-4, 2008. LNAI, vol. 5110, Springer 2008, pp. 28–42.

[18] G. F. Rose. Propositional calculus and realizability. Transactions of the American Mathematical Society, 75(1) (1953) 1–19.