A game semantics for proof search: Preliminary results [1]

Dale Miller and Alexis Saurin [2]

INRIA-Futurs and École Polytechnique
Palaiseau, France

Abstract

We describe an ongoing project in which we attempt to describe a neutral approach to proof and refutation. In particular, we present a language of *neutral expressions* which contains one element for each de Morgan pair of connectives in (linear) logic. Our goal is then to describe, in a neutral fashion, what it means to prove or refute. For this, we use games where moves are described as transitions between positions built with neutral expressions. In some settings, we can then relate winning a game with provability or with validity.

Key words: proof theory, game semantics, neutral approach to proof and refutation.

1 Introduction

Connections between games and logic are rather old and numerous [13]. Dialog games support the intuitive ideal that if one has a proof of a proposition, one can always win against an opponent attempting to attack that proposition [6,16]. In the domain of linear logic, various categories of games have been designed to give an abstract meaning to proofs and to proof normalization [1,4,10,14]. We will be interested in linear logic here as well but from the *computation-as-proof-search* perspective [19]. As is often the case, the goals and semantics of proof search and proof normalization are quite different. Loddo studied in his PhD thesis [15] connections between game theory and Prolog computation (as well as CLP computation) while Pym and Ritter studied recently [21] connections between games theory and proof search, but our approach differs significantly from both of these works and comparisons are difficult to draw.

[1] This paper is an improved and corrected version of a paper presented at the workshop GaLoP'05. We are grateful for the support of the ACI grants GEOCAL and Rossignol.
[2] Email: `dale [at] lix.polytechnique.fr` and `saurin [at] lix.polytechnique.fr`

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

We attempt to use games to provide a neutral approach to proof and refutation. Consider, for example, that we are given a (Horn clause) logic program $\mathcal{P}$, and a query G. If $\mathcal{P}$ is a *noetherian* program, we expect an attempt to prove G in Prolog to return either a *finite success* or a *finite failure*. In the first case, we have a proof of G from $\mathcal{P}$ and in the second case we have a proof of $\neg G$ from (the completion of) $\mathcal{P}$. This simple observation is an interesting challenge to the basic premise of proof search. That is, proof search tells us to establish first what we plan to prove, namely, either G or $\neg G$, and then to set about to prove that. Our (idealized) Prolog interpreter, however, does one computation and concludes afterwords with either a proof of G or a proof of $\neg G$. This suggests that there might be a neutral approach to understanding proof search more generally.

We formalize this neutral approach to proof and refutation with Horn clauses in Sections 2 and 3 by introducing a language of *neutral expressions* which contains one constructor for each de Morgan pair of connectives in (linear) logic. Horn clauses are flat and represent only one "phase" in a computation: they support no alternation in polarities. In Section 4, we extend the language of neutral expressions to include a "switch" operator that specifies that its argument expression is to be processed by the *other player* allowing the mixing of polarities and identify the class of *simple expressions* for which game playing has a simpler and more manageable structure. In Section 5, we present examples of simple expressions and their games and relate winning strategies to proofs within a propositional setting. Section 6 shows that Hintikka's games for determining truth are captured by purely additive games. Sections 7 and 8 discuss possible extensions to games and other future work.

2 Horn clauses and Prolog revisited

A Horn clause (in the *iff-form*) for the predicate p is a first-order formula

$$\forall x_1 \ldots \forall x_n [p(x_1, \ldots, x_n) \equiv G]$$

where $n \geq 0$, p is an n-ary predicate symbol, and the clause's body, G, is a *Horn clause goal formula* whose free variables are in $\{x_1, \ldots, x_n\}$ and which is built following grammar

$$G ::= \top \mid \bot \mid A \mid G \wedge G \mid G \vee G \mid \exists x\, G.$$

The syntactic variable A denotes *atomic formulas*: that is, a formula with a predicate (a non-logical constant) as its head: the formulas $\bot$ and $\top$ are not atomic formulas since their top-level symbol will be defined using logic.

A Horn program is a finite set $\mathcal{P}$ of Horn clauses all for distinct predicates. If we define $p \prec q$ for two predicates when q appears in the body of the Horn clause for p, then $\mathcal{P}$ is *noetherian* if the transitive closure of $\prec$ is acyclic. Notice that when $\mathcal{P}$ is noetherian, it can be rewritten to a logically equivalent logic

program $\mathcal{P}'$ for which the relation $\prec$ is empty: that is, there are no atomic formulas in the body of clauses in $\mathcal{P}'$. In noetherian programs, atoms are not necessary.

One approach to the foundations of logic programming is based on *proof search* in which computation is modeled by the search for a cut-free (and goal-directed) proof of a given sequent [19]. As search progresses, the sequents for which proofs are attempted change and this change represents the dynamics of the computation modeled. Such a view of logic programming has been been productive and has allowed the development of a number of logic programming languages based on logics other than first-order classical logic.

The following observation appears, however, to be an interesting challenge to this approach to logic programming. Consider a simple "logic engine" that will attempt to prove a given goal G from a given Horn clause program, and assume that this engine is also complete for noetherian programs (many Prolog systems are examples of such engines). Thus, when given the goal G and a noetherian program $\mathcal{P}$, such an engine will return either `yes` or `no`. The proof search interpretation of these responses is clear: `yes` means that a proof of G has been found from $\mathcal{P}$ and `no` means that no such proof exists. But another interpretation of `no` is that a proof of $\neg G$ has been found from $\mathcal{P}$. This is the basic idea of the "negation by failure" approach in logic programming which has been formalized in proof theory thanks to the artifact of definitions [11,18,8]. Thus, we can say that Prolog has either proved G or refuted G.

Thus the challenge to proof search is this: as just described, Prolog did one computation, evidentially not committing to proving or refuting, and that computation provided implicitly the proof of G or the refutation of G (that is, a proof of $\neg G$). Proof search, however, assumes instead that one fixes at the beginning a sequent to be proved and that the failed attempt to prove, say, $\longrightarrow G$ is not a proof of $G \longrightarrow \bot$. A natural question then is to develop this theme of neutral computation behind proof search and to see if we can generalize it beyond the (too) simple setting of first-order Horn clauses.

3 Neutral expressions

Given the motivation above, when we start a search with, say, the goal $G_1 \wedge G_2$, we do not know if we will eventually be proving a conjunction or its de Morgan dual, the disjunction $\neg G_1 \vee \neg G_2$. Similarly, if we start the search with the existential quantifier $\exists x.G$, we do not know if we will eventually be proving an existential or a universal $(\forall x \neg G)$. Thus, we introduce a new language of neutral expressions on which to conduct search.

Neutral expressions are given using the following syntax.

$$N ::= \mathbb{0} \mid \mathbb{1} \mid \dot{p}\, t_1 \ldots t_n \mid N + N \mid N \times N \mid \mathbf{Q}x.N$$

Here, $\mathbb{0}$ and $\mathbb{1}$ are the units of $+$ and $\times$, respectively. The variable x in the expression $\mathbf{Q}x.N$ is bound in the scope of N: usual rules for bound variables in syntax are assumed. When we translate neutral expressions back to logic, the neutral expression constructor $\dot{p}$ of arity n will correspond to the predicate p of the same arity.

A *first-order model* $\mathcal{M}$ is defined in the usual way: we write $|\mathcal{M}|$ for the domain of quantification of the model and we shall assume that for every $c \in |\mathcal{M}|$ there is a *parameter* $\bar{c}$ in the language of the logic. Function symbols will be interpreted in the model is the obvious way: the function symbol f is interpreted as the function $f^{\mathcal{M}}$ such that the term $f(t_1, \ldots, t_n)$ is interpreted as $f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}})$. An atomic formula $p(t_1, \ldots, t_n)$ is true if the n-tuple $\langle t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}} \rangle$ is a member of the n-ary relation that $\mathcal{M}$ provides for the predicate p.

A model of particular interest for us is the Herbrand model for signature Σ: this is a model $\mathcal{H}_\Sigma$ such that $|\mathcal{H}_\Sigma|$ is the set of closed terms built from Σ and in which the sole predicate that is interpreted is equality: $\mathcal{H}_\Sigma \models t = s$ if and only if t and s are identical closed terms. In such a model, there is no need to distinguish between c and $\bar{c}$.

Multisets of neutral expressions can be rewritten nondeterministically as follows. Given a model $\mathcal{M}$, the binary relation $\mapsto$ between finite multisets of neutral expressions is given as follows:

$$\mathbb{1}, \Gamma \mapsto \Gamma \qquad N \times M, \Gamma \mapsto N, M, \Gamma$$
$$N + M, \Gamma \mapsto N, \Gamma \qquad \mathbf{Q}x.N, \Gamma \mapsto N[\bar{c}/x], \Gamma, \text{ where } c \in |\mathcal{M}|$$
$$N + M, \Gamma \mapsto M, \Gamma \qquad \dot{p}(t_1, \ldots, t_n), \Gamma \mapsto \Gamma, \text{ where } \mathcal{M} \models p(t_1, \ldots, t_n)$$

Let $\mapsto^*$ be the reflective and transitive closure of $\mapsto$. If we consider the size of a multiset of neutral expressions to be the total number of occurrences of constructors in expressions in that multiset, then the size of multisets decreases as they are rewritten. As a result, rewriting always terminates. Rewriting can also be infinitely branching given the rule for $\mathbf{Q}$ and the assumption that there are infinitely many closed terms t.

We shall be interested in whether or not an expression N (considered as a singleton multiset) rewrites (via $\mapsto^*$) to $\{\}$. If $\mathbb{0} \in \Gamma$ or if $\dot{p}(t_1, \ldots, t_n) \in \Gamma$ where $p(t_1, \ldots, t_n)$ is false in $\mathcal{M}$ then Γ cannot reduce to $\{\}$.

In Figure 1, the positive and negative translations of neutral expressions into logic are provided. Notice that the only logical connectives in the range of $[\cdot]^+$ are positive (synchronous), while those in the range of $[\cdot]^-$ are negative (asynchronous), using the terminology of, say, [3,9].

In order to state a formal connection between the multiset rewriting above and provability, we introduce a proof system for first-order MALL (multiplicative, additive, linear logic) with equality. This proof system is the standard one-sided presentation [7] for the propositional connectives and for the quantifiers ($\forall$ introduction, for example, employs eigenvariables in the usual way). In order to deal with equality, however, we use the following inference rules,

N	$[N]^+$	$[N]^-$
$\mathbb{0}$	0	$\top$
$\mathbb{1}$	1	$\bot$
$\dot{p}(t_1,\ldots,t_n)$	$p(t_1,\ldots,t_n)$	$\neg p(t_1,\ldots,t_n)$
$N_1 + N_2$	$[N_1]^+ \oplus [N_2]^+$	$[N_1]^- \,\&\, [N_2]^-$
$N_1 \times N_2$	$[N_1]^+ \otimes [N_2]^+$	$[N_1]^- \parr [N_2]^-$
$\mathbf{Q}x.N$	$\exists x.[N]^+$	$\forall x.[N]^-$

Fig. 1. Translations of neutral expressions.

which have been developed in a number of recent papers [5,8,17,23]:

$$\frac{}{\vdash t = t} \qquad \frac{\vdash \Delta\theta}{\vdash \neg(t = s), \Delta}\ \dagger \qquad \frac{}{\vdash \neg(t = s), \Delta}\ \ddagger$$

The proviso † requires that t and s are unifiable and θ is their most general unifier ($\Delta\theta$ is the multiset resulting from applying θ to all formulas in Δ). The proviso ‡ requires that t and s are not unifiable. The free variables of a sequent are also called *eigenvariables*.

Notice that the equality rule with success of unification is treated multiplicatively while failed unification is treated additively (third rule). Notice moreover that each case of application of a rule corresponds to a unit of linear logic: the rule for the success of equality corresponds to 1 while the rule for the success of inequality corresponds to $\bot$. On the other hand, the rule for the failure of inequality corresponds to the $\top$ rule while the (absence of) rule for the failure of equality corresponds to the (absence of) rule for 0.

A slight novelty of our proof system for MALL is that we do not need the cut rule (since it is admissible for the usual reasons) and we do not need the initial rule. This last result is slightly more surprising since usually the best one can do in getting rid of the initial rule is to use it only for atomic formulas. But since we have introductions for equality and not other predicates are assume in this version of MALL with equality, there are no atoms: hence, the initial rule is not needed at all. The fact that we can actually use a proof without cut and initial is important for our exercise here since those are the only two rules that mention the same formula twice and at two different polarities. In the game setting, the thing that corresponds most closely to inference rules is the non-deterministic rewriting of expressions, and such rewritings are free of reference to polarities.

Proposition 3.1 *Let Σ be a fixed signature and assume that multiset rewriting is defined for $\mathcal{H}_\Sigma$. Let N be a neutral expression over Σ and $\doteq$ and assume provability is for first-order MALL over equality. If $N \mapsto^* \{\}$ then $\vdash [N]^+$. If N cannot be rewritten to $\{\}$ then $\vdash [N]^-$.*

Proof. Let $k \geq 0$ and define $\mapsto^k$ to be the k-fold join of $\mapsto$ (in particular, $\mapsto^0$ is multiset equality). We prove by induction on k that if $n \geq 0$ and $\{N_1, \ldots, N_n\} \mapsto^k \{\}$ then for all $j \in \{1, \ldots, n\}$, $\vdash [N_j]^+$. If $k = 0$ then $n = 0$ then the conclusion is immediate. Assume that $k > 0$ and that $\{N_1, \ldots, N_n\} \mapsto^k \{\}$. Consider the cases for the first step of this rewriting. The result follows easily by noticing that the rewriting rules for neutral expressions correspond to right-introduction rules for their positive translations.

Next we show that by induction on the size of multisets of neutral expressions (where one counts the number of occurrences of constructors of such expressions as their size) that if $\{N_1, \ldots, N_n\}$ is a multiset of *open* neutral expressions, all of whose free variables are in the set $\mathcal{V}$, and for every ground substitution θ with domain containing $\mathcal{V}$, we have $\{N_1\theta, \ldots, N_n\theta\} \not\mapsto \{\}$ then $\vdash_{\mathcal{V}} [N_1]^-, \ldots, [N_n]^-$. This invariant makes the rule for the quantifier immediate. The main interesting case is the rule for equality. Assume that N_1 is $t \doteq s$ and that for every ground substitution θ, we have that $\{t\theta \doteq s\theta, N_2\theta, \ldots, N_n\theta\} \not\mapsto \{\}$. Now either t and s are unifiable or not. If they are not unifiable, there there is an immediate proof of $\vdash_{\mathcal{V}} \neg(t = s), [N_2]^-, \ldots, [N_n]^-$ by using the equality proof rule marked with ‡. If t and s are unifiable, let σ be a unifier for them. The sequent $\vdash_{\mathcal{V}} \neg(t = s), [N_2]^-, \ldots, [N_n]^-$ is provable using the equality proof rule marked with † if we can show that the sequent $\vdash_{\mathcal{V}'} [N_2\sigma]^-, \ldots, [N_n\sigma]^-$ is provable. Here, $\mathcal{V}'$ is results from $\mathcal{V}$ be removing members of the domain of σ and adding variables that are free in the range of σ. To prove this, let ρ be any ground substitution for the domain $\mathcal{V}'$ and show that $\{(N_2\sigma)\rho, \ldots, (N_n\sigma)\rho\} \not\mapsto \{\}$. If there was, in fact, a path from $\{(N_2\sigma)\rho, \ldots, (N_n\sigma)\rho\}$ to $\{\}$, then there would be a path from $\{(t\sigma)\rho \doteq (s\sigma)\rho, (N_2\sigma)\rho, \ldots, (N_n\sigma)\rho\}$. But this is a contradiction since $\sigma \circ \rho$ is a closed substitution for $\mathcal{V}$. Since $\{(N_2\sigma)\rho, \ldots, (N_n\sigma)\rho\} \not\mapsto \{\}$ and this multiset is smaller, we know by the inductive assumption, that $\vdash_{\mathcal{V}'} [N_2\sigma]^-, \ldots, [N_n\sigma]^-$ and, hence, so too is $\vdash_{\mathcal{V}} \neg(t = s), [N_2]^-, \ldots, [N_n]^-$. □

We have now successfully characterized a neutral approach to proof search in the noetherian Horn clause setting. The neutral computation involves the search of the non-deterministic multiset rewriting tree. If the empty multiset is found, then a proof of a positive translation has been found; in fact, the rewritings along the path to the empty multiset can be used directly to build the proof. If, on the other hand, the search completes without finding the empty multiset, then the entire search tree is used to build the proof of the negative translation. This search in the rewriting tree can make use of variables and unification in order to avoid infinitely branching search. As a result, it is decidable whether or not a given expression can be rewritten to the empty multiset.

Notice that we have started with Horn clause logic which allowed for only two propositional connectives, $\wedge$ and $\vee$, and then ended with a proposition involving four connectives. If one thinks of Horn clauses truth functionally this seems odd since the set $\{\wedge, \vee\}$ is closed by taking de Morgan duals. A

standard proof search analysis of Horn clauses embedded within linear logic maps classical conjunction $\wedge$ to $\otimes$ and the classical disjunction to $\oplus$ (see, for example, [12, Section 4]): their duals are thus two additional connectives $\parr$ and $\&$. In this way, the four binary connectives used in Figure 1 are anticipated.

4 Neutral expressions for two players

The output of $[\cdot]^+$ contains only synchronous connectives and the output of $[\cdot]^-$ contains only asynchronous connectives. So far, there is no accounting for formulas containing both kinds of connectives. To account for formulas in which polarities can switch from one of these kinds to the other, we will add to the language of neutral expressions an operator that switches polarity: in particular, the language of neutral expressions is extended as follows

$$N ::= \ldots \mid \updownarrow N.$$

If Γ is the multiset $\{N_1, \ldots, N_n\}$ then $\updownarrow\Gamma$ is the multiset $\{\updownarrow N_1, \ldots, \updownarrow N_n\}$ ($n \geq 0$). The functions for translating neutral expressions into formulas are extended in the obvious way: $[\updownarrow N]^- = [N]^+$ and $[\updownarrow N]^+ = [N]^-$. Notice that adding this operator is close to the standard approach in games theory in which the difference between the two players is not in the rules they can apply but their alternation during plays.

In the non-deterministic rewriting of neutral expressions, a switched expression does not rewrite. Such expressions represent expressions that are to be given to the *other* player to rewrite. To be more precise, we shall consider the question of whether or not the expression N can be rewritten to a set of the form $\updownarrow\Gamma$. Such rewritings mean that the current player has successfully completed all of her rewritings and is prepared to offer to the next player the expressions in Γ on which to work.

A useful measure of a neutral expression is the maximum number of switched expressions that it can yield on some non-deterministic rewriting. Consider the function $\natural(N)$ from an expression N to a natural number defined as follows: $\natural(\mathbb{0}) = \natural(\mathbb{1}) = \natural(A) = 0$ for any atomic expression A; $\natural(\updownarrow N) = 1$; $\natural(\mathbf{Q}x.N) = \natural(N)$; $\natural(N_1 + N_2) = \max(\natural(N_1), \natural(N_2))$; and $\natural(N_1 \times N_2) = \natural(N_1) + \natural(N_2)$. An expression N is *simple* if $\natural(N) \leq 1$ and for every subexpression $\updownarrow N'$ of N, N' is simple. A multiset $\{N_1, \ldots, N_k\}$ (where $k \geq 0$) of expressions is *simple* if $N_1 \times \cdots \times N_k$ is simple. Clearly, for any expression N, $\natural(N) = 0$ if and only if N does not contain a switch. Alternatively, simple expressions can be defined by the grammar:

$$Z ::= A \mid \mathbb{0} \mid \mathbb{1} \mid Z + Z \mid Z \times Z \mid \mathbf{Q}x.Z$$

$$S ::= Z \mid S + S \mid Z \times S \mid S \times Z \mid \mathbf{Q}x.S \mid \updownarrow S,$$

where A, S, and Z are syntactic variables ranging over atomic expressions, simple expressions, and expressions without occurrences of $\updownarrow$, respectively.

Lemma 4.1 *Let Γ be a simple multiset. If $\Gamma \mapsto \Delta$ then Δ is simple. If $\Gamma \mapsto^* \updownarrow\Delta$ then Δ is simple and contains at most one element.*

5 Games for simple expressions

We first present some basic notions of games as they will be applied here. Let P be a set of positions and let ρ be a binary relationship on P that describes moves from one position to another. The pair $\langle P, \rho\rangle$ is an *arena*. A play is a sequence $P_1.P_2.\ldots.P_n$ of ρ-related moves: that is, for all $i = 0, \ldots, n-1$, $P_i\ \rho\ P_{i+1}$. For now we shall assume that ρ is *noetherian*: no infinite plays are possible. If σ is a set of plays then the set σ/N is defined to be $\{S \mid N.S \in \sigma\}$.

A *$\forall\exists$-strategy for N* is a prefixed closed set σ of plays such that $N \in \sigma$ and for all M such that $N\ \rho\ M$, the set σ/N is a $\exists\forall$-strategy for M. A *$\exists\forall$-strategy for N* is a prefixed closed set σ of plays such that $N \in \sigma$ and for at most one position M such that $N\ \rho\ M$, the set σ/N is a $\forall\exists$-strategy for M. Notice that if σ is a strategy for N ($\forall\exists$-strategy or $\exists\forall$-strategy) then σ/N may be empty (that is, $\sigma = \{N\}$) but for different reasons: if σ is a $\forall\exists$-strategy, then σ/N may be empty only if there is no M such that $N\ \rho\ M$ while if σ is an $\exists\forall$-strategy, σ/N may be empty for arbitrary positions.

A *winning $\forall\exists$-strategy* σ is a $\forall\exists$-strategy such that all maximal sequences in σ are of odd length (i.e., contain an even number of moves). A *winning $\exists\forall$-strategy* σ is a $\forall\exists$-strategy such that all maximal sequences in σ are of even length (i.e., contain an odd number of moves). We shall sometimes call a winning $\forall\exists$-strategy (resp. a winning $\exists\forall$-strategy) a *$\forall\exists$-win* (resp., *$\exists\forall$-win*).

We do not name our players as the opponent and the player but prefer instead to name them more symmetrically as the *current player* and the *other player*. Elsewhere, a $\forall\exists$-strategy is called simply a strategy and $\exists\forall$-strategy is called a counter-strategy.

We consider games based on simple expressions in this paper. Games based on non-simple expressions is a subject for future work (see Section 7.1).

Let the set of positions P be the set of neutral expressions. The move relation, ρ, is defined as the smallest relation such that $N\ \rho\ \mathbb{0}$ if $N \mapsto^* \{\}$ and $N\ \rho\ M$ if $N \mapsto^* \{\updownarrow M\}$. We now consider the nature of winning $\forall\exists$-strategies and winning $\exists\forall$-strategies based on this move relation.

If N does not contain $\updownarrow$ then there is either no move from N or the only move possible is to $\mathbb{0}$. In the first case, there exists a winning $\forall\exists$-strategy for N. In the second case, there is a winning $\exists\forall$-strategy for N (since the second player can make no move from $\mathbb{0}$).

Since all plays are finite and all terminal positions for a game are classified as a win for one player or the other, games for simple expressions are *determinate*: that is, given a simple expression N, there is either a winning $\forall\exists$-strategy or a winning $\exists\forall$-strategy for N.

The following two examples assume that the first-order model used to determine games is the Herbrand universe over a signature containing the

constants z and $s(\cdot)$ (used to encode non-negative integers).

Example 5.1 We code the natural numbers $z, s(z), s(s(z)), \ldots$ using zero and successor. Let A be the finite set $\{n_1, \ldots, n_k\}$ of natural numbers. The expression $x \doteq n_1 + \cdots + x \doteq n_k$ with one free variable encodes the extension of this set and is denoted as $A(x)$. Notice that $n \in A$ if and only if the expression $A(n)$ has a winning $\exists\forall$-strategy. In that case, the positive translation of this expression is provable: $(n = n_1) \oplus \cdots \oplus (n = n_k)$. Furthermore, n is not in A if and only if the expression $A(n)$ has a winning $\forall\exists$-strategy. In that case, the negative translation is provable: $\neg(n = n_1)\ \&\ \cdots\ \&\ \neg(n = n_k)$. If $A(x)$ and $B(x)$ are two expressions encoding the two finite sets of natural numbers A and B, then the expressions $A(x) + B(x)$ and $A(x) \times B(x)$ encode in the same way the intersection and union, respectively, of the sets A and B.

Example 5.2 Let $A(x)$ and $B(x)$ be two expressions encoding the extension of two finite sets of natural numbers A and B. The expression $\mathbf{Q}x.(A(x) \times \updownarrow B(x))$ will be denoted by $A \subseteq B$. Let P be the set $\{z, s(s(z))\}$ and let Q be the set $\{z, s(z), s(s(z))\}$. It is easy to check that the simple expression labeled $P \subseteq Q$, namely,

$$\mathbf{Q}x.([(x \doteq z) + (x \doteq s(s(z)))] \times \updownarrow[(x \doteq z) + (x \doteq s(z)) + (x \doteq s(s(z)))])$$

has a winning $\forall\exists$-strategy, and the following (equivalent) formulas (its negative translation) are provable:

$$\forall x.([\neg(x = z)\ \&\ \neg(x = s(s(z)))]\ \parr\ [(x = z) \oplus (x = s(z)) \oplus (x = s(s(z)))])$$

$$\forall x.([(x = z) \oplus (x = s(s(z)))] \multimap [(x = z) \oplus (x = s(z)) \oplus (x = s(s(z)))]).$$

Similarly, the simple expression labeled $Q \subseteq P$, namely,

$$\mathbf{Q}x.([(x \doteq z) + (x \doteq s(z)) + (x \doteq s(s(z)))] \times \updownarrow[(x \doteq z) + (x \doteq s(s(z)))])$$

has a winning $\exists\forall$-strategy, and its positive translation is also provable:

$$\exists x.([(x = z) \oplus (x = s(z)) \oplus (x = s(s(z)))] \otimes [\neg(x = z)\ \&\ \neg(x = s(s(z)))]).$$

Notice that the use of names to denote expressions in the examples above are mathematic-level conveniences. We are avoiding here an explicit formal treatment of definitions in this context, although the approach of, say, [17] should apply here as well.

Figure 2 presents a straightforward translation $[F]^n$ of MALL-formula F into neutral expressions. This translation is a kind of converse to the translation given in Figure 1 and is used in the following Proposition.

This proposition extends the result of Proposition 3.1 to the case of simple expressions without quantification and atoms.

Proposition 5.3 *Let N be a simple expression involving no quantification nor atoms. If there is a winning $\forall\exists$-strategy for N then $\vdash [N]^-$. If there is a*

$[F]^n = [F]^{n-}$ if F is negative.
$[F]^n = [F]^{n+}$ if F is positive.

- $[F]^{n-} = \updownarrow[F]^{n+}$ for F positive.
- $[\top]^{n-} = \mathbb{0}$
- $[\bot]^{n-} = \mathbb{1}$
- $[A \parr B]^{n-} = [A]^{n-} \times [B]^{n-}$
- $[A \,\&\, B]^{n-} = [A]^{n-} + [B]^{n-}$
- $[\forall x.A]^{n-} = \mathbf{Q}x.[A]^{n-}$
- $[\neg p(t_1, \ldots, t_n)]^{n-} = \dot{p}(t_1, \ldots, t_n)$

- $[F]^{n+} = \updownarrow[F]^{n-}$ for F negative.
- $[0]^{n+} = \mathbb{0}$
- $[1]^{n+} = \mathbb{1}$
- $[A \otimes B]^{n+} = [A]^{n+} \times [B]^{n+}$
- $[A \oplus B]^{n+} = [A]^{n+} + [B]^{n+}$
- $[\exists x.A]^{n+} = \mathbf{Q}x.[A]^{n+}$
- $[p(t_1, \ldots, t_n)]^{n+} = \dot{p}(t_1, \ldots, t_n)$

Fig. 2. Translation from MALL formula to a neutral expression.

winning $\exists\forall$-strategy for N then $\vdash [N]^+$. More precisely, one can build a proof from a winning strategy. Conversely, let F be a MALL formula built using only units and binary connectives such that $[F]^n$ is a simple expression. If $\vdash F$ and F is positive (resp. negative) then $[F]^n$ has a winning $\exists\forall$-strategy (resp. $\forall\exists$-strategy).

Proof. Given a winning strategy (of any kind: $\forall\exists$-win or $\exists\forall$-win) we build by induction a proof for either $\vdash [N]^-$ or $\vdash [N]^+$ depending on the sort of strategy we have. Notice that in winning strategies all the branches have same parity. We thus reason on the length of the largest branch in the strategy, which we refer to as the *size* of the strategy.

Base case: strategy σ has size 0. We are trying to build a proof for $\vdash [N]^-$ given that σ is a $\forall\exists$-win. Thus N is such that $N \not\rightarrowtail^* \updownarrow M$ nor $N \not\rightarrowtail^* \{\}$, that is any maximal internal derivation from N ends up with a multiset of expressions that contains some non-switched expressions and which cannot be rewritten (by maximality). These multisets are made of possibly one switched expression (but not more because N is simple) and of at least one $\mathbb{0}$: they are the only kind of multiset that cannot be rewritten and that are not legal positions: $\{(\updownarrow M), \mathbb{0}, \ldots, \mathbb{0}\}$.

Since $\vdash [N]^-$ is a negative sequent, it is possible to generate part of a proof tree by applying negative logical rules in any order up to a point where no negative rule can be applied (with the additional constraint that the $\top$ rule is applied only when all the negative formulas have been decomposed). Such a tree is actually a proof. Indeed if any branch of the tree leads to a non justified sequent, that means that either a sequent made of exactly one positive formula or an empty sequent is reached. In any other situation, the branch would be extendable. These two situations contradict the fact that no internal derivation from N can lead to a legal position: it is straightforward to see that any branch of the proof tree from $\vdash [N]^-$ to one of the two kinds of sequents just mentioned can be transformed into an internal derivation from

N to a legal position. Thus the negative tree is a proof.

Base case: strategy σ has size 1. We are now building a proof for $\vdash [N]^+$ given that σ is a $\exists\forall$-win. Thus σ starts with a move: $N\ \rho\ M$. There are two cases: either M is $\mathbb{0}$ and $N \mapsto^* \{\}$ or M is any expression and $N \mapsto^* \updownarrow M$: in this case, the move is extended by a $\forall\exists$-win σ' of length 0 for M. In both cases we pick any internal derivation justifying the move and use it to inductively build a proof for $\vdash [N]^+$. In the second case we will additionally refer to the base case for size 0 in order to have a proof of $\vdash [M]^-$.

We reason by induction on the length of the considered internal derivation maintaining the following invariant: if the derivation starts with $N_1, \ldots, N_k$ then the sequents $\vdash [N_1]^+, \ldots, \vdash [N_k]^+$ are all provable. The invariant is true for empty derivations: by hypothesis, we have either an empty multiset, and the assertion is trivial, or the multiset consists in a singleton $\updownarrow M$ and the invariant requires $\vdash [\updownarrow M]^+$ to be provable and base case for size 0 on σ' tells us that $\vdash [M]^-$ is provable. If the induction hypothesis is true for a derivation of length n it is also true for a derivation of length $n+1$. Indeed let ρ be an internal derivation of length $n+1$ from multiset $N_1, \ldots, N_k$ that is $\rho : N_1, \ldots, N_k \mapsto_r N'_1, \ldots, N'_l \mapsto^* \ldots$. Induction hypothesis for the derivation starting with $N'_1, \ldots, N'_l$ ensures that $\vdash [N'_1]^+, \ldots, \vdash [N'_l]^+$ are all provable. We now reason by considering the possible cases for rule r:

(i) r **is $\times$.** We have that $N_1 = N^1_1 \times N^2_1$ and $N'_1, \ldots, N'_l = N^1_1, N^2_1, N_2, \ldots, N_k$ and thus $\vdash [N^1_1]^+, \vdash [N^2_1]^+, \vdash [N_2]^+ \ldots \vdash [N_k]^+$ are all provable with proofs $\Pi_0, \ldots, \Pi_k$ by induction hypothesis. We have

$$\dfrac{\begin{array}{cc}\Pi_0 & \Pi_1\\ \vdash [N^1_1]^+ & \vdash [N^2_1]^+\end{array}}{\vdash [N^1_1]^+ \otimes [N^2_1]^+}\otimes \qquad \begin{array}{ccc}\Pi_2 & & \Pi_k\\ \vdash [N_2]^+ & \ldots & \vdash [N_k]^+\end{array}$$

(ii) r **is $+$.** We have that $N_1 = N^1_1 + N^2_1$ and $N'_1, \ldots, N'_l = N^i_1, N_2, \ldots, N_k$ and thus $\vdash [N^i_1]^+, \vdash [N_2]^+ \ldots \vdash [N_k]^+$ are all provable with proofs $\Pi_1, \ldots, \Pi_k$ by induction hypothesis. We have

$$\dfrac{\begin{array}{c}\Pi_1\\ \vdash [N^i_1]^+\end{array}}{\vdash [N^1_1]^+ \oplus [N^2_1]^+}\oplus_i \qquad \begin{array}{ccc}\Pi_2 & & \Pi_k\\ \vdash [N_2]^+ & \ldots & \vdash [N_k]^+\end{array}$$

(iii) r **is $\mathbb{1}$.** We have that $N_1 = \mathbb{1}$ and $N'_1, \ldots, N'_l = N_2, \ldots, N_k$ so that $\vdash [N_2]^+ \ldots \vdash [N_k]^+$ are all provable with proofs $\Pi_2, \ldots, \Pi_k$ by induction hypothesis. We finally have

$$\dfrac{}{\vdash 1}\,1 \qquad \begin{array}{ccc}\Pi_2 & & \Pi_k\\ \vdash [N_2]^+ & \ldots & \vdash [N_k]^+\end{array}$$

Induction case. Two cases need to be considered: σ has even size $n+1$ (it is a $\forall\exists$-win) or σ has odd size $n+1$ (it is a $\exists\forall$-win).

(i) *σ is of odd size.* This case is similar to the base case for strategies of size

1 when σ starts with a move $N\ \rho\ M$ and yields σ' as an $\forall\exists$-win for M. Using the induction hypothesis on σ' we obtain a proof

$$\begin{array}{c}\Pi\\ \vdash [M]^-\end{array}$$

and by picking some internal derivation $\rho : N \mapsto^*_\rho \updownarrow M$ we build an open positive derivation rooted in $\vdash [N]^+$:

$$\dfrac{\dfrac{}{\vdash [\updownarrow M]^+}}{\vdots}\\ \vdash [N]^+$$

that we can close with Π since $[\updownarrow M]^+ = [M]^-$.

(ii) *σ is of even size.* Once more, the proof is close to the base case. Considering a negative tree built thanks to the same process as the one described previously, the same three cases might occur: either the branch is closed, that is the corresponding internal derivation does not lead to a legal position ($\mathbb{0}$s appear in the multiset and on the proof-theoretical side a $\top$ rule is used), or we get to a sequent made of a positive formula F (there is a move $N\ \rho\ M$ with $[M]^+ = F$, the induction hypothesis allows us to proceed), or we get to an empty sequent (but this case is impossible for parity reasons on the length of the strategy which is required to be winning).

We give only a sketch of the proof for the second part of the proposition which essentially relies on the completeness of focused proofs [3]. We additionally require that the rule for $\top$ is used when no other rule can be applied. We use such a proof to build a winning strategy for $[F]^n$. Each positive layer of the proof results in a first move of a $\exists\forall$-strategy (the detail of this positive part of the proof can be mapped to an internal derivation justifying the move). Each negative layer of the proof results in a $\forall$-branching in the strategy. More precisely an additive slice of a negative layer results in a move of a $\forall\exists$-strategy (or an absence of move, see below). It is easy to check that all these moves are justified by internal derivations and that the $\forall\exists$-branching is complete. Concerning the parity condition, it is due to the fact that the proof can end with either a 1 rule or a $\top$ rule. In the first case, the corresponding position in the game is an empty position ($\{\}$ also denoted by $\mathbb{0}$) immediately following an $\exists\forall$-move that is a position from which no move is possible. In the second case the last multiset of a maximal internal derivation corresponding to this branch of the proof contains a $\mathbb{0}$ so that it cannot justify a move (it cannot be of the form $\updownarrow\Gamma$). □

This corollary follows immediately from the preceding Proposition.

Corollary 5.4 *Let N be a simple expression involving no quantification nor atoms. Either $\vdash [N]^-$ or $\vdash \neg[N]^-$ is provable.*

$$
\begin{array}{cc}
f(B \wedge C) = f(B) + f(C) & h(B \wedge C) = \updownarrow f(B \wedge C) \\
f(B \vee C) = \updownarrow h(B \vee C) & h(B \vee C) = h(B) + h(C) \\
f(\top) = \mathbb{0} & h(\top) = \updownarrow f(\top) \\
f(\bot) = \updownarrow h(\bot) & h(\bot) = \mathbb{0} \\
f(\forall x.B) = \mathbf{Q}x.f(B) & h(\forall x.B) = \updownarrow f(\forall x.B) \\
f(\exists x.B) = \updownarrow h(\exists x.B) & h(\exists x.B) = \mathbf{Q}x.h(B) \\
f(\neg(p(t_1, \ldots, t_n))) = \dot{p}(t_1, \ldots, t_n) & h(\neg A) = \updownarrow f(A) \\
f(A) = \updownarrow h(A) & h(p(t_1, \ldots, t_n)) = \dot{p}(t_1, \ldots, t_n)
\end{array}
$$

Fig. 3. Translating classical formulas into additive neutral expressions. The syntactic variable A ranges over atomic formula.

6 Additive games and truth

As described in, say [13], Hintikka presented a game to determine the truth of a formula in classical first-order logic. In that game, two players P and O, play with the same formula: if that formula is a conjunction, then player P would choose one of the conjuncts; if is a universal quantifier, then player P would pick an instance; if the formulas is a disjunction, then player O picks a disjunct; and if the formula is an existential quantifier, play O picks an instance.

In our setting, such a game is purely additive: that is, the neutral expressions determining the game contain no occurrences of $\times$ and $\mathbb{1}$. Figure 3 defines two mappings, $f(\cdot)$ and $h(\cdot)$, that take classical formulas in negation normal form (formulas where negations have only atomic scope) into additive neutral expressions.

Lemma 6.1 *Let $\mathcal{M}$ be a model and E be a closed first-order formula. The expression $f(E)$ has an $\forall\exists$-win iff $h(E)$ has an $\exists\forall$-win, and the expression $f(E)$ has an $\exists\forall$-win iff $h(E)$ has an $\forall\exists$-win.*

Proof. Proof is by induction on the structure of $f(E)$. Notice that the difference between $f(E)$ and $h(E)$ is that one is the $\updownarrow$ of the other. □

The following proposition shows that the additive fragment of our games corresponds to Hintikka's game theoretic approach to defining truth.

Proposition 6.2 *Let $\mathcal{M}$ be a model and let $f(E) = N$, where E is a closed first-order formula. The formula E is true in $\mathcal{M}$ if and only if there is a $\forall\exists$-win for N.*

Proof. We prove by induction on the structure of E that if $f(E) = N$ then

E is true in $\mathcal{M}$ iff N has an $\forall\exists$-win and E is false in $\mathcal{M}$ iff N has an $\exists\forall$-win.

Case: E is $\top$. Thus, $f(\top) = \mathbb{0}$ which has a $\forall\exists$-win.

Case: E is $\bot$. Thus, $f(\bot) = \updownarrow\mathbb{0}$ which has a $\exists\forall$-win.

Case: E is the atom A and $f(A) = \updownarrow\dot{p}(t_1, \ldots, t_n)$. Thus, A is true in the model if and only if $\updownarrow\dot{p}(t_1, \ldots, t_n)$ has an $\forall\exists$-win. Otherwise, A is false in the model if and only if $\updownarrow\dot{p}(t_1, \ldots, t_n)$ has an $\exists\forall$-win.

Case: E is a negative atom $\neg A$ and $f(\neg A) = \dot{p}(t_1, \ldots, t_n)$. Thus, $\neg A$ is true in the model if and only if A is false in the model if and only if $\dot{p}(t_1, \ldots, t_n)$ has an $\forall\exists$-win. Otherwise, $\neg A$ is false in the model if and only if A is true in the model if and only if $\dot{p}(t_1, \ldots, t_n)$ has an $\exists\forall$-win.

Case: E is $E_1 \wedge E_2$ and $f(E_1 \wedge E_2) = f(E_1) + f(E_2)$. Now, E is true if E_1 and E_2 are true. By induction, this holds if and only if both $f(E_1)$ and $f(E_2)$ have $\forall\exists$-wins but this is if and only if $f(E_1) + f(E_2)$ has a $\forall\exists$-win. Now, E is false iff either E_1 or E_2 is false iff either $f(E_1)$ or $f(E_2)$ have $\exists\forall$-win iff $f(E_1) + f(E_2)$ has a $\exists\forall$-win.

Case: E is $E_1 \vee E_2$ and $f(E_1 \vee E_2) = \updownarrow(h(E_1) + h(E_2))$. E is true if and only if either E_1 or E_2 is true if and only if either $f(E_1)$ or $f(E_2)$ has an $\forall\exists$-win if and only if either $h(E_1)$ or $h(E_2)$ has an $\exists\forall$-win if and only if $h(E_1) + h(E_2)$ has an $\exists\forall$-win if and only if $\updownarrow(h(E_1) + h(E_2))$ has an $\forall\exists$-win.

Case: E is $\forall x.E'$ and $f(E) = \mathbf{Q}x.f(E')$. The formula E is true if and only if $E'[\bar{c}/x]$ is true for all $c \in |\mathcal{M}|$ if and only if $f(E'[\bar{c}/x])$ has an $\forall\exists$-win for all $c \in \mathcal{M}$ if and only if $\mathbf{Q}x.f(E')$ has an $\forall\exists$-win.

Case: E is $\exists x.E'$ and $f(E) = \updownarrow\mathbf{Q}x.h(E')$. The formula E is true if and only if $E'[\bar{c}/x]$ is true for some $c \in \mathcal{M}$ if and only if $f(E'[\bar{c}/x])$ has an $\forall\exists$-win for some $c \in |\mathcal{M}|$ if and only if $h(E'[\bar{c}/x])$ has an $\exists\forall$-win for some $c \in \mathcal{M}$ if and only if $\mathbf{Q}x.h(E')$ has an $\exists\forall$-win if and only if $\updownarrow\mathbf{Q}x.h(E')$ has an $\forall\exists$-win.□

7 Extensions

We plan to consider a number of different ways to extend neutral expressions so that we can leave the restrictive setting of MALL. For example, we plan to consider adding a modal operator to neutral expressions in order to recover the modal operators ! and ?. Extending this framework to study higher-order (predicate) quantification also appears most natural.

Here, we illustrate two other extensions we plan to consider: namely, the extension to the non-simple case and an extension that allows recursion.

7.1 Motivations for the generalization to arbitrary expressions

An important goal for a project that proposes to describe a game semantics for proof search is to show how fundamental results for games can be used to provide new proofs for fundamental results in proof theory. While we leave a full development of this topic for future work, we note that one would expect

the following kinds of results to hold in game theory.

- For any N, it is natural to expect that $N \times \updownarrow N$ has a $\forall\exists$-win, in which case, all occurrences of the initial sequent $\vdash [N]^-, [N]^+$ are admissible.
- For any expressions M, N, and P, if $M \times N$ and $(\updownarrow N) \times P$ have $\forall\exists$-wins, then $M \times P$ should have a $\forall\exists$-win. From this result, cut-elimination for MALL should follow.

In both of these cases, narrowing ourselves to simple expressions results in the restriction that N must not contain the switch operator in the first case and that P (and M if we require N to be general) must not contain the switch in the second case; that is, the formulas involved in the statements above initial and cut would be neutral expressions from Section 3 and these expressions explain proof search with Horn clauses only. Thus, to allow ourselves to eventually obtain the proof theory results concerning the admissibility of cut and initial, we must consider the richer form of games based on more general expressions.

One more reason for such an extension is the fact that for the moment, only a weak form of multiplicative connectives is treated by the simple case: if the formula $N \times M$ is simple, only one among the two immediate subformula can contain an alternation of polarity. Understanding exactly which kind of multiplicatives we capture with the simple expressions will be of great interest.

Notice that the structure of games will certainly be more complex for non-simple expressions than for simple expressions: one illustration of this fact is that games will no longer be determinate, even for the quantifier-free case. For example, the non-simple neutral expression $\updownarrow\mathbb{1} \times \updownarrow\mathbb{1}$ translates positively to $\bot \otimes \bot$ and negatively to $1 \parr 1$: neither of these formulas are provable in linear logic. Finding extensions of simple expressions for which games remain determinate seems like an interesting project.

It is also clear that the mix inference rule [7] does not fit well with this game semantics approach since there are formulas B such that mix can be used to prove B and $\neg B$ (for example, take B to be either $\bot \otimes \bot$ or $1 \parr 1$). Adding mix would make a generalization of Proposition 5.3 impossible.

7.2 Adding fixpoints

Consider extending the language of neutral expressions with the constants $\{\mathit{fix}_n\}_{n\geq 0}$, where the expression $(\mathit{fix}_n \lambda P \lambda x_1 \ldots \lambda x_n . M)$ denotes an expression in which the bound variable P is an n-ary recursive call. If the syntactic type of neutral expressions is *exp* and the syntactic type of terms is i, then the syntactic type of fix_n is $(\alpha_n \to \alpha_n) \to \alpha_n$ where α_n is $i \to \ldots \to i \to \mathit{exp}$ (where i occurs n times). The rule for extending $\mapsto$ for such expressions is simply

$$(\mathit{fix}_n F t_1 \ldots t_n), \Gamma \mapsto (F(\mathit{fix}_n F) t_1 \ldots t_n), \Gamma$$

where we assume that λ-conversion is automatically applied to simplify syntactic expressions involving β-redexes.

All examples in this section assume that the first-order model used to determine games is the Herbrand universe over a signature containing the constants z and $s(\cdot)$ (used to encode non-negative integers).

Example 7.1 The expression

$$(\mathit{fix}_2\ \lambda leq \lambda n \lambda m[(n \doteq z) + \mathbf{Q}p\mathbf{Q}q.(n \doteq s(p) \times m \doteq s(q) \times leq(p,q))])$$

has syntactic type $i \to i \to exp$. This expression, named L, denotes a recursive expression on two arguments. Since the expression L contains no $\updownarrow$ operators, then $L(n,m) \mapsto \updownarrow\Gamma$ only if Γ is empty (assuming that n and m are terms denoting natural numbers). The expression L can be used to compute the "less-than-or-equal-to" relations on natural numbers in the following sense. The expression $L(n,m)$ has an $\exists\forall$-win if and only if $L(n,m) \mapsto^* \updownarrow\Gamma$ if and only if the number represented by the term n is less than or equal to the number represented by the term m. Similarly, $L(n,m)$ has an $\forall\exists$-win if and only if $L(n,m) \mapsto^* \updownarrow\Gamma$ is not possible if and only if the number represented by the term n is greater than the number represented by the term m.

Example 7.2 We can now define the maximum of a set of numbers. Let A be a non-empty set of numbers and let $A(n)$ denote the expression encoding this set (as described in Example 5.1). Let L be the recursive expression given in Example 7.1 for the less-than-or-equal-to relation. Let $\mathit{max}A(n)$ be the following expression with the one free variable n:

$$A(n) \times \updownarrow\mathbf{Q}m(A(m) \times \updownarrow L(m,n))$$

Developing the moves for this (simple) expression yields the following. The expression $\mathit{max}A(n)$ has an $\forall\exists$-win if and only if n is not in A or it is not the largest member of A. Similarly, $\mathit{max}A(n)$ has a $\exists\forall$-win if and only if n is the largest member of A.

Example 7.3 Inductive defined sets can be described as games [2]. Let D be a set and let Φ be a (finite) set of pairs $\langle X, x\rangle$ where X is a (finite) subset of D and $x \in D$. Let I_Φ be the smallest subset of D closed under the rules in Φ: that is, I_Φ is the smallest set Y such that if $\langle X, x\rangle \in \Phi$ and $X \subseteq Y$ then $x \in Y$. Let $x_0 \in D$. Consider the following game to determine membership of x_0 in I_Φ. After $n \geq 0$ rounds of the game, the first player is required to pick a set X_n such that $\langle X_n, x_n\rangle \in \Phi$ and the second player is required an element $x_{n+1} \in X_n$. If a player cannot move, then the other player wins. If a play has infinite length, it is a loss for the first player. It is now the case that $x_0 \in I_\Phi$ if and only if the first player has a winning strategy. The following recursive neutral expression describes this game for checking membership of y in I_Φ.

$$(\mathit{fix}_1 \lambda I \lambda y.\ \sum_{\langle X,x\rangle \in \Phi} (x \doteq y \times \updownarrow \sum_{w \in X} \updownarrow (I\ w)))$$

Example 7.4 Consider an abstract transition system given by the triple $\langle \Lambda, S, \delta \rangle$, where Λ is a non-empty set of actions, S is a non-empty set of states, and $\delta \subseteq S \times \Lambda \times S$. Assume that actions and states are made into individual constants and that δ is a finite set. Let $\delta(x, y, z)$ be the following expression with the three free variables x, y, and z:

$$\sum_{(p,a,q)\in\delta} (x \doteq p \times y \doteq a \times z \doteq q).$$

This encoding of a ternary relation is an immediate generalization of the encoding of finite sets in Example 5.1. Bisimulation between two states can be defined using the following recursive expression

$$\begin{array}{l}(\mathit{fix}_2\ \lambda \mathit{bisim} \lambda p \lambda q.\ [\mathbf{Q}a\mathbf{Q}p'.\delta(p,a,p') \times \updownarrow \mathbf{Q}q'(\delta(q,a,q') \times \updownarrow \mathit{bisim}(p',q'))] + \\ \qquad\qquad [\mathbf{Q}a\mathbf{Q}q'.\delta(q,a,q') \times \updownarrow \mathbf{Q}p'(\delta(p,a,p') \times \updownarrow \mathit{bisim}(p',q'))])\end{array}$$

Let *Bisim* be the above expression (of syntactic type $i \to i \to \mathit{exp}$) and let p and q be two states (members of S). The plays that start with the expression $\mathit{Bisim}(p, q)$ are exactly those used to describe bisimulation in, say, [24] and [22, Chapter 2]. This encoding of bisimulation is also close to the encoding using a proof theoretic notion of definitions [18].

As these examples illustrate, the proper translation of the *fix* combinator should be the greatest fixed point operator ν in the negative translation and the least fixed point operation μ in the positive translation.

$$[\mathit{fix}_n \lambda P \lambda x_1 \ldots \lambda x_n.N]^- = \nu_n \lambda P \lambda x_1 \ldots \lambda x_n.[N]^-$$

$$[\mathit{fix}_n \lambda P \lambda x_1 \ldots \lambda x_n.N]^+ = \mu_n \lambda P \lambda x_1 \ldots \lambda x_n.[N]^+$$

For example, the positive translation of L, which defines the less-than-or-equal-to relation, is

$$(\mu_2\ \lambda \mathit{leq} \lambda n \lambda m[(n = z) \oplus \exists p \exists q.(n = s(p) \otimes m = s(q) \otimes \mathit{leq}(p, q))]).$$

This expression corresponds to the "Clark completion" of the Prolog program

```
leq(z,N).
leq(s(P),s(Q)) :- leq(P,Q).
```

The negative translation of the *Bisim* neutral expression would then be

$$\begin{array}{l}(\nu_2\ \lambda \mathit{bisim} \lambda p \lambda q.\ [\forall a \forall p'.[\delta(p,a,p')]^+ \multimap \exists q'([\delta(q,a,q')]^+ \otimes \mathit{bisim}(p',q'))]\ \&\ \\ \qquad\qquad [\forall a \forall q'.[\delta(q,a,q')]^+ \multimap \exists p'([\delta(p,a,p')]^+ \otimes \mathit{bisim}(p',q'))])\end{array}$$

and this corresponds to the greatest fixed point of the "definition"

$$\begin{array}{rl}\mathit{bisim}(p,q) \triangleq & [\forall a \forall p'.\mathit{one}(p,a,p') \supset \exists q'(\mathit{one}(q,a,q') \wedge \mathit{bisim}(p',q'))] \wedge \\ & [\forall a \forall q'.\mathit{one}(q,a,q') \supset \exists p'(\mathit{one}(p,a,p') \wedge \mathit{bisim}(p',q'))]),\end{array}$$

which has been studied in [18,26].

Formal results analogous to those in Propositions 5.3 for the cases involving recursion will certainly be more involved. Soundness holds: that is, if $\vdash [N]^-$ (resp, $\vdash [N]^+$) for simple expressions N then there is a $\forall\exists$-win ($\exists\forall$-win). The converse is, of course, not possible since winning strategies will certainly require declaring infinite paths to be wins (for greatest fixed point) or losses (for least fixed points) and recognizing such infinite wins and losses will not be recursively enumerable in general. Of course, one can imagine strengthening the proof system of MALL to contain induction and co-induction principles for μ and ν formulas along the lines described in [20,26].

8 Future work

We need to understand better possible connections between this work and that done by Girard in his recent work on *Ludics* [10] and by the work on games for linear logic in the tradition of [1,4,14], for example. These approaches focus more on the role of cut-elimination and interaction than the program that we have outlined here. Of course, we are interested in an analysis of cut-elimination but from the game theoretical approach outlined in Section 7.1: such an analysis should lead us to consider interactions between strategies.

Extending neutral expressions to account for modals and higher-order quantification seems like a natural task to consider next. Clearly, developing the ideas we outlined for fixed point expressions should allow us to link together a number of computer science motivated problems (such as bisimulation, model checking, etc) to specifications written in logic.

There are a number of implementation-related questions that are interesting to follow. One is the basic problem of exploring arenas in order to find either winning $\forall\exists$-strategies or winning $\exists\forall$-strategies. Another problem is determining how to use logic variables and unification to determine quantification instantiations when exploring the moves. Tiu has implemented a system, *Level 0/1* [25], that can automatically search for winning strategies when recursively defined formulas are restricted to at most one alternation of polarities: for example, this system will successfully prove or refute bisimulations for noetherian transition systems as described above. This system employs a simple generalization of the way Prolog mixes unification and proof search and is incomplete for some two player games. We plan to re-examine the role of unification in the more general setting.

References

[1] S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. In *14th Annual Symposium on Logic in Computer Science*, pages 431–442. IEEE Computer Society Press, 1999.

[2] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, Amsterdam, 1977.

[3] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[4] A. Blass. A game semantics for linear logic. *Annals Pure Appl. Logic*, 56:183–220, 1992. Special Volume dedicated to the memory of John Myhill.

[5] L.-H. Eriksson. A finitary version of the calculus of partial inductive definitions. In L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, editors, *Proceedings of the Second International Workshop on Extensions to Logic Programming*, volume 596 of *LNAI*, pages 89–134. Springer-Verlag, 1991.

[6] W. Flescher. Dialogues, strategies and intuitionistic provability. *Annals of Mathematical Logic*, 28:217–254, 1985.

[7] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[8] J.-Y. Girard. A fixpoint theorem in linear logic. An email posting to the mailing list linear@cs.stanford.edu, February 1992.

[9] J.-Y. Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[10] J.-Y. Girard. Locus solum. *Mathematical Structures in Computer Science*, 11(3):301–506, June 2001.

[11] L. Hallnäs and P. Schroeder-Heister. A proof-theoretic approach to logic programming. II. Programs as definitions. *Journal of Logic and Computation*, 1(5):635–660, October 1991.

[12] J. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.

[13] W. Hodges. Logic and games. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2004.

[14] F. Lamarche. Games semantics for full propositional linear logic. In *LICS*, pages 464–473, 1995.

[15] J.-V. Loddo. *Généralisation des Jeux Combinatoires et Applications aux Langages Logiques*. PhD thesis, Université Paris VII, 2002.

[16] P. Lorenzen. Ein dialogisches konstruktivitätskriterium. In *Infinitistic Methods: Proceed. Symp. Foundations of Math.*, pages 193–200. PWN, 1961.

[17] R. McDowell and D. Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000.

[18] R. McDowell, D. Miller, and C. Palamidessi. Encoding transition systems in sequent calculus. *Theoretical Computer Science*, 294(3):411–437, 2003.

[19] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[20] A. Momigliano and A. Tiu. Induction and co-induction in sequent calculus. In M. C. Stefano Berardi and F. Damiani, editors, *Post-proceedings of TYPES 2003*, number 3085 in LNCS, pages 293 – 308, January 2003.

[21] D. Pym and E. Ritter. A games semantics for reductive logic and proof-search. In D. Ghica and G. McCusker, editors, *GaLoP 2005: Games for Logic and Programming Languages*, pages 107–123, 2005.

[22] D. Sangiorgi and D. Walker. *π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[23] P. Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *Eighth Annual Symposium on Logic in Computer Science*, pages 222–232. IEEE Computer Society Press, IEEE, June 1993.

[24] C. Stirling. Games for bisimulation and model checking. Notes for Mathfit Workshop on Finite Model Theory, University of Wales, Swansea, July 1996.

[25] A. Tiu. *Level 0/1 Prover: A tutorial*, September 2004. Available online.

[26] A. Tiu. *A Logical Framework for Reasoning about Logical Specifications*. PhD thesis, Pennsylvania State University, May 2004.