City University of New York (CUNY)
CUNY Academic Works

Computer Science Technical Reports | Graduate Center

2005

TR-2005012: Propositional Games with Explicit Strategies

Bryan Renne

Follow this and additional works at: http://academicworks.cuny.edu/gc_cs_tr

 Part of the Computer Sciences Commons

Recommended Citation
Renne, Bryan, "TR-2005012: Propositional Games with Explicit Strategies" (2005). *CUNY Academic Works.*
http://academicworks.cuny.edu/gc_cs_tr/267

This Technical Report is brought to you by CUNY Academic Works. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@gc.cuny.edu.

Propositional Games with Explicit Strategies

Bryan Renne
Computer Science
CUNY Graduate Center
365 Fifth Avenue, Room 4319
New York NY 10016

http://bryan.renne.org/

May 21, 2007

Abstract

This paper presents a game semantics for LP, Artemov's Logic of Proofs. The language of LP extends that of propositional logic by adding formula-labeling terms, permitting us to take a term t and an LP formula A and form the new formula $t:A$. We define a game semantics for this logic that interprets terms as winning strategies on the formulas they label, so $t:A$ may be read as "t is a winning strategy on A." LP may thus be seen as a logic containing in-language descriptions of winning strategies on its own formulas.

We apply our semantics to show how winnable instances of certain extensive games with perfect information may be embedded into LP. This allows us to use LP to derive a winning strategy on the embedding, from which we can extract a winning strategy on the original, non-embedded game. As a concrete illustration of this method, we compute a winning strategy for a winnable instance of the well-known game Nim.

1 Introduction

Propositional Verification is a game played by two players, who we call *True* and *False*. The game requires two input parameters: a formula A in the language of propositional logic and a background model M that interprets atomic formulas. To play the game, the players begin on the formula A and take turns choosing an immediate subformula instance of the current formula, with True choosing at positive subformula instances of A and False choosing at negative subformula instances of A. Eventually an atomic formula p is reached, at which point the game is over. True wins in two cases: (1) p is true in M and positive in A, and (2) p is false in M and negative in A. False wins exactly when True loses.

Propositional Verification can be used to define a notion of truth for propositional formulas: to say A is *true* means that for every model M interpreting atomic formulas, True has a winning strategy in the Propositional Verification Game on A with background model M. In this context, a *strategy* is just a function that specifies the choices True should make when it is his turn to move, and a *winning strategy* is a strategy that True can follow so as to guarantee himself a win, no matter the moves of False. Identifying the true formulas as those for which True has a winning strategy in Propositional Verification gives us a *game semantics* for classical propositional logic.

Propositional Verification may be extended to the language of first-order logic, yielding a *First-Order Verification Game*.[1] Hintikka and Sandu introduced *partial information* extensions of First-Order Verification in order to provide a semantics for *Independence-Friendly* (or *IF*) logic, a logic that allows for arbitrary dependencies between quantifiers and logical connectives in a first-order language [9]. Research in IF logics has centered on identifying these dependencies and understanding their influences on logic (see Sandu's paper [14] for a flavor of this work, along with his papers [13, 15] for an impression of IF logic applications in linguistics).

Verification games have been used to provide semantics for many other logics, including intuitionistic logic and modal logic (see Hodges' overview of games in logic [10]).

In this paper, we define a game semantics for the logic LP, Artemov's Logic of Proofs [4]. LP is a conservative extension of classical propositional logic with a language obtained from that of propositional logic by adding formula-labeling terms. If t is such a term and A is an LP formula, then $t:A$ is also an LP formula. Terms have a structure that mimics deduction in the system in the sense of Artemov's *Internalization Theorem*: each LP theorem A has a term t such that $t:A$ is also an LP theorem [4]. It is in this sense we say that LP *internalizes* its theorems—thereby providing a reason for each theorem's veracity—leading us to the informal reading of $t:A$ as "A for reason t." This in-language notion of justification has led to various multi-modal extensions of LP, which have been used as logics of multi-agent knowledge with reasons. These logics are now grouped under the name *Justification Logic* [2, 3, 1].

This paper defines a game semantics for the most basic Justification Logic, LP itself. Our game semantics adds to the list of known semantics for LP, which presently include an arithmetic semantics [4], a minimal semantics, [11], and a Kripke-style semantics [6]. We define this game semantics by extending Propositional Verification to the language of LP, interpreting LP terms as winning strategies on the formulas they label. We may thus assign to the LP formula $t:A$ the informal reading "t is a winning strategy on A."

Since terms are interpreted as winning strategies in LP Verification, the LP Internalization Theorem implies that winning strategies in LP Verification can be described within the LP Verification Game itself. We will use this in the end of the paper to show how the existence of a winning-strategy–preserving embedding of certain extensive games G of with perfect information into Propositional Verification (and hence into LP Verification) allows us to use the Internalization Theorem to build a winning strategy on the embedded version of a

[1] The basic ideas of this extension go back to Peirce [12].

winnable G-instance, from which we can then extract a winning strategy on the original, non-embedded G-instance itself. For concreteness, we provide an example strategy extraction for the well-known game of Nim [5].

But before we can do this, we must describe LP and its game semantics. So let us begin by introducing LP, Artemov's Logic of Proofs [4].

2 The Language and Theory of LP

For present purposes, the *language of propositional logic* consists of a countable number of propositional variables, the propositional constant $\top$ for truth, the propositional constant $\bot$ for falsehood, and the logical connective $\supset$ for implication. The *atoms*, also called *atomic formulas*, consist of the propositional letters and the propositional constants. The *propositional formulas* are obtained from the atoms by closure under the following rule of formula formation: if A and B are propositional formulas, then $A \supset B$ is also a propositional formula. A propositional formula written using other logical connectives is understood as an abbreviation for the appropriate formula in this language.

The *language of* LP is obtained from that of propositional logic by adding a countable number of *constant* symbols, a countable number of *variable* symbols, the binary function symbols $+$ and $\cdot$, and the unary function symbol $!$. The *atomic terms* consist of the constants and the variables. *Terms* are built-up from the atomic terms using the function symbols.

Notation. The letters t, u, and v will be used as metavariables ranging over terms.

The LP *formulas* are obtained from the propositional formulas by closure under both the rule of propositional formula formation and also the following rule: if A is an LP formula and t is a term, then $t\!:\!A$ is also an LP formula. In the remainder of the paper, unqualified use of the word *formula* refers to an LP formula.

Notation. Use of letters as metavariables:

- A, B, C, and D will be used for formulas.
- p will be used for atoms (propositional letters, $\top$, or $\bot$).

Definition 2.1. A *term-labeled formula* is a formula of the form $t\!:\!A$.

Definition 2.2. The *theory of* LP is given by the following axiom schemas and rules of inference.

- **Axiom Schemas**

LP0.	Axiom schemas for classical propositional logic
LP1.	$t\!:\!(A \supset B) \supset (s\!:\!A \supset (t \cdot s)\!:\!B)$
LP2.	$t\!:\!A \supset !t\!:\!(t\!:\!A)$
LP3.	$t\!:\!A \supset (t+s)\!:\!A$
	$s\!:\!A \supset (t+s)\!:\!A$
LP4.	$t\!:\!A \supset A$

- **Rules of Inference**
  - The rule of *Modus Ponens*:
$$\frac{A \quad A \supset B}{B}$$
  - The rule of *Constant Necessitation*:
$$\frac{c \text{ is a constant, } A \text{ is an } \mathsf{LP} \text{ axiom}}{c{:}A}$$

Theorem 2.3 (Artemov's Internalization Theorem [4])**.** Each LP theorem A has a term t such that $t{:}A$ is also an LP theorem. Further, the term t does not contain variables.

Proof. By induction on the length of a derivation of A. In case A is an axiom, then, letting c be a constant, $c{:}A$ is an LP theorem. Otherwise, A is obtained by Modus Ponens or Constant Necessitation. If A is obtained from the theorems B and $B \supset A$ by Modus Ponens, then the induction hypothesis yields terms u and v such that $v{:}B$ and $u{:}(B \supset A)$ are both theorems, and so it follows from **LP1** and Modus Ponens that $(u \cdot v){:}A$ is a theorem. If $c{:}A$ is obtained by Constant Necessitation, it follows from **LP2** and Modus Ponens that $!c{:}(c{:}A)$ is a theorem. □

3 Strategies and Strategy Maps

Before we introduce the LP Verification Game, we must first describe our notion of strategy for a formula. Since we will assign a strategy to each term occurring in a formula, we need a means of simultaneously specifying all of these assignments. A strategy map is a function that does this for us. But before we can define a strategy map, we first need some basic definitions.

Definition 3.1. The *construction tree* of A, written $\mathcal{T}(A)$, is the labeled binary tree constructed as follows:

- the root is labeled A;
- each node labeled $B \supset C$ has two children: a left child labeled B and a right child labeled C; and
- each node labeled $t{:}B$ has one child: a right child labeled B.

Definition 3.2. A *path* in a tree T is a sequence $n_1, n_2, n_3, \ldots, n_k$ of nodes such that n_1 is the root of T and, for each $i < k$, the node n_{i+1} is a child of n_i. A path is said to *end on* the node n iff n is the last node in the sequence.

Definition 3.3.

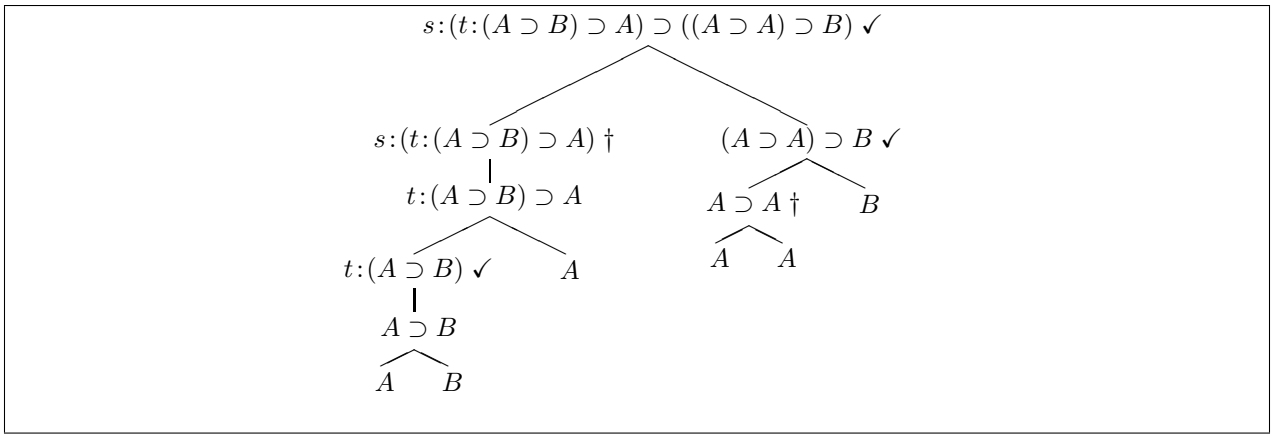

Figure 1. A formula construction tree. The choice points are marked by a check (✓) and the response points are marked by a dagger (†). Recall the child-naming convention from Definition 3.1: if a node has exactly one child, then this child is a right child.

- A *subformula instance* of A is a node in $\mathcal{T}(A)$.
- A subformula instance n of A is *immediate* iff n is a child of the root of $\mathcal{T}(A)$.
- A subformula instance n of A is *positive* iff the path in $\mathcal{T}(A)$ ending on n contains the left child of an even number of nodes.
- A subformula instance n of A is *negative* iff n is not positive.
- To say that the subformula instances n_1 and n_2 of A have the *same polarity* means that n_1 is positive iff n_2 is positive.
- The subformula instances n_1 and n_2 of A have *opposite polarity* iff n_1 and n_2 do not have the same polarity.
- A subformula instance n of A is *term-labeled* iff the formula labeling n is term-labeled.
- When referring to a formula B as a *subformula instance* of A, the referent is a subformula instance n of A (so n is a node in $\mathcal{T}(A)$) that is labeled by B. The adjectives *immediate*, *positive*, and *negative* have their corresponding meaning when applied to a formula B that is a subformula instance of A.

Definition 3.4. A *choice point* of A is a non-leaf positive subformula instance of A that does not have a positive term-labeled ancestor.

Definition 3.5. A *response point* of A is a non-leaf negative subformula instance of A that does not have a negative term-labeled ancestor.

Example 3.6. Figure 1 indicates the choice points and response points in a complicated construction tree.

Example 3.7. The only choice point of $t{:}A$ is the root of $\mathcal{T}(t{:}A)$.

Definition 3.8. A *strategy* on A is a function mapping each choice point B of A to an immediate subformula instance of B.

Definition 3.9. A *counter-strategy* on A is a function mapping each response point B of A to an immediate subformula instance of B.

Notation. Suppose that s^* is a strategy on A, that s_* is a counter-strategy on A, that B^- is a response point of A, and that C^+ is a choice point of A.

- $s^* \restriction B^-$ is the counter-strategy on B^- that maps each response point D of B^- to the immediate subformula instance $s^*(D)$ of D.
- $s^* \restriction C^+$ is the strategy on C^+ that maps each choice point D of C^+ to the immediate subformula instance $s^*(D)$ of D.
- $s_* \restriction B^-$ is the strategy on B^- that maps each choice point D of B^- to the immediate subformula instance $s_*(D)$ of D.
- $s_* \restriction C^+$ is the counter-strategy on C^+ that maps each response point D of C^+ to the immediate subformula instance $s_*(D)$ of D.

Definition 3.10. We will call the function with empty domain both the *trivial strategy* and also the *trivial counter-strategy*.

Remark 3.11. The trivial strategy is a strategy on any formula that does not have choice points, and the trivial counter-strategy is a counter-strategy on any formula that does not have response points. For example: the trivial strategy is a strategy on each atom p; the trivial counter-strategy is a counter-strategy on each atom p.

Definition 3.12. A *possible strategy map* is a partial function S that maps the term-formula pair (t, A) to a strategy $S(t, A)$ on A. Notation: $S(t, A)\downarrow$ means (t, A) is in the domain of S, and $S(t, A)\uparrow$ means (t, A) is not in the domain of S.

Definition 3.13. A possible strategy map S is *axiomatically saturated* iff $S(c, A)\downarrow$ for each constant c and axiom A.

Remark 3.14. It is sometimes convenient to restrict the rule of Constant Necessitation, thereby yielding a weaker theory. While our present task is to define a semantics for the full theory of LP, which allows unrestricted use of the rule of Constant Necessitation, our semantics will work just as well for these weaker theories if one appropriately redefines the meaning of axiomatically saturated, Definition 3.13. Namely, if we impose a restriction R on the use of the rule of Constant Necessitation, we would then call the possible strategy map S *axiomatically saturated* iff $S(c, A)\downarrow$ whenever $c{:}A$ is derivable via the rule of Constant Necessitation with restriction R. With this modified definition of axiomatically saturated, the work of the paper could then be carried out as-is.

Definition 3.15. A *strategy map* is a possible strategy map S that satisfies each of the following conditions:

1. *Product.* If $S(u, A \supset B)\downarrow$ and $S(v, A)\downarrow$, then both
   (a) $S(u \cdot v, B) = S(u, A \supset B) \restriction B$, and
   (b) $S(u, C \supset B)\downarrow$ and $S(v, C)\downarrow$ implies $S(u, C \supset B) \restriction B = S(u, A \supset B) \restriction B$.
2. *Proof Checker.* $S(t, A)\downarrow$ implies $S(!t, t\!:\!A)\downarrow$.
3. *Sum.*
   (a) $S(t, A)\downarrow$ implies $S(t + u, A) = S(t, A)$.
   (b) $S(t, A)\uparrow$ and $S(u, A)\downarrow$ implies $S(t + u, A) = S(u, A)$.

Remark 3.16. Condition 1b in the definition of strategy map (Definition 3.15) ensures that S is well-defined on term-formula pairs whose term is of the form $u \cdot v$. This condition is otherwise unused.

Definition 3.17. Suppose that n_1 and n_2 are each ancestors of a node n in the tree $\mathcal{T}(A)$. n_1 is *nearer* to n than n_2 iff the length of the path in $\mathcal{T}(A)$ ending on n_1 is greater than the length of the path in $\mathcal{T}(A)$ ending on n_2. The ancestors of n are linearly ordered by this notion of nearness.

Definition 3.18. Let S be a strategy map. A path $n_1, n_2, n_3, \ldots, n_k$ of $\mathcal{T}(A)$ is *S-admissible* iff for each $i < k$, we have each of the following:

- if n_i is labeled $t\!:\!B$ and $S(t, B)\uparrow$, then $i = k$;
- if n_i is positive and n_i has a positive term-labeled ancestor, then, letting $u\!:\!C$ be the label of n_i's nearest positive term-labeled ancestor, we have $n_{i+1} = S(u, C)(n_i)$; and
- if n_i is negative and n_i has a negative term-labeled ancestor, then, letting $v\!:\!D$ be the label of n_i's nearest negative term-labeled ancestor, we have $n_{i+1} = S(v, D)(n_i)$.

Definition 3.19. Let S be a strategy map, let A^* be a strategy on A, let A_* be a counter-strategy on A, and let $n_1, n_2, n_3, \ldots, n_k$ be a path in $\mathcal{T}(A)$.

- This path is *in accordance with* the strategy A^* iff the path is S-admissible and we have $n_{i+1} = A^*(n_i)$ for each choice point n_i of A.
- This path is *in accordance with* the counter-strategy A_* iff the path is S-admissible and we have $n_{i+1} = A_*(n_i)$ for each response point n_i of A.

Definition 3.20. Given a strategy map S, a *play* of a formula A is an S-admissible path in $\mathcal{T}(A)$ of maximal length. Notation: if τ is a play of A and B occurs along τ, then $\tau \restriction B$ is the play of B obtained from τ by deleting those nodes not in $\mathcal{T}(B)$.

Lemma 3.21 (Play Lemma)**.** Let S be a strategy map. If A^* is a strategy on A and A_* is a counter-strategy on A, then there is a unique play τ of $\mathcal{T}(A)$ such that τ is in accordance with A^* and with A_*.

Proof. To construct τ, we produce a sequence $n_1, n_2, n_3, \ldots, n_k$ of nodes in $\mathcal{T}(A)$. First, let n_1 be the root of $\mathcal{T}(A)$. Once n_i has been specified, if we have either that n_i is a leaf or else that n_i is labeled $v : C$ with $S(v, C)\uparrow$, then we are done. Otherwise, n_i is a non-leaf, and we specify a child n_{i+1} of n_i according to the following case analysis.

- Case: n_i is a choice point of A.

  Set $n_{i+1} := A^*(n_i)$.

- Case: n_i is a response point of A.

  Set $n_{i+1} := A_*(n_i)$.

- Case: n_i is neither a choice point of A nor a response point of A.

  Let $u : B$ be the nearest term-labeled ancestor of n that has the same polarity as n and then set $n_{i+1} := S(u, B)(n_i)$.

Each node in $\mathcal{T}(A)$ is either a choice point, a response point, or else has a nearest term-labeled ancestor of the same polarity. Therefore, these three cases are mutually exclusive and exhaust all possibilities. Further, one may argue by induction on i that

- the path $n_1, n_2, n_3, \ldots, n_i$ is in accordance with A^* and with A_*, and
- if n_i is a non-leaf and it is not the case that n_i is labeled $v : C$ with $S(v, C)\uparrow$, then n_i satisfies one of the three construction cases.

It follows that the construction indeed specifies a play τ of $\mathcal{T}(A)$ in accordance with A^* and with A_*. The uniqueness of τ follows from the fact that any play τ' in $\mathcal{T}(A)$ in accordance with A^* and with A_* must also be built using this construction, so $\tau' = \tau$. □

4 Truth

Now that we have seen how a strategy, a counter-strategy, and a strategy map work together to specify a unique play of each formula, we introduce a model that interprets atomic formulas. Since our atoms are propositional letters and propositional constants, we take a rather simple model: the set of atoms that are true in the model.

Definition 4.1. A *valuation set* is any set obtained as a union of $\{\top\}$ with a (possibly empty) set of propositional letters.

In our description in the introduction of the Propositional Verification Game on A, we described how the first player ("True") wins on an atom p in one of two cases: (1) p is true in the background model and positive in the formula A, and (2) p is false in the background model and negative in the formula A. The next definition provides a name both for atoms that match one of these two cases and also for atoms that match neither of these two cases. We then define the notion of winning (counter-)strategy.

Definition 4.2. Given a valuation set V, to say that a leaf n of $\mathcal{T}(A)$ is *matching in* A means that n is positive iff the label of n is in V. If a leaf is not matching in A, it is called *mismatching in* A.

Definition 4.3 (Winning Strategies, Winning Counter-Strategies)**.** Let V be a valuation set and S be a strategy map.

- A a strategy A^* on a formula A is *winning* iff for each play τ of $\mathcal{T}(A)$ in accordance with A^*, the end of τ is either a negative non-leaf or else a leaf matching in A.
- A counter-strategy A_* on A is *winning* iff for each play τ of $\mathcal{T}(A)$ in accordance with A_*, the end of τ is either a positive non-leaf or else a leaf mismatching in A.

Given a valuation set V and a strategy map S, we may think of a play of A as a path in $\mathcal{T}(A)$ chosen during the LP Verification Game on A. As before, the first player ("True") moves on positive subformula instances and the second player ("False") moves on negative subformula instances. A player's move consists of choosing an immediate subformula instance of the current formula, subject to the restriction that the path in $\mathcal{T}(A)$ ending on the chosen node is S-admissible. Notice that a play may end on a non-leaf n—this happens when n is a term-labeled formula $t\!:\!B$ with $S(t, B)\!\uparrow$ (Definition 3.20)—so a player is sometimes unable to make a move. The game ends when a player cannot make a move or when a leaf is reached; that is, the game ends when a play of A has been constructed. The first player wins iff a leaf matching in A is reached or the second player is unable to make a move, and the second player wins iff a leaf mismatching in A is reached or the first player is unable to move.

A strategy tells the first player how he should move on those positive subformula instances in which a certain move is not forced upon him by the S-admissibility constraint. Similarly, a counter-strategy tells the second player how he should move at those negative subformula instances where his move is not forced by S-admissibility. Thus a winning strategy on A is just a strategy that guarantees the first player a win in the LP Verification Game on A. Likewise, a winning counter-strategy on A guarantees the second player a win in this game.

We will use the existence of a winning strategy to define a notion of truth for the formula A: call A *true* exactly when the first player ("True") always has a winning strategy in the LP Verification Game on A. Similarly, A is called *false* exactly when the second player ("False") always has a winning counter-strategy in this game. The remainder of this section sets out the formal details of these definitions and shows that they provide a well-behaved notion of truth for LP formulas. In particular, we now show that no formula is both true and false.

Lemma 4.4 (Win-Loss Lemma)**.** Let V be a valuation set and S be a strategy map.

- If A^* is a winning strategy on A, then no counter-strategy A_* on A is winning.
- If B_* is a winning counter-strategy on B, then no strategy B^* on B is winning.

Proof. Assume that A^* is a winning strategy on A. Given a counter-strategy A_* on A, the Play Lemma (Lemma 3.21) then yields a unique play τ of $\mathcal{T}(A)$ that is in accordance with

A^* and with A_*. It follows from the meaning of a winning strategy that τ ends either on a negative non-leaf or else on a leaf matching in A, so A_* is not a winning counter-strategy on A. This completes the proof of the first item. The second item is proved similarly. □

So while no formula may be both true and false, it might be the case that some formula might not be assigned a truth value at all. The next lemma eliminates this possibility by showing that every formula is indeed assigned a truth value.

Lemma 4.5 (Determinacy Lemma)**.** Given a valuation set V and a strategy map S, each formula A has either a winning strategy or else a winning counter-strategy.

Proof. This lemma is a special case of the Gale-Stewart Theorem [7] (see Hodges' exposition [10]), but it is useful for us to prove the result directly. Let us proceed.

It follows from the Win-Loss Lemma (Lemma 4.4) that a formula A cannot have both a winning strategy and a winning counter-strategy, so we only need show that each formula A has at least one of the two. Accordingly, we prove by induction on formula construction that each formula has a winning strategy or a winning counter-strategy.

1. Base case: the formula is an atom p.

   A strategy on p is winning iff $p \in V$, and a counter-strategy on p is winning iff $p \notin V$. Since we have either that $p \in V$ or that $p \notin V$, the result follows.

2. Inductive case: the formula is an implication $B \supset C$.

   Let us first prove two claims.

   (a) Claim: if there is a winning counter-strategy on B or a winning strategy on C, then there is a winning strategy on $B \supset C$.

      i. Case: B_* is a winning counter-strategy on B.
      Let s^* be the strategy that chooses the immediate subformula instance B of $B \supset C$ and thereafter acts according to B_*; that is, let $s^*(B \supset C) = B$, let $s^* \restriction B = B_*$, and let $s^* \restriction C$ be an arbitrary strategy on C. If τ is a play of $B \supset C$ in accordance with s^*, it follows that $\tau \restriction B$ is a play of B in accordance with B_*. Since B_* is a winning counter-strategy on B, we have that $\tau \restriction B$ ends either on a positive term-labeled subformula instance of B or else on a leaf mismatching in B. But a negative subformula instance of B is a positive subformula instance of $B \supset C$, and so it follows that τ ends either on a negative term-labeled subformula instance of $B \supset C$ or else on a leaf matching in $B \supset C$. Since τ was an arbitrary play of $B \supset C$ in accordance with s^*, it follows that s^* is a winning strategy on $B \supset C$.

      ii. Case: C^* is a winning strategy on C.
      Let s^* be the strategy such that $s^*(B \supset C) = C$, that $s^* \restriction C = C^*$, and that $s^* \restriction B$ is an arbitrary counter-strategy on B. Since C^* is a winning strategy on C, it follows that s^* is a winning strategy on $B \supset C$ by an argument like that in the previous case.

(b) Claim: if there is a winning strategy on B and a winning counter-strategy on C, then there is a winning counter-strategy on $B \supset C$.

Suppose that B^* is a winning strategy on B and C_* is a winning counter-strategy on C. Define the counter-strategy s_* on $B \supset C$ by setting $s_* \restriction B = B^*$ and $s_* \restriction C = C_*$. By an argument analogous to those in the cases of the previous claim, it follows that s_* is a winning counter-strategy on $B \supset C$.

Now that these claims have been proven, we apply the induction hypothesis: there is either a winning strategy on B or else a winning counter-strategy on B, and there is either a winning strategy on C or else a winning counter-strategy on C. It then follows from the claims that there is a winning strategy on $B \supset C$ or a winning counter-strategy on $B \supset C$, which is what we wished to show.

3. Inductive case: the formula is the term-labeled formula $t\!:\!B$.

We consider three cases.

(a) Case: $S(t, B)\!\downarrow$ and $S(t, B)$ is a winning strategy on B.

Let s^* be the strategy on $t\!:\!B$ given by $s^*(t\!:\!B) = B$. Since $S(t, B)$ is a winning strategy on B, it follows that s^* is a winning strategy on $t\!:\!B$.

(b) Case: $S(t, B)\!\downarrow$ but $S(t, B)$ is not a winning strategy on B.

If $S(t, B)$ is not a winning strategy on B, then there is a play τ of $t\!:\!B$ such that the play $\tau \restriction B$ of B is in accordance with $S(t, B)$ and τ ends either on a positive term-labeled formula or else on a leaf mismatching in $t\!:\!B$. Let s_* be the counter-strategy on $t\!:\!B$ that maps each response point n in τ to the child of n in τ and otherwise maps each response point n not in τ to an arbitrary child of n. It follows that if a play τ' of $t\!:\!B$ is in accordance with s_*, then $\tau' = \tau$, and so s_* is a winning counter-strategy on $t\!:\!B$.

(c) Case: $S(t, B)\!\uparrow$.

In this case, every play of $t\!:\!B$ ends on the positive subformula instance $t\!:\!B$ and thus any counter-strategy on $t\!:\!B$ is winning.

Since these cases exhaust all possibilities, there is a winning strategy on $t\!:\!B$ or a winning counter-strategy on $t\!:\!B$, which is what we wished to show. □

It was our intention to assign to the formula $t\!:\!A$ the the informal reading "t is a winning strategy on A." The following definition classifies the strategy maps that make good on this intention.

Definition 4.6. A strategy map S is *good* for a valuation set V iff $S(t, A)\!\downarrow$ implies that $S(t, A)$ is a winning strategy on A.

To complete our definition of truth, we want to make sure that there is a good strategy map that respects the rule of Constant Necessitation. This is the task of the next theorem.

Theorem 4.7. For each valuation set V, there is an axiomatically saturated strategy map S that is good for V.

Proof. In this proof, a *set* is just a set of formulas. The *conjunction* of a finite set is the conjunction whose conjuncts are the members of that finite set. Call a set *consistent* iff no conjunction of a finite subset implies $\bot$ in the theory LP. A set is *inconsistent* iff it is not consistent. A set is called *maximal consistent* iff it is consistent and the addition of any formula not already in the set would make the resulting set inconsistent. Using a Lindenbaum argument, any consistent set may be extended to a maximal consistent set.

Let $V' := V \cup \{p \supset \bot \mid p \notin V\}$. V' is consistent and so it may be extended to a maximal consistent set T. Define a map W that takes each non-atom A to an immediate subformula instance of A as follows:

$$W(B \supset C) := \begin{cases} B & \text{if } B \notin T, \\ C & \text{otherwise;} \end{cases}$$

$$W(t{:}B) := B \ .$$

For each formula A, let A^* be the strategy on A that follows W; that is, for each choice point B of A, set $A^*(B) := W(B)$. Note that for each atom p, the strategy p^* on p is just the trivial strategy. Observe that if B is a choice point of the formula A, then $A^* \restriction B = B^*$.

We now define a possible strategy map S: the domain of S consists of all term-formula pairs (t, A) such that $t{:}A \in T$, and, for such a term-formula pair, we set $S(t, A) := A^*$. It follows from the maximal consistency of T that S is axiomatically saturated.

We now show that S is in fact a strategy map.

1. *Product.* Suppose that $S(u, A \supset B)\!\downarrow$ and $S(v, A)\!\downarrow$.

   (a) We show $S(u \cdot v, B) = S(u, A \supset B) \restriction B$.

   By definition of S, we have $u{:}(A \supset B), v{:}A \in T$ and thus that $(u \cdot v){:}B \in T$ by **LP1** and the maximal consistency of T. It follows from the definition of S that $S(u \cdot v, B)\!\downarrow$ and, in particular, that $S(u \cdot v, B) = B^*$. Since B is a choice point of $A \supset B$, we have $B^* = (A \supset B)^* \restriction B$, where the right-hand side of this equation is just $S(u, A \supset B) \restriction B$ by our definition of S. The result follows.

   (b) We show $S(u, C \supset B)\!\downarrow$ and $S(v, C)\!\downarrow$ implies $S(u, C \supset B) \restriction B = S(u, A \supset B) \restriction B$.

   By the definition of S, we have $S(u, C \supset B) \restriction B = (C \supset B)^* \restriction B$ and $S(u, A \supset B) \restriction B = (A \supset B)^* \restriction B$. Since B is a choice point of both $A \supset B$ and $C \supset B$, we have $B^* = (A \supset B)^* \restriction B$ and $B^* = (C \supset B)^* \restriction B$. The result follows.

2. *Proof Checker.* Suppose that $S(t, A)\!\downarrow$.

   By the definition of S, this means $t{:}A \in T$ and thus $!t{:}(t{:}A) \in T$ by **LP2** and the maximal consistency of T. Thus we have $S(!t, t{:}A)\!\downarrow$ by our definition of S.

3. *Sum.*

(a) We show $S(t,A)\!\downarrow$ implies $S(t+u,A) = S(t,A)$.
Suppose $S(t,A)\!\downarrow$. By the definition of S, this means $t\!:\!A \in T$ and thus $(t+u)\!:\!A \in T$ by **LP3** and the maximal consistency of T. It then follows from the definition of S both that $S(t+u,A) = A^*$ and that $S(t,A) = A^*$, so the result follows.

(b) We show $S(t,A)\!\uparrow$ and $S(u,A)\!\downarrow$ implies $S(t+u,A) = S(u,A)$.
This follows by an argument similar to the previous case.

So we have shown that S is indeed a strategy map.

What remains is to show that S is good for V. To prove this, we first assume what we call the *WS Property*: $A \in T$ implies A^* is a winning strategy on A, and $A \notin T$ implies there is a winning counter-strategy on A. We will prove the WS Property in a moment, but let us first show that S is good for V. That is, we prove $S(t,A)\!\downarrow$ implies that $S(t,A)$ is a winning strategy on A.

So suppose that $S(t,A)\!\downarrow$. By our definition of S, it follows that $t\!:\!A \in T$ and thus that $A \in T$ by **LP4** and the maximal consistency of T. By the WS Property, it follows that A^* is a winning strategy on A. By our definition of S, we have $S(t,A) = A^*$. We have thus shown that S is good for V.

We now complete the proof by proving the WS Property: $A \in T$ implies A^* is a winning strategy on A, and $A \notin T$ implies there is a winning counter-strategy on A. We prove this by induction on the construction of A.

- Base case: A is an atom p.
  - Suppose $p \in T$.
    T was constructed as a maximal consistent extension of the set
    $$V' := V \cup \{p \supset \bot \mid p \notin V\} \ ,$$
    and so $p \in T$ implies $p \in V$ by the consistency of T. So $p \in V$, and thus the trivial strategy p^* is winning on p.
  - Suppose $p \notin T$.
    Since T is a maximal consistent extension of V', we have that $p \notin T$ implies $p \notin V$. Thus the trivial counter-strategy is winning on p.
- Inductive case: A is the formula $B \supset C$.
  - Suppose $B \supset C \in T$.
    By the maximal consistency of T, we have that $B \notin T$ or $C \in T$. We handle each case in turn.
    - Case: $B \notin T$.
      By the inductive hypothesis, there is a winning counter-strategy B_* on B. By Claim 2(a)i of the proof of the Determinacy Lemma (Lemma 4.5) and the definition of W, it follows that $(B \supset C)^*$ is a winning strategy on $B \supset C$.

* Case: $C \in T$.
  By the inductive hypothesis, C^* is a winning strategy on C. By Claim 2(a)ii of the proof of the Determinacy Lemma and the definition of W, it follows that $(B \supset C)^*$ is a winning strategy on $B \supset C$.

– Suppose $B \supset C \notin T$.
  By the maximal consistency of T, we have that $B \in T$ and that $C \notin T$. By the inductive hypothesis, B^* is a winning strategy on B and there is a winning counter-strategy C_* on C. By Claim 2b of the proof of the Determinacy Lemma, there is a winning counter-strategy on $B \supset C$.

- Inductive case: A is the formula $t\!:\!B$.

  – Suppose $t\!:\!B \in T$.
    By the **LP4** and the maximal consistency of T, it follows that $B \in T$. Applying the inductive hypothesis, B^* is a winning strategy on T. By our definition of S, we have $S(t, B) = B^*$. By Case 3a of the proof of the Determinacy Lemma and the definition of W, it follows that $(t\!:\!B)^*$ is a winning strategy on $t\!:\!B$.

  – Suppose $t\!:\!B \notin T$.
    By the definition of S, we have that $S(t, B)\!\uparrow$. Applying Case 3c of the proof of the Determinacy Lemma, there is a winning counter-strategy on $t\!:\!B$. □

Finally, we may identify the true formulas as those for which there exists a winning strategy in the LP Verification Game.

Definition 4.8 (Truth)**.** Let V be a valuation set and A be a formula.

- A is *true in* V iff for every axiomatically saturated strategy map S that is good for V, there is a winning strategy on A.

- A is *false in* V iff A is not true in V.

To say a formula B is *true* means that B is true in every valuation set, and to say that B is *false* means that B is not true.

The next lemma shows that our definition of truth is compositional in a Tarskian sense.

Lemma 4.9 (Compositionality Lemma)**.** Let V be a valuation set.

1. An atom p is true in V iff $p \in V$.

2. An implication $A \supset B$ is true in V iff A is false in V or B is true in V.

3. A term-labeled formula $t\!:\!A$ is true in V iff for each axiomatically saturated strategy map S that is good for V, we have that $S(t, A)\!\downarrow$ and that $S(t, A)$ is a winning strategy on A.

Proof. We prove each item in turn.

1. This was shown in the base case, Item 1, of the proof of the Determinacy Lemma (Lemma 4.5).

2. The right-to-left direction is Claim 2a from the proof of the Determinacy Lemma. We now argue the left-to-right direction. First, if it is not the case that A is false in V or B is true in V, then it follows from the Determinacy Lemma and the Win-Loss Lemma (Lemma 4.4) that A is true in V and B is false in V. Applying Claim 2b from the proof of the Determinacy Lemma, $A \supset B$ is false in V. By the Determinacy and Win-Loss Lemmas, this is equivalent to saying that $A \supset B$ is not true in V. This completes the left-to-right direction.

3. The right-to-left direction is Case 3a from the proof of the Determinacy Lemma. The left-to-right direction is Cases 3b and 3c from the proof of the Determinacy Lemma. □

Finally, we show that our notion of truth exactly captures the provable LP formulas.

Theorem 4.10. The true formulas are exactly those formulas provable in LP.

Proof. If A is not provable in LP, then we may follow the proof of Theorem 4.7 to construct a valuation set V and an axiomatically saturated strategy map S good for V such that A is false in V. In particular, the non-provability of A implies that $\{A \supset \bot\}$ is consistent and so this set may be extended to a maximal consistent set T. We then set $V := \{p \mid p \in T\}$. Given this V and this T, the construction in the proof of Theorem 4.7 gives us an axiomatically saturated strategy map good for V and, as $A \supset \bot \in T$ and the maximal consistency of T implies $A \notin T$, the WS Property ensures that there is a winning counter-strategy on A.

So what remains is to show that each LP-provable formula is true. We do this by an induction on the length of derivation in LP. Proceeding, let V be an arbitrary valuation set.

- Base case: each LP axiom is true.
  - **LP0.** Classical propositional logic

    We use the following axiomatization of classical propositional logic [8].

    1. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$
    2. $A \supset (B \supset A)$
    3. $((A \supset \bot) \supset \bot) \supset A$

    For each of the axioms, it follows from the Determinacy Lemma (Lemma 4.5) that each of the subformula instances A, B, and C is itself true in V or false in V. Note that the trivial counter-strategy is a winning strategy on $\bot$, and therefore $\bot$ is false. Applying the Compositionality Lemma (Lemma 4.9), it is easy to see that each of these axioms is true in V.

– **LP1.** $t{:}(A \supset B) \supset (s{:}A \supset (t \cdot s){:}B)$

By the Determinacy and Compositionality Lemmas, we may assume that $t{:}(A \supset B)$ and $s{:}A$ are both true in V. Applying the Compositionality Lemma, if S is an axiomatically saturated strategy map that is good for V, then we have that $S(t, A \supset B)\downarrow$, that $S(s, A)\downarrow$, that $S(t, A \supset B)$ is a winning strategy on $A \supset B$, and that $S(s, A)$ is a winning strategy on A. Were it the case that the value of $S(t, A \supset B)$ on $A \supset B$ is the immediate subformula instance A of $A \supset B$, then $S(t, A \supset B) \restriction A$ is a winning counter-strategy on A, and the Win-Loss Lemma implies that this contradicts the fact that A has a winning strategy. Therefore the value of $S(t, A \supset B)$ on $A \supset B$ is the immediate subformula instance B of $A \supset B$, and so $S(t, A \supset B) \restriction B$ is a winning strategy on B. The Product Property of strategy maps ensures that $S(t \cdot s, B) = S(t, A \supset B) \restriction B$, so $S(t \cdot s, B)$ is a winning strategy on B. Since S was an arbitrary axiomatically saturated strategy map good for V, it follows from the Compositionality Lemma that $(t \cdot s){:}B$ is true in V. Applying the Compositionality Lemma again, **LP1** is true in V.

– **LP2.** $t{:}A \supset !t{:}(t{:}A)$

By the Determinacy and Compositionality Lemmas, we may assume that $t{:}A$ is true in V. By the Compositionality Lemma, if S is an axiomatically saturated strategy map that is good for V, then we have that $S(t, A)\downarrow$ and that $S(t, A)$ is a winning strategy on A. The Proof Checker property of strategy maps ensures that $S(!t, t{:}A)\downarrow$, and so a play of $t{:}A$ in accordance with $S(!t, t{:}A)$ is just a play of $t{:}A$. But if τ is a play of $t{:}A$, then $\tau \restriction A$ is a play in accordance with the winning strategy $S(t, A)$ on A, so the end of $\tau \restriction A$ is either a negative non-leaf or else a leaf matching in A. Since A is a positive subformula instance of $t{:}A$ and τ was an arbitrary play of $t{:}A$ in accordance with $S(!t, t{:}A)$, it follows that $S(!t, t{:}A)$ is a winning strategy on $t{:}A$. Since S was an arbitrary axiomatically saturated strategy map good for V, the Compositionality Lemma then implies that $!t{:}(t{:}A)$ is true in V. Applying the Compositionality Lemma again, **LP2** is true in V.

– **LP3.** $t{:}A \supset (t + s){:}A$, $t{:}A \supset (s + t){:}A$

We show only that the left **LP3** axiom is true in V, the proof for the right **LP3** axiom is similar. By the Determinacy and Compositionality Lemmas, we may assume that $t{:}A$ is true in V. If S is an axiomatically saturated strategy map good for V, the Compositionality Lemma implies that $S(t, A)\downarrow$ and that $S(t, A)$ is a winning strategy on A. Applying the Sum Property of strategy maps, $S(t{+}s, A) = S(t, A)$ and so $S(t{+}s, A)$ is also a winning strategy on A. Since S was an arbitrary axiomatically saturated strategy map good for V, the Compositionality Lemma implies that $(t + s){:}A$ is true in V, so the Compositionality Lemma then implies that **LP4** is true in V.

– **LP4.** $t{:}A \supset A$

By the Determinacy and Compositionality Lemmas, we may assume that $t{:}A$

is true in V. Applying the Compositionality Lemma, if S is an axiomatically saturated strategy map good for V, then $S(t, A)\downarrow$ and $S(t, A)$ is a winning strategy on A. A is thus true in V, and so it follows from the Compositionality Lemma that **LP4** is also true in V.

- Inductive case: truth is closed under the rule of Modus Ponens.

  If A and $A \supset B$ are both true in V, then it follows from the Compositionality Lemma, the Win-Loss Lemma, and the Determinacy Lemma that B is true in V.

- Inductive case: truth is closed under the rule of Constant Necessitation.

  Suppose that S is an axiomatically saturated strategy map good for V, that c is a constant, and that A is an axiom. Since S is axiomatically saturated, we have that $S(c, A)\downarrow$, so it follows that $S(c, A)$ is a winning strategy on A by our assumption that S is good for V. Since the only choice point of $c\!:\!A$ is the root of $\mathcal{T}(c\!:\!A)$ and $S(c, A)$ is a winning strategy on A, it then follows that the unique strategy on $c\!:\!A$ is a winning strategy on $c\!:\!A$. As S was an arbitrary axiomatically saturated strategy map good for V, the Compositionality Lemma implies that $c\!:\!A$ is true in V.

Since V was arbitrary, we have shown that each LP theorem is true. □

5 Embeddings and the Internalization Theorem

The concept of *extensive game with perfect information* was introduced by von Neumann and Morgenstern [17]. Following some of Sevenster's notation and naming conventions [16], we define an *extensive game with perfect information* as a tuple $G = (N, \Sigma, \mathcal{H}, p, \{u_i\}_{i\in N})$, where

- N is a nonempty set whose elements are called *players*.

- Σ is a nonempty set.

- $\mathcal{H}$ is a nonempty prefix-closed set of strings over the alphabet Σ such that there is a unique shortest string r in $\mathcal{H}$. (r is typically the empty string ϵ.)

  Members of $\mathcal{H}$ are called *histories*. A string h_1 is a *prefix* of a string h iff there is a string h_2, which may be ϵ, such that $h = h_1h_2$. To say that $\mathcal{H}$ is *prefix-closed* means that if h' is a non-ϵ prefix of h and $h \in \mathcal{H}$, then $h' \in \mathcal{H}$. A string h_2 is a *suffix* of a string h iff there is a string h_1, which may be ϵ, such that $h = h_1h_2$. If h is a history, $a \in \Sigma$, and ha is a history, then ha is called a *move at* h. Notation: $\mathcal{H}_t$ is the subset of $\mathcal{H}$ containing all *terminal* histories, which are those histories h such that there is no move at h. Also, Σ^* is the set of all strings over the alphabet Σ, including ϵ.

- p is a function $(\mathcal{H} - \mathcal{H}_t) \rightarrow N$ that maps each non-terminal history to a player. To say that the history h is a *player* i *position* means that $p(h) = i$.

- u_i is a function $\mathcal{H}_t \to \mathbb{R}$ that maps each terminal history to a payoff for player i.

G is called *finite* iff the sets N, Σ, and $\mathcal{H}$ are all finite. G is *two-player* iff $N = \{1, 2\}$ and $p(r) = 1$ if $r \notin \mathcal{H}_t$. If G is two-player, then G is *win-loss* exactly when for each $h \in \mathcal{H}_t$, we have that $u_1(h) = -u_2(h)$ and that $|u_1(h)| = |u_2(h)| = 1$. All of the extensive games with perfect information we discuss will be finite, two-player, and win-loss, so we make the following definition.

Definition 5.1. A *verification-like extensive game* is an extensive game with perfect information that is finite, two-player, and win-loss.

In verification-like extensive games, players take turns at each non-terminal history h, with player $p(h)$ choosing some move at h. Once a terminal history is reached, the game is over, and the winner is the player whose payoff at that terminal history is 1; the other player loses.

A *strategy* in a verification-like extensive game G is a function that maps each player 1 position h to a move at h. A history h is *in accordance* with a strategy s^* iff for each player 1 position h' such that h' is a prefix of h with $h' \neq h$, we have that $s^*(h')$ is also a prefix of h. To say a strategy s^* is *winning* means that for every terminal history h in accordance with s^*, we have $u_1(h) = 1$.

We define the notion of *counter-strategy* in G as we just did for a strategy in G, except that the references to player 1 are all replaced by player 2. The meaning of a history *in accordance* with a counter-strategy is given in the same way. A counter-strategy s_* is *winning* iff for every terminal history h in accordance with s_*, we have $u_1(h) \neq 1$.

Since verification-like extensive games are finite, it follows from the Gale-Stewart Theorem that each verification-like extensive game has either a winning strategy or a winning counter-strategy (and not both).[2]

If we fix a formula A, a valuation set V, and an axiomatically saturated strategy map S that is good for V, then the LP Verification Game on A with V and S may be viewed as a verification-like extensive game. However, this statement is also true in the other direction.

Proposition 5.2. For each verification-like extensive game

$$G = (N, \Sigma, \mathcal{H}, p, \{u_i\}_{i \in N}) \ ,$$

there is a formula A such that A is true iff there is a winning strategy in G.

Proof. This proof argues that the game tree of G can be faithfully represented by a formula in the language of propositional logic. For transparency of the argument, we will perform a few winning-strategy–preserving operations that modify G, allowing us to assume that G is in a desirable form. To say that these operations are *winning-strategy–preserving* means that there is a winning strategy on G iff there is a winning strategy on the verification-like extensive game that results by applying these operations in order on G. We now describe these operations and argue that each of them is winning-strategy–preserving.

[2]See Hodges' exposition [10] of the Gale-Stewart Theorem [7].

- Collapse tails until each terminal history has a tail of length 1.

  The *tail* of a terminal history h is the longest suffix h_2 of h such that, if $h = h_1 h_2$, then for each non-ϵ prefix h' of h_2, there is at most one move to make at $h_1 h'$. The reason h_2 is called a tail: ordering h_2's non-ϵ prefixes $h^{(1)}, h^{(2)}, h^{(3)}, \ldots, h^{(n)}, h_2$ by increasing length, there is at exactly one move to make at each non-terminal history in the sequence

  $$h_1 h^{(1)}, h_1 h^{(2)}, h_1 h^{(3)}, \ldots, h_1 h^{(n)}, h_1 h_2 \ ,$$

  so this sequence traces out a tail-like path.

  If $a h_2$ is a tail of the terminal history $h_1 a h_2$, with $a \in \Sigma$, then to *collapse* $a h_2$ is to define the verification-like extensive game

  $$G' = (N, \Sigma, \mathcal{H}', p', \{u'_i\}_{i \in N}) :$$

  - $\mathcal{H}' := \{h \in \mathcal{H} \mid (\forall h' \in \Sigma^*)(h \neq h_1 a h')\} \cup \{h_1 a\}$;
  - $p'(h) := p(h)$ if $h \in \mathcal{H}$ and $h \notin (\mathcal{H}_t \cap \mathcal{H}'_t)$;
  - $u'_i(h) := \begin{cases} u_i(h) & \text{if } h \in (\mathcal{H}_t \cap \mathcal{H}'_t), \\ u_i(h_1 a h_2) & \text{if } h = h_1 a. \end{cases}$

  In G', the tail a of terminal history $h_1 a$ is of length 1. Further, a strategy s^* in G induces a strategy s'^* in G' that takes each player 1 position $h \in \mathcal{H}'$ to a move at h:

  $$s'^*(h) := s^*(h) \ .$$

  Since $a h_2$ is the tail of $h_1 a h_2$ in G, the terminal history $h_1 a h_2$ in G is in accordance with a strategy s^* in G iff the terminal history $h_1 a$ in G' is in accordance with the strategy s'^* in G' induced by s^*. It follows that s^* is a winning strategy in G iff s'^* is a winning strategy in G'. We may therefore collapse tails one by one until each terminal history has a tail of length 1, after which we are assured that there is a winning strategy in the resulting verification-like extensive game iff there was a winning strategy in the original verification-like extensive game.

- Remove all only-child double-moves from G.

  A *double-move* is a non-terminal history h' that is a move at another history h with $p(h) = p(h')$; h' is said to be a *double-move at* h. A *only-child double-move* is a double-move h' at h such that h' is the unique move at h. h' is called an only-child double-move because player $p(h)$ has only the one move h' at h and h' is a double-move at h.

  An only-child double-move can be removed from the game G in the following way. First, suppose $h_1 a$ is an only-child double-move, where $a \in \Sigma$. We define the verification-like extensive game

  $$G' = (N, \Sigma, \mathcal{H}', p', \{u'_i\}_{i \in N}) :$$

– $\mathcal{H}' := \{h \in \mathcal{H} \mid (\forall h' \in \Sigma^*)(h \neq h_1 a h')\} \cup \{h_1 h' \mid h_1 a h' \in \mathcal{H}\}$;

– $p'(h) := \begin{cases} p(h) & \text{if } h \in \mathcal{H} \text{ and } h \notin (\mathcal{H}_t \cap \mathcal{H}'_t), \\ p(h_1 a h') & \text{if } h = h_1 h' \text{ and } h_1 a h' \in (\mathcal{H} - \mathcal{H}_t); \end{cases}$

– $u'_i(h) := \begin{cases} u_i(h) & \text{if } h \in (\mathcal{H}_t \cap \mathcal{H}'_t), \\ u_i(h_1 a h') & \text{if } h = h_1 h' \text{ and } h_1 a h' \in \mathcal{H}_t. \end{cases}$

G' has one less only-child double-move than G does. Further, a strategy s^* in G induces a strategy s'^* in G' that takes each player 1 position $h \in \mathcal{H}'$ to a move at h:

$$s'^*(h) := \begin{cases} s^*(h) & \text{if } h \neq h_1, \\ h_1 h_2 & \text{if } h = h_1 h', \, h_1 a h' \in \mathcal{H}, \text{ and } s^*(h_1 a h') = h_1 a h_2. \end{cases}$$

Since $h_1 a$ is an only-child double-move in G, a history $h_1 a h'$ is in accordance with a strategy s^* in G iff the history $h_1 h'$ is in accordance with the strategy s'^* in G' induced by s^*. It follows that s^* is a winning strategy in G iff s'^* is a winning strategy in G'. So we may remove all only-child double-moves from G in this way, one by one, and we are assured the existence of a winning strategy in the resulting verification-like extensive game iff there was a winning strategy in the original verification-like extensive game.

Since we removed only-child double-moves from G after collapsing tails until all tails are of length 1, the verification-like extensive game resulting from these two operations contains no only-child double-moves and has all its tails of length 1.

- Convert each three-plus fork to a two-fork.

  A history h is called a *three-plus fork* iff there are at least three moves at h, and h is called a *two-fork* iff there are exactly two moves at h. If $h_1 a$ and $h_1 b$ are both moves at the three-plus fork h_1, then we can reduce by one the number of moves at h_1 by defining the verification-like extensive game

  $$G' = (N, \Sigma', \mathcal{H}', p', \{u'_i\}_{i \in N}) :$$

  – For some $c \notin \Sigma$, let $\Sigma' := \Sigma \cup \{c\}$;

  – $\mathcal{H}' := \{h \in \mathcal{H} \mid (\forall h' \in \Sigma^*)(h \neq h_1 a h' \text{ and } h \neq h_1 b h')\} \cup \{h_1 c\} \cup \{h_1 c a h' \mid h_1 a h' \in \mathcal{H}\} \cup \{h_1 c b h' \mid h_1 b h' \in \mathcal{H}\}$;

  – $p'(h) := \begin{cases} p(h) & \text{if } h \in \mathcal{H} \text{ and } h \notin (\mathcal{H}_t \cap \mathcal{H}'_t), \\ p(h_1) & \text{if } h = h_1 c, \\ p(h_1 a h') & \text{if } h = h_1 c a h' \text{ and } h_1 a h' \in (\mathcal{H} - \mathcal{H}_t), \\ p(h_1 b h') & \text{if } h = h_1 c b h' \text{ and } h_1 b h' \in (\mathcal{H} - \mathcal{H}_t); \end{cases}$

  – $u'_i(h) := \begin{cases} u_i(h) & \text{if } h \in (\mathcal{H}_t \cap \mathcal{H}'_t), \\ u_i(h_1 a h') & \text{if } h = h_1 c a h' \text{ and } h_1 a h' \in \mathcal{H}_t, \\ u_i(h_1 b h') & \text{if } h = h_1 c b h' \text{ and } h_1 b h' \in \mathcal{H}_t. \end{cases}$

There is one less move at history h_1 in G' than there is at h_1 in G. Further, a strategy s^* in G induces a strategy s'^* in G' that takes each player 1 position $h \in \mathcal{H}'$ to a move at h:

$$s'^*(h) := \begin{cases} s^*(h) & \text{if } s^*(h) \neq h_1a \text{ and } s^*(h) \neq h_1b, \\ h_1c & \text{if } s^*(h) = h_1a \text{ or } s^*(h) = h_1b, \\ h_1ca & \text{if } h = h_1c \text{ and } s^*(h_1) = h_1a, \\ h_1cb & \text{if } h = h_1c \text{ and } s^*(h_1) = h_1b, \\ h_1cah_2 & \text{if } h = h_1cah' \text{ and } s^*(h_1ah') = h_1ah_2, \\ h_1cbh_2 & \text{if } h = h_1cbh' \text{ and } s^*(h_1bh') = h_1bh_2. \end{cases}$$

It follows from the construction of G' that the history h_1cah' in G' is in accordance with the strategy s'^* in G' induced by a strategy s^* in G iff the history h_1ah' in G is in accordance with s^*. The same result holds with respect to the history h_1cbh' in G' and the corresponding history h_1bh' in G. We thus have that s^* is a winning strategy in G iff s'^* is a winning strategy in G'. So, by repeatedly performing this operation on three-plus forks until no more three-plus forks remain, we produce a verification-like extensive game in which there exists a winning strategy iff there was a winning strategy in the original verification-like extensive game.

In performing this operation after the previous two, we made all tails of length 1, we then removed all only-child double-moves, and we then incrementally reduced the number of moves at three-plus forks until there were no more three-plus forks. The resulting verification-like extensive game thus has tails all of length 1, contains no only-child double-moves, and contains no three-plus forks. In fact, calling a history a *fork* iff there are at least two moves at that history, every fork in the resulting verification-like extensive game is a two-fork.

- Incrementally reduce the degree of each two-fork parity point until no two-fork is a parity point.

  A *parity point* is a history h_1 such that there is a non-terminal move h_2 at h_1 satisfying $p(h_2) \neq p(h_1)$. h_1 is called a parity point because the player-to-move can flip from $p(h_1)$ to the other of the two players. The *degree* of a history h is equal to the number of non-terminal moves h' at h such that $p(h') \neq p(h)$. Thus a history of nonzero degree is a parity point.

  Assume that h_1 is a two-fork parity point and that h_1a is a non-terminal history with $p(h_1a) \neq p(h_1)$, where $a \in \Sigma$. We decrease by one the degree of h_1 by adding a double-move between h_1 and h_1a. To do this, we define the verification-like extensive game

  $$G' = (N, \Sigma', \mathcal{H}', p', \{u'_i\}_{i \in N}) :$$

  - For some $b \notin \Sigma$, let $\Sigma' := \Sigma \cup \{b\}$;
  - $\mathcal{H}' := \{h \in \mathcal{H} \mid (\forall h' \in \Sigma^*)(h \neq h_1ah')\} \cup \{h_1b\} \cup \{h_1bah' \mid h_1ah' \in \mathcal{H}\}$;

– $p'(h) := \begin{cases} p(h) & \text{if } h \in \mathcal{H} \text{ and } h \notin (\mathcal{H}_t \cap \mathcal{H}_t), \\ p(h_1) & \text{if } h = h_1 b, \\ p(h_1 a h') & \text{if } h = h_1 b a h' \text{ and } h_1 a h' \in (\mathcal{H} - \mathcal{H}_t); \end{cases}$

– $u'_i(h) := \begin{cases} u_i(h) & \text{if } h \in (\mathcal{H}_t \cap \mathcal{H}'_t), \\ u_i(h_1 a h') & \text{if } h = h_1 b a h' \text{ and } h_1 a h' \in \mathcal{H}_t. \end{cases}$

The degree of h_1 in G' is one less than the degree of h_1 in G. Further, a strategy s^* in G induces a strategy s'^* in G' that takes each player 1 position $h \in \mathcal{H}'$ to a move at h:

$$s'^*(h) := \begin{cases} s^*(h) & \text{if } h \in \mathcal{H} \text{ and } h \neq h_1, \\ h_1 b & \text{if } h = h_1, \\ h_1 b a & \text{if } h = h_1 b, \\ h_1 b a h_2 & \text{if } h = h_1 b a h', \, h_1 a h' \in \mathcal{H}, \text{ and } s^*(h_1 a h') = h_1 a h_2. \end{cases}$$

It follows from our construction that the history $h_1 b a h'$ in G' is in accordance with the strategy s'^* in G' induced by a strategy s^* in G iff the history $h_1 a h'$ in G is in accordance with s^*. We thus have that s^* is a winning strategy in G iff s'^* is a winning strategy in G'. Proceeding in this way, we incrementally reduce the degree of each two-fork parity point until there are no more two-fork parity points, and we are assured that the resulting verification-like extensive game has a winning strategy iff the original extensive game had a winning strategy. Notice that since we only perform this operation at two-fork parity points, we never introduce only-child double-moves.

So after performing the operations above in order, we end up with a verification-like extensive game satisfying each of the following properties:

1. every fork is a two-fork,
2. every terminal history is a move at a two-fork (because all tails are of length 1),
3. no two-fork is a parity point, and
4. there are no only-child double-moves.

We may thus assume without loss of generality that G satisfies each of these properties.

We now proceed with our construction of the formula A, the formula in the statement of this proposition. First, call a terminal history *positive* in G iff it has an even number of ancestors that are parity points; call a terminal history is *negative* G iff it is not positive in G. Working our way backward from terminal histories, we define a function f that takes each history h to a formula $f(h)$ according to the following case analysis.

- If h is a positive terminal history, then

$$f(h) := \begin{cases} \top & \text{if } u_1(h) = 1, \\ \bot & \text{if } u_1(h) \neq 1. \end{cases}$$

- If h is a negative terminal history, then

$$f(h) := \begin{cases} \bot & \text{if } u_1(h) = 1, \\ \top & \text{if } u_1(h) \neq 1. \end{cases}$$

- If h is a parity point (and hence non-terminal), then

$$f(h) := \neg f(h') \ ,$$

  where h' is the unique move at h (uniqueness follows from Properties 1 and 3).

- If h is neither a parity point nor a terminal history, then

$$f(h) := f(h_1) \vee f(h_2) \ ,$$

  where h_1 and h_2 are the two moves at h. To see why there are exactly two moves at h, first notice that there is a move h_1 at h because h is non-terminal. Next observe that h_1 is a double-move because h is not a parity point. But Property 4 implies that h_1 cannot be an only-child double-move, so h is a fork. Applying Property 1, it follows that h is in fact a two-fork.

In this way, f maps each history to a formula, and we let the formula A in the statement of this proposition be the formula $f(r)$, where r is the unique shortest string in $\mathcal{H}$.

A is in the language of propositional logic and, further, it is *letterless*, meaning that each atomic formula appearing in A is a propositional constant. It thus follows that the winning (counter-)strategy on A in LP Verification is independent of any valuation set V and any axiomatically saturated strategy map S good for V. So, in fact, we do not need any of the special features of LP Verification—Propositional Verification will do—but we will need LP Verification later when we apply the Internalization Theorem, so we will nonetheless use LP Verification. What remains is to show that A is true iff there is a winning strategy in G.

For the moment, it will be convenient to assume that the language of propositional logic contains the symbol $\neg$ for negation and $\vee$ for disjunction and that formula formation (for propositional and LP formulas) is closed both under forming the negation $\neg B$ of a formula B and under forming the disjunction $B \vee C$ of two formulas B and C. The construction tree $\mathcal{T}(B)$ of an LP formula B is then built using the rules from before in addition to the following rules: a node labeled $\neg C$ has the left child C, and a node labeled $C \vee D$ has the right child C and the further-right child D. (The terminology "further-right" comes from the fact that we do not want to change our definition of positive subformula instance: a subformula instance C of B is *positive* iff the path in $\mathcal{T}(B)$ ending on C contains the left child of an even number of nodes.) That the work of the previous sections can be carried out in a language that also has negations and disjunctions ought to be apparent, so we proceed, letting $\mathcal{T}(A)$ be the construction tree for our formula A in this larger language.

We now argue that f is a *tree-isomorphism* between $\mathcal{H}$ and $\mathcal{T}(A)$, by which we mean that three conditions are satisfied. We list each condition along with an argument as to why that condition is true.

1. $f(r) = A$, where r is the unique shortest string in $\mathcal{H}$.

   This condition is satisfied by our definition of A.

2. If h' is a move at h, then $f(h')$ is an immediate subformula instance of $f(h)$.

   This follows by inspection of the way we defined f on each history h.

3. If B is an immediate subformula instance of $f(h)$, then there is a move h' at h such that $f(h') = B$.

   By our definition of f, that the formula $f(h)$ has immediate subformula instances implies that $f(h)$ is either a disjunction or a negation. If $f(h)$ is a disjunction, then $f(h)$ was defined by forming the disjunction of the formulas $f(h_1)$ and $f(h_2)$, where h_1 and h_2 are moves at h. If $f(h)$ is a negation, then $f(h)$ was defined by forming the negation of the formula $f(h_3)$, where h_3 is the unique move at h. So in either case, we have for each immediate subformula instance B of $f(h)$ a move h' at h such that $f(h') = B$.

We may thus view f as a bijection between histories and subformula instances of A, so it makes sense to talk of the history $f^{-1}(B)$ obtained as the inverse image of f on the subformula instance B of A. It follows that terminal histories are in one-to-one correspondence with leaves of $\mathcal{T}(A)$. Observe that for an immediate subformula instance C of a subformula instance B of A, we have that B and C are of opposite polarity only if $f^{-1}(B)$ is a parity point. We therefore have that an atomic subformula instance p of A is positive in A iff $f^{-1}(p)$ is positive in G. Looking back to how we specified the leaves in $\mathcal{T}(A)$ as images of terminal histories, it then follows that that player 1 wins at terminal history h in G iff the first player ("True") wins the play of A ending on $f(h)$ in the LP Verification Game on A. It then follows immediately that there is a winning strategy in G iff there is a winning strategy on A. Since A is letterless, it follows that there is a winning strategy in G iff A is true. □

So the proof of Proposition 5.2 defines a winning-strategy–preserving embedding that maps each verification-like extensive game G to a letterless propositional formula A_G. Now let us assume for a moment that there is a winning strategy in G, and so A_G is true and hence provable in LP. Applying the Internalization Theorem (Theorem 2.3), there is a term t containing no variables such that $t{:}A_G$ is also a theorem of LP. We now show how the inductive construction of t in the proof of the Internalization Theorem tells us how to build a winning strategy A^*_G on A_G, which we may view as the interpretation of t in the LP Verification Game.

- Suppose we used a constant c to internalize the axiom A.

  We proved in Theorem 4.7 that each axiom has a winning strategy, so choose a winning strategy A^* on A (any will do). Take A^* as the interpretation of c.

- Suppose we used the term $u \cdot v$ to internalize the conclusion B obtained from Modus Ponens on B and $C \supset B$, where we already constructed terms u and v such that both $u{:}(C \supset B)$ and $v{:}B$ are theorems.

We have already determined a winning strategy B^* on B that interprets v and a winning strategy $(C \supset B)^*$ on $C \supset B$ that interprets u. It follows from the Compositionality Lemma (Lemma 4.9) that $(C \supset B)^* \restriction B$ is a winning strategy on B, so take $(C \supset B)^* \restriction B$ as the interpretation of $u \cdot v$.

- Suppose we used the term $!c$ to internalize the conclusion $c{:}A$ obtained from Constant Necessitation on the axiom A.

  We have already determined a winning strategy A^* on A interpreting c. But notice that there is then only one winning strategy $(c{:}A)^*$ on $c{:}A$ because the only choice point of $c{:}A$ is the root of $\mathcal{T}(c{:}A)$. So $(c{:}A)^*$ simply maps $c{:}A$ to the immediate subformula instance A. Let this strategy interpret $!c$.

In this way, we obtain a winning strategy A^*_G on A_G that interprets t. However, since the game tree of G is essentially the same as the LP Verification game tree on A_G, the winning strategy A^*_G induces a winning strategy s^* on G. Here the word "essentially" is used to indicate that we manipulated G during the proof of Proposition 5.2 in our construction of A_G; however, these manipulations are invertible, which allows us to convert the winning strategy on the manipulated G to a winning strategy on the original, non-manipulated G. In this way, the Internalization Theorem provides a means of constructing winning strategies on winnable instances of verification-like extensive games.

5.1 Example: Obtaining Winning Strategies in Nim

The well-known game of Nim [5] may be viewed as a verification-like extensive game. The initial setup for a play of Nim consists of three separate piles of stones, each pile being of finite size. A move consists of selecting one pile and then removing any nonzero number of stones from that pile, leaving the other two piles alone. The removed stones are then discarded, as they are no longer part of the game. Two players take alternate turns moving in this way until all stones are removed. The player that picks up the last stone is the winner, and so the player that has no stone to pick up is the loser.

We represent a *Nim instance* as a triple (a, b, c) of non-negative integers. The Nim instance (a', b', c') stands in *one-move relation* to the Nim instance (a, b, c), written $(a, b, c) \rightarrow_1 (a', b', c')$, iff the primed triple is obtained from the unprimed triple by one legal move in the Nim game. Notice that no Nim instance stands in one-move relation to $(0, 0, 0)$ because $(0, 0, 0)$ marks the end of every play of Nim. We write $(a, b, c) \rightarrow_* (a', b', c')$ iff the Nim instance (a', b', c') may be obtained from the Nim instance (a, b, c) by zero or more legal moves in the Nim game, so $\rightarrow_*$ is just the reflexive-transitive closure of $\rightarrow_1$. A Nim instance (a, b, c) may be viewed as the verification-like extensive game $G(a, b, c) = (N, \Sigma, \mathcal{H}, p, \{u_i\}_{i \in N})$, where the components of this tuple are given as follows.

- $N := \{1, 2\}$.
- $\Sigma := \{(a', b', c') \mid (a, b, c) \rightarrow_* (a', b', c')\}$.

- $\mathcal{H}$ is defined as follows. First, set $\mathcal{H}_0 := \{(a, b, c)\}$. Once $\mathcal{H}_k$ is defined, define $\mathcal{H}_{k+1}$ as the set

$$\{h(a_1, b_1, c_1)(a', b', c') \mid h(a_1, b_1, c_1) \in \mathcal{H}_k \text{ and } (a_1, b_1, c_1) \rightarrow_1 (a', b', c')\} \ ,$$

  where h is a metavariable ranging over Σ^*. Finally, let $\mathcal{H} := \bigcup_{i=0}^{a+b+c} \mathcal{H}_i$.

- p is defined as follows. For each history h, define the *length* of h as the number of elements of Σ contained in h. Example: in $G(2, 1, 0)$, the non-terminal history

$$(2, 1, 0)(1, 1, 0)(1, 0, 0)$$

  has length three. Now if h is a non-terminal history, then set $p(h) := 1$ if the length of h is odd; otherwise, if the length of h is even, then set $p(h) := 2$.

- u_1 is defined as follows. For each terminal history h, set $u_1(h) := 1$ if the length of h is even; set $u_1(h) := -1$ if the length of h is odd.

- u_2 is defined as follows. For each terminal history h, set $u_2(h) := 1$ if the length of h is odd; set $u_2(h) := -1$ if the length of h is even.

Now that we have seen how a Nim instance may be viewed as a verification-like extensive game, we will work out an example that shows how to use the Internalization Theorem to construct a winning strategy on the winnable Nim instance $(2, 1, 0)$.

First observe that player 1 can guarantee himself a win in $G(2, 1, 0)$ iff his move at the history $(2, 1, 0)$ is $(2, 1, 0)(1, 1, 0)$. This move corresponds to the first player picking up one stone from the first pile. The second player then picks up one stone from either of the first or second piles, leaving the first player to pick up the last stone for the win.

So there is indeed a winning strategy in $G(2, 1, 0)$. We will now embed $G(2, 1, 0)$ into the LP Verification Game according to the construction in the proof of Proposition 5.2. We will then see how the Internalization Theorem allows us to extract the winning strategy in $G(2, 1, 0)$. Initially, $G(2, 1, 0)$ has the form of the tree in Figure 2. We then perform in order the operations on $G(2, 1, 0)$ described in the proof of Proposition 5.2. (As we proceed, we will use the terminology from this proof, though we also restate the definitions of the most important terminology. The reader may find it convenient to keep track of where we are in the bulleted list at the beginning of the proof of Proposition 5.2, which itemizes these operations in order and provides formal definitions of the terminology.)

The first operation calls for us to collapse tails until each terminal history's tail is of length 1. The result of this operation is the tree in Figure 3.

The next operation calls for us to remove each only-child double-move; however, the tree in Figure 3 does not have only-child double-moves, so this operation causes no change in the tree. Moving to the next operation, we are called to convert three-plus forks to two-forks. The result of this operation is the tree in Figure 4.

The next operation calls for us to ensure that no two-fork is a parity point. The result of this operation is the tree in Figure 5.

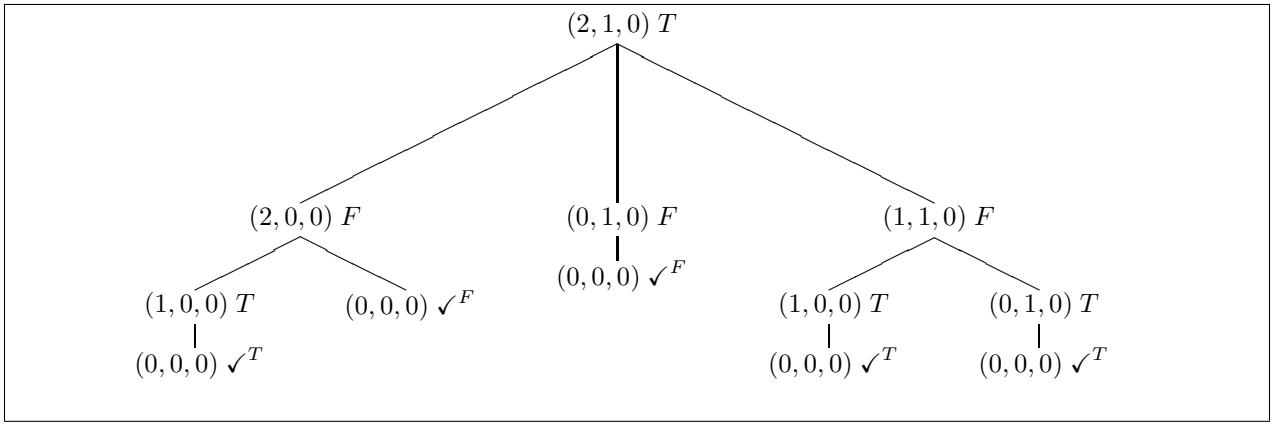

Figure 2. A tree representing $G(2, 1, 0)$. Positions at which the first player moves are marked by a T and positions at which the second player moves are marked by an F. The winning states for the first player are marked by the symbol $\checkmark^T$ and the winning states for the second player are marked by the symbol $\checkmark^F$.

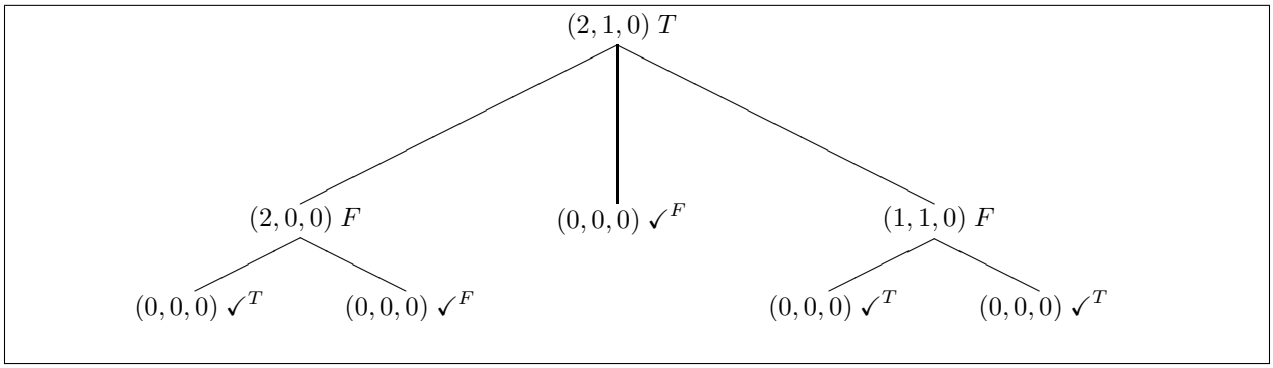

Figure 3. The tree of Figure 2 after tails are collapsed to length 1. (Here a *tail* is path that proceeds upward from a leaf and that is of maximal length subject to the restriction that no node appearing in the path has more than one child.)

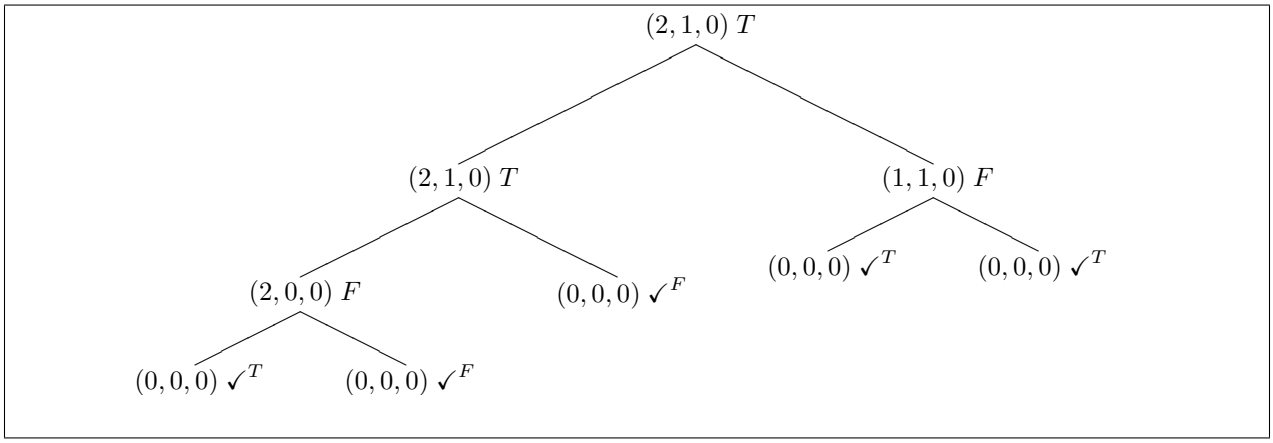

Figure 4. This tree results from converting three-plus forks to two-forks in the tree of Figure 3. To complete this operation, we introduced a redundant node southwest of the root of the tree in Figure 3. (Here a *fork* is a node with more than one child, a *three-plus fork* is a fork with at least three children, and a *two-fork* is a fork with exactly two children.)

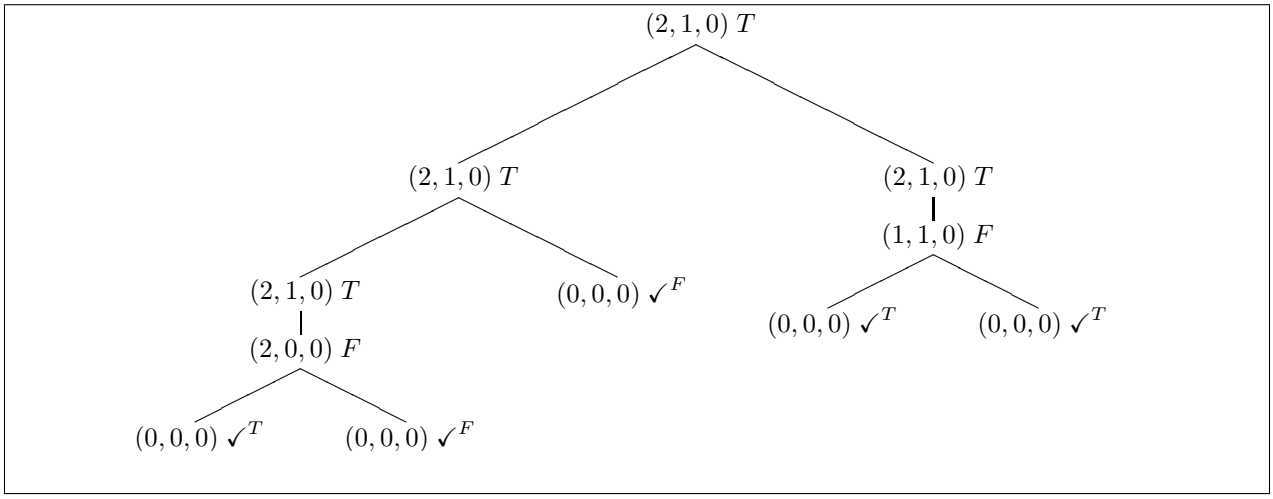

Figure 5. This tree results from introducing double-moves into the tree of Figure 4 so that no two-fork is a parity point. To complete this operation, we introduced one double-move southeast of the root and another double-move southwest of the southernmost $(2,1,0)$ position. (Here a *double-move* is a node whose parent has the same player-to-play, and a *parity point* is a node with a non-leaf child at which the player-to-play differs.)

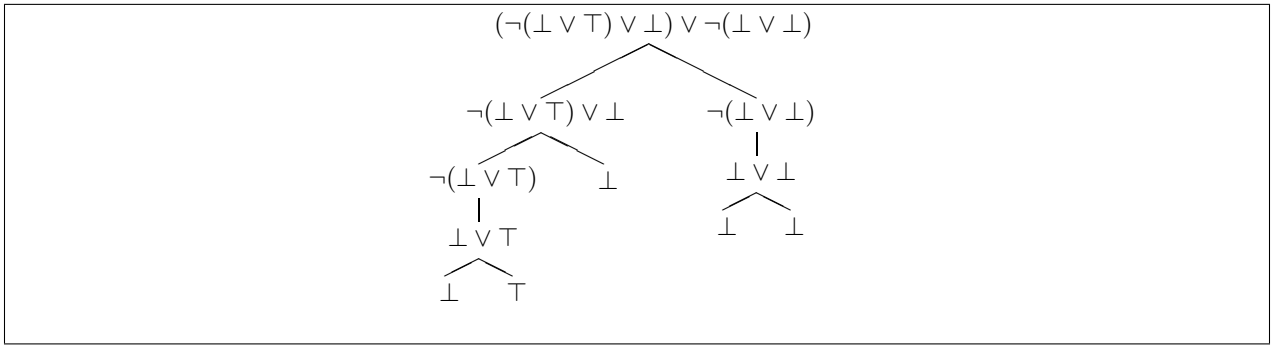

Figure 6. The formula construction tree of $A_{G(2,1,0)}$, a construction tree created from the tree in Figure 5 according to the proof of Proposition 5.2.

In Figure 5, we have that every fork is a two-fork, that every terminal history is a move at a two-fork, that no two-fork is a parity point, and that there are no only-child double-moves. We are led to the formula construction in Figure 6 by the construction in the proof of Proposition 5.2.

The formula at the root of the Figure 6 construction tree is the formula $A_{G(2,1,0)}$. This formula is a theorem of propositional logic (and hence of LP). Here is a proof of $A_{G(2,1,0)}$.

1. $\bot \supset \bot$
   by Axiom $\bot \supset A$
2. $(\bot \supset \bot) \supset ((\bot \supset \bot) \supset (\bot \vee \bot \supset \bot)$
   by Axiom $(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C))$
3. $\bot \vee \bot \supset \bot$
   by Modus Ponens 1,2
4. $(\bot \vee \bot \supset \bot) \supset \neg(\bot \vee \bot)$
   by Axiom $(A \supset \bot) \supset \neg A$
5. $\neg(\bot \vee \bot)$
   by Modus Ponens 3,4
6. $\neg(\bot \vee \bot) \supset ((\neg(\bot \vee \top) \vee \bot) \vee \neg(\bot \vee \bot))$
   by Axiom $A \supset B \vee A$
7. $(\neg(\bot \vee \top) \vee \bot) \vee \neg(\bot \vee \bot)$
   by Modus Ponens 5,6

Here is the above proof internalized in LP.

1. $a{:}(\bot \supset \bot)$
2. $b{:}((\bot \supset \bot) \supset ((\bot \supset \bot) \supset (\bot \vee \bot \supset \bot))$
3. $((b \cdot a) \cdot a){:}(\bot \vee \bot \supset \bot)$
4. $c{:}((\bot \vee \bot \supset \bot) \supset \neg(\bot \vee \bot))$
5. $(c \cdot ((b \cdot a) \cdot a)){:}(\neg(\bot \vee \bot))$
6. $d{:}(\neg(\bot \vee \bot) \supset ((\neg(\bot \vee \top) \vee \bot) \vee \neg(\bot \vee \bot)))$
7. $(d \cdot (c \cdot ((b \cdot a) \cdot a))){:}((\neg(\bot \vee \top) \vee \bot) \vee \neg(\bot \vee \bot))$

To determine the winning strategy described by the term $d \cdot (c \cdot ((b \cdot a) \cdot a))$, it suffices to determine the winning strategy described by d on the axiom it labels in line 6. This strategy is given as follows:

- Map $\neg(\bot \vee \bot) \supset ((\neg(\bot \vee \top) \vee \bot) \vee \neg(\bot \vee \bot))$ to its consequent, which is the formula $A_{G(2,1,0)}$.
- Map $A_{G(2,1,0)}$ to its immediate subformula instance $\neg(\bot \vee \bot)$.
- Map $\neg(\bot \vee \bot)$ to its immediate subformula instance $\bot \vee \bot$.

And so the strategy on $A_{G(2,1,0)}$ described by $d \cdot (c \cdot ((b \cdot a) \cdot a))$ consists of the strategy d restricted to its consequent $A_{G(2,1,0)}$, a strategy consisting of just the last two of the three bullets above. This strategy on $A_{G(2,1,0)}$ specifies a strategy on the tree in Figure 5, which

induces the strategy "at $(2,1,0)$, choose $(1,1,0)$" on the tree in Figure 4. The strategy on the tree in Figure 4 then induces the same strategy on the tree of Figure 3. And the strategy on the tree of Figure 3 induces the following strategy on the tree of Figure 2:

- "At $(2,1,0)$, choose $(1,1,0)$."
- "At $(1,0,0)$, choose $(0,0,0)$."
- "At $(0,1,0)$, choose $(0,0,0)$."

We see immediately that this is indeed a winning strategy in $G(2,1,0)$. And as we had hoped, this strategy on the tree of Figure 2 corresponds to the following strategy for the Nim instance $(2,1,0)$: "remove one stone from the first pile, wait for the other player to respond, and then remove the remaining stone."

6 Conclusion

We have defined a game semantics for LP in which terms are interpreted as winning strategies on the formulas they label. This interpretation allows us to view LP as a logic of explicit strategies for its own verification game. Of particular interest is the Internalization Theorem (Theorem 2.3), which we may read as asserting that LP describes a winning strategy on each of its theorems. Notice that there is no requirement for LP to be *complete* with respect to the class of winning strategies in its verification game (meaning that for a fixed valuation set V and axiomatically saturated strategy map S that is good for V, if s^* is a winning strategy on a formula A, then there is a term t such that $S(t,A) = s^*$).

It may be of interest to determine how such strategic completeness can be imposed, whether semantically (perhaps a trivial matter of definition) or syntactically (via a language extension). Of course, such an imposition ought not disturb the Internalization Theorem, since it is this theorem that lets us exploit the winning-strategy–preserving embedding of verification-like extensive games into LP Verification in order to construct winning strategies on winnable instances of verification-like extensive games. (We showed how this is done for the Nim instance $(2,1,0)$.)

But it might be the case that strategic *incompleteness* is of greater interest. In particular, by carefully managing those winning strategies that terms may and may not express—something we might call *strategic expressivity*—our LP Verification Game might be extended to the various (multi-)modal extensions of LP [2, 3, 1]. This extension would have a player who is to make a move at the modal formula $\Box A$ choose a term t and then continue playing as if the current formula were $t:A$. Thus the set

$$\{s^* \mid (\exists t)(S(t,A)\!\downarrow \text{ and } S(t,A) = s^*)\}$$

of strategies on A expressible by a term would determine the modal theory for the modality $\Box$. Such a study of term expressivity seems promising as a direction for research aimed at defining a game semantics for all of Justification Logic [2, 3, 1].

References

[1] Sergei Artemov. Justified common knowledge. *Theoretical Computer Science*, 357:4–22, 2006.

[2] Sergei Artemov and Elena Nogina. Introducing justification into epistemic logic. *Journal of Logic and Computation*, 15(6):1059–1073, 2005.

[3] Sergei Artemov and Elena Nogina. On epistemic logic with justification. In Ron van der Meyden, editor, *Theoretical Aspects of Rationality and Knowledge: Proceedings of the Tenth Conference (TARK X)*, pages 279–294. ACM Digital Library, 2005.

[4] Sergei N. Artemov. Explicit provability and constructive semantics. *The Bulletin of Symbolic Logic*, 7(1):1–36, 2001.

[5] Charles L. Bouton. Nim, a game with a complete mathematical theory. *The Annals of Mathematics*, 3 (2nd Series):35–39, 1901-1902.

[6] Melvin Fitting. The logic of proofs, semantically. *Annals of Pure and Applied Logic*, 132(1):1–25, 2005.

[7] David Gale and F. M. Stewart. Infinite games with perfect information. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume II, pages 245–266. Princeton University Press, Princeton NJ, 1953.

[8] Leon Henkin. The completeness of the first-order functional calculus. *The Journal of Symbolic Logic*, 14(3):159–166, 1949.

[9] Jakko Hintikka and Gabriel Sandu. Game-theoretical semantics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 361–410. Elsevier, 1997.

[10] Wilfrid Hodges. Logic and games. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2006.

[11] Alexey Mkrtychev. Models for the logic of proofs. In Sergei I. Adian and Anil Nerode, editors, *Logical Foundations of Computer Science, Proceedings of the 4th International Symposium*, volume 1234 of *Lecture Notes in Computer Science*, pages 266–275. Springer, 1997.

[12] Ahti Pietarinen. Peirce's game-theoretic ideas in logic. *Semiotica*, 144:33–47, 2003.

[13] Gabriel Sandu. On the theory of anapohora: Dynamic predicate logic vs. game-theoretical semantics. *Linguistics and Philosophy*, 20:147–174, 1997.

[14] Gabriel Sandu. Signalling in languages with imperfect information. *Synthese*, 127:21–34, 2001.

[15] Gabriel Sandu. Two notions of scope. Manuscript, February 2006.

[16] Merlijn Sevenster. *Branches of Imperfect Information: Logic, Games, and Computation*. PhD thesis, Universiteit van Amsterdam, 2006.

[17] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.