Propositional Games with Explicit Strategies

Bryan Renne
Computer Science
CUNY Graduate Center
365 Fifth Avenue, Room 4319
New York, NY
USA
http://bryan.renne.org/

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 those subformula instances of A that are either positive non-conjunctions or else negative conjunctions and False choosing at those subformula instances

of A that are either negative non-conjunctions or else positive conjunctions. (Special case: if A contains no conjunctions, then True chooses at positive subformula instances of A and False chooses at negative subformula instances of A.) In this way, the players choose immediate subformula instances of the current formula until an atomic formula p is reached, at which point the game is over. True wins in two cases: (1) if p is true in M and positive in A, or (2) if 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 in a background model* M means that 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. The notion of truth can then be extended to a notion of validity: call a formula *valid* exactly when the formula is true in every background model. In this way, we obtain 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 [14]. Research in IF logics has centered on identifying these dependencies and understanding their influences on logic (see Sandu's paper [21] for a flavor of this work).

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 [15]).

In this paper, we define a game semantics for the logic LP, Artemov's Logic of Proofs [7]. 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 [7]. 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." Extensions and variations of this in-language notion of justification have recently been used for studying evidence and justification from an epistemic perspective, leading to the study of a family of logics grouped together under the name *Justification Logic* [1, 2, 3, 5, 6, 10, 11, 17, 20].

This paper defines a game semantics for one of the basic Justification Logics, LP itself. Our game semantics adds to the list of known semantics for LP, which presently includes an arithmetic semantics [7], a minimal semantics [18], and a Kripke-style semantics [9]. 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

[1] The basic ideas of this extension go back to Peirce [13, 19].

existence of a winning-strategy–preserving embedding of certain extensive games of 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 winnable game instance, from which we can then extract a winning strategy on the original, non-embedded game instance itself. For concreteness, we will use this method at the end of the paper to extract a winning strategy for a winnable instance of the well-known game of Nim [8].

But before we can do any of this, we must first describe LP and its game semantics. So let us begin by introducing LP, Artemov's Logic of Proofs [7].

2 The Language and Theory of LP

For present purposes, the *language of propositional logic* consists of a countable number of propositional letters, the propositional constant $\top$ for truth, the propositional constant $\bot$ for falsehood, and the following logical connectives: binary implication (written $\supset$), binary conjunction (written $\wedge$), binary disjunction (written $\vee$), and unary negation (written $\neg$). The *atoms*, also called *atomic formulas*, consist of the propositional letters and the propositional constants. The *propositional formulas* are obtained in the usual way from the atoms using the logical connectives.

To say that a formula is *conjunctive* means that the formula is of the form $B \wedge C$, and to say that a formula is *non-conjunctive* means that the formula is not conjunctive. Further, a *conjunction* is a conjunctive formula, and a *non-conjunction* is a non-conjunctive formula.

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 2.1. 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 rules 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 2.2. 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 2.3. The *theory of* LP is given as follows.

- Axiom Schemes

  LP0. Axiom schemes for classical propositional logic

  LP1. $u\!:\!(A \supset B) \supset \big(v\!:\!A \supset (u \cdot v)\!:\!B\big)$

LP2. $u{:}A \supset !u{:}(u{:}A)$

LP3. $u{:}A \vee v{:}A \supset (u+v){:}A$

LP4. $u{:}A \supset A$

- Rule of *Modus Ponens:* if $A \supset B$ and A are provable, then B is provable.
- Rule of *Constant Necessitation:* if c is a constant and A is an axiom of LP, then $c{:}A$ is provable.

The intended reading of the formula $t{:}A$ is "t is a proof of A" [7]. Here we are to think of the term t as an abstract representation of an actual proof in the theory LP of the formula A. Let us see how the LP-specific axiom schemes and rules provide an intuitive support for this reading.

- The scheme $u{:}(A \supset B) \supset \big(v{:}A \supset (u \cdot v){:}B\big)$ says that in case u is a proof of an implication and v is a proof of that implication's antecedent, then $u \cdot v$ is a proof of that implication's consequent. So the function symbol $\cdot$ is used to represent applications of Modus Ponens.
- The scheme $u{:}A \supset !u{:}(u{:}A)$ says that if u is a proof of A, then $!u$ checks that u is indeed a proof of A. So the function symbol ! provides a means of verifying a proof assertion.
- The scheme $u{:}A \vee v{:}A \supset (u+v){:}A$ says that if one or more of u and v is a proof of A, then $u+v$ is also a proof of A. So the function symbol $+$ is a monotonic combination of proofs, in that $u+v$ proves all those things proved by either u or v.
- The scheme $u{:}A \supset A$ says that if u is a proof of A, then A is true. This tells us that our system of proof is veridical: anything that is proven is in fact true.
- The rule of Constant Necessitation says that we use constants as unanalyzed proofs of our most basic assertions, the axioms.

The following theorem, due to Artemov [7], describes the way in which LP is able to reason about its own proofs. This theorem bolsters the intuitive reading of $t{:}A$ as "t is a proof of A."

Theorem 2.4 (Artemov's Internalization Theorem [7])**.** For each LP theorem A, there is 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 by Constant Necessitation. Otherwise, if A is not an axiom, then A is obtained by Modus Ponens or Constant Necessitation. If A is obtained from the theorems $B \supset A$ and B by Modus Ponens, then the induction hypothesis yields variable-free terms u and v such that $u{:}(B \supset A)$ and $v{:}B$ are both theorems, and so it follows by LP1 and Modus Ponens that $(u \cdot v){:}A$ is also a theorem. If $c{:}A$ is obtained by Constant Necessitation, it follows by LP2 and Modus Ponens that $!c{:}(c{:}A)$ is a theorem. □

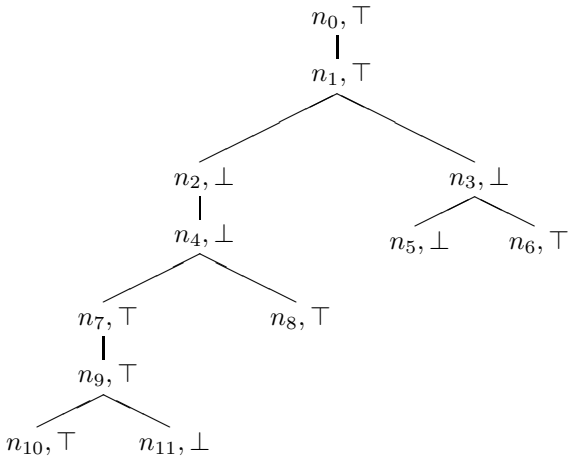

Figure 1. A pebble game for players $\top$ and $\bot$.

While we have focused on the reading "t is a proof of A" for the formula $t\!:\!A$, it will be our task now to describe how this formula may also be read as "t is a winning strategy on A." To make sense of the latter reading, we will define a two-player game called the LP *Verification Game*. A formula will be used to generate the game board for a particular play of the LP Verification Game, and terms will be used as schematic descriptions of strategies in the game. We will then see that if the strategies described by the terms respect the axiomatics of LP, then we are guaranteed that the formula $t\!:\!A$ is provable in LP if and only if t is a schematic description of a winning strategy on the board generated by A. This will justify our reading $t\!:\!A$ as "t is a winning strategy on A."

3 Pebble Games with Explicit Strategies

The LP Verification Game is based on a rather simple game that we call the *pebble game*.[2] The pebble game is a game played by two players. The game board consists of a finite tree that has had a pebble placed at its root and has had each of its nodes labeled by the name of one or the other of the players. The game is then played in the following way. If the pebble is located on a leaf, then the game is over, and the player whose name is written on that leaf wins. If the pebble is located on a non-leaf, then the player whose name is written on that non-leaf must move the pebble to a child of that non-leaf. In this way, the players move the pebble in a sequence of parent-to-child moves until a leaf is finally reached. The object of the game is for a player to have this final leaf be one on which his name is written.

Example 3.1. Figure 1 is a pebble game for players $\top$ and $\bot$. The board consists of twelve nodes, n_0 through n_{11}, each of which is labeled by the name of one or the other of the players. Concerning the leaves: nodes n_{10}, n_8, and n_6 are winning positions for $\top$, while nodes n_{11}

[2]The pebble game is not our creation; it is in fact a certain kind of extensive game with perfect information [23] (see §6 for details). Pebble games and adaptations of pebble games have been used to define game semantics for a number of logics [14, 15].

and n_5 are winning positions for $\bot$. Concerning the non-leaves: nodes n_0, n_1, n_7 and n_9 are positions at which $\top$ must make a move, while nodes n_2, n_3, and n_4 are positions at which $\bot$ must make a move. So an example play of the game might go as follows. The pebble begins at the root n_0. Since n_0 is labeled by $\top$, player $\top$ gets the first move. To make this move, player $\top$ must move the pebble to a child of n_0. So suppose player $\top$ moves the pebble from n_0 to the child n_1. The pebble then rests on n_1 and, since this node is also labeled by $\top$, player $\top$ must again make a move. To make this move, player $\top$ must move the pebble from n_1 to a child of n_1. So suppose player $\top$ moves the pebble from n_1 to the child n_3. The pebble then rests on n_3 and, since this node is labeled by $\bot$, player $\bot$ must make a move. To make this move, player $\bot$ must move the pebble from n_3 to a child of n_3. So suppose player $\bot$ moves the pebble from n_3 to the child n_5. The pebble then rests on n_5, which is a leaf and hence the game is over with player $\bot$ the winner by the fact that n_5 is labeled by $\bot$. Note that this sequence n_0, n_1, n_3, n_5 is only one of five possible plays that can occur in this particular pebble game. (The other four plays are obtained by taking each of the leaves other than n_5 and then enumerating a parent-to-child sequence of nodes that begins at the root n_0 and ends at the chosen leaf.)

A variation of the pebble game, which we call the *pebble game with explicit strategies*, describes the essential underlying structure of the LP Verification Game. So let us now discuss the pebble game with explicit strategies.

Take a game board of the pebble game. This game board consists of a finite tree that has had each of its nodes labeled by the name of one or the other of the players. Intuitively, for a player to play by a strategy, he is to make his moves according to a preconceived plan. This plan simply specifies a move for the player to make at each of those positions at which he must make a move. This leads us to the following formal definition: for a player P and a game board G, a *strategy for P in G* is a function that maps each P-labeled non-leaf in G to a child of that non-leaf.[3] And for a P-labeled node n in G, a *strategy for P at n* is a function that maps each non-leaf P-labeled descendant of n to a child of that non-leaf descendant. Important point: we adopt the convention that *a node is not a descendant of itself*; accordingly, if a node n is labeled by P, then a strategy for player P at a node n *does not* provide a move for P at the node n itself.

Remark 3.2. We have made a distinction between the notion of a strategy *in a pebble game* and the notion of a strategy *at a node* in a pebble game. To see the difference, observe that a strategy *in a pebble game* specifies a move at every node in the game at which the player could possibly have to move; in particular, a strategy for P in a pebble game G will specify

[3]Notice that we require strategies to be *complete*, in the sense that they must tell the player what to do in every position at which he might possibly have to move. Thus a strategy must say what to do for every possible play of the game, not just for a particular play (in which some positions may not be reached). As an example: in the pebble game from Figure 1, a strategy for player $\top$ must choose a child of node n_9 even if this very strategy chooses the child n_3 of the node n_1 as its second move (a move that makes it impossible for the pebble to ever land on node n_9).

This choice of complete strategies is not essential to our setup; indeed, in weighing the consequences of having non-complete strategies (more complexity in the notion of strategy, less complexity in specifying particular strategies) versus having complete strategies (less complexity in the notion of strategy, more complexity in specifying particular strategies), we chose the latter route in the interest of keeping our basic concepts as simple as possible. But this choice could have easily been made the other way around.

a move at the root of G whenever the root is labeled by P. In contrast, a strategy *at a node* n in a pebble game G only specifies a move at the descendants of n in G; in particular, a strategy at the root of G does *not* specify a move at the root (because the root is not a descendant of itself). We distinguish these two notions of strategy for technical reasons, and we will point out later where it is that this distinction arises. But for now our focus will be on the notion of strategy *at a node*.

Example 3.3. Consider the following description of choices to be made by player $\top$ in the pebble game from Figure 1:

- at node n_1, choose the child n_2;
- at node n_7, choose the child n_9;
- at node n_9, choose the child n_{10}.

Let us see that this description is a strategy for player $\top$ at the root n_0; that is, we are to show that this description specifies a function that maps each non-leaf descendant of n_0 that is labeled by $\top$ to a child of that non-leaf descendant. We will do this by observing three points. First, our description clearly specifies a function by the fact that it does not specify two different choices for one and the same node. Second, for each of the nodes for which our description makes a choice, the choice is always a child of the given node. Third, of the non-leaf $\top$-labeled nodes that are descendants of n_0—which, by inspection of Figure 1, consist of n_1, n_7, and n_9—our description makes a choice at each such node. (Recall that a node is a not descendant of itself and so n_0 is a not a descendant of n_0.) Taken together, these three points show that our description is indeed a strategy for player $\top$ at the root n_0.

Example 3.4. Consider the following description of choices to be made by player $\bot$ in the pebble game from Figure 1:

- at node n_4, choose the child n_7.

By an argument similar to that in Example 3.3, we have that this description is a strategy for player $\bot$ at the node n_2; that is, this description specifies a function that maps each non-leaf descendant of n_2 that is labeled by $\bot$ to a child of that non-leaf descendant.

Example 3.5. Consider the following description of choices to be made by player $\top$ in the pebble game from Figure 1:

- at node n_9, choose the child n_{11}.

By an argument similar to that in Example 3.3, we have that this description is a strategy for player $\top$ at the node n_7; that is, this description specifies a function that maps each non-leaf descendant of n_7 that is labeled by $\top$ to a child of that non-leaf descendant.

The reader has perhaps observed that our definitions of strategy in a pebble game and strategy at a node in a pebble game do not rule out the *empty strategy*, which we define as the empty function (that is, the function whose domain is empty). In fact, the empty strategy is a strategy in any one-node pebble game G because there are no non-leaves in G.

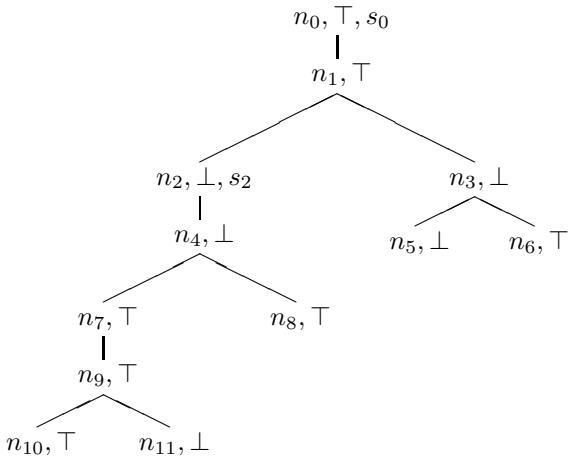

Figure 2. A pebble game with explicit strategies for players $\top$ and $\bot$.

Similarly, the empty strategy is a strategy at any node satisfying the property that each of the node's children is a leaf; after all, such a node has no non-leaf descendants. The reader bored by this discussion of the empty strategy need not worry: while this concept may come up from time to time, it will not be of significant concern to us beyond the paragraph we have just finished.

Let us extend the labeling of game board nodes in the following way: if a node n is labeled by the player P—meaning that player P either wins at n (if n is a leaf) or that player P must move at n (if n is a non-leaf)—then n may also be labeled by a strategy for P at n.[4] Now consider the following rule, called the *Strategy Rule*: if the pebble lands on a node n labeled by a strategy s for a player, then s controls the player's moves at the descendants of n. Adding this rule to the list of rules of the basic pebble game has the following effect: a player is allowed to make his moves however he wishes as long as the pebble has not yet landed on a node labeled by one of his strategies; however, once the pebble does land on a node n labeled by one of his strategies, then this strategy thereafter completely controls his moves at the descendants of n.

Example 3.6. Let s_0 be the strategy for player $\top$ at node n_0 defined in Example 3.3 and let s_2 be the strategy for player $\bot$ at node n_2 defined in Example 3.4. Now consider the game board in Figure 2. In this game, the pebble begins at the root n_0. Since n_0 is labeled by $\top$ and by the strategy s_0, the Strategy Rule applies: s_0 controls player $\top$'s moves at the descendants of n_0. Since n_0 is not a descendant of itself (recall our convention that a node is not a descendant of itself), player $\top$ still has to choose his move at n_0, though the fact that n_0 has only one child ends up trivializing this choice. So player $\top$ moves the pebble from n_0 to the child n_1. Since n_1 is labeled by $\top$, it is again player $\top$'s turn to move. But

[4]It is not quite correct for us to say that a node is *labeled by a strategy*; we should instead say that a node is *labeled by the name of a strategy.* But the distinction between these two statements is not important for what follows, so, in the interest of brevity, we will generally conflate a strategy with its name. (For the same reason, we have also at times conflated players with their names, a practice that we will continue when we find it both convenient and also unlikely to cause confusion.)

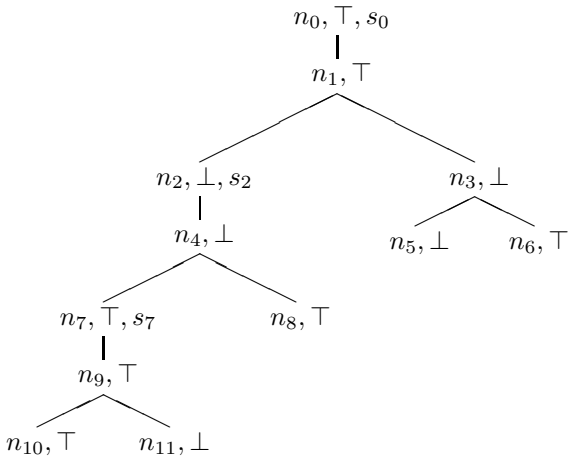

Figure 3. A pebble game with explicit strategies for players $\top$ and $\bot$.

n_1 is a $\top$-labeled descendant of n_0 and so the Strategy Rule has s_0 make the move at this node: s_0 moves the pebble from n_1 to the child n_2 (in accordance with the definition of s_0 from Example 3.3). Since n_2 is labeled by $\bot$ and by s_2, the Strategy Rule again applies: s_2 controls player $\bot$'s moves at the descendants of n_2. Since n_2 is not a descendant of itself, player $\bot$ still has to choose his move at n_2, though his choice is similarly trivialized, so he moves n_2 to the child n_4. Since n_4 is labeled by $\bot$, it is player $\bot$'s turn again. But n_4 is a $\bot$-labeled descendant of n_2 and so the Strategy Rule has s_2 make the move: s_2 moves the pebble from n_4 to the child n_7 (in accordance with the definition of s_2 from Example 3.5). But n_7 is a $\top$-labeled descendant of n_0 and so the Strategy Rule has s_0 make the move at this node: s_0 moves the pebble from n_7 to the child n_9 (in accordance with the definition of s_0 from Example 3.3). Since n_9 is also a $\top$-labeled descendant of n_0, the Strategy Rule again has s_0 make the move: s_0 moves the pebble from n_9 to the child n_{10} (also in accordance with the definition of s_0 from Example 3.3). Since n_{10} is a leaf, the game is over with player $\top$ the winner by the fact that n_{10} is labeled by $\top$.

Perhaps the reader has realized that if we do not place additional restrictions on how nodes may be labeled by strategies, then the Strategy Rule can run afoul of itself. To see why, suppose that the pebble lands on a node n labeled by a strategy s for a player. Applying the Strategy Rule, the strategy s then controls how the player plays at the descendants of n. But in playing the remainder of the game, the pebble might land on another node n' that is labeled by yet another strategy s' for the same player. By another application of the Strategy Rule, the strategy s' then also controls how the player plays the game at the descendants of n', which are themselves descendants of n. Since each of the strategies s and s' is to control how the player plays at the descendants of n', a problem may arise in the following way. If the pebble should end up on a descendant of n' at which the player in question must make a move and the strategies s and s' disagree as to the move to make, then any move will violate at least one of the strategies s or s'. In such a situation, it impossible for the player to act in compliance with the Strategy Rule.

Example 3.7. Let s_0 be the strategy for player $\top$ at node n_0 defined in Example 3.3, let s_2 be the strategy for player $\bot$ at node n_2 defined in Example 3.4, and let s_7 be the strategy for player $\top$ at node n_7 defined in Example 3.5. Now consider the game board in Figure 3. Suppose the game has been played so that the pebble has reached node n_9. The pebble was thus moved from n_0 to n_1 to n_2 to n_4 to n_7 to n_9 (observe that each of these moves is in accord with the Strategy Rule). Since the pebble rests on n_9, it is then player $\top$'s turn to play, except that now each of the strategies s_0 and s_7 is to control his moves his moves according to the Strategy Rule because n_9 is a $\top$-labeled descendant of both n_0 and n_7. But here we have a problem: strategy s_0 chooses the child n_{10}, whereas strategy s_7 chooses the child n_{11}. Since no choice of a child of n_9 can satisfy each of these strategies, no choice of a child of n_9 will be in compliance with the Strategy Rule.

We wish to avoid such situations in which it is impossible for a player to act in compliance with the Strategy Rule. In so doing, we want to keep the general spirit of our initial setup, by which we mean that we will not investigate "exotic" solutions that call for non-trivial mechanisms that negotiate an agreement between conflicting strategies. This leaves us with two routes we may take to solve the problem of strategy conflicts.

The first route is to simply forbid all strategy labelings that result in strategy conflicts. Formally, this means that whenever a node n labeled by player P and by strategy s has a descendant n' labeled by player P and by strategy s', then strategies s and s' must agree on all descendants of n'. Said informally, whenever two or more strategies are to control a player's moves, they must all together agree on the moves that the player is to make. While this restriction does eliminate the possibility of strategy conflicts, it has the unfortunate consequence of diminishing the role of later-encountered strategies. After all, a strategy says how the player is to play the remainder of the game, so once one strategy has been encountered, this strategy then completely determines the moves that any subsequently encountered strategies may choose. We find this undesirable—especially in the specific context of the to-be-defined LP Verification Game—so we will not pursue this first route.

Example 3.8. Suppose we were to adopt the first route, whereby we forbid all strategy labelings that result in strategy conflicts. The labeling in Figure 3 with strategies defined as in Example 3.7 would then be forbidden because we showed in Example 3.7 that this labeling can lead to a strategy conflict. We could of course change the labeling in Figure 3 so as to eliminate strategy conflicts. In particular, we could replace the strategies s_0 and s_7 by other strategies such that the resulting labeling would have no conflicts. In doing this, we might like to leave strategy s_0 in place because it takes player $\top$ to one of his winning positions. This reduces the problem to one of finding a strategy s_7' to replace s_7 in order to produce a board without strategy conflicts. But now notice that these two goals—keeping s_0 as part of the labeling (goal one) and choosing s_7' so as to eliminate strategy conflicts (goal two)—together determine the strategy s_7'. Namely, s_7' is the strategy obtained by restricting the domain of s_0 to the descendants of n_7; that is, s_7' must move at node n_9 as does s_0. It is in this way that the earlier-encountered strategy s_0 completely determines the later-encountered strategy s_7', thereby diminishing the role of this later-encountered strategy. Since we find this phenomenon undesirable, we will not pursue this route.

The second route one may take to solve the problem of strategy conflicts—and it is this route that we will take—is to limit the scope of a strategy's control over a player's moves so

as to guarantee that no more than one strategy is in control of a player's moves at any given time. To do this, we will replace the Strategy Rule with a new rule called the *Strategy Hand Off Rule*: if the pebble lands on a node n labeled by a strategy s for a player, then control of the player's moves at the descendants of n is immediately relinquished to strategy s, no matter whether it was another strategy or the player himself that previously had control of the player's moves at these descendants. So we see that the Strategy Hand Off Rule has the player or any strategy that is controlling a player's moves "hand off" its control at descendant nodes to the next-encountered strategy for that same player.

The Strategy Hand Off Rule guarantees that at most one strategy will be in control of a player's moves at any given time. This eliminates the possibility of strategy conflicts because such conflicts can only arise in situations in which two or more strategies simultaneously control a player's moves. In addition, the Strategy Hand Off Rule does not have the undesirable property whereby the first-encountered strategy for a player diminishes the role of a later-encountered strategy for that same player (in the sense that the first-encountered strategy completely determines the later-encountered strategy, as described above). In fact, a later-encountered strategy will still have a genuine role to play, in that it will also get its chance to control.

Example 3.9. Consider again the game board in Figure 3, with the strategies s_0, s_2, and s_7 defined respectively as in Examples 3.3, 3.4, and 3.5. Let us now examine how the game is played when the rules consist of those for the basic pebble game in addition to the Strategy Hand Off Rule. In this game, the pebble proceeds as in Example 3.7 from the root n_0 to the node n_7; after all, the effects of the Strategy Hand Off Rule and of the Strategy Rule coincide up to the point where the pebble reaches node n_7. But then there is a crucial difference: since n_7 is labeled by $\top$ and by strategy s_7, the Strategy Hand Off Rule relinquishes control of $\top$'s moves at the descendants of n_7 to the strategy s_7. So we see that at n_7, the strategy s_0 still gets to move (since n_7 is not a descendant of itself), so s_0 moves the pebble from n_7 to the child n_9 (in accordance with the definition of s_0 from Example 3.3). But then the pebble rests on n_9, a $\top$-labeled descendant of n_7, so the Strategy Hand Off Rule says that it is strategy s_7—and not strategy s_0—that determines this particular move: s_7 moves the pebble from n_9 to the child n_{11}. Since n_{11} is a leaf, the game is then over with player $\bot$ the winner by the fact that n_{11} is labeled by $\bot$.

We have described almost all of the concepts needed to understand the notion of a *pebble game with explicit strategies*. What is missing are two additional concepts that will be important later in the specific context of the LP Verification Game. These missing concepts are the *forfeit move* and the *forfeit strategy*.

A *forfeit move* is a new kind of move that we allow a player to make whenever it is his turn to move. When a player *forfeits* (that is, when he makes a forfeit move), game play stops immediately and the forfeiting player loses.

Closely related to the concept of the forfeit move is the concept of *forfeit strategy*: a forfeit strategy is a special label that we use to designate non-leaf nodes at which the player-to-move must immediately forfeit, no matter whether a strategy is currently controlling his moves.[5] (When it is clear from context, we will generally omit the plural or singular of the

[5]In case a node is labeled both by a (regular) strategy s and by the special symbol designating a forfeit

words "move" and "strategy" when we discuss forfeit moves or forfeit strategies, using the plural or singular of the word "forfeit" to refer to either of these concepts.) Whenever forfeit moves and strategies are made part of the game, we will say that the game is a game *with forfeits*.

Forfeits are the final ingredients we need to complete the following definition: a *pebble game with explicit strategies* is a pebble game with forfeits to whose rules we add the Strategy Hand Off Rule.

Example 3.10. Consider again the game board in Figure 3. Define the strategy s_0 as in Example 3.3 and the strategy s_2 as in Example 3.4. But in a change from before, let the strategy s_7 at node n_7 designate a forfeit strategy. With these respective definitions of s_0, s_2, and s_7, let us suppose that the pebble game with explicit strategies on the game board in Figure 3 has reached a point where the pebble rests on node n_7. Since n_7 is labeled both by $\top$ and by the forfeit strategy s_7, player $\top$ must then forfeit, and the game ends with player $\bot$ the winner. Note that game play does not make it any further than node n_7, as this is the node at which player $\top$ forfeits. As such, this particular play of the game, which consists of the sequence n_0, n_1, n_2, n_4, n_7 of game tree nodes, ends not on a leaf but instead on the node at which the forfeit move was made.

Example 3.10 demonstrates the following fact: a play of the pebble game with explicit strategies ends on a leaf if and only if no forfeit move was made;[6] furthermore, if a forfeit move is made during a play of this game, then this forfeit move is the unique forfeit move made during the play and the node at which this forfeit move was made is the node that ends the play. Taking note of this fact will help to make sense of the forthcoming definition of a *play* in the LP Verification Game (Definition 4.13).

In the next section, we will introduce the LP Verification Game, a game whose essential underlying structure matches that of the pebble game with explicit strategies.

4 The LP Verification Game

The LP Verification Game, whose name we often shorten to LP *Verification*, is the game that we will use to define a semantics for the theory of LP. It will be our task in this section to describe how this game provides us with a notion of formula validity that makes formal sense of the reading "t is a winning strategy on B" for the formula $t\!:\!B$. We will later argue that this notion of validity is correct, meaning that the formulas valid according to this notion are exactly those formulas that are provable in the theory of LP.

The idea of the LP Verification Game is rather simple. To determine the validity of a formula A, we construct a game board tree $\mathcal{T}_S^V(A)$ for a pebble game with explicit strategies. Other than the formula A, the game board $\mathcal{T}_S^V(A)$ depends on two parameters: a parameter V that varies the winning conditions of the leaves and a parameter S that varies the labeling of nodes by strategies. So once we fix a particular pair (V, S), our two players $\top$ ("True") and $\bot$ ("False") can play the pebble game with explicit strategies on the game board $\mathcal{T}_S^V(A)$.

strategy, then the forfeit strategy always takes precedence: the player-to-move must immediately forfeit, no matter how good it might otherwise have been for the player to follow the (regular) strategy s.

[6] Recall that forfeits may only occur at non-leaves.

We then define a notion of truth and a notion of validity as follows. To say that A is *true* in the parameter pair (V, S) means that there is a winning strategy for $\top$ ("True") in the pebble game with explicit strategies on game board $\mathcal{T}_S^V(A)$.[7] And to say that A is *valid* means that A is true in each admissible parameter pair (V, S). So our notions of truth and validity do not depend on how well True happens to play a particular round of the game. Instead, these notions depend on how well True can play the game in the best of circumstances; that is, the notions of truth and validity depend on whether it is possible for True to guarantee himself a win in the game, which is just what it means to say that there is a winning strategy for True.

So to begin, we need to say how the formula A and the parameter pair (V, S) determine the game board $\mathcal{T}_S^V(A)$. To do this, we first define a finite tree that provides the underlying structure of the game board. This tree, written $\mathcal{T}(A)$, is built by breaking down the formula A according to its inductive construction.

Definition 4.1. The *construction tree* of A, written $\mathcal{T}(A)$, is the labeled binary tree built as follows.

- The root of $\mathcal{T}(A)$ is labeled A.
- If c is a binary logical connective, then each node in $\mathcal{T}(A)$ labeled $B\ c\ C$ has exactly two children: a left child labeled B and a right child labeled C. (Example: a node labeled $B \supset C$ has a left child labeled B and a right child labeled C.)
- Each node in $\mathcal{T}(A)$ labeled $\neg B$ has a unique child labeled B.
- Each node in $\mathcal{T}(A)$ labeled $t{:}B$ has a unique child labeled B.

Whenever it is convenient, we will identify occurrences of subformulas of A with nodes in $\mathcal{T}(A)$. As an example, to say that "B is an occurrence of a subformula of A" is to refer to a node n in $\mathcal{T}(A)$ that is labeled by B.

This provides us with the finite tree $\mathcal{T}(A)$. But in order to have a game board for a pebble game with explicit strategies, we still need to label each of the nodes of this tree by the name of one or the other of the players. We will accomplish this labeling in two stages. In the first stage, we will use the structure of A to label the non-leaves of $\mathcal{T}(A)$. In the second stage, we will use the input parameter V in addition to the structure of A to label the leaves (so varying V will vary the labeling of the leaves). Let us now describe each of these labeling stages.

In the first stage, we label the non-leaves of $\mathcal{T}(A)$ using the structure of A itself. We do this as follows: label positive non-conjunctive non-leaves in $\mathcal{T}(A)$ and negative conjunctive non-leaves in $\mathcal{T}(A)$ by $\top$, and label negative non-conjunctive non-leaves in $\mathcal{T}(A)$ and positive conjunctive non-leaves in $\mathcal{T}(A)$ by $\bot$. In this way, our labeling of the non-leaves of $\mathcal{T}(A)$ will be given by the notion of polarity (of a subformula occurrence).

[7] Our notion of truth uses the notion of strategy *in a pebble game (with explicit strategies)*, as opposed to the notion of strategy *at a node* in a pebble game (with explicit strategies). However, when we use the parameter S to label a node n in $\mathcal{T}_S^V(A)$ with a strategy, then, as before, we will label n with a strategy for a player *at the node*. See Remark 3.2 on Page 6 for the difference between these two kinds of strategy.

Definition 4.2. A *polarity* is an assignment of either *positive* or *negative* to some object. Given a polarity, the *opposite polarity* is the assignment consisting of the other polarity. We assign polarities to the nodes of $\mathcal{T}(A)$ as follows.

- The root of $\mathcal{T}(A)$ is positive.
- If a node in $\mathcal{T}(A)$ labeled $B \supset C$ has already been assigned a polarity, then the left child B is assigned the opposite polarity and the right child C is assigned the same polarity.
- If a node in $\mathcal{T}(A)$ labeled either $B \wedge C$ or $B \vee C$ has already been assigned a polarity, then the left child B and the right child C are each assigned this same polarity.
- If a node in $\mathcal{T}(A)$ labeled $\neg B$ has already been assigned a polarity, then the child B is assigned the opposite polarity.
- If a node in $\mathcal{T}(A)$ labeled $t{:}B$ has already been assigned a polarity, then the child B is assigned the same polarity.

Following our convention that identifies occurrences of subformulas of A with nodes in $\mathcal{T}(A)$, we make the following definition: to say that an occurrence of a subformula B of A has a certain polarity in A means that the node in $\mathcal{T}(A)$ corresponding to this subformula occurrence has that very polarity.

We have said that we will label positive non-conjunctive non-leaves in $\mathcal{T}(A)$ and negative conjunctive non-leaves in $\mathcal{T}(A)$ by $\top$, and we will label negative non-conjunctive non-leaves in $\mathcal{T}(A)$ and positive conjunctive non-leaves in $\mathcal{T}(A)$ by $\bot$. This labeling is based on the following special kind of polarity called the *position-polarity*.

Definition 4.3. The *position-polarity* is a polarity that we assign to each node n in $\mathcal{T}(A)$ in the following way.

- If n is a non-conjunction, then the position-polarity assigned to n in $\mathcal{T}(A)$ is the same as the (regular) polarity that is assigned to n in $\mathcal{T}(A)$ (according to Definition 4.2).
- If n is a conjunction, then the position-polarity of n in $\mathcal{T}(A)$ is the opposite of the (regular) polarity that is assigned to n in $\mathcal{T}(A)$ (according to Definition 4.2).

Using the notion of position-polarity, our labeling of non-leaves n in $\mathcal{T}(A)$ can be described this way: n is labeled by $\top$ if the node has positive position-polarity, and n is labeled by $\bot$ if n has negative position-polarity. So True is to move at the non-leaves with positive position-polarity, and False is to move at the non-leaves with negative position-polarity. Let us establish some terminology reflecting this arrangement.

Definition 4.4. A *$\top$-position (in $\mathcal{T}(A)$)* is a non-leaf in $\mathcal{T}(A)$ that has positive position-polarity, and a *$\bot$-position (in $\mathcal{T}(A)$)* is a non-leaf in $\mathcal{T}(A)$ that has negative position-polarity.

This completes the first stage of labeling nodes of $\mathcal{T}(A)$ by player names. To complete the second stage, we need to label the leaves of $\mathcal{T}(A)$ by the name of one or the other of the players. To do this, we will make use of the input parameter V, which is called a *valuation set*.

Definition 4.5. A *valuation set* is any set obtained as a union of $\{\top\}$ with a possibly empty set of propositional letters.

Think of a valuation set V in the following way. An occurrence of an atom p in A has a polarity according to Definition 4.2. The membership assertion of p in V also has a polarity: the *positive membership assertion* is "$p \in V$" and the *negative membership assertion* is "$p \notin V$". The label of this particular occurrence of p, whether $\top$ ("True") or $\bot$ ("False"), says whether these two polarities match. If it is true that these polarities match, then this occurrence of p is to be labeled by $\top$; if it is false that these polarities match, so they in fact mismatch, then this occurrence of p is to be labeled by $\bot$.

Definition 4.6. Let V be a valuation set. To say that a leaf l in $\mathcal{T}(A)$ is *matching (under V)* means that l is positive if and only if the formula that labels l is a member of V. If a leaf in $\mathcal{T}(A)$ is not matching under V, then this leaf is said to be *mismatching (under V).*

We will label leaves of $\mathcal{T}(A)$ according to this notion of matching: leaves that match are labeled $\top$ and leaves that mismatch are labeled $\bot$. This completes the labeling of the leaves of $\mathcal{T}(A)$, which is the second and final stage of labeling the nodes of $\mathcal{T}(A)$ by the names of one or the other of the players.

Taken together, our two labeling stages tell us how the formula A and the valuation set V induce a labeling of the nodes of $\mathcal{T}(A)$ by the names of one or the other of the players. We call this induced labeling the *player labeling.*

Definition 4.7. Let V be a valuation set. The *player labeling of $\mathcal{T}(A)$ (under V)* is the function L^V that maps each node of $\mathcal{T}(A)$ to the set $\{\top, \bot\}$ according to the following.

- For each non-leaf n in $\mathcal{T}(A)$, we have

$$L^V(n) := \begin{cases} \top & \text{if } n \text{ is a } \top\text{-position in } \mathcal{T}(A), \\ \bot & \text{if } n \text{ is a } \bot\text{-position in } \mathcal{T}(A). \end{cases}$$

- For each leaf l in $\mathcal{T}(A)$, we have

$$L^V(l) := \begin{cases} \top & \text{if } l \text{ is matching under } V, \\ \bot & \text{if } l \text{ is mismatching under } V. \end{cases}$$

We define $\mathcal{T}^V(A)$ to be the tree obtained from $\mathcal{T}(A)$ by adding the label $L^V(n)$ to each node n in $\mathcal{T}(A)$.

The formula A and the valuation set V give rise to the finite tree $\mathcal{T}^V(A)$ whose nodes are labeled by the name of one or the other of the players. While this will suffice as a game board for the pebble game with explicit strategies, it does not address the issue of how a formula $t{:}B$ can be assigned the reading "t is a winning strategy on B." To address this issue, we introduce an additional parameter S in the setup of the LP Verification Game. This new parameter is called a *strategy map.*

Definition 4.8 (Strategy, Strategy Map)**.** A *strategy on B* is a function mapping each $\top$-position in $\mathcal{T}(B)$ to a child of that $\top$-position.[8] A *strategy map (on B)* is a partial function that maps each term-formula pair (t, C) in the domain of S to a strategy $S(t, C)$ on C. Notation: $S(t, C)\downarrow$ means that (t, C) is in the domain of S, and $S(t, C)\uparrow$ means that (t, C) is not in the domain of S.

Given our finite tree $\mathcal{T}(A)$, we will use a strategy map S on A to label the nodes of $\mathcal{T}(A)$ by strategies for the players. To do this, we will take an occurrence of a subformula $t\!:\!B$ of A and examine whether (t, B) is in the domain of S. If $S(t, B)\uparrow$, then we will take this as tantamount to the specification of a forfeit strategy, and so we will label the node in $\mathcal{T}(A)$ corresponding to this occurrence of $t\!:\!B$ by a forfeit strategy. If $S(t, B)\downarrow$, then we will label the node in $\mathcal{T}(A)$ corresponding to this occurrence of $t\!:\!B$ by the strategy $S(t, B)$ on B. In this way, the term t in the formula $t\!:\!B$ does name a strategy on B (though it is not necessarily a good strategy, an issue we will address shortly).

Definition 4.9. Let S be a strategy map on A. The *strategy labeling of $\mathcal{T}(A)$ (under S)* is the partial function L_S that maps nodes of $\mathcal{T}(A)$ to strategies. We define L_S in three stages, with each stage to be completed before proceeding to the next stage.

1. For each node n labeled $t\!:\!B$ such that $S(t, B)\uparrow$, define $L_S(n)$ to be a forfeit strategy.
2. For each node n labeled $t\!:\!B$ such that $S(t, B)\downarrow$, define $L_S(n)$ to be the strategy $S(t, B)$.
3. For each node n such that n was not labeled in any of the previous stages, n is not in the domain of L_S.

Notation: $L_S(n)\downarrow$ means that n is in the domain of L_S, and $L_S(n)\uparrow$ means that n is not in the domain of L_S. We define $\mathcal{T}_S(A)$ to be the tree obtained from $\mathcal{T}(A)$ by adding the label $L_S(n)$ to each node n in $\mathcal{T}(A)$ that is in the domain of L_S.

Combining the player labeling and the strategy labeling gives us a game board $\mathcal{T}_S^V(A)$ for the pebble game with explicit strategies.

Definition 4.10. Let V be a valuation set and let S be a strategy map. The tree $\mathcal{T}_S^V(A)$ is obtained from $\mathcal{T}(A)$ using the player labeling L^V (Definition 4.7) and the strategy labeling L_S (Definition 4.9) as follows.

- For each node n in $\mathcal{T}(A)$, add $L^V(n)$ to the label of n.
- For each node n in $\mathcal{T}(A)$ that is in the domain of L_S, add $L_S(n)$ to the label of n.

[8] So a strategy on B is a strategy *in the pebble game (with explicit strategies)*. Note that this is *not* the same notion as the notion of strategy *at the root* of a game board tree based on $\mathcal{T}(B)$. See Remark 3.2 on Page 6, which describes the difference between the notions of strategy *in a pebble game (with explicit strategies)* and strategy *at a node* in a pebble game (with explicit strategies). In Definition 4.8, we are interested in the first notion: the strategy *in a pebble game (with explicit strategies)*.

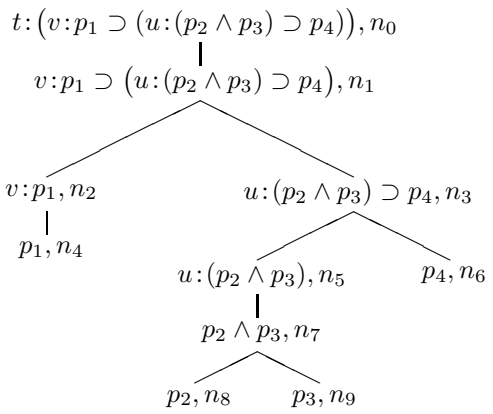

Figure 4. The construction tree $\mathcal{T}(A)$ from Example 4.11. Nodes names n_0 through n_9 have been added for convenience.

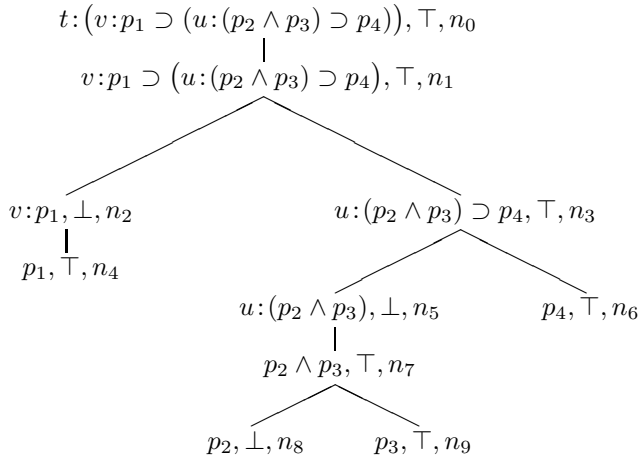

Figure 5. The labeled tree $\mathcal{T}^{V(2,4)}(A)$ from Example 4.11.

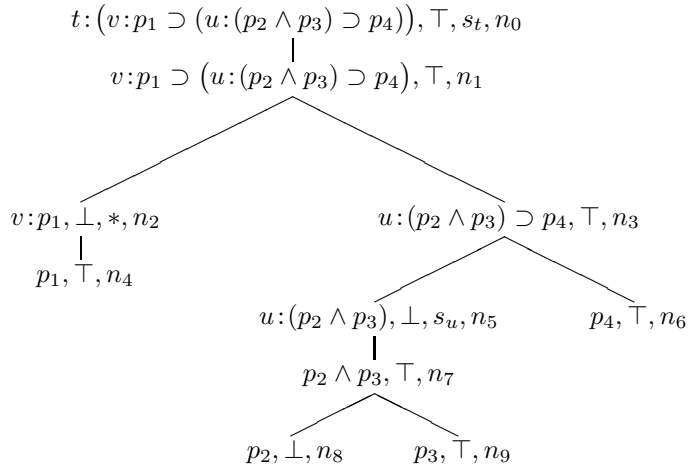

Figure 6. The labeled tree $\mathcal{T}_S^{V(2,4)}(A)$ from Example 4.11. An asterisk ("$*$") denotes a forfeit strategy.

Example 4.11. Let $\bar{4} := \{1, 2, 3, 4\}$, let p_i be a propositional letter for each $i \in \bar{4}$, and let A denote the formula

$$t : \big(v : p_1 \supset (u : (p_2 \wedge p_3) \supset p_4)\big) \ .$$

Figure 4 depicts the construction tree $\mathcal{T}(A)$; for convenience, we have named the nodes in this figure using the names n_0 through n_9. Observe that the subformula occurrence p_4 is positive in A, while the subformula occurrences p_1, p_2, and p_3 are each negative in A. Thus for a valuation set V, the player labeling L^V will assign $\top$ to the node n_4 in $\mathcal{T}(A)$ corresponding to p_4 if $p_4 \in V$ (because the positive polarity of p_4 matches the polarity of this positive membership assertion), and L^V will assign $\bot$ to this node if $p_4 \notin V$ (because the positive polarity of p_4 mismatches the polarity of this negative membership assertion). Further, for each $i \in \{1, 2, 3\}$, the player labeling L^V will assign $\top$ to the node in $\mathcal{T}(A)$ corresponding to p_i if $p_i \notin V$ (because the negative polarity of p_i matches the polarity of this negative membership assertion), and it will assign $\bot$ to this node if $p_i \in V$ (because the negative polarity of p_i mismatches the polarity of this positive membership assertion). So if we let $V(2, 4)$ be the valuation set $\{\top, p_2, p_4\}$, then we obtain the labeled tree $\mathcal{T}^{V(2,4)}(A)$ depicted in Figure 5. (Observe that n_7 is labeled by $\top$ because n_7 is a negative conjunctive non-leaf in $\mathcal{T}(A)$, from which it follows by Definitions 4.3 and 4.4 that n_7 is a $\top$-position in $\mathcal{T}(A)$.) Let us now see how a strategy map adds strategy labels to this tree. Let B denote the formula

$$v : p_1 \supset (u : (p_2 \wedge p_3) \supset p_4) \ .$$

We define the strategy s_t on B using the node names from Figure 5 in the following way:

- at node n_1, choose the child n_3;
- at node n_3, choose the child n_5;
- at node n_7, choose the child n_9.

Now define the strategy s_u on $p_2 \wedge p_3$ to be the empty strategy; this is all right: there are no $\top$-positions in $\mathcal{T}(p_2 \wedge p_3)$. Finally, let S be a strategy map such that $S(t, B) = s_t$, $S(v, p_1)\!\uparrow$, and $S(u, p_2 \wedge p_3) = s_u$. Using the player labeling $L^{V(2,4)}$ and the strategy labeling L_S, we obtain the labeled tree $\mathcal{T}_S^{V(2,4)}(A)$ depicted in Figure 6. The players $\top$ ("True") and $\bot$ ("False") may now play the pebble game with explicit strategies on the game board $\mathcal{T}_S^{V(2,4)}(A)$. In this game, the pebble begins at the root n_0. Since n_0 is labeled by $\top$ and by s_t, the Strategy Hand Off Rule applies: control of True's moves at descendants of n_0 is immediately relinquished to s_0. Since n_0 is not a descendant of itself (by our convention on descendants), True has to move at n_0, though his move is trivialized by the fact that n_0 has only one child. So True moves the pebble from n_0 to the child n_1. But n_1 is a $\top$-labeled descendant of n_0, so the Strategy Hand Off Rule has s_t move: s_t moves the pebble from n_1 to the child n_3 (in accordance with the definition of s_t). Since n_3 is a $\top$-labeled descendant of n_0, the Strategy Hand Off Rule again has s_t move: s_t moves the pebble from n_3 to the child n_5 (also in accordance with the definition of s_t). Since n_5 is labeled by $\bot$ and by s_u, the Strategy Hand Off Rule applies yet again: control of False's moves at descendants of n_5 is immediately relinquished to s_u. Since n_5 is not a descendant of itself, False has to move at n_5, though his move is likewise trivialized. So False moves the pebble from n_5 to the child

n_7. But n_7 is a $\top$-labeled descendant of n_0, so the Strategy Hand Off Rule then has s_t move once again: s_t moves the pebble from n_7 to the child n_9 (in accordance with the definition of s_t). Since n_9 is a leaf, the game is over with player $\top$ the winner by the fact that n_9 is labeled by $\top$. This concludes Example 4.11.

Example 4.11 shows how a play of LP Verification may be identified with a play of a pebble game with explicit strategies. This identification uses a strategy map S to assign a strategy $S(t, B)$ on the formula B, thereby justifying a reading of $t\!:\!B$ as "t is a strategy on B." While this moves us closer to achieving our proposed reading of $t\!:\!B$ as "t is a winning strategy on B," we are missing a requirement that the strategy map S assign a *winning* strategy $S(t, B)$ on B in case $S(t, B)\!\downarrow$. In order to address this missing requirement, let us introduce some additional terminology.

Definition 4.12. A *partial play* of A is a nonempty sequence $\{n_i\}_{i=0}^{k}$ of nodes in $\mathcal{T}(A)$ such that n_0 is the root of $\mathcal{T}(A)$ and n_{i+1} is a child in $\mathcal{T}(A)$ of n_i for each non-negative integer $i < k$. A partial play $\{n_i\}_{i=0}^{k}$ of A is said to *end on* the node n_k at the end of the sequence $\{n_i\}_{i=0}^{k}$ that makes up the partial play.

A partial play of A is just a path in the tree $\mathcal{T}(A)$ that begins at the root. In the pebble game with explicit strategies, it is the Strategy Hand Off Rule that determines whether a given partial play of A is legal according to the rules of the game. In particular, for a partial play of A to be legal it must satisfy the following: if a node n occurring in the partial play is labeled by a player P and by a non-forfeit strategy $L_S(n)$ and $L_S(n)$ does not relinquish control to another strategy before reaching a non-leaf P-labeled descendant n' of n occurring somewhere before the end of the partial play, then the node following n' in the partial play must be the node specified by the strategy $L_S(n)$. So we see that as long as each such *strategy-coupled pair* (n, n') occurring in a partial play of A satisfies the property that the node following n' in the partial play is chosen by the strategy $L_S(n)$, then the partial play is in accord with the Strategy Hand Off Rule, and so the partial play is legal according to the rules of the pebble game with explicit strategies. This leads us to the following definition of what constitutes a (full and legal) play of the game.

Definition 4.13. Let S be a strategy map.

- To say that (n_a, n_b) is a *strategy-coupled pair (under S)* occurring in a partial play $\{n_i\}_{i=0}^{k}$ of A means we have that $0 \leq a \leq b \leq k$ and that each of the following properties is satisfied.
  - The strategy labeling L_S labels node n_a by a non-forfeit strategy: $L_S(n_a)\!\downarrow$ and $L_S(n_a)$ is not a forfeit.
  - n_b is a descendant of n_a; that is, $a < b$.[9]
  - n_b is not the last member of the sequence; that is, $b < k$.[10]
  - The player who moves at n_a is the same player that moves at n_b; that is, n_a and n_b have the same position-polarity in $\mathcal{T}(A)$. (Position-polarity is defined in Definition 4.3.)

[9] Recall our convention that a node is not a descendant of itself.

[10] Note that this requirement implies that n_b is a non-leaf.

– In the pebble game with explicit strategies, the strategy $L_S(n_a)$ does not relinquish control to a strategy labeling an intermediate node because no intermediate node of the same position-polarity is labeled by a strategy; that is, if $a < c < b$ and node n_c has the same position-polarity in $\mathcal{T}(A)$ as does n_a, then $L_S(n_c)\uparrow$.[11]

- To say that a node n in $\mathcal{T}(A)$ is *forfeited (under S)* means that n is labeled by $t\!:\!B$ and $S(t, B)\uparrow$.

- An *S-play of A* is a partial play $\{n_i\}_{i=0}^k$ of A satisfying each of the following properties.

  – If a node n_j in the sequence is forfeited under S, then this node is last node in the sequence (that is, $j = k$).[12]

  – If no node in the sequence is forfeited under S, then the node n_k at the end of the sequence is a leaf in $\mathcal{T}(A)$.

  – The sequence is *in accord with* the strategy map S, by which we mean the following: for each strategy coupled pair (n_a, n_b) occurring in $\{n_i\}_{i=0}^k$, if we let s denote the strategy $L_S(n_a)$ that labels n_a, then $n_{b+1} = s(n_b)$.

  When it ought not cause confusion, we may refer to an S-play as a *play*.

- To say that an S-play $\{n_i\}_{i=0}^k$ is *forfeited* means that the last node n_k in the sequence $\{n_i\}_{i=0}^k$ is forfeited under S.

Given the above definition, we may now specify what it means for a strategy to be winning. Intuitively, to say that a strategy is *winning* means that a player who plays according to the strategy is guaranteed a win, no matter the moves made by the opponent. So in the specific situation of the pebble game with explicit strategies on the game board $\mathcal{T}_S^V(A)$, where a strategy on A is a strategy for True (Definition 4.8), this amounts to the following: for each play in which True made his moves by following a winning strategy up to the point at which the Strategy Hand Off Rule passed control of True's moves to some other strategy (should such a point exist), the play in question ends either on a non-leaf at which False forfeits or else on a leaf that is matching in A (meaning that the leaf is labeled by $\top$). To formalize this, let us first give a name to those nodes at which a player has no control over his moves (by the fact that the Strategy Hand Off Rule has passed control of his moves to some strategy).

Definition 4.14. To say that a node n in $\mathcal{T}(A)$ is *determined* means that n has an ancestor n' in $\mathcal{T}(A)$ such that n' is labeled by a formula of the form $t\!:\!B$ and n' has the same position-polarity as does n in $\mathcal{T}(A)$. Important point: we adopt the convention that *a node is not an ancestor of itself*. To say that a node in $\mathcal{T}(A)$ is *undetermined* means that the node is not determined.

[11] Note that $L_s(n_c)\uparrow$ implies that n_c is labeled neither by a forfeit strategy nor by a (regular) strategy (see Definition 4.9).

[12] Note that this implies that there is at most one node in the sequence that is forfeited under S.

Our intention is that a non-leaf is *determined* when a move at that non-leaf is controlled by a labeling strategy as per the Strategy Hand Off Rule. Hence the determined nodes ought to be just those nodes that have a same–position-polarity ancestor labeled by a strategy. According to our strategy labeling (Definition 4.9), a node will have a same–position-polarity ancestor labeled by a strategy if and only if the node has a same–position-polarity ancestor labeled by a formula of the form $t:B$. This is the reason why we defined determined nodes as above.

Since a player cannot control his moves at determined nodes, any strategy that he uses to try and win will only affect his moves at *undetermined* nodes. So the influence of a given strategy (for True) is limited to the choices it makes at undetermined nodes (at which True is to move). And a strategy is *winning* exactly when True is guaranteed a win whenever he follows this strategy. We are lead to the following definition.

Definition 4.15 (Winning Strategy)**.** Let V be a valuation and S be a strategy map.

- To say that a partial play $\{n_i\}_{i=0}^{k}$ of A *follows* a strategy A^* on A means that for each non-negative integer $i < k$ such that n_i is an undetermined $\top$-position in $\mathcal{T}(A)$, we have that $n_{i+1} = A^*(n_i)$.

- To say that a strategy A^* on A is *winning under* (V, S) means that each S-play of A that follows A^* ends either on a $\bot$-position or else on a leaf that is matching under V.

We observe that a play of A that ends on a $\bot$-position in $\mathcal{T}(A)$ is a play of A that ended with a forfeit by False (Definitions 4.7, 4.9, and 4.13), and True is to win such a play. This is the reason for the case distinction in Definition 4.15's consideration of plays that follow a winning strategy.

The reader has perhaps noticed that a strategy on a formula A (Definition 4.8) is really a strategy for True. After all, a strategy on A specifies moves at $\top$-positions of $\mathcal{T}(A)$ and it is True that is to move at these positions (Definition 4.7). So in the interest of having a notion of strategy for False, we make the following definition.

Definition 4.16. A *counter-strategy* on B is a function mapping each $\bot$-position in $\mathcal{T}(B)$ to a child of that $\bot$-position.

There is a natural duality between strategies and counter-strategies. To describe this duality, we introduce the following notation.

Notation 4.17. If s is a strategy or counter-strategy on A and B is an occurrence of a subformula of A, then $s \restriction B$ denotes the function obtained by restricting the domain of s to the nodes of the subtree $\mathcal{T}(B)$ of $\mathcal{T}(A)$.

The duality between strategies and counter-strategies is then characterized by the following lemma, whose proof is straightforward.

Lemma 4.18. Let B be an occurrence of a subformula of A.

- Suppose A^* is a strategy on A.
  - If B is positive in A, then $A^* \restriction B$ is a strategy on B.

– If B is negative in A, then $A^* \restriction B$ is a counter-strategy on B.

- Suppose A_* is a counter-strategy on A.
  - If B is positive in A, then $A_* \restriction B$ is a counter-strategy on B.
  - If B is negative in A, then $A_* \restriction B$ is a strategy on B.

Following the lead in Definition 4.15 (definition of a winning strategy), we define what it means for a counter-strategy to be *winning* (for False).

Definition 4.19. Let V be a valuation set and S be a strategy map.

- To say that a partial play $\{n_i\}_{i=0}^k$ of A *follows* a counter-strategy A_* on A means that for each non-negative integer $i < k$ such that n_i is an undetermined $\bot$-position in $\mathcal{T}(A)$, we have that $n_{i+1} = A_*(n_i)$.

- To say that a counter-strategy A_* on A is *winning under* (V, S) means that each S-play of A that follows A_* ends either on a $\top$-position or else on a leaf that is mismatching under V.

We observe that a play of A that ends on a $\top$-position in $\mathcal{T}(A)$ is a play of A that ends with a forfeit by True (Definitions 4.7, 4.9, and 4.13), and False is to win such a play. This is the reason for the case distinction in Definition 4.19's consideration of plays that follow a winning counter-strategy.

To gain the reading "t is a winning strategy on B" for the formula $t\!:\!B$ under a valuation set V, we will restrict our attention to those strategy maps that are *good for* our valuation set, in the sense of the following definition.

Definition 4.20. Let V be a valuation set. To say that a strategy map S is *good for* V means that whenever $S(t, B)\!\downarrow$, we have that $S(t, B)$ is a winning strategy under (V, S) on B.

All that remains is for us to restrict attention to those strategy maps that assign strategies in a way that respects the intended meaning of the term-forming functions in the language of LP.

Definition 4.21. To say that a strategy map S is *proper* means that S satisfies each of the following conditions.

1. *Product.* If $S(u, B \supset C)\!\downarrow$ and $S(v, B)\!\downarrow$, then both
   (a) $S(u \cdot v, C) = S(u, B \supset C) \restriction C$, and
   (b) $S(u, D \supset C)\!\downarrow$ and $S(v, D)\!\downarrow$ implies $S(u, D \supset C) \restriction C = S(u, B \supset C) \restriction C$.[13]
2. *Proof Checker.* $S(t, B)\!\downarrow$ implies $S(!t, t\!:\!B)\!\downarrow$.
3. *Sum.*

[13] This condition is included to ensure that $S(u \cdot v, C)$ is well-defined. The condition is otherwise unused.

(a) $S(u, B)\downarrow$ implies $S(u + v, B) = S(u, B)$.

(b) $S(u, B)\uparrow$ and $S(v, B)\downarrow$ implies $S(u + v, B) = S(v, B)$.

4. *Constant Necessitation.* If c is a constant and B is an axiom of LP, then $S(c, B)\downarrow$.

A proper strategy map S provides an interpretation for the term-forming functions in the language of LP, in the sense that a strategy $S(t, B)$ assigned to a term t will depend on the strategies $S(u, C)$ assigned to the terms u that make up t. Proper strategy maps are the final ingredient we need in order to define a notion of *model* for use in our semantics for the language of LP.

Definition 4.22. A *model* is a pair (V, S) consisting of a valuation set V and a proper strategy map S that is good for V.

A model (V, S) provides the parameters we need in order to determine whether a formula is true. But before we give the definition of truth, let us first prove that the concept of model is not an empty concept.

Theorem 4.23. For each valuation set V, there is a proper strategy map that is good for V.

Proof. In the context of this proof, the *conjunction* of a finite set of formulas is the conjunction whose conjuncts consist of the formulas in that set. To say that a set of formulas is *consistent* means that for no conjunction C of a finite subset do we have that $C \supset \bot$ is provable in LP. To say that a set of formulas is *inconsistent* means that the set is not consistent. To say that a set of formulas is *maximal consistent* means that the set 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 of formulas may be extended to a maximal consistent set. Letting V be a fixed valuation set, we define $V' := V \cup \{\neg p : p \notin V\}$, where p is a metavariable ranging over atoms (propositional letters, $\top$, and $\bot$). The set V' is consistent and so may be extended to a maximal consistent set T.

If c is a binary logical connective, then we let $\mathcal{B}(B\ c\ C)$ abbreviate the biconditional statement "$B \in T$ if and only if B is a positive subformula occurrence of $B\ c\ C$." Then for each non-atomic formula A, we define functions $W^\top$ and $W^\bot$ that map the root r of $\mathcal{T}(A)$ to a child of r according to the following.

- For a binary logical connective c, we define

$$W^\top(B\ c\ C) := \begin{cases} B & \text{if } \mathcal{B}(B\ c\ C), \\ C & \text{otherwise;} \end{cases}$$

$$W^\bot(B\ c\ C) := \begin{cases} C & \text{if } \mathcal{B}(B\ c\ C), \\ B & \text{otherwise.} \end{cases}$$

- $W^\top(\neg B) := B$ and $W^\bot(\neg B) := B$.
- $W^\top(t{:}B) := B$ and $W^\bot(t{:}B) := B$.

For each formula A, we define the strategy A^* on A and the counter-strategy A_* on A according to the following.

- The domain of A^* is the set of $\top$-positions in $\mathcal{T}(A)$, and for each $\top$-position B in $\mathcal{T}(A)$, we define

$$A^*(B) := \begin{cases} W^\top(B) & \text{if this occurrence of } B \text{ is positive in } A, \\ W^\bot(B) & \text{if this occurrence of } B \text{ is negative in } A. \end{cases}$$

- The domain of A_* is the set of $\bot$-positions in $\mathcal{T}(A)$, and for each $\bot$-position B in $\mathcal{T}(A)$, we define

$$A_*(B) := \begin{cases} W^\top(B) & \text{if this occurrence of } B \text{ is negative in } A, \\ W^\bot(B) & \text{if this occurrence of } B \text{ is positive in } A. \end{cases}$$

We now state and prove three properties of the functions A^* and A_*.

- For each atom p, we have that p^* is the empty strategy on p and that p_* is the empty counter-strategy on p.[14]

  The domain of p^* is the set of $\top$-positions in $\mathcal{T}(p)$. But a $\top$-position is a non-leaf (Definition 4.4) and the one and only node in $\mathcal{T}(p)$, the root, is a leaf. It follows that the domain of p^* is the empty set, which implies that p^* is the empty function. But this is what it means to say that p^* is the empty strategy.

  The argument that the function p_* is the empty counter-strategy is similar.

- If B is an occurrence of a positive subformula of A, then $A^* \restriction B = B^*$ and $A_* \restriction B = B_*$.

  We argue that $A^* \restriction B = B^*$. First, the domain of $A^* \restriction B$ consists of the $\top$-positions in A that are also in $\mathcal{T}(B)$ (Notation 4.17). But a $\top$-position of A that is in $\mathcal{T}(B)$ is itself a $\top$-position in B because B is an occurrence of a positive subformula of A. Similarly, a $\top$-position in $\mathcal{T}(B)$ is itself a $\top$-position in $\mathcal{T}(A)$ because B is an occurrence of a positive subformula of A. It follows that the domains of $A^* \restriction B$ and B^* are identical. Second, for each $\top$-position C in $\mathcal{T}(B)$, we have by the definition of B^* that

$$B^*(C) = \begin{cases} W^\top(C) & \text{if this occurrence of } C \text{ is positive in } B, \\ W^\bot(C) & \text{if this occurrence of } C \text{ is negative in } B. \end{cases}$$

  But if C is an occurrence of a positive subformula of B, then C is an occurrence of a positive subformula of A because B is an occurrence of a positive subformula of A. Similarly, if C is an occurrence of a negative subformula of B, then C is an occurrence of a negative subformula of A because B is an occurrence of a positive subformula of A. So it follows that

$$B^*(C) = \begin{cases} W^\top(C) & \text{if this occurrence of } C \text{ is positive in } A, \\ W^\bot(C) & \text{if this occurrence of } C \text{ is negative in } A. \end{cases}$$

[14] The *empty strategy* and the *empty counter-strategy* are names for the empty function (the function with empty domain).

But then we have that $B^*(C) = A^*(C) = (A^* \restriction B)(C)$, where the rightmost equality follows by the fact that C is in $\mathcal{T}(B)$. Since C was an arbitrary $\top$-position in $\mathcal{T}(B)$, we have shown that $A^* \restriction B = B^*$. The argument that $A_* \restriction B = B_*$ is shown similarly.

- If B is an occurrence of a negative subformula of A, then $A^* \restriction B = B_*$ and $A_* \restriction B = B^*$.

  We argue that $A^* \restriction B = B_*$. First, the domain of $A^* \restriction B$ consists of the $\top$-positions in A that are also in $\mathcal{T}(B)$ (Notation 4.17). But a $\top$-position of A that is in $\mathcal{T}(B)$ is itself a $\bot$-position in B because B is an occurrence of a negative subformula of A. Similarly, a $\bot$-position in B is itself a $\top$-position in A because B is an occurrence of a negative subformula of A. It follows that the domains of $A^* \restriction B$ and B_* are identical. Second, for each $\bot$-position C in $\mathcal{T}(B)$, we have by the definition of B_* that

$$B_*(C) = \begin{cases} W^\top(C) & \text{if this occurrence of } C \text{ is negative in } B, \\ W^\bot(C) & \text{if this occurrence of } C \text{ is positive in } B. \end{cases}$$

  But if C is an occurrence of negative subformula of B, then C is an occurrence of a positive subformula of A because B is an occurrence of a negative subformula of A. Similarly, if C is an occurrence of a positive subformula of B, then C is an occurrence of a negative subformula of A because B is an occurrence of a negative subformula of A. So it follows that

$$B_*(C) = \begin{cases} W^\top(C) & \text{if this occurrence of } C \text{ is positive in } A, \\ W^\bot(C) & \text{if this occurrence of } C \text{ is negative in } A. \end{cases}$$

  But then we have that $B_*(C) = A^*(C) = (A^* \restriction B)(C)$, where the rightmost equality follows by the fact that C is in $\mathcal{T}(B)$. Since C was an arbitrary $\bot$-position in $\mathcal{T}(B)$, we have shown that $A^* \restriction B = B_*$. The argument that $A_* \restriction B = B^*$ is shown similarly.

We will make frequent use of the above properties in the remainder of this proof.

We now define a strategy map S. The domain of S consists of all term-formula pairs (t, A) such that $t{:}A \in T$, and for each pair (t, A) in the domain of S, we set $S(t, A) := A^*$. We now argue that S is proper.

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

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

   By the definition of S, we have that $u{:}(B \supset C) \in T$ and that $v{:}B \in T$. It follows that $(u \cdot v){:}C \in T$ by LP1 and the maximal consistency of T. Applying the definition of S, we have that $S(u \cdot v, C)\!\downarrow$ and that $S(u \cdot v, C) = C^*$. Since C is an occurrence of a positive subformula of $B \supset C$, we have that $C^* = (B \supset C)^* \restriction C$. But $(B \supset C)^* \restriction C = S(u, B \supset C) \restriction C$ by the definition of S, and hence $S(u \cdot v, C) = S(u, B \supset C) \restriction C$.

   (b) We show that $S(u, D \supset C)\!\downarrow$ and $S(v, D)\!\downarrow$ together imply that $S(u, D \supset C) \restriction C = S(u, B \supset C) \restriction C$.

   By the definition of S, we have that $S(u, D \supset C) \restriction C = (D \supset C)^* \restriction C$ and $S(u, B \supset C) \restriction C = (B \supset C)^* \restriction C$. Since C is an occurrence of a positive subformula of $B \supset C$ and of $D \supset C$, we have $C^* = (B \supset C)^* \restriction C$ and $C^* = (D \supset C)^* \restriction C$.

2. *Proof Checker.* Suppose that $S(u, B)\downarrow$. We show that $S(!u, u{:}B)\downarrow$.

   By the definition of S, we have $u{:}B \in T$ and thus that $!u{:}(u{:}B) \in T$ by LP2 and the maximal consistency of T. It follows that $S(!u, u{:}B)\downarrow$ by the definition of S.

3. *Sum.*

   (a) We show that $S(u, B)\downarrow$ implies $S(u + v, B) = S(u, B)$.

   Suppose $S(u, B)\downarrow$. By the definition of S, we then have that $u{:}B \in T$ and thus that $(u + v){:}B \in T$ by LP3 and the maximal consistency of T. It then follows from the definition of S both that $S(u + v, B) = B^*$ and that $S(u, B) = B^*$.

   (b) We show that $S(u, B)\uparrow$ and $S(v, B)\downarrow$ implies $S(u + v, B) = S(v, B)$.

   This follows by an argument similar to the previous case.

4. *Constant Necessitation.* We show that for each constant c and each axiom B of LP, we have that $S(c, B)\downarrow$.

   If c is a constant and B is an axiom of LP, it follows by the rule of Constant Necessitation and the maximal consistency of T that $c{:}B \in T$. Applying the definition of S, we then have that $S(c, B)\downarrow$.

So S is indeed a proper 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 under (V, S) on A, and $A \notin T$ implies A_* is a winning counter-strategy under (V, S) on A. We will prove the WS Property in a moment, but let us first show that the WS Property implies that S is good for V. That is, we prove that the WS Property and $S(t, A)\downarrow$ together imply that $S(t, A)$ is a winning strategy under (V, S) on A.

So suppose that the WS Property holds and that $S(t, A)\downarrow$. By our definition of S, $S(t, A)\downarrow$ implies that $S(t, A) = A^*$ and $t{:}A \in T$. But $t{:}A \in T$ implies that $A \in T$ by LP4 and the maximal consistency of T. Applying the WS Property, $A \in T$ implies that A^* is a winning strategy under (V, S) on A. But then $A^* = S(t, A)$ is a winning strategy under (V, S) on A. We have therefore shown that the WS Property and $S(t, A)\downarrow$ together imply that $S(t, A)$ is a winning strategy under (V, S) on A. But this is what it means to say that the WS Property implies that S is good for V.

So we complete the proof by proving the WS Property: $A \in T$ implies A^* is a winning strategy under (V, S) on A, and $A \notin T$ implies A_* is a winning counter-strategy under (V, S) on A. We prove the WS Property by induction on the construction of formulas.

- Base case: the formula is an atom p.

  T was constructed as a maximal consistent extension of $V' := V \cup \{\neg p : p \notin V\}$. As such, we have that $p \in T$ if and only if $p \in V$. Thus $p \in T$ implies $p \in V$, which implies that the empty strategy p^* is winning under (V, S) on p. Similarly, $p \notin T$ implies $p \notin V$, which implies that the empty counter-strategy p_* is winning under (V, S) on p.

- Inductive case: the formula is of the form $B \supset C$.

  Assume that $(B \supset C) \in T$. It follows from the maximal consistency of T that $B \notin T$ or $C \in T$. We consider each case in turn.

  - Case: $B \notin T$.

    Since $B \notin T$ and B is an occurrence of a negative subformula of $B \supset C$, we have that $\mathcal{B}(B \supset C)$ is true and thus that $(B \supset C)^*(B \supset C) = W^{\top}(B \supset C) = B$. Further, $(B \supset C)^* \restriction B = B_*$ by the fact that B is an occurrence of a negative subformula of $B \supset C$. But $B \notin T$, so the induction hypothesis implies that B_* is a winning counter-strategy under (V, S) on B. So we see that $(B \supset C)^*$ is the following strategy for player $\top$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(B \supset C)$: at the $\top$-position $B \supset C$, choose the $\bot$-position B; at the $\bot$-position B, play the winning counter-strategy B_*; at the $\top$-position C, play the strategy C^*. Conclusion: $(B \supset C)^*$ is a winning strategy under (V, S) on $B \supset C$.

  - Case: $B \in T$ and $C \in T$.

    Since $B \in T$ and B is an occurrence of a negative subformula of $B \supset C$, it follows that $\mathcal{B}(B \supset C)$ is false and thus that $(B \supset C)^*(B \supset C) = W^{\top}(B \supset C) = C$. Further, $(B \supset C)^* \restriction C = C^*$ by the fact that C is an occurrence of a positive subformula of $B \supset C$. But $C \in T$, so the induction hypothesis implies that C^* is a winning strategy under (V, S) on C. So we see that $(B \supset C)^*$ is the following strategy for player $\bot$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(B \supset C)$: at the $\top$-position $B \supset C$, choose the $\top$-position C; at the $\bot$-position B, play the counter-strategy B_*; at the $\top$-position C, play the winning strategy C^*. Conclusion: $(B \supset C)^*$ is a winning strategy under (V, S) on $B \supset C$.

  Now assume that $(B \supset C) \notin T$. It follows from the maximal consistency of T that $B \in T$ and $C \notin T$. Since B and C are occurrences of negative and positive subformulas of $B \supset C$ (respectively), we have that $(B \supset C)_* \restriction B = B^*$ and $(B \supset C)_* \restriction C = C_*$. But $B \in T$ and $C \notin T$, so the induction hypothesis implies that B^* is a winning strategy under (V, S) on B and that C_* is a winning counter-strategy under (V, S) on C. So we see that $(B \supset C)_*$ is the following strategy for player $\bot$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(B \supset C)$: at the $\bot$-position B, play the winning strategy B^*; at the $\top$-position C, play the winning counter-strategy C_*. Conclusion: $(B \supset C)_*$ is a winning counter-strategy under (V, S) on $B \supset C$.

- Inductive case: the formula is of the form $B \wedge C$.

  Assume that $(B \wedge C) \in T$. It follows by the maximal consistency of T that $B \in T$ and $C \in T$. Since each of B and C is an occurrence of a positive subformula of $B \wedge C$, we have that $(B \wedge C)^* \restriction B = B^*$ and that $(B \wedge C)^* \restriction C = C^*$. But $B \in T$ and $C \in T$, so the induction hypothesis implies that B^* is a winning strategy under (V, S) on B and C^* is a winning strategy under (V, S) on C. So we see that $(B \wedge C)^*$ is the following strategy for player $\top$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(B \wedge C)$: at the $\top$-position B, play the winning strategy B^*; at the $\top$-position C, play the winning strategy C^*. Conclusion: $(B \wedge C)^*$ is a winning strategy under (V, S) on $B \wedge C$.

Now assume that $(B \wedge C) \notin T$. It follows by the maximal consistency of T that $B \notin T$ or $C \notin T$. We consider each case in turn.

- Case: $B \notin T$.

  Since $B \notin T$ and B is an occurrence of a positive subformula of $B \wedge C$, we have that $\mathcal{B}(B \wedge C)$ is false and thus that $(B \wedge C)_*(B \wedge C) = W^{\bot}(B \wedge C) = B$. Since B is an occurrence of a positive subformula of $B \wedge C$, we have that $(B \wedge C)_* \restriction B = B_*$. But $B \notin T$, so the induction hypothesis implies that B_* is a winning counter-strategy under (V, S) on B. So we see that $(B \wedge C)_*$ is the following strategy for player $\bot$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(B \wedge C)$: at the $\bot$-position $B \wedge C$, choose the $\top$-position B; at the $\top$-position B, play the winning counter-strategy B_*; at the $\top$-position B, play the counter-strategy C_*. Conclusion: $(B \wedge C)_*$ is a winning-counter strategy under (V, S) on $B \wedge C$.

- Case: $B \in T$ and $C \notin T$.

  Since $B \in T$ and B is an occurrence of a positive subformula of $B \wedge C$, we have that $\mathcal{B}(B \wedge C)$ is true and thus that $(B \wedge C)_*(B \wedge C) = W^{\bot}(B \wedge C) = C$. Since C is an occurrence of a positive subformula of $B \wedge C$, we have that $(B \wedge C)_* \restriction C = C_*$. But $C \notin T$, so the induction hypothesis implies that C_* is a winning counter-strategy under (V, S) on C. So we see that $(B \wedge C)_*$ is the following strategy for player $\bot$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(B \wedge C)$: at the $\bot$-position $B \wedge C$, choose the $\top$-position C; at the $\top$-position B, play the counter-strategy B_*; at the $\top$-position C, play the winning counter-strategy C_*. Conclusion: $(B \wedge C)_*$ is a winning counter-strategy under (V, S) on $B \wedge C$.

- Inductive case: the formula is of the form $B \vee C$.

  As in the argument for the case $B \supset C$, though with the necessary changes made in the appropriate places.

- Inductive case: the formula is of the form $\neg B$.

  Assume that $\neg B \in T$. It follows from the maximal consistency of T that $B \notin T$. Since B is an occurrence of a negative subformula of $\neg B$, we have that $(\neg B)^* \restriction B = B_*$. But $B \notin T$, so the induction hypothesis implies that B_* is a winning counter-strategy under (V, S) on B. Since $(\neg B)^*(\neg B) = W^{\top}(\neg B) = B$, we see that $(\neg B)^*$ is the following strategy for player $\top$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(\neg B)$: at the $\top$-position $\neg B$, choose the $\bot$-position B; at the $\bot$-position B, play the winning counter-strategy B_*. Conclusion: $(\neg B)^*$ is a winning strategy under (V, S) on $\neg B$.

  Now assume that $\neg B \notin T$. It follows from the maximal consistency of T that $B \in T$. Since B is an occurrence of a negative subformula of $\neg B$, we have that $(\neg B)_* \restriction B = B^*$. But $B \in T$, so the induction hypothesis implies that B^* is a winning strategy under (V, S) on B. So we see that $(\neg B)^*$ is the following strategy for player $\bot$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(\neg B)$: at the $\bot$-position B, play the winning strategy B^*. Conclusion: $(\neg B)_*$ is a winning counter-strategy under (V, S) on $\neg B$.

- Inductive case: the formula is of the form $t : B$.

Assume that $t{:}B \in T$. By our definition of S, we then have that $S(t,B)\!\downarrow$ and $S(t,B) = B^*$. Further, $S(t,B)\!\downarrow$ implies that the root of $\mathcal{T}_S^V(t{:}B)$ is labeled by the strategy $S(t,B) = B^*$. But $t{:}B \in T$ implies $B \in T$ by LP4 and the maximal consistency of T, from which it follows by the induction hypothesis that $B^* = S(t,B)$ is a winning strategy under (V,S) on B. In addition, we observe that $(t{:}B)^*(t{:}B) = W^\top(t{:}B) = B$. But then the strategy $(t{:}B)^*$ for player $\top$ in the pebble game with explicit strategies on $\mathcal{T}_S^V(t{:}B)$ has the following effect: at the $\top$-position $t{:}B$, the $\top$-position B is chosen; however, since the $\top$-position $t{:}B$ is labeled by the winning strategy $S(t,B) = B^*$ and the $\top$-position B is a descendant of the $\top$-position $t{:}B$, the winning strategy B^* takes control of player $\top$'s moves beginning at the $\top$-position B. It follows that $(t{:}B)^*$ is a winning strategy under (V,S) on $t{:}B$.

Now assume that $t{:}B \notin T$. By the definition of S, we then have that $S(t,B)\!\uparrow$, which implies that the root of $\mathcal{T}_S^V(t{:}B)$ is labeled by a forfeit strategy. Player $\top$ therefore forfeits on his first move in the pebble game with explicit strategies on $\mathcal{T}_S^V(t{:}B)$. It follows that any counter-strategy on $t{:}B$ is winning under (V,S) on $t{:}B$. Hence $(t{:}B)_*$ is a winning counter-strategy under (V,S) on $t{:}B$.

So the WS Property indeed holds, and the proof of this theorem is therefore complete. □

So we see that our concept of *model* is not an empty concept. We now use this concept to define our game semantics for the language of LP.

Definition 4.24 (Truth, Validity)**.** Let (V,S) be a model. To say that A is *true in* (V,S), written $V,S \models A$, means that there is a winning strategy under (V,S) on the formula A. To say that A is *valid*, written $\models A$, means that A is true in every model.

Saying that A is true in a model (V,S) means just that there is a winning strategy under (V,S) on A, which is equivalent to saying that there is a winning strategy for $\top$ ("True") in the pebble game with explicit strategies on the game board $\mathcal{T}_S^V(A)$. Identifying the LP Verification Game on the formula A with model (V,S) with the latter pebble game with explicit strategies, we see how it is that LP Verification provides a notion of truth for LP formulas.

Definition 4.25. Let (V,S) be a model. The LP *Verification Game on* A *under* (V,S) is the pebble game with explicit strategies on the game board $\mathcal{T}_S^V(A)$ with players $\top$ ("True") and $\bot$ ("False").

Note that in the particular case of the formula $t{:}B$, we have that $t{:}B$ is true in a model (V,S) if and only if $S(t,B)$ is a winning strategy under (V,S) on B. Thinking of the term t as naming the strategy $S(t,B)$ on B, we are led to our reading "t is a winning strategy on B" for the formula $t{:}B$.

5 Correctness

While we have defined a notion of truth for formulas in the language of LP (Definition 4.24), we still need to check that this notion behaves appropriately; that is, we will prove that

our notion of truth satisfies a *compositionality property*: the truth value of a formula A in a model can be determined using the truth values of the formulas that make up A along with certain properties of the model.

As our first step toward showing that our notion of truth is well-behaved, we prove that there is always exactly one winner in the pebble game with explicit strategies on the game board $\mathcal{T}_S^V(A)$ with model (V, S).

Lemma 5.1 (Determinacy Lemma)**.** For each model (V, S) and each formula A, there is either a winning strategy under (V, S) on A or else a winning counter-strategy under (V, S) on A.

Proof. This lemma can be viewed as a special case of the Gale-Stewart Theorem [12, 15]; nonetheless, it will be instructive for us to prove the result directly for LP Verification. Proceeding, let (V, S) be a model. We show by induction on the construction of the formula A that there is either a winning strategy under (V, S) on A or else a winning counter-strategy under (V, S) on A.

- Base case: the formula is an atom p.

  The empty strategy is winning under (V, S) on p if and only if $p \in V$, and the empty counter-strategy is winning under (V, S) on p if and only if $p \notin V$. Since we have either that $p \in V$ or that $p \notin V$, it follows that there is either a winning strategy under (V, S) on p or else a winning counter-strategy under (V, S) on p.

- Inductive case: the formula is of the form $B \supset C$.

  By the induction hypothesis, there is a either a winning strategy under (V, S) on B or a winning counter-strategy under (V, S) on B; likewise, there is a either a winning strategy under (V, S) on C or a winning counter-strategy under (V, S) on C. We consider three cases.

  - B_* is a winning counter-strategy under (V, S) on B.

    First, let us fix an arbitrary strategy C^* on C. We then define the strategy $(B \supset C)^*$ on $B \supset C$ as follows: at the $\top$-position $B \supset C$, choose the $\bot$-position B; at the $\bot$-position B, play the winning counter-strategy B_*; at the $\top$-position C, play the strategy C^*. It is not hard to see that $(B \supset C)^*$ is a winning strategy under (V, S) on $B \supset C$.

  - C^* is a winning strategy under (V, S) on C.

    First, let us fix an arbitrary counter-strategy B_* on B. We then define the strategy $(B \supset C)^*$ on $B \supset C$ as follows: at the $\top$-position $B \supset C$, choose the $\top$-position C; at the $\bot$-position B, play the counter-strategy B_*; at the $\top$-position C, play the winning strategy C^*. It is not hard to see that $(B \supset C)^*$ is a winning strategy under (V, S) on $B \supset C$.

  - B^* is a winning strategy under (V, S) on B and C_* is a winning counter-strategy under (V, S) on C.

Define the counter-strategy $(B \supset C)_*$ on $B \supset C$ as follows: at the $\bot$-position B, play the winning strategy B^*; at the $\top$-position C, play the winning counter-strategy C_*. It is not hard to see that $(B \supset C)^*$ is a winning counter-strategy under (V, S) on $B \supset C$.

Conclusion: there is either a winning strategy under (V, S) on $B \supset C$ or else a winning counter-strategy under (V, S) on $B \supset C$.

- Inductive case: the formula is of the form $B \wedge C$.

  By the induction hypothesis, there is a either a winning strategy under (V, S) on B or a winning counter-strategy under (V, S) on B; likewise, there is a either a winning strategy under (V, S) on C or a winning counter-strategy under (V, S) on C. We consider three cases.

  - B^* is a winning strategy under (V, S) on B and C^* is a winning strategy under (V, S) on C.

    Define the strategy $(B \wedge C)^*$ on $B \wedge C$ as follows: at the $\top$-position B, play the winning strategy B^*; at the $\top$-position C, play the winning strategy C^*. It is not hard to see that $(B \wedge C)^*$ is a winning strategy under (V, S) on $B \wedge C$.

  - B_* is a winning counter-strategy under (V, S) on B.

    First, let us fix an arbitrary counter-strategy C_* on C. We then define the counter-strategy $(B \wedge C)_*$ on $B \wedge C$ as follows: at the $\bot$-position $B \wedge C$, choose the $\top$-position B; at the $\top$-position B, play the winning counter-strategy B_*; at the $\top$-position C, play the counter-strategy C_*. It is not hard to see that $(B \wedge C)_*$ is a winning counter-strategy under (V, S) on $B \wedge C$.

  - C_* is a winning counter-strategy under (V, S) on C.

    First, let us fix an arbitrary counter-strategy B_* on B. We then define the counter-strategy $(B \wedge C)_*$ on $B \wedge C$ as follows: at the $\bot$-position $B \wedge C$, choose the $\top$-position C; at the $\top$-position B, play the counter-strategy B_*; at the $\top$-position C, play the winning counter-strategy C_*. It is not hard to see that $(B \wedge C)_*$ is a winning counter-strategy under (V, S) on $B \wedge C$.

  Conclusion: there is either a winning strategy under (V, S) on $B \wedge C$ or else a winning counter-strategy under (V, S) on $B \wedge C$.

- Inductive case: the formula is of the form $B \vee C$.

  As in the argument for the case $B \supset C$, though with the necessary changes made in the appropriate places.

- Inductive case: the formula is of the form $\neg B$.

  By the induction hypothesis, there is either a winning strategy under (V, S) on B or else a winning counter-strategy under (V, S) on B. Let us consider each case in turn.

  Suppose B^* is a winning strategy under (V, S) on B. Define the counter-strategy $(\neg B)_*$ on $\neg B$ as follows: at the $\bot$-position B, play the winning strategy B^*. It is not hard to see that $(\neg B)_*$ is a winning counter-strategy under (V, S) on $\neg B$.

Now suppose B_* is a winning counter-strategy under (V, S) on B. Define the strategy $(\neg B)^*$ on $\neg B$ as follows: at the $\top$-position $\neg B$, choose the $\bot$-position B; at the $\bot$-position B, play the winning counter-strategy B_*. It is not hard to see that $(\neg B)^*$ is a winning strategy under (V, S) on $\neg B$.

Conclusion: there is either a winning strategy under (V, S) on $\neg B$ or else a winning-counter strategy under (V, S) on $\neg B$.

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

  We have either that $S(t, B)\!\downarrow$ or else that $S(t, B)\!\uparrow$. Let us examine each case in turn.

  Suppose $S(t, B)\!\downarrow$. Since (V, S) is a model, we have that S is good for V, which implies that $S(t, B)$ is a winning strategy under (V, S) on B. Define the strategy $(t\!:\!B)^*$ on $t\!:\!B$ as follows: at the $\top$-position $t\!:\!B$, choose the $\top$-position B; at the $\top$-position B, play the winning strategy $S(t, B)$. It is not hard to see that $(t\!:\!B)^*$ is a winning strategy under (V, S) on $t\!:\!B$.[15]

  Now suppose $S(t, B)\!\uparrow$. Let $(t\!:\!B)_*$ be an arbitrary counter-strategy on $t\!:\!B$. It is not hard to see that $(t\!:\!B)_*$ is a winning counter-strategy under (V, S) on $t\!:\!B$. After all, the root of $\mathcal{T}_S^V(t\!:\!B)$ is to be labeled by a forfeit strategy by the fact that $S(t, B)\!\uparrow$ (Definitions 4.9 and 4.10).

  Conclusion: there is either a winning strategy under (V, S) on $t\!:\!B$ or else a winning-counter strategy under (V, S) on $t\!:\!B$. □

The Determinacy Lemma (Lemma 5.1) suggests the following definition.

Definition 5.2. Let (V, S) be a model. To say that A is *false in* (V, S), written $V, S \not\models A$, means that there is a winning counter-strategy under (V, S) on the formula A.

Given this definition, we may restate the Determinacy Lemma (Lemma 5.1) in the following way: each formula is either true or false in a model, but never both. In addition, it follows quite easily from the proof of the Determinacy Lemma that our semantics has a compositionality property.

Lemma 5.3 (Compositionality Lemma)**.** Let (V, S) be a model.

- $V, S \models p$ if and only if $p \in V$, where p is an atom.
- $V, S \models B \supset C$ if and only if $V, S \not\models B$ or $V, S \models C$.
- $V, S \models B \vee C$ if and only if $V, S \models B$ or $V, S \models C$.
- $V, S \models B \wedge C$ if and only if $V, S \models B$ and $V, S \models C$.
- $V, S \models \neg B$ if and only if $V, S \not\models B$.

[15] If B^* is a strategy on B satisfying $B^* \neq S(t, B)$, then we could just as well have defined $(t\!:\!B)^*$ as follows: at the $\top$-position $t\!:\!B$, choose the $\top$-position B; at the $\top$-position B, play the strategy B^*. Since the root of $\mathcal{T}_S^V(t\!:\!B)$ will be labeled by the strategy $S(t, B)$ (Definitions 4.9 and 4.10), the Strategy Hand Off Rule will require True play the strategy $S(t, B)$ at the $\top$-position B, even in the case when $(t\!:\!B)^* \restriction B \neq S(t, B)$. So we see that the result does not depend on the way we define $(t\!:\!B)^*$ on the subtree $\mathcal{T}_S^V(B)$ of $\mathcal{T}_S^V(t\!:\!B)$.

- $V, S \models t : B$ if and only if $S(t, B)\downarrow$.

Thus we see that our semantics for the language of LP is well-behaved. All that remains is for us to verify the correctness of our semantics with respect to the theory of LP; that is, we show that the formulas that are valid according to our semantics are exactly the formulas that are provable in LP.

Theorem 5.4. *A is valid if and only if A is a theorem of LP.*

Proof. By induction on the length of a derivation in LP, we show that each theorem of LP is valid. In this induction, we will make frequent use of the compositionality of our semantics (Lemma 5.3).

- LP0. Each scheme for classical propositional logic is valid.

  This follows from the compositionality of our semantics by the usual truth-table arguments for classical propositional logic.

- LP1. $u : (A \supset B) \supset \big(v : A \supset (u \cdot v) : B\big)$ is valid.

  Suppose (V, S) is a model satisfying $V, S \models u : (A \supset B)$ and $V, S \models v : A$. It follows by compositionality that $S(u, A \supset B)\downarrow$ and $S(v, A)\downarrow$. Since S is proper, it follows that $S(u \cdot v, B)\downarrow$. Applying compositionality, we then have that $V, S \models (u \cdot v) : B$.

- LP2. $u : A \supset !u : (u : A)$ is valid.

  Suppose (V, S) is a model satisfying $V, S \models u : A$. It follows by compositionality that $S(u, A)\downarrow$. Since S is proper, it follows that $S(!u, u : A)\downarrow$. Applying compositionality, we then have that $V, S \models !u : (u : A)$.

- LP3. $u : A \vee s : A \supset (u + v) : A$ is valid.

  Suppose (V, S) is a model satisfying $V, S \models u : A \vee s : A$. It follows by compositionality that $V, S \models u : A$ or $V, S \models v : A$. By another application of compositionality, we have that $S(u, A)\downarrow$ or $S(v, A)\downarrow$. Since S is proper, it follows that $S(u + v, A)\downarrow$. Applying compositionality once more, we have shown that $V, S \models (u + v) : A$.

- LP4. $u : A \supset A$ is valid.

  Suppose (V, S) is a model satisfying $V, S \models u : A$. It follows by compositionality that $S(u, A)\downarrow$. Since S is a model, we have that S is good for V, which means that $S(u, A)\downarrow$ implies $S(u, A)$ is a winning strategy under (V, S) on A. Conclusion: $V, S \models A$.

- *Modus Ponens:* if $A \supset B$ and A are valid, then so is B.

  Suppose $\models A \supset B$ and $\models A$. If (V, S) is a model, then it follows from our assumptions that $V, S \models A \supset B$ and $V, S \models A$. Applying compositionality, we then have that $V, S \models B$. Since the model (V, S) was chosen arbitrarily, we have shown that $\models B$.

- *Constant Necessitation:* if c is a constant and A is an axiom of LP, then $c : A$ is valid.

  Let (V, S) be an arbitrary model. Since S is proper, we have that $S(c, A)\downarrow$. Applying compositionality, it follows that $V, S \models c : A$. Since the model (V, S) was chosen arbitrarily, we have shown that $\models c : A$.

We have thus shown that A is valid if A is a theorem of LP.

To prove the converse, we adopt the terminology relating to the notion of *consistency* as defined in the first paragraph of the proof of Theorem 4.23. We then assume that A is not a theorem of LP. It follows that $\{\neg A\}$ is consistent and so may be extended to a maximal consistent set T. Define $V := \{p : p \in T\}$, where p ranges over all atoms, and then define $V' := V \cup \{\neg p : \neg p \in T\}$. Given this particular T, this particular V, and this particular V', we then use the construction that begins at the second paragraph of the proof of Theorem 4.23 to produce a proper strategy map S that is good for V. It follows from the maximal consistency of T that $A \notin T$ and thus that A_* is a winning counter-strategy under (V, S) on A by the WS Property (see the proof of Theorem 4.23 for definitions). Applying the Determinacy Lemma (Lemma 5.1), we have shown that there is a model in which A is not true. It follows that A is not valid if A is not a theorem of LP. □

This completes our verification that our game semantics for LP is both correct and well-behaved.

6 Embeddings and the Internalization Theorem

The concept of *extensive game with perfect information* was introduced by von Neumann and Morgenstern [23]. Following some of Sevenster's notation and naming conventions [22], 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 if and only if there is a string h_2, which may be ϵ, such that $h = h_1 h_2$. 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 if and only if there is a string h_1, which may be ϵ, such that $h = h_1 h_2$. 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 $(\mathcal{H} - \mathcal{H}_t) \to N$ 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* if and only if the sets N, Σ, and $\mathcal{H}$ are all finite. G is *two-player* if and only if $N = \{1, 2\}$ 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 6.1. 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^* if and only if 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* if and only if 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).[16]

If we fix a propositional formula A and a model (V, S), then the pebble game with explicit strategies on the game board $\mathcal{T}_S^V(A)$ is a verification-like extensive game. Accordingly, each instance of the LP Verification Game on a propositional formula can be viewed as a verification-like extensive game. But there is also a sense in which a verification-like extensive game G can be viewed as an instance of the LP Verification Game on a propositional formula A_G whose construction tree faithfully represents the game tree of G.

Proposition 6.2. For each verification-like extensive game

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

there is a propositional formula A_G such that A is valid if and only if there is a winning strategy in G, a winning strategy on A_G (in an arbitrary model) is convertible to a winning strategy in G using four basic operations (defined in the proof below), and a winning strategy in G is convertible to a winning strategy on A_G (in an arbitrary model) using the inverse of these four operations.

Proof. This proof argues that the game tree of G can be faithfully represented by a formula A_G 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 if and only if 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.

[16] See Hodges' exposition [15] of the Gale-Stewart Theorem [12].

- 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 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

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

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

  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 if and only if 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 if and only if 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 if and only if 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

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

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

– $p'(h) := \begin{cases} p(h) & \text{if } h \neq h_1 h', \\ p(h_1 a h') & \text{if } h = h_1 h'; \end{cases}$

– $u'_i(h) := \begin{cases} u_i(h) & \text{if } h \neq h_1 h', \\ u_i(h_1 a h') & \text{if } h = h_1 h'. \end{cases}$

G' has one fewer only-child double-move than does 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:

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

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 if and only if 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 if and only if 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 if and only if 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* if and only if there are at least three moves at h, and h is called a *two-fork* if and only if 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 $\Sigma' := \Sigma \cup \{c\}$;

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

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

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

  There is one fewer 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 if and only if 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 if and only if 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 if and only if 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* if and only if 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 $\Sigma' := \Sigma \cup \{b\}$;
  - $\mathcal{H}' := \{h \in \mathcal{H} \mid (\forall h' \in \Sigma^*)(h \neq h_1ah')\} \cup \{h_1b\} \cup \{h_1bah' \mid h_1ah' \in \mathcal{H}\}$;
  - $p'(h) := \begin{cases} p(h) & \text{if } h \in \mathcal{H}, \\ p(h_1) & \text{if } h = h_1b, \\ p(h_1ah') & \text{if } h = h_1bah'; \end{cases}$

– $u'_i(h) := \begin{cases} u_i(h) & \text{if } h \neq h_1bah', \\ u_i(h_1ah') & \text{if } h = h_1bah'. \end{cases}$

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:

$$s'^*(h) := \begin{cases} s^*(h) & \text{if } s^*(h) \neq h_1a, \\ h_1b & \text{if } s^*(h) = h_1a, \\ h_1ba & \text{if } h = h_1b, \\ h_1bah_2 & \text{if } h = h_1bah' \text{ and } s^*(h_1ah') = h_1ah_2. \end{cases}$$

It follows from our construction that the history h_1bah' in G' is in accordance with the strategy s'^* in G' induced by a strategy s^* in G if and only if the history h_1ah' in G is in accordance with s^*. We thus have that s^* is a winning strategy in G if and only if 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 if and only if 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_G, the formula in the statement of this proposition. First, call a terminal history *positive* in G if and only if it has an even number of ancestors that are parity points; call a terminal history is *negative* G if and only if 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_G in the statement of this proposition be the formula $f(r)$, where r is the unique shortest string in $\mathcal{H}$.

A_G is *letterless*, which means that each atomic formula appearing in A_G is a propositional constant. It thus follows that the winning (counter-)strategy on A_G in LP Verification is independent of any model (V, S). Further, A_G is in the language of propositional logic, which implies that 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_G is valid if and only if there is a winning strategy in G. To do this, we will argue that f is a *tree-isomorphism* between $\mathcal{H}$ and $\mathcal{T}(A_G)$, by which we mean that three conditions are satisfied. We list each condition along with an argument as to why the condition is true.

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

   This condition is satisfied by our definition of A_G.

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_G, 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_G. It follows that terminal histories are in one-to-one correspondence with

leaves of $\mathcal{T}(A_G)$. Observe that for an immediate subformula instance C of a subformula instance B of A_G, we have that B and C are of opposite polarity (as subformula instances of A_G) if and only if $f^{-1}(B)$ is a parity point. We therefore have that an atomic subformula instance p of A_G is positive in A_G if and only if $f^{-1}(p)$ is positive in G. Looking back to how we specified the leaves in $\mathcal{T}(A_G)$ as images of terminal histories, it then follows that player 1 wins at terminal history h in G if and only if True wins the play of A_G ending on $f(h)$ in the LP Verification Game on A_G. It then follows immediately that there is a winning strategy in G if and only if there is a winning strategy on A_G (in an arbitrary model). Since A_G is letterless, it follows that there is a winning strategy in G if and only if A_G is valid. □

So the proof of Proposition 6.2 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 valid and hence provable in LP. Applying the Internalization Theorem (Theorem 2.4), 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 LP axiom B.

  We proved in Theorem 5.4 that each axiom has a winning strategy, so choose a winning strategy B^* on B (any will do). Take B^* as the interpretation of c.

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

  We have already determined a winning strategy $(B \supset C)^*$ on $B \supset C$ that interprets u and a winning strategy B^* on B that interprets v. Since each of $(B \supset C)^*$ and B^* is a winning strategy, it is not hard to see that $(B \supset C)^* \restriction C$ is a winning strategy on C, so take $(B \supset C)^* \restriction C$ as the interpretation of $u \cdot v$.

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

  We have already determined a winning strategy B^* on B interpreting c. But it then follows that any strategy on $c{:}B$ is winning. So choose an arbitrary strategy on $c{:}B$ to 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 6.2 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.

6.1 Example: Obtaining Winning Strategies in Nim

The well-known game of Nim [8] may be viewed as a verification-like extensive game. The initial setup for a play of Nim consists of three separate piles of stones (or other objects of any kind), with each pile having 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 $(a, b, c) \rightarrow_1 (a', b', c')$, if and only if 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 $(a, b, c) \rightarrow_* (a', b', c')$ if and only if 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 $G(2, 1, 0)$ if and only if 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 the first or the second pile, 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 6.2. 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 7. We then perform in order the operations on $G(2,1,0)$ described in the proof of Proposition 6.2. (As we proceed, we will use the terminology from this proof. 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 6.2, 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 8.

The next operation calls for us to remove each only-child double-move; however, the tree in Figure 8 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 9.

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 10.

In Figure 10, 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 tree in Figure 11 by the construction in the proof of Proposition 6.2.

The formula at the root of the Figure 11 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.

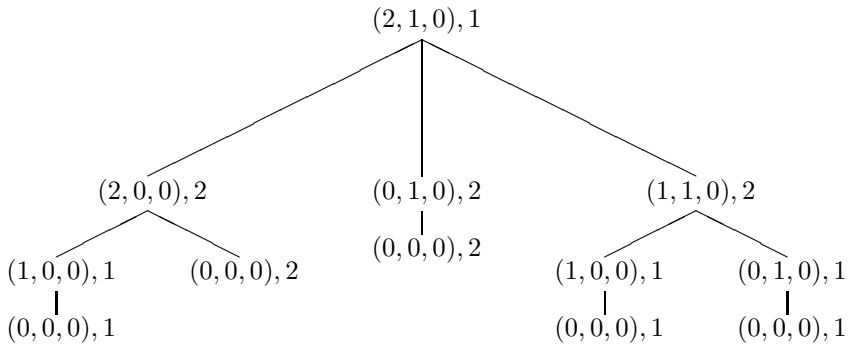

Figure 7. A tree representing $G(2, 1, 0)$ with players 1 and 2.

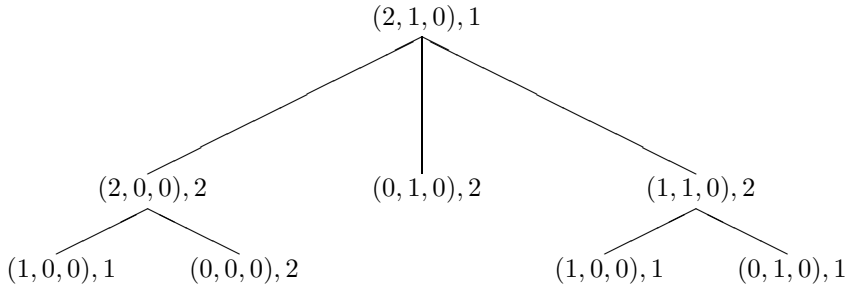

Figure 8. The tree of Figure 7 after tails are collapsed to length 1.

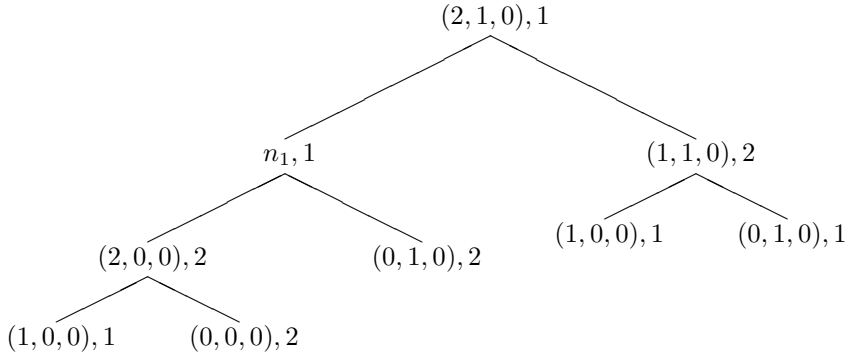

Figure 9. This tree results from converting three-plus forks to two-forks in the tree of Figure 8. This conversion introduced the new node n_1.

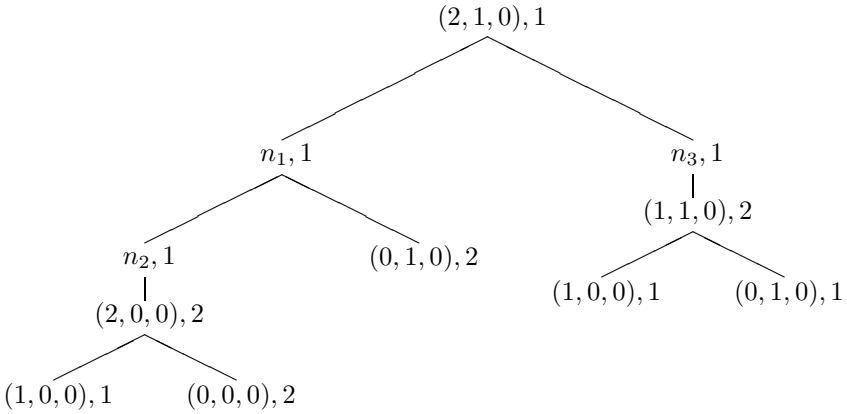

Figure 10. This tree results from introducing double-moves (the nodes n_2 and n_3) into the tree of Figure 9 so that no two-fork is a parity point.

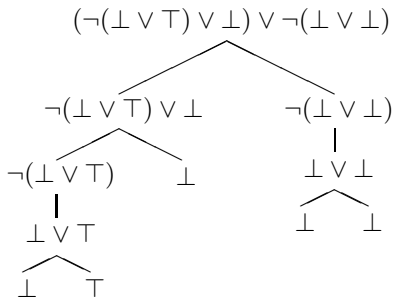

Figure 11. The formula construction tree of $A_{G(2,1,0)}$, a construction tree created from the tree in Figure 10 according to the proof of Proposition 6.2.

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

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 $\bot \vee \bot$;
- map those positive subformulas not already handled in one of the three items above to an arbitrary immediate subformula instance.

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 three of the four bullets above. This strategy on $A_{G(2,1,0)}$ specifies a strategy on the tree in Figure 10, which induces a strategy on the tree in Figure 9 that calls for the move "at $(2,1,0)$, choose $(1,1,0)$" (among other moves). The strategy on the tree in Figure 9 then induces a strategy on the tree of Figure 8, which itself induces a strategy on the tree of Figure 7 that calls for the following moves (among others):

- 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 7 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."

7 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 2.4), 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 model (V, S), 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 express—something we might call *expressivity of strategies*—our LP Verification Game might be extended to the various (multi-)modal extensions of LP [1, 2, 3, 5, 6, 17, 20]. Such extensions 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.

Since LP Verification is an extension of the pebble game, a game that is extended and and massaged to provide semantics for many logics [15], it would be interesting to see how LP Verification might itself be extended or massaged so as to handle interesting fragments of more general frameworks (such as Computability Logic [16]) or other frameworks whose underlying logics are essentially different than that of classical propositional logic (examples include IF logic [14] and the Basic Intuitionistic Logic of Proofs [4]). But we leave these investigations for future work.

8 Acknowledgements

The author would like to thank Sergei Artemov, Roman Kuznets, Evan Goris, two anonymous referees, and the members of the CUNY Computational Logic Seminar for their helpful comments, criticisms, and suggestions.

References

[1] Evangelia Antonakos. Justified and common knowledge: Limited conservativity. In Sergei N. Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, volume 4514 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2007.

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

[3] Sergei Artemov. Justification logic. Technical Report TR-2007019, CUNY Graduate Center Ph.D. Program in Computer Science, 2007.

[4] Sergei Artemov and Rosalie Iemhoff. The basic intuitionistic logic of proofs. *The Journal of Symbolic Logic*, 72(2):439–451, 2007.

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

[6] 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.

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

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

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

[10] Melvin Fitting. Justification logics and conservative extensions. Technical Report TR-2007015, CUNY Ph.D. Program in Computer Science, 2007.

[11] Melvin Fitting. Explicit logics of knowledge and conservativity. Proceeings of the Tenth International Symposium on Artificial Intelligence and Mathematics, 2008.

[12] 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.

[13] Risto Hilpinen. On C.S. Peirce's theory of proposition: Peirce as a precursor to game-theoretic semantics. *The Monist*, 65:182–188, 1982.

[14] 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.

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

[16] Giorgi Japaridze. Introduction to computability logic. *Annals of Pure and Applied Logic*, 123:1–99, 2003.

[17] Roman Kuznets. *Complexity Issues in Justification Logic*. PhD thesis, The City University of New York, 2008.

[18] 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.

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

[20] Bryan Renne. *Dynamic Epistemic Logic with Justification*. PhD thesis, The City University of New York, 2008.

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

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

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