Playing Games with Diagrams: Truth Diagrams and Game Semantics

Can Başkent(✉)

Department of Computer Science, Middlesex University, London, UK
c.baskent@mdx.ac.uk
https://canbaskent.net/logic

Abstract. In this paper we discuss the connection between truth diagrams and game semantics. Truth diagrams offer a diagrammatic way to represent truth in propositional logic. Game semantics, on the other hand, offers a strategic and game theoretical way to establish the truth value of a given formula. By establishing a relation between the two, we offer another diagrammatic reasoning for game semantics, beyond game trees; and characterise various operations on truth diagrams game theoretically.

Keywords: Truth diagrams · Game Semantics · Propositional Logic · Iterated elimination of strictly dominated strategies

1 Introduction

Logic has always enjoyed a variety of different semantics. Semantic tools have included naive sets, topologies, graphs, diagrams and games. In this paper, we establish a connection between a diagrammatic and a game theoretical semantics for propositional logic. Particularly, for the former we focus on the truth diagrams of Cheng and for the latter the game semantics of Henkin and Hintikka.

Cheng's truth diagrams (TDs, for short) suggest a diagrammatic way to understand truth in propositional logic [7]. Diagrammatic reasoning in logic, as Cheng underlined, is not recent, and can be traced back to Frege and Wittgenstein.

Game semantics (GS, for short), on the other hand, suggests an intuitive and strategic methodology to compute the truth value of a given formula in a given model. In a game semantic model, the truth values of the simplest formulas, that is propositions, are known, yet the truth value of the given formula is *computed* by playing a game. The *semantic verification game* for propositional logic is a two-player, sequential (players take turns to make moves), zero-sum (one player wins, the other loses), pure strategy (players do not roll a die), determined (one player always has a winning strategy) and competitive (players do not cooperate) game with rationality (players make moves and strategise with the purpose of winning). It is played by two players, the Verifier and the Falsifier. Their goal is to win the game and establish the truth value of the given formula. If the

J. Lemanski et al. (Eds.): Diagrams 2024, LNAI 14981, pp. 300–315, 2024.
https://doi.org/10.1007/978-3-031-71291-3_24

Verifier wins, then the formula is true. If the Falsifier wins, on the other hand, the formula is false. Moreover, the converses of these theorems also hold. True formulas force the Verifier to have a winning strategy, whereas false formulas force the Falsifier to have a winning strategy.

In this paper, we demonstrate the relationship between game semantics and truth diagrams. What we achieve is two-fold: (i) we present a novel domain for truth diagrams, laden with strategic reasoning, (ii) we develop a new research direction for game semantics. As a result, we *gamify* Cheng's truth diagrams. This establishes a straight-forward way to identify TDs with semantic games, opening up further possibilities between different TDs and game semantics for various other logical systems, including non-classical and modal logics.

2 Brief Review of Truth Diagrams and Game Semantics

We start by reviewing the fundamentals of truth diagrams and game semantics.

2.1 Truth Diagrams

Truth diagrams have three elements: (i) *letters* are used to represent countably-many propositional variables $p, q \ldots$, (ii) *nodes* identify the positions around the letters, and (iii) *connectors* link the nodes to express truth or falsity. In classical logic, there are two nodes for each letter: high and low. The high node is for truth and the low one for falsity. Connectors represent the truth values and are colour-coded: black represents truth (T) whereas grey represents falsity (F).

Figure 1 summarises the TDs for the basic logical connectives using the propositional variables p and q: negation, conjunction, disjunction and implication, respectively. The formula $p \rightarrow q$, for example, fails in classical logic when p is true and q is false. Therefore, there is a grey connector, representing F, between the high node of p and the low node of q. And all the other connectors are black as the formula $p \rightarrow q$ is true for all other truth values of p and q.

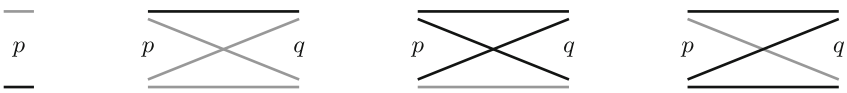


Fig. 1. Truth diagrams for negation, conjunction, disjunction and implication, respectively.

As another example, let us consider the formula $(p \wedge q) \vee p$, given in Fig. 2. The truth table for the aforementioned formula suggests that it is true only when p is true, and false otherwise. The TD reflects this fact diagrammatically.

It is important to note that in a TD, the number of letters and connectors do not depend on the number of subformulas of the given formula, but rather depend on the number of letters (that is propositional variables) appearing in

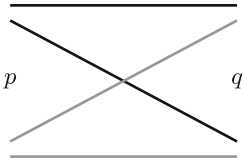


Fig. 2. Truth Diagram for $(p \wedge q) \vee p$ in propositional logic.

the formula. As such, the TD given in Fig. 2 also represents the formula $(p \wedge q) \vee p \vee p \vee p$.

Unlike truth tables, TDs are not necessarily constructed recursively. In truth tables, every subformula needs to be considered by assigning them a column, and the truth values of the given formula depend on the said subformulas. TDs are not recursive, they simply demonstrate the final result of the computation in the truth table by skipping the columns for the subformulas.[1]

We can now define TDs formally, first the elements of TDs, and then their semantics, based on [7].

Definition 1 (Elements of Truth Diagrams). *Elements of truth diagrams for classical logic are given as follows.*

- *A TD is composed of* letters, nodes *and* connectors.
- *Letters are arranged horizontally (with regular spacing for readability).*
- *Nodes are small areas, one above and one below the letters.*
- *Connectors are lines linking nodes. Each connector intersects just one node at each letter and has straight segments that span pairs of immediately adjacent letters.*
- *One connector for each possible combination of high or low nodes of each (type of) letter is permitted: the shape of each connector in a TD is unique.*
- *The style of the connectors is solid and either black or grey.*
- *A TD can contain more than one instance of a letter.*
- *The horizontal order of the letters is arbitrary.*
- *Letters in separate TDs are not linked by connectors.*
- *A connector intersects the nodes at the same level for each instance of the same letter.*

Definition 2 (Semantics for Truth Diagrams). *Semantics of truth diagrams for classical logic is defined as follows.*

- *Letters are propositional* variables.
- *Each node represents a truth-value for the variable: high-node T, low-node F.*
- *The number of the distinct types of variables is the arity of the TD.*

[1] This is an important point from an intuitionistic perspective. If truth is proof, and if proofs are computations, and if computations are strategies, then said proofs, computations and strategies are not visible in TDs.

– *A connector is a* case*: it constitutes a unique set of truth-value assignments to the variables.*
– *Connector style represents the overall truth-value assigned to its case. A black connector assigns T, grey connector assigns F.*

TDs reflect the fundamental properties of classical logic directly. Connectors are functional as propositional logic is. They cannot skip a letter, they cannot double a letter. We must have a connector between every possible truth value combinations of letters, and we cannot have more than one connector between each combination, as truth value gaps and gluts, respectively, are not allowed in classical logic. As such, they demonstrate the strengths and limitations of classical propositional logic directly.

2.2 Game Semantics

The semantic verification game is a sequential, zero-sum, pure strategy, determined and competitive game between two players, the Verifier and the Falsifier. The game is played to establish the truth value of a given formula in a given model. The game is played on a model where the truth values of the propositional letters appearing in the formula are known. What is not known is the truth value of the formula in the model. Players take turn and make moves to play the verification game. The goal of the Verifier is to establish the truth of the formula, and the Falsifier aims at establishing the falsity of the formula. We assume that the players are rational, thus forcing the game to an end for their goal.

The rules of the verification game are given based on the form of the given formula. At conjunctions, the Falsifier makes a move and chooses a subformula, and the game continues with the chosen subformula. Similarly, at disjunctions, the Verifier makes a move. The implication $p \rightarrow q$ is considered as an abbreviation for $\neg p \vee q$ and treated as such. The negation, however, is tricky. At a negation $\neg\varphi$, the players switch their roles and the game carries on with the new roles at φ.

During the game, the given formula is broken down into subformulas step by step. The game terminates when it reaches the subformulas which cannot be broken further down into subformulas and when the players do not have any further move to make – that is the game terminates when it reaches propositional variables. If the game terminates at a propositional letter which is given true at the model, the Verifier (that is the player with the role of the Verifier) wins. Otherwise, the Falsifier wins. The correctness theorem of game semantics shows that the Verifier has a winning strategy in the verification game *if and only if* the given formula is true in the given model. Similarly, the Falsifier has a winning strategy *if and only if* the formula is false.

Let us now formally describe game semantics for classical logic, following [17]. First, some notation. For a given formula φ, the set $Subf(\varphi)$ denotes the set of subformulas of φ, defined in the usual way. For example, for the formula $(p \wedge q) \vee p$, formulas q or $p \wedge q$ belong to $Subf((p \wedge q) \vee p)$. For a sequence of

moves $\sigma = (a_1, \ldots, a_n)$ and a move a, the concatenation of $\sigma + a$ is equal to $(a_1, \ldots, a_n, a)$.

Definition 3. *For a formula φ of propositional logic, and a model M, the semantic game $G(\varphi, M)$ is a two-player, competitive, zero-sum game with perfect information played on the set of histories, defined as follows.*

- *The game has two players: the Verifier and the Falsifier*
- *The set of histories H is defined as $H = \bigcup\{H_\psi : \psi \in Subf(\varphi)\}$ where H_ψ is defined recursively as follows:*
  - *$H_\varphi = \{\varphi\}$,*
  - *If φ is $\neg\psi$, then $H_\psi = \{h + \psi : h \in H_{\neg\psi}\}$,*
  - *If φ is $\psi_1 \vee \psi_2$, then $H_{\psi_i} = \{h + \psi_i : h \in H_{\psi_1 \vee \psi_2}\}$,*
  - *If φ is $\psi_1 \wedge \psi_2$, then $H_{\psi_i} = \{h + \psi_i : h \in H_{\psi_1 \wedge \psi_2}\}$,*
- *Once the game reaches a propositional variable, the game terminates,*
- *If $\varphi = \neg\psi$, then the players switch roles and the game continues with $G(\psi, M)$*
- *If $\varphi = \psi_1 \vee \psi_2$, then the Verifier makes a move and chooses one of ψ_1 or ψ_2, and the game continues with the chosen disjunct $G(\psi_i, M)$*
- *If $\varphi = \psi_1 \wedge \psi_2$, then the Falsifier makes a move and chooses one of ψ_1 or ψ_2, and the game continues with the chosen conjunct $G(\psi_i, M)$*
- *The Verifier wins if the game terminates at a propositional letter p that is true in M, the Falsifier wins if the game terminates at p that is false in M.*

Game trees diagrammatically represent semantic games. Let us consider the formula $(p \wedge q) \vee p$ given in the TD in Fig. 2. Let us further assume that in our model, where we will play the verification game, p is true and q is false. The game tree for this game is given in Fig. 3.

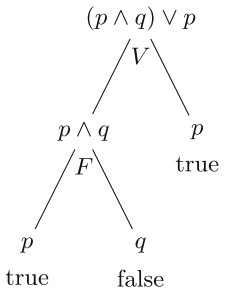


Fig. 3. The game tree for the verification game for the formula $(p \wedge q) \vee p$ where p is true, q is false.

Here, V denotes the player Verifier, and F denotes the player Falsifier. And the truth values of the propositional letters are also specified in the tree. Now, it is easy to see that the Verifier has a winning strategy by choosing p at the very beginning when it is her turn to make a choice. Because otherwise, if the Verifier

chooses $p \wedge q$, then the Falsifier can simply choose a false proposition (that is q) to win the game. This shows that the formula $(p \wedge q) \vee p$ ends up being true, which can also be verified by a simple truth table.

2.3 A Brief Literature Review

Truth diagrams and game semantics are relatively disjoint research areas.

Diagrams, however, are fundamental elements in game theory in terms of representing epistemic, rational and strategic reasoning, including extensive-form game trees (unlike normal-form games). In GS, extensive-form trees are used widely, as given in Fig. 3.

Earlier work on GS (or dialogical logics) goes back to the 1950s where Lorenzen published his first insights [14]. The work of Paul Lorenzen, later with Lorenz, set the foundations of modern game semantics [15]. Leon Henkin also did foundational work in semantic games [13]. Similarly, an earlier work by Parikh discusses similar ideas (much later published as [19]). The popularity of game semantics, however, is largely due to Hintikka and Helsinki school researchers, including its various extensions [12,17,20]. Recently, a broad variety of logics have been given a game semantical characterisation [2,3,5,10,11].

Another stream of research focuses on game semantics for programming languages and proofs [1]. Arguably, this line of research benefits largely from diagrammatic reasoning due to its wide use of category theory, which is a branch of mathematics that relies heavily on diagrammatic reasoning.

Truth diagrams, on the other hand, have been a recent development by Cheng, published in 2020 [7]. However recent they are, they stand on the shoulders of the earlier work of Frege, Peirce and Wittgenstein. TDs for a variety of non-classical logics were recently given to explore how non-classical logics and TDs interact [4]. Diagrammatic reasoning in mathematics and logic is, most certainly, not new, and can easily be traced back to Euclid. As argued by Macbeth, diagrammatic representation in mathematical disciplines offers a "real extension to our knowledge" [16].

3 From Diagrams to Games, and From Games to Diagrams

What GS achieves is that it starts from the truth values of the propositional variables appearing in the formula, and constructs a way to establish the truth value of the formula in question. Take the simple formula given in Fig. 2. GS suggests that, by playing a verification game, when, for example, p is true and q is false, it is possible to establish the truth value of the given formula. So, this is a bottom-to-top approach. Is it possible to introduce a top-to-bottom approach to game semantics?

By using TDs, it is easy to identify under what conditions a certain player would admit a winning strategy. This becomes even easier once we identify the

colour of a connector with a player. Black connector is for the Verifier, and the grey connector is for the Falsifier. In short, black connectors identify the winning strategies of the Verifier, grey connectors identify the winning strategies of the Falsifier. This identification ranges over all possible different inputs for the propositional variables.

In this section we describe a natural methodology to identify connectors with strategies. First, we examine the TD to GS direction, then the GS to TD direction.

Given a TD, it is possible to identify what combinations of truth values render the formula true. These combinations are the models where the Verifier has a winning strategy. Therefore, the black connectors identify the verification games where the Verifier has a winning strategy. Similarly, the grey connectors identify the games where the Falsifier has a winning strategy.

Notice that since the TDs do not deal with subformulas (recall the formula in Fig. 2), they do not carry the information of how the players may win the game nor what the winning strategy is for the winning player. Strategic elements of a game tree cannot be retrieved from a TD.

Conversely, given a game tree and a winning strategy for a player, it is possible to identify a connector in the associated TD. For this, one needs to consider truth value distributions of the propositional letters. In those games, where the Verifier has a winning strategy, the given truth value distribution identifies a black connector. Similarly, when the Falsifier has a winning strategy, the distribution identifies a grey connector. Therefore, each different game identifies one connector. In order to construct a complete TD, one needs to consider, at the worst case, all possible 2^n semantic games for possible combinations of truth value distributions over n distinct propositional letters, appearing in the formula.[2]

These observations can easily be extended to multi-valued truth diagrams, where TDs may admit more than two colours for the connectors. This is akin to allowing more than two players in GS, corresponding to multiple truth values in non-classical logics [2].

What is more interesting now is the game theoretical interpretations of certain operations on truth diagrams.

4 Truth Diagram Operations on Game Trees

As van Benthem argued, extensive-form trees of semantic games of logically equivalent formulas are good examples of the "problem of game equivalence" [6]. Given two game trees for the logically equivalent formulas $(p \wedge q) \vee p$ and $(p \vee p) \wedge (q \vee p)$ in Fig. 4, how can we determine if these games are equivalent? In the first game, the Verifer makes the first move and wins. In the second one, however, the Verifier has to wait for the first move by the Falsifier. In each case the Verifier still has a winning strategy, but the strategies are different.

[2] Some games can have the same output, therefore may not need to be considered.

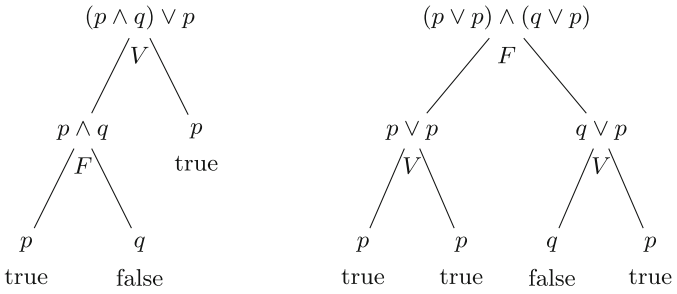


Fig. 4. The game trees for the verification games for the formulas $(p \wedge q) \vee p$ and $(p \vee p) \wedge (q \vee p)$, respectively.

This is a major problem, as it is possible to approach game equivalence from at least two perspectives. The pragmatic approach only focuses on the results, rather than *how* the results are achieved. For pragmatics, the two games above are equivalent. The behaviourist approach, however, considers the way the games are played. For behaviourists, the games are not equivalent as different players take different turns throughout the games and follow different strategies.

We can now ask how TDs can contribute to the understanding of this problem. TDs admit various operations on the diagrams, and by exploring this operations, we can get close to understanding diagrammatic game equivalence. Our goal is to move from TDs to GS by means of diagrammatic operations. In order to achieve this, we start with examining two categories of TD operations: "letter relocation operations" and "composition operations" [7].

Letter Operators. "Letter relocation operator", as Cheng argued, "exploits the arbitrariness of the relative horizontal location of letters in a TD" [ibid]. Using this operator, one can swap, repeat or remove letters as seen in Fig. 5.

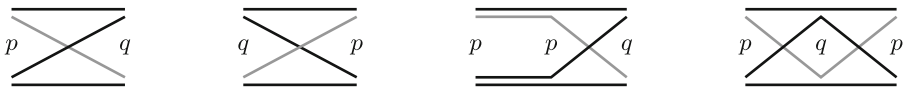


Fig. 5. Relocation operations for the TD for the formula $p \rightarrow q$, as given in [7].

From the viewpoint of game semantics, the first letter relocation operator (switching the positions of p and q for example) does not generate an entirely new game – left moves become right moves, right moves become left moves. Other operators which duplicate the letters introduce new branches to the game as given in Fig. 6. These branches do not introduce any additional strategic power to the game – no player ends up better off after a relocation operation is applied to the game.

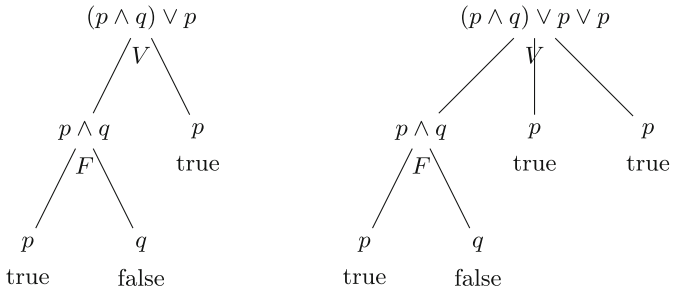


Fig. 6. The game trees for the verification games for the formulas $(p \wedge q) \vee p$ and $(p \wedge q) \vee p \vee p$, respectively, where the latter is obtained after a letter relocation operation is applied to the former one.

Remark 1. Letter relocation operations do not introduce any strategic advantage to any players in semantic games.

Composition Operators. "Composition operators", on the other hand, compose TDs to generate new TDs, as given in Fig. 7 [ibid][3].

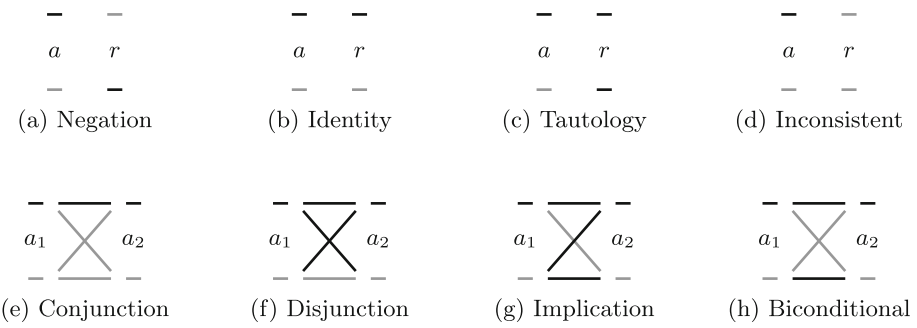


Fig. 7. Unary (top) and binary composition operators, as given in [7].

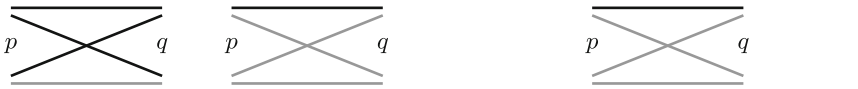
Composition operators take one (if unary) or more inputs, but produces only one output as they are functional. The unary inputs are represented with a whereas the binary ones are a_1 and a_2. As detailed in [7], composition operators describe how TDs can be "composed" using Boolean connectives. Figure 8 shows the composition of $\neg p$ and q using the conjunction operator, which is given in Fig. 7. In the TD for $\neg p \wedge q$, obtained by composition using the conjunction operator, the only black connector occurs between the black input nodes. In this case, they are the bottom node for $\neg p$ and the top node for q. Hence, the black connector in the output TD for $\neg p \wedge q$ is between these nodes.

[3] Cheng places composition operators in dotted-line rectangles which we omit here for simplicity.

(a) Inputs TDs for $\neg p$ and q (b) Output TD for $\neg p \wedge q$

Fig. 8. Composing $\neg p$ and q using the conjunction operator to obtain the TD for $\neg p \wedge q$.

It is also possible to compose two binary formulas. Figure 9 shows the composition of $(p \vee q) \wedge (p \wedge q)$.



(a) Inputs TDs for $p \vee q$ and $p \wedge q$. (b) Output TD for $(p \vee q) \wedge (p \wedge q)$.

Fig. 9. Composing the TDs for $p \vee q$ and $p \wedge q$ using the conjunction operator to obtain the TD for $(p \vee q) \wedge (p \wedge q)$.

The composition operators are relevant from the view point of semantic games. In the sequel, we explore how composition operators can be interpreted game semantically.

Let us first consider the semantic games of $p \vee q$ and $p \wedge q$, and see how we can then generate the semantic game of $(p \vee q) \wedge (p \wedge q)$ using the TD-based composition operations. Semantic game trees for these formulas are given in Fig. 10 in a model where p is true and q is false.

In these game trees, the model is fixed: p is true and q is false. Therefore, one only needs to consider the connectors between the top node of p and the bottom node of q in Fig. 9a. The conjunction operator rule given in Fig. 7e suggests that only black connectors produce a black connector. Therefore, in the given model, under conjunction, the output formula $(p \vee q) \wedge (p \wedge q)$ for top-node of p and the bottom-node for q admits a grey connector, hence false, suggesting the Falsifier has a winning strategy as expected.

Let us now systematically explain this process.

First, composition rules give the first move to certain players: game trees composed under the conjunction operator yield the first move to the Falsifier, those composed under the disjunction operator yield the first move to the Verifier.

Second, as argued in Sect. 3, a global look at the input TDs explain under what conditions and presuppositions, players admit winning strategies. As such, determining whether a player has a winning strategy for the composed game directly depends on who has a winning strategy for the composed games.

Now, in this case, for the formula $(p \vee q) \wedge (p \wedge q)$ in Fig. 9, the conjunction operator gives the first move to the Falsifier. He makes the choice and chooses

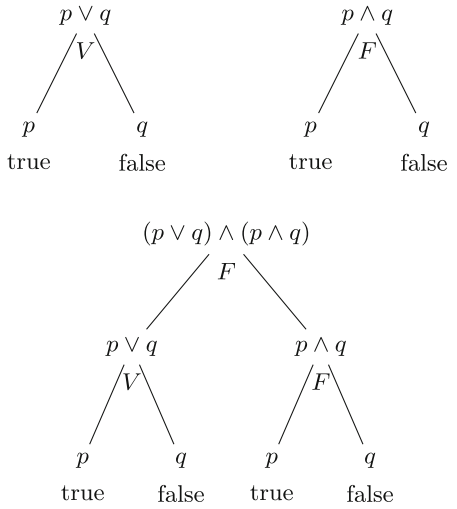


Fig. 10. The game trees for the verification games for the formulas $p \vee q$ and $p \wedge q$, and $(p \vee q) \wedge (p \wedge q)$ respectively.

one of the sub-games, that is one of the inputs. He needs to be able to choose a grey connector to guarantee a win in the subgame.

Let us now consider all four possible input types in order to see whether the Falsifier can win the game under these circumstances. In order for the Falsifier to win, he needs to pick the grey connector.

- If p and q are both true: In both input TDs, the connectors for these cases are black. The Falsifier cannot win, the Verifier has a winning strategy in this case.
- If p and q are both false: In both input TDs, now the connectors are both grey. Therefore, whichever choice the Falsifier makes, he still has a guaranteed win. Hence, in the composed game, the Falsifier has a winning strategy.
- If p is true and q is false: In one of the input TDs (the one for $p \wedge q$), we have a grey connector that the Falsifier can choose. This forms his winning strategy in the composed game.
- If p is false and q is true: In one of the input TDs (the one for $p \wedge q$), we have a grey connector that the Falsifier can choose. This forms his winning strategy in the composed game.

We can now generalise this methodology recursively in order to play semantic games on truth diagrams with connector operations. By doing so, we can game theoretically characterise the TD operations.

Let us start with the rules.

- **Propositional letters**: For a propositional letter p, the Verifier wins for the black connector, the Falsifier wins for the grey connector.

– **Negation**: Players switch colours.
– **Conjunction**: The falsifier makes a choice and selects one of the input games.
– **Disjunction**: The verifier makes a choice and selects one of the input games.

As before, we consider implication as an abbreviation: $p \rightarrow q \equiv \neg p \vee q$. And, similarly, the game terminates when a player reaches a propositional letter and has no more moves to make.

This is how we define GS for operations on TDs. For a given formula φ in a model M, the semantic game on truth diagrams is denoted by $G_{TD}(\varphi, M)$.

Theorem 1. *For a given formula φ in a model M, in the truth diagram semantic game $G_{TD}(\varphi, M)$, $M \models \varphi$ if and only if the Verifier has a winning strategy. Consequently, $M \not\models \varphi$ if and only if the Falsifier has a winning strategy.*

Proof. The proof is by induction on the complexity of φ. We will only show it for the base case, negation and conjunction as the case for disjunction is similar.

For φ is a propositional letter p: If $\varphi = p$, then the game terminates. The Verifier wins if and only if $M \models p$. Similarly for the Falsifier.

For φ is a negation $\varphi = \neg\psi$: For $\varphi = \neg\psi$, players switch colours. The composition operator given in Fig. 7a switches the colours, too. By the induction assumption, then, the Falsifier has a winning strategy in the game $G_{TD}(\psi, M)$. Then, the Verifier has a winning strategy in $G_{TD}(\neg\psi, M)$; hence in $G_{TD}(\varphi, M)$.

The argument for the Falsifier is similar.

For φ is a conjunction $\varphi = \psi_1 \wedge \psi_2$: Let $M \models \psi_1 \wedge \psi_2$. Thus, Let $M \models \psi_1$ and $M \models \psi_2$. By the induction hypothesis, the Verifier has winning strategies for both $G_{TD}(\psi_1, M)$ and $G_{TD}(\psi_2, M)$. For the game, $G_{TD}(\psi_1 \wedge \psi_2, M)$, according to the game rules, the Falsifier makes a choice. Whatever choice he makes, the Verifier ends up having a winning strategy at that choice. Thus, the Verifier has a winning strategy for the game $G_{TD}(\psi_1 \wedge \psi_2, M)$ and consequently for $G_{TD}(\varphi, M)$.

Conversely, let the Verifier have a winning strategy for $G_{TD}(\psi_1 \wedge \psi_2, M)$. It is, however, the Falsifier turn to make a move at a conjunction. Therefore, whatever move the Falsifier makes, the Verifier can still win. Therefore, the Verifier has winning strategies for both $G_{TD}(\psi_1, M)$ and $G_{TD}(\psi_2, M)$. By the induction hypothesis, this means that $M \models \psi_1$ and $M \models \psi_2$, suggesting that $M \models \psi_1 \wedge \psi_2$. Consequently, $M \models \varphi$.

The argument for the Falsifier is similar.

This concludes the proof. □

Note that the statement of Theorem 1 benefits from well-defined TD operations given in Fig. 7. What about the converse? Can we make use of game semantical methods to define new composition operations for TDs?

This question pushes us back to the original question regarding game tree equivalences which we mentioned at the beginning of this section. For the example given in Fig. 4, another way to approach the problem is to determine the operations which preserve the strategic power of the players, even before deciding what makes game trees equivalent. This question is deep and challenging. However, we have one answer.

Iterated Elimination of Strictly Dominated Strategies. In the sequel, we explore a well-known solution method in game theory: *iterated elimination of strictly dominated strategies*.

Iterated elimination of strictly dominated strategies (IESDS, for short) is a solution method in games where those strategies which are strictly dominated by other strategies are removed step by step from the game [18]. A strictly dominant strategy is the one that brings higher pay-off to a player, irrespective of how the opponents play. In short, what IESDS does is that once there are alternatives, the strategies which are *strictly* dominated by the others are not the rational moves to make as the *dominant* strategy would bring a higher pay-off. Thus, the strictly dominated ones can be eliminated.

Consequently, IESDS bears significance for the following reason.

Remark 2 ([18]). An action of a player in a finite strategic game is a never-best response if and only if it is strictly dominated.

The same idea applies to logic, particularly to non-classical logics [3]. For example, in classical logic, under disjunction, the truth value True dominates the truth value False. Therefore, the strategy for the sub-game for the False can be eliminated. This idea can be extended to multi-valued logics, too [3,8,9]. Various non-classical logics contain "a non-classical truth value that is 'contaminating' in the sense that a formula must be assigned that value whenever any of its subformulae are assigned the contaminating value" [8]. The game semantic interpretation of this observation is that the contaminating non-classical truth-value is forced by a player whose strategy becomes *strictly dominant* [3]. And this allows us to take advantage of IESDS in semantic games. TDs for such non-classical logical systems fall outside the scope of the current paper, and are thus left for future work.

What is then the equivalent of IESDS in TD games? Can we eliminate some connectors when we are composing TDs?

Let us start with reconsidering the example of the conjunction operator given in Fig. 9. Under the binary conjunction composition rule (Fig. 7e), the Falsifier can eliminate strategies as follows.

- If connectors between same nodes (such as between high p node and low q node) are of different colour, then the Falsifier eliminates the dominated strategy of the Verifier. In the new TD, the grey connectors dominate for the Falsifier.
- If both connectors that are being composed are grey, the Falsifier maintains his winning strategies and does not eliminate any.
- If both connectors that are being composed are black, the Falsifier has no winning strategy under these circumstances.

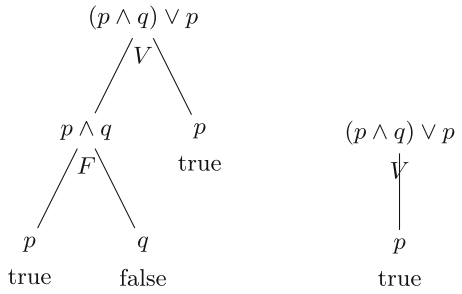


Fig. 11. Iterated elimination of strictly dominated strategies applied to the winning strategy of the Verifier in the game tree of $(p \wedge q) \vee p$ in the given model. The tree on the left turns to the tree on the right after the strategy elimination.

The same methodology applies to the Verifier for disjunction.

- If connectors between same nodes (such as between high p node and low q node) are of different colour, then the Verifier eliminates the dominated strategy of the Falsifier. In the new TD, the black connectors dominate for the Verifier.
- If both connectors that are being composed are black, the Verifier maintains his winning strategies and does not eliminate any.
- If both connectors that are being composed are grey, the Verifier has no winning strategy under these circumstances.

Notice that the Verifier cannot eliminate any strictly dominated strategy under conjunction, and the Falsifier cannot eliminate any strictly dominated strategy under disjunction. The reason for this is that it is not their turn to make a move at those connectives.

Theorem 2. *Game semantics for truth diagrams for classical logic admits the solution method of iterated elimination of strictly dominated strategies as follows.*

- *If connectors between same nodes are of different colour, then the Falsifier eliminates the dominated strategy of the Verifier. In the new TD, the grey connectors dominate for the Falsifier.*
- *If connectors between same nodes are of different colour, then the Verifier eliminates the dominated strategy of the Falsifier. In the new TD, the black connectors dominate for the Verifier.*

Other cases are given as above.

The proof of this argument follows immediately from the discussion above.

The process of IESDS relates to "decomposition" operators in TDs. The decomposition operators "split apart" the arguments of TDs [7]. Classical logic, however, is not mathematically complex enough to generate composition and decomposition rules for classical TDs. We leave it to future work to examine

how various multi-valued logics and their TDs may be characterised by some solution methods in game theory, including the iterated eliminated of *weakly* dominated strategies.

5 Conclusion

In this paper, we established a connection between semantic games and truth diagrams for propositional logic, particularly by focusing on operations on truth diagrams. The immediate extensions of this work is to consider (classical) modal and first-order logics. However portable this methodology is, it is far from trivial to extend truth diagrams to modal or first-order cases by means of game semantics. However, there are benefits. Considering the extensions of propositional logic by means of game semantics provides an alternative to model theoretical methodology to extend truth diagrams to higher order or modal logics. One can ask how the propositional semantic games would change, what additional game theoretical tools would be required, and how strategies need to be advanced to cover modal and higher-order cases.

Non-classical logics is another direction to pursue for truth diagram games. Non-classical logics are often motivated by various philosophical arguments [21]. Diagrammatic reasoning and diagrams themselves are relatively new to non-classical logics, and consequently this is a direction that requires more work.

A direct extension of game semantics is a logical system called "independence-friendly logic" (IF logic, for short) [17]. In IF logic variable dependency of first-order logic can be characterised game theoretically. This is yet another methodology to address the question of truth diagrams for first-order logic by means of focusing on variable dependence (or independence [22]).

Modal and non-classical logics have direct applications in computer science, which can also benefit from diagrammatic reasoning, particularly from truth diagrams. Dynamic epistemic logics (and its power in solving puzzles), knowledge representation (and its power in reasoning with multi-agent systems), and distributed computing (and its power in resource allocation) are some of the areas which can directly benefit from truth-diagrammatic reasoning and representation.

We leave such work for the future.

Disclosure of Interests. The author has no competing interests to declare that are relevant to the content of this article.

References

1. Abramsky, S., McCusker, G.: Game semantics. In: Berger, U., Schwichtenberg, H. (eds.) Computational Logic. NATO ASI Series, vol. 165, pp. 1–55. Springer, Cham (1999). https://doi.org/10.1007/978-3-642-58622-4_1
2. Başkent, C.: Game theoretical semantics for some non-classical logics. J. Appl. Non-Classical Logics **26**(3), 208–39 (2016). https://doi.org/10.1080/11663081.2016.1225488

3. Başkent, C.: A game theoretical semantics for a logic of nonsense. In: Raskin, J.F., Bresolin, D. (eds.) Proceedings of the 11th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2020). Electronic Proceedings in Theoretical Computer Science, vol. 326, pp. 66–81 (2020)
4. Başkent, C.: Truth diagrams for some non-classical and modal logics. J. Appl. Non-Classical Logics (to appear)
5. Başkent, C., Henrique Carrasqueira, P.: A game theoretical semantics for a logic of formal inconsistency. Logic J. IGPL **28**(5), 936–952 (2020). https://doi.org/10.1093/jigpal/jzy068
6. van Benthem, J.: Logic in Games. MIT Press, Cambridge (2014)
7. Cheng, P.C.H.: Truth diagrams versus extant notations for propositional logic. J. Logic Lang. Inform. **29**, 121–161 (2020)
8. Ciuni, R., Ferguson, T.M., Szmuc, D.: Logics based on linear orders of contaminating values. J. Log. Comput. **29**(5), 631–663 (2019). https://doi.org/10.1093/logcom/exz009
9. Ferguson, T.M.: Logics of nonsense and parry systems. J. Philos. Log. **44**(1), 65–80 (2015). https://doi.org/10.1007/s10992-014-9321-y
10. Fermüller, C.G.: Semantic games for fuzzy logics. In: Cintula, P., Fermüller, C.G., Noguera, C. (eds.) Handbook of Mathematical Fuzzy Logic, vol. 3, pp. 969–1029. College Publications (2016)
11. Fermüller, C.G., Majer, O.: On semantic games for Łukasiewicz logic. In: van Ditmarsch, H., Sandu, G. (eds.) Jaakko Hintikka on Knowledge and Game-Theoretical Semantics. OCL, vol. 12, pp. 263–278. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-62864-6_10
12. Hintikka, J., Sandu, G.: Game-theoretical semantics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, pp. 361–410. Elsevier (1997). https://doi.org/10.1016/B978-044481714-3/50009-6
13. Hodges, W.: Logic and games. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy (2013). http://plato.stanford.edu/archives/spr2009/entries/logic-games
14. Lorenzen, P.: Einführung in die operative Logik und Mathematik. Springer, Cham (1955)
15. Lorenzen, P., Lorenz, K.: Dialogische Logik. WBG (1978)
16. Macbeth, D.: Realizing Reason. Oxford University Press, Oxford (2014)
17. Mann, A.L., Sandu, G., Sevenster, M.: Independence-Friendly Logic. Cambridge University Press, Cambridge (2011)
18. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
19. Parikh, R.: D structures and their semantics. In: Gerbrandy, J., Marx, M., de Rijke, M., Venema, Y. (eds.) JFAK, ILLC, UvA (1999). http://www.illc.uva.nl/j50/
20. Pietarinen, A., Sandu, G.: Games in philosophical logic. Nord. J. Philos. Log. **4**(2), 143–173 (2000)
21. Priest, G.: An Introdiction to Non-Classical Logic. Cambridge University Press, Cambridge (2008)
22. Väänänen, J.: Dependence Logic. Cambridge University Press, Cambridge (2007)