Propositions, Boolean Functions, and Paradoxes

David Wyde

January 23, 2024

Abstract

To negate logical connectives such as implication, or gates such as AND and OR in computer science, we flip the output entries of the original function's truth table. The present paper explores whether a similar approach to negation can apply to paradoxes such as the liar, the truth-teller, the card paradox, and Yablo's paradox. The two main ideas are that the liar and the truth-teller are negations of each other, and a general approach to determining the truth of systems of sentences that refer to each other.

1 Introduction

1.1 Background

A previous paper[1] studied how different approaches to self-reference and negation might be able to resolve certain paradoxes related to truth. The present paper aims to provide more support for the earlier paper's claim that the liar and the truth-teller are negations of each other, and to describe a general way of negating, and evaluating the truth of, systems of sentences that refer to each other.

1.2 Propositional Functions

Propositional functions are functions that produce a proposition as output when their free variables are resolved[2]. Propositions that don't refer to any sentences can be seen as 0-ary propositional functions. In classical logic, such propositions have a constant truth value of either true or false.

Propositions that refer to themselves, or to other propositions, can be seen as propositional functions with at least one argument. These references must be replaced before the propositional function can be evaluated as a true or false proposition.

1.3 Negation and Paradoxes

In classical logic, $p \rightarrow q$ is equivalent to $\neg p \vee q$. Its negation, $p \wedge \neg q$, has a truth table in which every output entry is flipped from true to false. In Boolean

algebra and computer science, the negation of an `AND` gate is a `NAND` gate, and the negation of an `OR` gate is a `NOR` gate[3]. In each of those pairs of negated operations, the original function has a truth table in which all of the output entries of its negation's truth table are flipped.

The laws of excluded middle and noncontradiction require that, given a sentence and its negation, exactly one is true. Using classical logic, along with the aforementioned techniques for propositional functions and negation, we will attempt to resolve several paradoxes.

1.4 Outline of This Paper

The present paper treats several paradoxes as propositional functions and considers how to negate them. Section 2 examines how to negate the liar[4], the truth-teller[5], the card paradox[6], and Yablo's paradox[7]. Section 3 attempts to generalize the methods used on those paradoxes into an algorithm for evaluating the truth of, or negating, any system of sentences. Section 4 considers several related approaches. Section 5 summarizes and concludes.

2 Propositional Functions as Boolean Functions

2.1 The Main Idea

This section treats sentences, and systems of sentences, as propositional functions. The examples include basic propositions, sentences with one or two inputs and one output, and finite and infinite systems of sentences that refer to each other. Several of the examples are considered to be paradoxes.

Each example can also be seen as a Boolean function. To produce a truth table for a system of labeled sentences, the values of the references to other sentences are input bits, and the truth values of the sentences are output bits. There are multiple ways to allow sentences to refer to each other. The present paper uses explicit sentence labels, but a system such as Gödel numbering serves a similar purpose[8].

2.2 A Single Sentence

2.2.1 No Input Variables

A proposition that does not refer to any propositions can be seen as a Boolean function with no input variables and one output variable. Its output is constant, either true or false. For example,

a: $2 + 2 = 4$

is a true proposition. On the other hand,

b: $1 + 2 = 4$

is false over the integers.

2.2.2 One Input Variable

A single self-referential proposition corresponds to a Boolean function with one input and one output. There are four possible truth tables for such a function.

Table 1: The truth-teller

In	Out
1	1
T	T
F	F

An example truth-teller sentence is:

1: Tr(1),

where *Tr* is the truth predicate.

Table 2: The liar

In	Out
2	2
T	F
F	T

An example liar sentence is:

2: $\neg$Tr(2).

Table 3: Constant-true, 1 sentence

In	Out
3	3
T	T
F	T

A single sentence can also be constant-true. Examples include

3: Tr(3) $\vee$ $\neg$Tr(3)

and

4: Tr(4) $\vee$ $\top$.

Table 4: Constant-false, 1 sentence

In	Out
5	5
T	F
F	F

The final truth table for a single sentence is constant-false. Examples are

5: $\text{Tr}(5) \wedge \neg\text{Tr}(5)$

and

6: $\text{Tr}(6) \wedge \bot$.

Sentences with the same truth table are logically equivalent. For example,

1: Tr(1)

and

7: Tr(7)

are both instances of the truth-teller[1].

There may be multiple ways of representing a function. (3) and (4) have the same truth table, and are thus the same from the perspective of Boolean functions, but have different references. (3) refers to itself twice, as in a multigraph. On the other hand, (4) refers to itself and $\top$.

2.2.3 Negation

The logical connective $\neg$, like the `NOT` gate in computer science, is a 1-ary operation. $\neg$ takes a proposition as input and produces a proposition with the opposite truth table. `NOT` takes a Boolean variable and flips its value[3].

`NOT` gates and $\neg$ are two forms of negation. Applying a `NOT` gate to the output of a Boolean function negates the output and thus the function. Section 2.3 includes several examples of negation via `NOT`.

In the previous section, the truth-teller and the liar have opposite output truth-tables. Similarly, constant-true and constant-false have opposite truth-tables. Each pair can be seen as a group of two sentences that are negations of each other, given that negation is a function that flips the output entries of a truth table.

2.3 2-ary Functions with One Output

2.3.1 Logic Gates

Some logical connectives, like $\vee$ and $\wedge$, and the corresponding logic gates, are 2-ary Boolean functions with one output.

The $\wedge$ connective corresponds to an `AND` gate. Its negation is `NAND`:

Table 5: `AND` and `NAND` gates

In		Out	
p	q	`AND`	`NAND`
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

`AND` and `NAND` have opposite values for each output entry in their truth table, since they are negations of each other.

The $\vee$ connective corresponds to an `OR` gate. Its negation is `NOR`:

Table 6: `OR` and `NOR` gates

In		Out	
p	q	`OR`	`NOR`
T	T	T	F
T	F	T	F
F	T	T	F
F	F	F	T

Since `NOT` negates an bit, `NAND(p, q)` can also be represented as `NOT(AND(p, q))`, and `NOR(p, q)` as `NOT(OR(p, q))`.

2.3.2 Implication

Implication, $\rightarrow$, corresponds to $\neg p \vee q$ in classical logic. Its negation in classical logic is $p \wedge \neg q$.

As with $\wedge$ and $\vee$, all of the output truth table entries are flipped between the two negated sentences:

Table 7: p $\rightarrow$ q, and its negation

In		Out	
p	q	p $\rightarrow$ q	p $\wedge \neg$q
T	T	T	F
T	F	F	T
F	T	T	F
F	F	T	F

2.4 Two-Sentence Systems

2.4.1 The Card Paradox

The card paradox consists of the following two sentences:

8: Tr(9)
9: ¬Tr(8)

The paradox is that there does not seem to be a consistent assignment of (8) and (9):

Table 8: The card paradox

In		Out		Evaluation	
8	9	8	9	8	9
T	T	T	F	T	Liar
T	F	F	F	Liar	F
F	T	T	T	Liar	T
F	F	F	T	F	Liar

For the Evaluation columns, an entry is labeled `Liar` when the input bit does not match the corresponding output bit.

In the present paper's approach, output bits depend only on input bits, not on other output bits. For the card paradox, Out(8) is true if and only if In(9) is true, and Out(9) is true if and only if In(8) is false.

With an approach in which output bits depend on other output bits, Out(8) depends on Out(9), and Out(9) depends on Out(8). If we try to pick one to use its input to resolve the corresponding output, then use that output for the other input, it is not obvious whether to start with Out(8) or Out(9). Otherwise, we seem to have a deadlock.

The negated system of sentences should have the following truth table:

Table 9: Negation of the card paradox

In		Out		Evaluation	
10	11	10	11	10	11
T	T	F	T	Liar	T
T	F	T	T	T	Liar
F	T	F	F	F	Liar
F	F	T	F	Liar	F

Two sentences meeting that expectation are:

10: ¬Tr(11).
11: Tr(10).

This negation is equivalent to negating each of the original sentences.

By the law of excluded middle, each of (8) and (9) must be true in the card paradox or its negation, and false in the other. So, exactly one of (8) and (10), and one of (9) and (11), should be true. In the original card paradox, Out(8) can be consistently assigned true when In(8) and In(9) are both true. In the negation, Out(11) can be consistently assigned true when In(10) and In(11) are both true.

2.4.2 Analysis of the Card Paradox

The techniques in the present paper make it possible to negate a system of sentences by negating each sentence in turn. Thus, individual sentences and systems of sentences can be treated in a similar fashion.

One drawback to negating the card paradox as above is that it is not possible to consistently assign truth values to both (8) and (9), or both (10) and (11), in either the original system or its negation. Whether that is a problem depends on how we try to generalize the law of excluded middle to systems of sentences, since each of (8) and (9) is separately true in one of the two negated systems.

An alternative approach is to say that the card paradox is a propositional function, with (8) and (9) as free variables. Until the free variables are resolved, it is impossible to convert propositional functions into propositions. In that view, the most complete description we can give of the card paradox is its truth table.

2.5 Infinite Systems of Sentences

Yablo's paradox is an infinite system of sentences:

S_1: For $i > 1$, S_i is not true.
S_2: For $i > 2$, S_i is not true.
...
S_j: For $i > j$, S_i is not true.
...

The above system does not seem to have a consistent truth assignment. We will determine the negation of Yablo's paradox and attempt to find an assignment for that negation.

To get a feel for Yablo's paradox, we can evaluate it after a fixed, finite number of sentences:

Table 10: Yablos's paradox, $i = 2$

In		Out		Evaluation	
S_1	S_2	S_1	S_2	S_1	S_2
T	T	F	?	Liar	?
T	F	T	?	T	?
F	T	F	?	F	?
F	F	T	?	Liar	?

Table 11: Yablos's paradox, $i = 3$

In			Out			Evaluation		
S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
T	T	T	F	F	?	Liar	Liar	?
T	T	F	F	T	?	Liar	T	?
T	F	T	F	F	?	Liar	F	?
T	F	F	T	T	?	T	Liar	?
F	T	T	F	F	?	F	Liar	?
F	T	F	F	T	?	F	T	?
F	F	T	F	F	?	F	F	?
F	F	F	T	T	?	Liar	Liar	?

The all-true and all-false inputs have a liar-like result for every output bit. The pattern holds for any number of sentences in a version of Yablo's paradox. If all of the inputs are taken to be true, then none of the sentences below any given sentence will be false, so every output will be false. If all of the inputs are taken to be false, then all of the sentences below any given sentence will be false, and each output bit will be true.

We can expect that the negation of Yablo's paradox, produced by flipping all of the output truth table entries, will have consistent assignments at those two entries. All of the other entries have at least one non-liar output in the original Yablo's paradox, and will thus have a liar-like result in the negation. In the original system, there will be at least one false input with a true after it, unless the first entry is true and the rest are false, in which case the first sentence is true.

To flip all of the output truth table entries and thereby negate Yablo's paradox, we can negate each sentence in the system:

R_1: For $i > 1$, at least one R_i is true.
R_2: For $i > 2$, at least one R_i is true.
...
R_j: For $i > j$, at least one R_i is true.
...

For example, the original S_2 is

S_2: $\neg\mathrm{Tr}(S_3) \wedge \neg\mathrm{Tr}(S_4) \wedge \neg\mathrm{Tr}(S_5) \wedge \ldots$.

The negation of S_2 is thus

R_2: $\mathrm{Tr}(R_3) \vee \mathrm{Tr}(R_4) \vee \mathrm{Tr}(R_5) \vee \ldots$,

by de Morgan's laws.

In the negated system, there are two consistent truth assignments: all sentences are true, and all sentences are false. As in the truth-teller, it is not clear which assignment to pick. A similar problem was alluded to in [9]. It is possible that the (*) argument from [1], described in Section 4.4, can resolve the issue.

3 An Algorithm for Truth

3.1 A General Approach

This section describes a general algorithm for evaluating the truth of systems of sentences. The sentences may refer to each other or themselves. The result may be a single, consistent truth assignment, or a propositional function that treats different truth assignments for sentences as free variables.

Each output bit is set to either true or false in a given truth table row. The value of an output bit may conflict with the corresponding input bit, which corresponds to a liar-like sentence. Different logics may use different techniques to evaluate such truth table entries.

3.2 The Algorithm

Input: a list of sentences.

Output: a truth table, with one row for each possible truth assignment.

Steps:

1) Create a truth table with one input bit and one output bit for each sentence referred to, or from, within the input sentences.

2) Fill in all possible combinations of input bits in the truth table: 2^n entries for n input bits.

3) Fill in each row of output bits one-at-a-time.

   i) Substitute the input truth value for each reference in a given sentence.

   ii) Evaluate the resulting logical expression to set that sentence's output bit.

   iii) Once an output bit is set, leave it alone: do not recurse by treating it as an updated input bit.

3.3 Negation in the Algorithm

To negate a system of sentences, first negate all of the sentences, then run the above algorithm on the negated sentences. Every output bit should be flipped from running the algorithm on the original system of sentences.

When negating, a self-referential sentence binds to "this sentence" rather than to its original label. This was the technique used for Yablo's paradox in Section 2.5. [1] argues that such an approach avoids paradoxes like the liar, the truth-teller, and Curry's paradox, while maintaining classical logic.

For example, the negation of

1: Tr(1)

is

2: ¬Tr(2),

rather than

12: ¬Tr(1).

With this approach, it is more important to negate the original truth table than it is to refer to the original labels within the negated sentences.

3.4 Algorithm Example

Consider the following pair of related sentences:

13: ¬Tr(13)
14: Tr(13)

There are at least two different ways to compute the truth table for this system of sentences. One approach is to compute the output bits from only the input bits, as in the above algorithm.

Table 12: Compute Out(14) from In(13)

In		Out	
13	14	13	14
T	T	F	T
T	F	F	T
F	T	T	F
F	F	T	F

A different approach is to evaluate Out(13) from In(13), then compute Out(14) from Out(13). This way differs from the above algorithm.

Table 13: Compute Out(14) from Out(13)

In		Out	
13	14	13	14
T	T	F	F
T	F	F	F
F	T	T	T
F	F	T	T

As noted in Section 2.4.1, it is difficult to apply the second approach to the card paradox.

4 Related Work

4.1 Overview

Some alternative approaches to the topics in the present paper are Tarski's hierarchical approach to truth, Kripke's notion of grounded sentences, model theory, and modal logic. Other related ideas include many-valued logic, negation according to Boole, and circuits in computer science.

4.2 Tarski's Hierarchy

Tarski argued for a hierarchy of sentences as a way to avoid paradoxes such as the liar[10]. Each sentence can only refer to sentences below it in the hierarchy, so that no sentence can evaluate its own truth. Tarski used sentential functions, which become sentences when their free variables are resolved, in his analysis[11].

Like Tarski, the present paper's approach to propositional functions argues that certain sentences, which may look like propositions at first, are not propositions.

4.3 Grounded Sentences

4.3.1 Kripke's Approach

Kripke provided an approach for evaluating truth in systems of sentences[9]. The idea is to start from sentences with a definite truth value, which don't refer to truth. Then, fill in the truth values of sentences until there are no updates, at which point the process can stop.

Kripke's paper focused on a many-valued version of the above technique. If a sentence can eventually be assigned a truth value by the process he described, it is grounded. Otherwise, it is ungrounded and receives a third, gap truth value. Paradoxical sentences do not have a definite truth value after any number of updates.

Kripke also offered a variant of his approach that uses only true and false, by assigning a gap-valued sentence false and its negation true. A possible issue with using the two-valued version in classical logic is that, if a sentence and its

negation both receive the gap truth value, it may be hard to say that Sentence X is false and its negation is true: Sentence X could be seen as the negation and its original negation as the main sentence.

4.3.2 Analysis of Grounded Sentences

A sentence that is ungrounded is a propositional function with free variables. In some cases, such as the card paradox, it is not a proposition with a definite truth value until it has an assignment of variables. A simple example is

15: The team won the game.

The truth value of (1) depends on the values of the variables "the team" and "the game".

If a proposition is ungrounded but has a constant-true or constant-false truth-table, such as

3: $\mathrm{Tr}(3) \vee \neg\mathrm{Tr}(3)$,

it may be possible to resolve its truth value. A possible analogy is to $f(x) = x^2$, with $x^2 = 0$, over the integers. We can say it is true only if we know whether $x = 0$. On the other hand, $x^2 + 1 = 0$ is constant-false over the integers.

These examples can be seen as related to model theory and modal logic.

4.4 Model Theory

Model theory studies what values make a particular thing true[12]. It draws a distinction between valid propositions, which are true in all models, and propositions that satisfy a model, which are true in that model but may be false in other models[13].

The notions of validity and satisfaction seem relevant to the present paper's discussion of propositional functions. Truth table input rows can be seen as possible models. If a propositional function is constant-true, it can be seen as valid. If it is constant-false, it has no model. Otherwise, it satisfies some models but not others.

The liar and the truth-teller seem to present a borderline case between a propositional function with free variables, and a constant-like function with forced outputs. The argument (*) from [1] can be seen as:

1) The truth-teller and the liar are negations of each other.

2) The truth-teller has a model, but the liar does not.

3) By the law of excluded middle, the truth-teller is true, and the liar is not true.

4.5 Modal Logic

Modal logic studies whether a sentence is true in all, some, or none of all possible worlds[14]. It evaluates if a sentence is necessary, or whether it is possible[15].

As with model theory, these notions from modal logic can be applied to propositional functions. It is necessary that a constant-true function is true and that a constant-false function is false. Otherwise, a sentence is possibly true and possibly false.

4.6 Many-Valued Logic

The idea that a sentence cannot be assigned truth or false is a common feature of many-valued logics[16]. To say that systems of sentences, such as the card paradox, are propositional functions rather than propositions, leads to not assigning truth values to them.

Both approaches lead to sentences not being assigned a value of true or false, though the approach in the present paper only allows that when the sentence is a propositional function rather than a proposition.

4.7 Boole and Circuits

Boole argued that to negate is to take the complement of a set[17], applying tools from another area to propositional logic. The approach to negation in the present paper, flipping output truth table entries, uses Boolean functions in a fashion similar to digital circuits in computer science.

5 Conclusion

The present paper showed that the liar and the truth-teller are negations of each other. Those two sentences have truth tables in which each of the output bits are flipped, which is a characteristic feature of negation in both propositional logic and digital circuits.

The present paper also gave a general approach to evaluating the truth of, and negating, systems of sentences that refer to each other. Examples of such systems include the card paradox and Yablo's paradox. Systems of sentences can be seen as propositional functions, then treated as Boolean functions. If a propositional function has free variables or multiple consistent assignments, the most complete way of evaluating its truth is with a truth table.

Some of the ideas in the present paper are similar to existing work, such as Tarski's hierarchical approach to truth and Kripke's grounded sentences. The current paper likewise hopes to resolve multiple paradoxes related to truth with a single approach[18].

References

[1] David Wyde. Preserve Self-Reference to Solve Paradoxes. `https://davidwyde.com/docs/wyde-self-reference.pdf`, 2023. Accessed: 2024-01-23.

[2] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica to *56*. Cambridge University Press, 2nd edition, 1999.

[3] Noam Nisan and Shimon Shocken. *The Elements of Computing Systems*. MIT Press, 2nd edition, 2021.

[4] Jc Beall, Michael Glanzberg, and David Ripley. Liar Paradox. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2023 edition, 2023.

[5] Chris Mortensen and Graham Priest. The Truth Teller Paradox. *Logique et Analyse*, 24(95/96):381–388, 1981.

[6] Laurence Goldstein. Fibonacci, Yablo, and the Cassationist Approach to Paradox. *Mind*, 115(460):867–890, 2006.

[7] Stephen Yablo. Paradox without Self-Reference. *Analysis*, 53(4):251–252, 1993.

[8] Ernest Nagel and James R. Newman. *Gödel's Proof*. NYU Press, 2001.

[9] Saul Kripke. Outline of a Theory of Truth. *The Journal of Philosophy*, 72(19):690–716, 1975.

[10] Anita Burdman Feferman and Solomon Feferman. *Alfred Tarski: Life and Logic*. Cambridge University Press, 2004.

[11] Alfred Tarski. *The Concept of Truth in Formalized Languages*. Hackett Publishing Company, 2nd edition, 1983. Translated by J. H. Woodger.

[12] Wilfrid Hodges. Model Theory. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2023 edition, 2023.

[13] C. C. Chang and H. Jerome Keisler. *Model Theory*. Dover, 3rd edition, 2012.

[14] James Garson. Modal Logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2024 edition, 2024.

[15] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2010.

[16] Leonard Bolc and Piotr Borowik. *Many-Valued Logics 1: Theoretical Foundations*. Springer, 1992.

[17] Edwin Mares. Propositional Function. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2019 edition, 2019.

[18] Graham Priest. The Structure of the Paradoxes of Self-Reference. *Mind*, 103(409):25–34, 1994.