Reference, paradoxes and truth

June 26, 2008

Abstract

We introduce a variant of pointer structures with denotational semantics and show its equivalence to systems of boolean equations: both have the same solutions. Taking paradoxes to be statements represented by systems of equations (or pointer structures) having no solutions, we thus obtain two alternative means of deciding paradoxical character of statements, one of which is the standard theory of solving boolean equations. To analyze more adequately statements involving semantic predicates, we extend propositional logic with the assertion operator and give its complete axiomatization. This logic is a sub-logic of statements in which the semantic predicates become internalized (for instance, counterparts of Tarski's definitions and T-schemata become tautologies). Examples of analysis of self-referential paradoxes are given and the approach is compared to the alternative ones.

1 Introduction

After Kripke's fixed-points approach, [14, 4], and the revision theories of truth, [10, 13, 1, 12], which have dominated the stage for the last three decades, in recent years two new approaches to paradoxes have appeared. One uses systems of boolean equations to represent sentences, [18, 15], and captures the paradoxical ones as systems possessing no solutions. It has not received much attention, probably, because its formalism has not been lifted to the level of philosophical reflection over the concept of truth. In the other approach sentences, or rather their tokens, are represented as pointer structures. These explicate often ambiguous referential structure present implicitly in the natural discourse. The semantic value is determined by applying rather complicated operational rules which traverse the pointer structure and assign the value of paradox when failing to assign any truth-value.[1] The clear consequences of the framework for the concept of truth have been elaborated in a series of publications [6, 7, 8].

On the face of it, the two are completely different. One of the first issues addressed in the paper is that, contrary to this appearance, they not only capture the same intuitions but are formally equivalent. This equivalence does not concern the pointer approach with all its technical choices. We will adjust a lot of its details preserving, however, the underlying intuitions. The adjustments result in a specific theory which narrows the range of possibilities left open by the pointer framework. In many cases, evaluation of particular sentences will yield more specific or even different results than do variants of the original proposal. The major difference is that we give denotational semantics to pointer structures which coincides with the solutions of the corresponding systems of boolean equations whenever no paradoxes are involved. This semantics provides a criterion of correctness for possible operational interpretations, eases formulations and proofs of meta-theorems and enables us to introduce a logic in which paradoxical statements can be identified and treated. All these differences notwithstanding, the arising theory of truth, although more specific and axiomatized, is very close to that of the pointer framework.

[1] "Truth-value" means in this article only 'true' or 'false'. The only other semantic value to be considered, 'paradox', is never intended by this name.

The main results and structure of the paper can be summarized as follows. Statements, involving possibly self-reference, can be represented either by systems of boolean equations, Section 2, or by graphs (pointer structures).[2] A natural definition of the admissible assignment of truth-values to a graph makes the representations equivalent in the sense that a statement is paradoxical (does not have a solution) in one representation if and only if it is such in the other, Section 3. Reflection over the intra-linguistic reference (i.e., reference to other statements, not to non-linguistic objects) involved in both representations shows the possibility of capturing the semantic predicates, "is true" and "is false". This requires slight adjustments of the propositional language used in boolean equations. We introduce the assertion operator, which is simply identity on the truth-values. However, in the presence of paradoxical statements, it lifts such statements to the level of propositions, namely, statements possessing a truth-value. We design propositional logic with the assertion operator, which internalizes classical properties of truth, like Tarski's definitions and T-schemata, Section 4. The complete approach is presented in Section 5. The predicates "is true" and "is false" remain internalized and coincide with the internal assertion and negation. The paradoxical character of a statement can be decided by analyzing the system of standard boolean equations or, alternatively, the corresponding graph representation. We give static or denotational, not operational, conditions on the admissible assignments to graphs which allow distribution of three semantic values among the involved statements (nodes of the graph). The theorem showing the existence of such an assignment for every graph implies semantic completeness: every statement and substatement is guaranteed to obtain a semantic value, and the set of possible values is easily decidable. The proof gives also a simple algorithm for constructing admissible assignments. Section 6 gives a series of examples and compares our approach to others, in particular, to pointer structures.

2 Boolean equations

In the literature one often finds examples expressed in the following form. Let x_1, x_2 be the following sentences:

$$\begin{aligned} x_1 &= \text{This sentence is false and the sentence } x_2 \text{ is true.} \\ x_2 &= \text{The sentence } x_1 \text{ is true.} \end{aligned} \tag{2.1}$$

Analyzing the possible cases, one arrives at the conclusion that both sentences must be false. Expressed in this way, the Liar becomes:

$$L = \text{The sentence } L \text{ is false.} \tag{2.2}$$

The notation is useful, if not necessary, for disambiguating sentences in natural language. The sentence "This sentence is false", for instance, can be used for expressing a completely different statement when "this" refers to another statement:

$$z = \text{The sentence } L \text{ is false.} \tag{2.3}$$

Intuitively, L is a paradox but z is false (claiming falsity of L which is not false). There is only a small step from such semi-formal notation to a fully formal representation of statements as boolean equations, as observed in [15, 18]. The example (2.1) is represented as the system of two equations:

$$\begin{aligned} x_1 &= \neg x_1 \wedge x_2 \\ x_2 &= x_1, \end{aligned} \tag{2.4}$$

while the liar statement as the single equation:

$$L = \neg L. \tag{2.5}$$

For the sake of simplicity, we will focus on a minimal propositional syntax with only negation, $\neg$, and conjunction, $\wedge$, and treat other connectives as defined in the standard way. Moreover, we will treat conjunction as n-ary connective, for any finite $n > 1$.

As illustrated by (2.2) and (2.3), the same sentence can be used for expressing different statements. Our formal development addresses only the latter.

[2] We distinguish sentences, which are always sentences in natural language, from statements which are expressed by sentences. This is the same distinction as that between sentence types and sentence tokens in the pointer approach.

A *statement* is a variable in a system of equations. In this paper we consider only systems with finitely many equations. We assume, again only for the sake of simplicity, the systems to be in a normal form, namely, the left hand side of every = consists of a single variable, and every variable occurs at most once on the left of =.

Applying the known techniques, e.g., [16, 15, 17], the existence of solutions for a system of boolean equations can be decided and one can also decide whether the solution is unique. Finding the actual solutions may be computationally more cumbersome but is conceptually equally simple. Thus, the system (2.4) has the unique solution $x_1 = x_2 = \mathbf{0}$, while the equation (2.5) has none.

We divide the class of statements into two disjoint subclasses. A *paradox* is a statement for which there is no solution, while a *proposition* is a statement for which there exists a solution. Propositions comprise thus much more than classical propositions, but they share with the latter the property of having truth-value. A *classical proposition* is a well-founded statement, i.e., one without any circular references. (The notion of reference will be explained in the following section.) A classical proposition, say, $a \wedge b$, is represented as $x = a \wedge b$. (The variables not occurring on the left of =, like a, b, are *sinks*.) By this definition, classical proposition is a proposition – any assignment to the sinks induces a solution to the system.

One may convince oneself about the intuitive plausibility of the above definition of paradox by analyzing a couple of examples. Liar is an obvious case. As another example, "If this statement is true then snow is black" can be represented by the equation $x = x \to b$, which rewritten in the minimal syntax becomes

$$x = \neg(x \wedge \neg b). \tag{2.6}$$

This equation has a solution, namely, $b = \mathbf{1} = x$. However, when $b = \mathbf{0}$ it has none, and this corresponds to the paradoxical character of the statement exactly in such situations.

The following modification of (2.1) is intuitively seen to be a paradox

x_1 = This statement is false and the statement x_2 is true.
x_2 = The statement x_1 is false.

Making x_1 true forces x_1 to be false (due to its first conjunct), while making it false, makes both its first conjunct and x_2 true, i.e., makes x_1 true. Lack of any solution for the system:

$$\begin{aligned} x_1 &= \neg x_1 \wedge x_2 \\ x_2 &= \neg x_1 \end{aligned}$$

gives a formal counterpart of this intuitive analysis. Section 6 analyses a series of classical examples using the full formalism developed in the following sections.

Conversely, given a system of equations with no solution, one can formulate paradoxical statement(s) in natural language. For instance, the system

$$\begin{aligned} x &= y \wedge \neg z \\ y &= \neg x \vee z \\ z &= x \wedge y \end{aligned}$$

has no solutions, [18]. It can represent the following three statements:
x = The statement y is true and the statement z is false.
y = Either the statement x is false or the statement z is true.
z = Both statements x and y are true.
Informal analysis shows the paradoxical character of these three statements.

Remark 2.7 Just as one sentence can be used to make different statements, there is no unique function from statements to sentences in natural language. For instance, the system
$x = \neg y, \quad y = x$
can be taken as representing two sentences, x = "The next sentence is false" and y = "The previous sentence is true", but also as representing a single sentence, for instance, y' = "It is true that this sentence is false" or x' = "This sentence is not true". Such ambiguities can be resolved by choosing explicitly one *root* of the system as the main statement, namely, any variable on the left of = which refers to, or is dependent on, every other variable in the system. This notion will not have much bearing on the formal development so we do not focus on it. A more significant ambiguity may be whether, instead of x', one should not rather consider the statement x'' = "This statement is false". This issue will be resolved when presenting the full approach in Section 5.

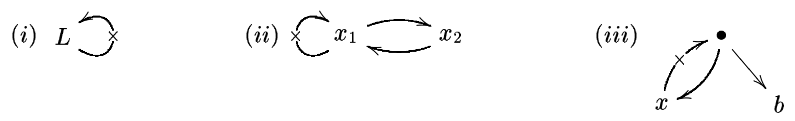
Boolean equations give thus straightforward means for analyzing many statements and deciding their possibly paradoxical character. We now look closer at one aspect of this representation which will be of central importance.

3 Reference and graphs

Names of variables provide means for representing *referential structure* of statements. Roughly, given an equation $x = A$, x refers to A but, transitively, also to each variable occurring in A and to whatever these variables refer. This idea of reference underlies Gaifman's pointer structures [6, 7, 8], where it is represented by a mapping of each variable x (statement name) to a statement $x\!\downarrow\, = A$, to which x refers/points. The graphical representation introduced below is very similar to pointer structures. But it has significantly simpler formalism and static (or denotational), rather than operational, semantic definitions. These make plain their equivalence to boolean equations and offer other advantages over pointer structures, which are discussed in Subsection 6.5.

The reference relation is naturally represented by paths in directed graphs. We let nodes represent variables (statements and substatements) and the outgoing edges conjunctions of confirming or denying references to (sub)statements represented by their targets. A graph $\mathcal{G}$ is a triple (N, EP, EN) where N is a finite set of nodes and $EP, EN \subseteq N \times N$ are collections of positive, respectively, negative edges (the latter marked with $\times$ on the drawings). Given a node x, we write $EP(x), EN(x)$ for the set of positive, respectively, negative neighbours of x, and let $E(x) = EP(x) \cup EN(x)$. The *sinks* of a graph $\mathcal{G}$ are all the nodes without outgoing edges, $sinks(\mathcal{G}) = \{x \in \mathcal{G} \mid E(x) = \varnothing\}$, and the *internal* nodes all others, $int(\mathcal{G}) = \{x \in \mathcal{G} \mid E(x) \neq \varnothing\}$. Considering various assignments to the nodes N of a graph $\mathcal{G}$, we will speak simply about assignments to $\mathcal{G}$.

The liar equation (2.5), the system (2.4) and (2.6) are represented by the respective graphs



(3.1)

Transformation of a finite system of equations (in normal form) into such a graph $\mathcal{G}$ starts by first rewriting right-hand side of every equation into an equivalent formula using only negations and conjunctions.[3] Thus, every equation obtains one of the three forms:

a) $x = y$, where y is a variable
b) $x = \neg B$
c) $x = y_1 \wedge ... \wedge y_n \wedge \neg B_1 \wedge ... \wedge \neg B_m$, where either $n \neq 0$ or $m \neq 0$.

All B, B_j's can be composite formulae. Iterate now through all equations and for each equation $x = ...$ do the following:

1. add the node x to $\mathcal{G}$, if x is not already in $\mathcal{G}$
2. when the equation is of the form
   - a) add a positive edge $x \longrightarrow y$ (add new y if it is not already in $\mathcal{G}$)
   - b) add a negative edge from x to
     - b.1) the node B if B is a variable (new if it is not already in $\mathcal{G}$) (3.2)
     - b.2) a new node u if B is a composite formula; in this case, add also a new equation in normal form (e.g., just after the current one): $u = B$
   - c) add n positive (and m negative) edges from x to the nodes y_i (resp. B_j), each obtained as in the step a), respectively, b).

For instance, given the system with two equations

$$\begin{aligned} x_1 &= \neg x_1 \vee x_2 \\ x_2 &= \neg x_1 \end{aligned} \tag{3.3}$$

[3]We consider only finite graphs as just described, with positive and negative edges, so will not qualify the word "graph" any further.
One can easily extend the definition of graphs annotating nodes by various connectives. This would give more compact graphs but would require treatment of additional cases in definitions and proofs, making the crucial points less transparent. Therefore, without any loss of generality, we formulate the basic theory with our minimal syntax.

we first rewrite $x_1 = \neg(x_1 \wedge \neg x_2)$. This equation gives, by b.2), $x_1 \xrightarrow{\times} u$ and the new equation $u = x_1 \wedge \neg x_2$. This new equation leads, by c), to adding the new node x_2 and two edges from u, namely, $\begin{tikzcd} x_1 \arrow[r, bend left, "\times"] & u \arrow[l, bend left] \arrow[r, "\times"] & x_2 \end{tikzcd}$. Finally, the equation $x_2 = \neg x_1$ gives, by b.1) the last edge and the resulting graph

$$\begin{tikzcd} & u \arrow[dl] \arrow[dr, "\times"] & \\ x_1 \arrow[ur, bend left, "\times"] & & x_2 \arrow[ll, "\times"] \end{tikzcd} \tag{3.4}$$

Conversely, from any graph we obtain a system of equations by letting nodes become variables and introducing an equation $x = Frm(x)$ for every internal node $x \in int(\mathcal{G})$. The formulae associated with the nodes give equations a), b) or c), depending on the outgoing edges:

a) if $E(x) = EP(x) = \{y\}$, i.e., x has only single positive edge to y, $Frm(x) = y$
b) if $E(x) = EN(x) = \{y\}$, i.e., x has only single negative edge to y, $Frm(x) = \neg y$
c) otherwise, when $|E(x)| > 1$, i.e., x has multiple outgoing edges, $Frm(x) = \bigwedge_{y \in EP(x)} y \ \wedge \bigwedge_{z \in EN(x)} \neg z$, i.e., the conjunction of the targets of the positive outgoing edges and the negations of the targets of the negative outgoing edges. (3.5)

For instance, from the graph in (3.4), we obtain the system of equations:

$$\begin{aligned} x_1 &= \neg u \\ u &= x_1 \wedge \neg x_2 \\ x_2 &= \neg x_1 \end{aligned} \tag{3.6}$$

The system is not identical to that in (3.3) from which the graph (3.4) was obtained. But the original system can be obtained from this one by substituting $x_1 \wedge \neg x_2$ for u in the first equation and the two are equivalent. Strictly speaking, two systems of equations $\mathcal{E}_1$ and $\mathcal{E}_2$ are *equivalent* iff their solutions are identical. This is not the case here, since variable u appears in one system but not in the other. But we can loosen the notion of equivalence and claim it in the sense that every solution for the latter contains a solution for the former (considering only assignments to variables occurring in the former), while every solution for the former determines a solution for the latter (by assigning to u whatever is determined by the assignment to x_1 and x_2). Conversely, starting with a graph $\mathcal{G}$ and constructing a system of equations $\mathcal{E}$ from it, then the graph $\mathcal{G}'$ obtained from $\mathcal{E}$ will be isomorphic to $\mathcal{G}$. Since we are interested primarily in the solutions of systems, no confusion should arise when, instead of speaking about one-to-one correspondence between isomorphism classes of graphs and equivalence classes of systems of equations, we simply speak about graphs and systems of equations corresponding to each other.

Graphs and systems of boolean equations are thus two equivalent representations of statements in natural language. This equivalence extends to their solutions. A *solution* for a graph $\mathcal{G}$ is a function $\alpha \in \{\mathbf{0},\mathbf{1}\}^{\mathcal{G}}$ satisfying the following condition for every internal node $x \in int(\mathcal{G})$:

$$\alpha(x) = \overline{\alpha}(Frm(x)), \tag{3.7}$$

where $\overline{\alpha}$ denotes the extension of the valuation α of variables $\mathcal{G}$ to the boolean expressions.

For instance, assigning $\alpha(x_2) = \mathbf{0}$ in the graph (ii) from (3.1), forces also $\alpha(x_1) = \mathbf{0}$ and this α satisfies the above condition. Assigning, on the other hand $\phi(x_2) = \mathbf{1}$ would require that $\phi(x_1) = \mathbf{1}$ iff $\phi(x_1) = \mathbf{0}$, since x_1 is its own negative neighbour. The equation $L = \neg L$ does not have a solution, and this is now captured the fact that solution for the graph (i) in 3.1) would have to make $\alpha(L) = \mathbf{1}$ iff $\alpha(L) \neq \mathbf{1}$.

The condition (3.7) makes explicit the fact that solution need not be obtained by an iterative process starting from some well-defined basis, as is the case in classical logic. It is holistic, putting *simultaneous and mutual* restrictions on all nodes of a graph. Consequently, and unlike in the classical logic, compositionality fails. The truth-value of "the whole" is not always determined by the truth-values of its components considered in isolation. The graph (i) in (3.1) does not have any solution but "embedded", as x_1, into (ii) it acquires a solution $x_1 = \mathbf{0}$, thanks to the wider context in which it now appears.

In general, a graph $\mathcal{G}$ obtained from a system of equations $\mathcal{E}$ may have more nodes than there are variables in $\mathcal{E}$ (e.g., (2.6) and graph (iii) in (3.1)). But this is only a consequence of $\mathcal{G}$ representing an expanded but equivalent system of equations, as explained in the paragraph below (3.6). We can thus treat solutions for $\mathcal{E}$ and the corresponding $\mathcal{G}$ as essentially the same functions. The condition (3.7) makes the following theorem obvious.

Theorem 3.8 *Every solution for a system of equations determines a unique solution for the corresponding graph and vice versa.*

In particular, a system of equations has a solution iff the corresponding graphs does. In the sequel, saying "system" we will not distinguish whether we speak about a graph or a system of equations, treating them as two equivalent representations.

3.1 Reference

The arising model gives a direct representation of intra-linguistic reference. It allows us to study phenomena of circularity at the propositional level, avoiding technicalities (extending first-order logic with arithmetics, Gödel numbering, etc.) which are otherwise necessary to address them. As we will see, this simplification does not impoverish the view of self-reference – rather, it only extracts its essential aspects.

Formally, in the graph representation, a statement (a node) refers to another statement (node) iff there is a path from the former to the latter. A statement refers to its substatements. E.g., $x = y \wedge \neg z$ refers positively to y and negatively to z. This purely syntactic aspect has the semantic counterpart expressed by the condition (3.7). According to it, syntactic reference reflects the truth-functional dependence of a statement on its substatements. This reference to substatements amounts to reference to their truth-value: x refers to y by claiming its truth and to z by claiming its falsity.

So far, this might be only an unusual reading of statements as *referring* to (the truth-values of) their arguments. One might object: a statement does not refer to (claiming truth or falsity of) its arguments, it *uses* them! This use, however, amounts exactly to the use of their truth-values, an implicit reference to these values. In natural discourse, we would not distinguish between (the truth-conditions of) the following statements:

(a) It is raining and the moon is not round.

(b) It is true that it is raining and it is true that the moon is not round.

(c) It is true that it is raining and the moon is not round.

We would not distinguish between them because they say exactly the same. The difference may be that in (a) the truth-values of the substatements are referred to only implicitly, while in (b) and (c) more explicitly. This, however, is at most a difference of emphasis. The contribution of these truth-values to the whole statement is in all cases the same. This reference to substatements is represented by the edges of our graphs and condition (3.7) makes it into truth-reference.

The explicit positive truth-reference, the claim "It is true that...", does not appear in propositional logic for the simple reason of its redundancy. As long as all variables take one of the only two truth-values, the assertion "It is true that..." acts simply as the identity and is definable as double negation. Now, a node x with only a single positive edge, $x \longrightarrow y$, can be said to claim truth of y, since x is true iff y is. But this is just the equation $x = y$. However, these two readings become different once an additional value (or lack of truth-value) is possible. The following system

$$\begin{tikzcd} x \arrow[r] & y \arrow[loop right, "\times"] \end{tikzcd} \tag{3.9}$$

has no solution, since there is no solution for y. Marking this by a third value, $y = \bot$, the equational reading would now make $x = \bot$, while the truth-referential one, according to which x says that y is true, would make $x = \mathbf{0}$.

Since graphs can represent statements having no truth-value, we must choose one of these readings. Choosing the equational one, there would not be much to add. There is an equivalence between systems of boolean equations and graphical representation and both are capable to distinguish between propositions and paradoxes. Theorem 3.8 summarizes this equivalence and the paper would end here.

Choosing the truth-referential interpretation, however, has reasons other than the mere wish to continue writing. For the first, it suggests the possibility of an explicit treatment of the "is true" predicate, which remains at best implicit and at worst absent in boolean equations. The statements (a), (b) and (c) above are represented by the same system of equations. This might be an advantage, since we claimed their equivalence. But this equivalence concerns only the solutions and their existence. It says nothing about cases when no solutions exist.

Having no solutions, (3.9) is equivalent, for instance, to $\begin{tikzcd} x \arrow[r, "\times" marking] & z \arrow[r] & y \arrow[loop right, "\times"] \end{tikzcd}$. Analysis of more subtle paradoxical phenomena would be hindered. The equation $x = \neg x$, for instance, would represent both the Liar and the Strengthened Liar. These shortcomings of the equational reading amount to the fact that, although systems like (3.9) can be declared paradoxical, one does not obtain any closer identification of the reasons of failure. A system either has a solution or not and, in the later case, it concerns the whole system, even if the failure occurs in some specific part of it. In cases like (3.9), however, one would like to express that failure occurs at y and not at x. These are the issues addressed in the rest of the paper. As far as the paradoxical character of statements is concerned, nothing will be changed nor added to the already stated results, which are summarized in Theorem 3.8 and which would remain the same also with the equational reading. But we will offer an analysis of statements which, even when paradoxical, may possess substatements with well-defined truth-values. This analysis will also yields an internal theory of the "is true" predicate.

So, from now on, we read every positive edge as saying "the target is true". This applies equally to single positive edges, like in (3.9), and to multiple ones. A node x with positive edges to y_1 and y_2 represents the statement $x =$ "y_1 is true and y_2 is true" which is identified with the simple conjunction $x =$ "y_1 and y_2". Identifying thus positive reference with the claim of truth, there is no reason to distinguish the two in the negative case. Reading negative edge as "the target is false", we identify the statements "z is false" and "not z". This identification of the semantic meta-predicates with the object-level connectives may raise serious concerns. Such concerns will, hopefully, disappear as we present the formal treatment of this reading.[4]

In the following section, we give a view of propositional logic as arising from natural discourse exactly by means of applying such "meta-predicates" to arbitrary statements. This extended logic will in Section 5 serve to generalize the condition (3.7) determining admissible assignments, to internalize the semantic predicates and to analyze arbitrary statements.

4 Propositional logic: assertions and denials

The fundamental feature distinguishing propositions from statements expressed in natural language is that the former always have one of the two truth-values, while the latter may fail to have any. Consequently, one views propositional logic as a subset of natural language and reasoning, comprising only classical propositions. This is hard to dispute, but one can also ask how this subset became isolated. We certainly won't inquire here into the history of the idea nor into technical aspects distinguishing classical propositions from others. But isolation of propositions from other statements can be plausibly assumed to be due to the interest in truth and falsity. The basic question, "What is truth?", involves various particular questions of the form "Is x true?" or "Is it true that x?", where x is some actual statement. Whatever the answer to such questions, they lift x from the sphere of natural discourse to the level of propositions. For no matter the truth-value of x, and even whether it has any truth-value at all, the statement "It is true that x" can be assigned a unique truth-value. The same happens when stating "It is false that x". Propositions arise from natural discourse by the claims of truth or falsity of arbitrary natural statements, by assertions or denials. The bivalence of propositions is due to this character of meta-statements. Claims of truth and falsity appear thus as the basis of propositions, not as meta-predicates to be added on top of them.

4.1 Semantics and properties of the truth-operator

Our propositional logic is only a formalization of the above picture. It admits the possibility of statements which are neither true nor false, but whenever such a statement x appears, it is immediately turned into a proposition by claiming $True(x)$ or $False(x)$. The third value, $\bot$, can appear only by being assigned to the variables, never as a value of a composite formula. $False(y)$ is identified with $\neg y$, and we denote $True(y)$ by $!y$. We also include usual

[4]This complete embedding of the semantic meta-predicates into the object-language is one minor difference from Gaifman's pointer structures, where object-negation 'not', $\neg$, is distinct from 'is false', Fa, representing meta-negation which, however, also belongs to the object-language. Even if, for instance, $\neg(Fa(x))$, $Fa(Fa(x))$, $Fa(\neg x)$ could be taken to represent different sentences from natural language, distinguishing between their truth-conditions seems unnecessary.

connectives, $\wedge, \vee, \rightarrow$. Their truth tables, motivated by the discussion in 3.1, are as follows:

	$!$	$\neg$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\perp$	$\mathbf{0}$	$\mathbf{0}$

$\wedge$	$\mathbf{1}$	$\perp$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\perp$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$

$\vee$	$\mathbf{1}$	$\perp$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$
$\perp$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$

$\rightarrow$	$\mathbf{1}$	$\perp$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\perp$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$

(4.1)

All the tables are classical when restricted only to the truth-values, $\{\mathbf{0}, \mathbf{1}\}$. The table for $\wedge$ reflects the implicit assertion of arguments, e.g., $\perp \wedge \mathbf{1} = \mathbf{0}$ since the first argument, being implicitly asserted, yields $\mathbf{0}$. The same applies to $\vee$. The table for $\rightarrow$ does not seem to conform to this pattern of asserting each argument. The reason is that implication does not assert them in the same way as conjunction and disjunction do. According to the classical definition, implication either asserts the consequent or denies the antecedent. The above tables make the equivalence $(A \rightarrow B) \leftrightarrow (\neg A \vee !B)$ valid (where biconditional $A \leftrightarrow B$ is defined as $(A \rightarrow B) \wedge (B \rightarrow A)$). Still, the values for the implication involving $\perp$ may rise some doubts, in particular, the fact that $\perp \rightarrow \perp = \mathbf{0}$. Informal reason might be that we do not want to conclude anything from a paradox unless it can be concluded without it (the case $\perp \rightarrow \mathbf{1} = \mathbf{1}$). This distinguishes paradox from contradiction in the propositional context. Another reason might be that claiming propositional equivalence, $A \leftrightarrow B$, we want to be sure that both sides are propositions. A more technical reason is that this makes $\rightarrow$ definable in the classical way from $\vee$ and $\neg$, **t1** below. Having also the equivalences $!A \leftrightarrow A \wedge A$ and $(A \vee B) \leftrightarrow \neg(\neg!A \wedge \neg!B)$, all the mentioned connectives are definable from $\neg$ and $\wedge$.[5]

The following examples of tautologies should give some feeling of the logic.[6]

t1. $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$, and
$(A \rightarrow B) \leftrightarrow (A \rightarrow !B)$
t2. $(A \wedge B) \rightarrow \neg(\neg A \vee \neg B)$
t3. $\neg(A \wedge \neg A)$
t4. $A \rightarrow \neg\neg A$
t5. $\neg\neg\neg A \leftrightarrow \neg A$
t6. $\neg A \rightarrow \neg!A$
t7. $!A \rightarrow A$
t8. $!A \vee \neg!A$, and $A \vee \neg!A$
t9. $!A \leftrightarrow !!A$
t10. $(A \vee \neg A) \leftrightarrow (!A \leftrightarrow A)$, and
$(A \vee \neg A) \leftrightarrow (!A \leftrightarrow \neg\neg A)$

(4.2)

In general, none of the above implications can be reversed. However, since the only deviance from the classical logic concerns the possible value $\perp$ at the atomic formulae, all classical tautologies remain valid for composite formulae.

As we see from **t3**, the principle of non-contradiction remains valid but, like in most three-valued logics, the excluded middle fails. **t8** give its counterparts. The law fails, of course, for $A = \perp$, for which we have $A \vee \neg A = \mathbf{0}$. This, however, leads to another interesting tautology, namely, the equivalence of the excluded middle and non-paradoxicality, **t10**. The equivalence $A \leftrightarrow !A$ holds whenever A is $\mathbf{0}$ or $\mathbf{1}$ and this reflects irrelevance, or redundancy, of such an assertion or truth operator in classical logic. The equivalence fails, however, iff A is paradoxical, and this is equivalent to A having no truth-value at all. Thus, Tarski's T-schema, $!A \leftrightarrow A$, can be applied internally whenever A has a boolean value. In general, however, only assertion of a statement implies the statement, **t7**, while the inverse implication, $A \rightarrow !A$, fails exactly when A is not making any proposition. We then do not view the implication as true, because paradox is not a proposition and does not enable us to conclude any proposition.

Thus the failure of the T-schema is equivalent to the failure of the excluded middle, as well as of the equivalence of assertion and double negation, as shown by the second equivalence in **t10**. This failure does not seem to be any shortcoming of the theory of truth as argued, for instance, in [5]. It rather expresses in a precise way the limitations of bivalent logic.[7]

[5] The formula $A \wedge A$ is not expressible in the graph language since we can not have multiple positive edges from one node to another. But this formula is equivalent to the assertion of A, i.e., $A \wedge A \leftrightarrow !A$, and thus corresponds to a single positive edge. Like in occasional daily situations, repetition amounts to assertion by emphasis.

[6] Only interpretation in the canonical structure over $\{\mathbf{0}, \mathbf{1}, \perp\}$ is considered, with the operations defined by the above tables and $\mathbf{1}$ being the only designated element. A is a tautology, $\models A$, iff it evaluates to $\mathbf{1}$ for every assignment $\alpha \in \{\mathbf{0}, \mathbf{1}, \perp\}^X$ to the variables X occurring in A.

[7] As $!A$ is true exactly when A is, the intuitions of the T-schema are preserved at the meta-level where we have the equivalence for every $\alpha : \alpha \models A$ iff $\alpha \models !A$.

One might miss in the above considerations the more general equivalence of two formulae which have the same, possibly paradoxical, value for every assignment to the variables, $A \overset{\perp}{\leftrightarrow} B$. This is definable by defining the predicate of being a paradox, $\perp A \leftrightarrow (\neg\neg A \wedge \neg!A)$, and then $(A \overset{\perp}{\leftrightarrow} B) \leftrightarrow ((\perp A \wedge \perp B) \vee (A \leftrightarrow B))$.

Another interesting feature of the above logic is internalization of Tarski's definitions of truth. The following table lists its clauses with the tautologies of our logic which are their internal versions. (X is the set of all relevant variables.)

	for every $\alpha \in \{\mathbf{0},\mathbf{1}\}^X$	for every $\alpha \in \{\mathbf{0},\mathbf{1},\bot\}^X$
(and)	$\alpha \models A \wedge B$ iff $\alpha \models A$ and $\alpha \models B$	$\alpha \models\ !(A \wedge B) \leftrightarrow (!A \wedge !B)$
(or)	$\alpha \models A \vee B$ iff $\alpha \models A$ or $\alpha \models B$	$\alpha \models\ !(A \vee B) \leftrightarrow (!A \vee !B)$
(neg)	$\alpha \models \neg A$ iff $\alpha \not\models A$	$\alpha \models\ !\neg A \leftrightarrow \neg A$
(imp)	$\alpha \models A \rightarrow B$ iff $\alpha \not\models A$ or $\alpha \models B$	$\alpha \models\ !(A \rightarrow B) \leftrightarrow (\neg A \vee !B)$

The intuitive obviousness of, say, the (and) clause reflects the common-sense understanding that to state a conjunction amounts to claiming truth of both arguments. What causes trouble, when seen for the first time, is the distinction between the object-language and the meta-language, between $\wedge$ and 'and'. Incorporating the assertion of truth into the basic language, assertion of conjunction is equivalent to the assertion of its arguments. The internal version of the (and) clause is the tautology shown in the right column. The case of the (or) clause is analogous. The situation with the clauses (neg) and (imp), involving negation, is slightly different. As long as the involved formulae obtain only truth-values, we have the corresponding internal equivalences, e.g., for the clause (neg), the tautology $(A \vee \neg A) \rightarrow (\neg !A \leftrightarrow\ !\neg A)$. But when A is $\bot$, the implication $\neg !A \rightarrow\ !\neg A$ fails. As in the case of the failure of the internal T-schema above, this is not a shortcoming of the described internalization, but rather a precise expression of the limitations of these classical definitions to the truth-valued context. Truth of negation, $!\neg A$, is internalized coinciding with the plain negation which carries the same meta-character as assertion.

We now give a sound and complete adjustment of Gentzen's system for the introduced logic, which makes identification of its tautologies close to trivial.

4.2 Propositional reasoning with "external" paradoxes

Paradoxes in the current context are "external" in the sense that they never result from composite formulae but appear only as possible values of variables. This means that we must pay special attention to the atomic subformulae. Roughly, whenever such a subformulae a is extracted on the left of $\vdash$ from a composite one (in a bottom-up application of a rule), it acquires $!a$, though there are some special cases with the negative occurrences. Lowercase letters $a, b, \ldots$ are used for atomic statements (variables) and uppercase $A, B, \ldots$ for arbitrary formulae. A sequent has the form $\Gamma \vdash \Delta$ where both sides are finite sets of formulae. The rules are only an adaptation of the standard Gentzen rules, with the special treatment of atomic (sub)formulae resulting in a modification and addition of some axioms.

Axioms :

$$\Gamma, !a \vdash a, \Delta \qquad\qquad \Gamma, !a, \neg a \vdash \Delta \qquad\qquad \Gamma, a, \neg a \vdash \Delta$$

Rules for non-atomic A :

$$! \vdash \quad \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad\qquad \neg \vdash \quad \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}$$

Rules for arbitrary subformulae :

$$\begin{array}{llll}
\vdash ! & \dfrac{\Gamma \vdash A, \Delta}{\Gamma \vdash !A, \Delta} & \vdash \neg & \dfrac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta} \\
\wedge \vdash & \dfrac{\Gamma, !A, !B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} & \vdash \wedge & \dfrac{\Gamma \vdash A, \Delta \ ; \ \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta} \\
\vee \vdash & \dfrac{\Gamma, !A \vdash \Delta \ ; \ \Gamma, !B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} & \vdash \vee & \dfrac{\Gamma, \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta} \\
\rightarrow \vdash & \dfrac{\Gamma, \neg A \vdash \Delta \ ; \ \Gamma, !B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta} & \vdash \rightarrow & \dfrac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}
\end{array}$$

The rule $! \vdash$ can be removed if we restrict all other rules, introducing $!$ in the assumption, by requiring that it is introduced only when the subformula acquiring this operator is atomic.

A sequent $A_1, ..., A_n \vdash B_1, ..., B_m$ is valid, $A_1, ..., A_n \models B_1, ..., B_m$, iff for every assignment either (at least) one A_i becomes $\mathbf{0}$ or (at least) one B_j becomes $\mathbf{1}$.[8] The completeness theorem has an easy and standard proof based on the semantic invertibility of the rules:

Theorem 4.3 *The system is sound and complete, i.e.,* $\Gamma \vdash \Delta \Leftrightarrow \Gamma \models \Delta$.

Example 4.4 *On the left, we have a proof of the tautology* $(a \vee \neg a) \to (\neg\mathbf{!}a \to \mathbf{!}\neg a)$*:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{!}a \vdash a, \neg a}{\mathbf{!}a \vdash \mathbf{!}a, \neg a}\ {\vdash \mathbf{!}} \qquad \dfrac{\dfrac{\neg a, a \vdash \mathbf{!}a}{\neg a \vdash \mathbf{!}a, \neg a}\ {\vdash \neg}}{\mathbf{!}\neg a \vdash \mathbf{!}a, \neg a}\ {\mathbf{!} \vdash}}{a \vee \neg a \vdash \mathbf{!}a, \neg a}\ {\vee \vdash}}{a \vee \neg a \vdash \mathbf{!}a, \mathbf{!}\neg a}\ {\vdash \mathbf{!}}}{a \vee \neg a, \neg\mathbf{!}a \vdash \mathbf{!}\neg a}\ {\neg \vdash}}{a \vee \neg a \vdash \neg\mathbf{!}a \to \mathbf{!}\neg a}\ {\vdash \to}}{\vdash (a \vee \neg a) \to (\neg\mathbf{!}a \to \mathbf{!}\neg a)}\ {\vdash \to} \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a \vdash a}{a \vdash \mathbf{!}a}}{\vdash \mathbf{!}a, \neg a}\ {\vdash \neg}}{\vdash \mathbf{!}a, \mathbf{!}\neg a}\ {\vdash \mathbf{!}}}{\neg\mathbf{!}a \vdash \mathbf{!}\neg a}\ {\neg \vdash}}{\vdash \neg\mathbf{!}a \to \mathbf{!}\neg a}\ {\vdash \to}$$

The right derivation is an attempted (bottom-up) proof of $A = \neg\mathbf{!}a \to \mathbf{!}\neg a$. *The countermodel* $a = \bot$ *for the irreducible top sequent is also, by the invertibility of the rules, a countermodel for* A.

4.3 Classical propositional logic

The above logic can be expressed in other known systems and, technically, does not present any major contribution. For instance, $\mathbf{!}$ has the same table as Bochvar's assertion operator and also the tables for $\wedge$ and $\vee$ coincide with his tables for the external conjunction and disjunction, [2]. However, the system is a minimal and sufficient extension of classical logic needed, on the one hand, for capturing the propositional aspects of our graphical representation and, on the other hand, for accounting for the properties of the truth-operator, as shown in 4.1. Its minimality allows comparison with *many* other systems which, therefore, will not be attempted here. We summarize only formal relations to the classical propositional logic. They illustrate, in a more technical manner, the initial picture of propositional logic arising from, and possibly functioning within, the context of paradoxical statements. In this context, assertion is not definable by double negation and appears naturally as an additional primitive operator.

Every classical tautology A has a tautological counterpart in our system. For instance, take A as a consequent of an implication with the conjunction of the disjunctions $x \vee \neg x$, for all variables x in A, as the antecedent. More interestingly, however, classical logic can be obtained not by coding its tautologies or connectives but by *limiting the values of statements to the truth-values.* This is achieved by asserting every variable. A consequence of Theorem 4.3 is the following.

Corollary 4.5 *A classical tautology remains a tautology*
(i) when each variable x *is asserted* $\mathbf{!}x$*, or else*
(ii) when every negative occurrence of each variable is asserted.

Point (i) follows easily by inspection of the rules of the calculus. If all atoms are asserted, these rules are equivalent to the rules of the classical system. The additional axioms, requiring the presence of $\neg a$ on the left of $\vdash$ can be now processed further since $\mathbf{!}a$ is non-atomic, and so $..., \neg\mathbf{!}a \vdash ...$, is obtained from $... \vdash \mathbf{!}a...$ by the rule ($\neg \vdash$). Point (ii) follows by a closer analysis of derivations, showing that only negative occurrences can end up on the left of $\vdash$ in irreducible sequents.

Thus, for instance, neither excluded middle nor Pierce's law are tautologies but, according to (i) both $\mathbf{!}a \vee \neg\mathbf{!}a$ and $((\mathbf{!}p \to \mathbf{!}q) \to \mathbf{!}p) \to \mathbf{!}p$ are, while according to (ii), also $a \vee \neg\mathbf{!}a$ and $((\mathbf{!}p \to q) \to \mathbf{!}p) \to p$ are. Point (i) justifies our claim that propositional logic can be viewed as arising from natural reasoning by asserting (or denying) involved statements. Point (ii) clarifies only that not all statements must be so asserted. When all are, the above system and the whole logic becomes the classical propositional one. Conversely, every tautology has a classical counterpart.

[8] Note that this is different from the validity of $(A_1 \wedge ... \wedge A_n) \to (B_1 \vee ... \vee B_m)$ which, provided $n > 1$, would allow assignment of $\bot$ to any A_i's.

Corollary 4.6 *Every tautology is classical when all assertions* **!** *are removed.*

E.g., from the tautology $!A \to A$ we obtain the classical one $A \to A$. Although many classical tautologies are no longer valid, we obtain more tautologies since every classical one has a well-defined counterpart, given by Corollary 4.5, in our system. The additional tautologies are obtained only from such ones by removing redundant assertions.

Thus, the only difference from the classical logic is the occasional need for explicit assertion. This need arises from the presence of paradoxes which, once asserted, enter the world of truth-values. The classical logic, assuming the excluded middle, can simply ignore this need and the hollowness of the predicate "is true" in its context reflects exactly this dismissal. The need for it, and possibly also its problematic status, becomes transparent only when statements may not possess any truth-value. This general situation is considered in the following section.

5 The proposed approach

We start by incorporating our propositional logic into systems of (extended) boolean equations, Subsection 5.1, and then present the corresponding graphical representation, Subsection 5.2. The latter is the same as before but now we consider three-valued assignments. The condition (3.7) for truth-valued solutions is generalized and specifies admissible three-valued assignments. Theorem 5.4 shows their existence for every graph, thus ensuring semantic completeness – every statement in every system has a semantic value. In Subsection 5.3, we present the arising logic of statements which combines our propositional logic with arbitrary statements. Semantic predicates remain internalized as in the propositional logic, but now one can also perform a detailed, logical analysis of paradoxical statements.

The notation in this section refers only to the generalized definitions introduced below. We overload it expecting that this will ease seeing the earlier concepts as special cases of the ones introduced here, without causing any confusion. For instance, lowercase Greek letters $\alpha, \beta, \ldots$ will now denote three-valued assignments, $\overline{\alpha}, \overline{\beta}, \ldots$ their extensions to the boolean expressions over our extended propositional logic, $Frm(n)$ such an extended boolean expression associated with a node n of a graph, etc.

5.1 Extended boolean equations

The extended propositional logic allows more accurate representation of statements expressed in natural language and using the semantic predicates. E.g.:

(i)	z is false.	$x_1 = \neg z$
(ii)	z is not true.	$x_2 = \neg !z$
(iii)	This statement is false.	$x_3 = \neg x_3$
(iv)	This statement is not true.	$x_4 = \neg !x_4$
(v)	It is true that this statement is false.	$x_5 = !\neg x_5$, i.e., $x_5 = \neg x_5$

The only restriction on such extended equations disallows single variable on the right-hand side of $=$. Such a single variable must be preceded by **!**. We thus do not have equations $x = y$ but, instead, $x = !y$, which represent statements $x =$ "y is true".

The propositional logic allows simplification by substitution of equivalent formulae, as exemplified in (v). Such simplifications yield equivalent systems of equations, namely, systems having the same solutions. Moreover, since **!** is identity on the truth-values, the system is also equivalent to the standard one with all **!** removed: both have the same solutions and, in particular, one has none iff the other has none. Thus, to find the solutions of a system of extended equations, we just work with the usual boolean equations. The differences occur when no solutions exist and these are addressed now.

5.2 Three-valued assignments

Statements can be now represented as systems of extended equations or, directly, as graphs. A graph is transformed into a system of equations in the same way as before, (3.5), except that in the case a), when x has only a single positive edge $x \longrightarrow y$, its formula is $Frm(x) = !y$, i.e., we obtain the equation $x = !y$. Transformation of an extended system into a graph is also essentially the same as before, (3.2), except that the cases a) and c) involve now a more general situation. The single (positive) variables in equations of the form a) or c) may now be asserted composite formulae, i.e., instead of $x = ...y...$, we may have $x = ...!A....$ These give rise to positive edges (in case 2.a), a single one) from the node x to a node u. If A is a variable, then u is A, and we continue processing next equation. If A is a composite formula, then u is a new node and we continue by processing first the new equation $u = A$. The rest of the transformation is as in (3.2). Consequently, as before, we speak about systems without differentiating whether these are graphs or systems of extended boolean equations.

By the above mentioned equivalence of such systems and of the standard ones, Theorem 3.8 remains true. The cases when only solutions are of interest can be treated exactly as before. We now generalize this to cases when no solutions exist by admitting three-valued assignments, $\alpha \in \{\mathbf{0}, \mathbf{1}, \bot\}^{\mathcal{G}}$. Such assignments must respect the boolean dependencies, i.e., when a node is assigned **1** then all its positive neighbours are assigned **1** and all negative ones **0**, while when it is assigned **0** then one of these conjuncts fails. Informally, an assignment is admissible iff it assigns correct truth-values to a maximal number of nodes and $\bot$ to the rest. Formalization of this maximality is given in the following definition.

Recall that, for $n \in int(\mathcal{G}) = \mathcal{G} \setminus sinks(\mathcal{G})$, $Frm(n)$ is n's formula, i.e., $Equ(n)$ is $n = Frm(n)$ (cf. (3.5) and the opening of the first paragraph of this section.) For $\alpha \in \{\mathbf{0}, \mathbf{1}, \bot\}^{Y}$, $\overline{\alpha}$ denotes evaluation of the propositional formulae under α using the tables (4.1).

Definition 5.1 *Given an $\alpha \in \{\mathbf{0}, \mathbf{1}, \bot\}^{\mathcal{G}}$, let $B(\alpha) = \{n \in int(\mathcal{G}) \mid \alpha(n) \neq \bot\}$.*
α is admissible for $\mathcal{G}$, $\alpha \in adm(\mathcal{G})$, iff

(B) $\forall n \in B(\alpha) : \alpha(n) = \overline{\alpha}(Frm(n)) \wedge$

(P) $\forall \beta \in \{\mathbf{0}, \mathbf{1}, \bot\}^{\mathcal{G}} : B(\alpha) \subset B(\beta) \Rightarrow \exists x \in B(\beta) : \beta(x) \neq \overline{\beta}(Frm(x))$.

The condition (P) states simply that an admissible α assigns truth-values to a maximal possible subset of variables, without violating the condition (B).[9]

Example 5.2 (i) *The Liar, $x = \neg x$, has no solution. Trying $\alpha(x) = \mathbf{1}$ we get $\overline{\alpha}(\neg x) \neq \alpha(x)$ and similarly for $\alpha(x) = \mathbf{0}$. So $\alpha(x) = \bot$ satisfies the condition (P) and is the only admissible assignment.*

(ii) *For the Strengthened Liar:*

$$\begin{tikzcd} x \arrow[r, bend left, "\times"] & y \arrow[l, bend left] \end{tikzcd} \qquad \begin{array}{l} Equ(x) : x = \neg y \\ Equ(y) : y = !x \end{array}$$

there are only two assignments, $\alpha(x) = \bot, \alpha(y) = \mathbf{0}$ and $\beta(x) = \mathbf{0}, \beta(y) = \bot$. Verification of the assignment of $\bot$ follows in each case since the system has no solutions. The impossibility of assigning $\bot$ to both nodes follows since the above two assignments provide counter-examples to the condition (P) in case we attempt to let $P(\alpha) = \{x, y\}$.

(iii) *A Contingent Strengthened Liar is, for instance:*

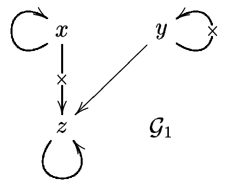
$$\begin{tikzcd} x \arrow[r, bend left, "\times"] & y \arrow[l, bend left] \arrow[r] & z \end{tikzcd} \qquad \begin{array}{l} Equ(x) : x = \neg y \\ Equ(y) : y = !x \wedge z \end{array}$$

There are two admissible assignments extending $\gamma(z) = \mathbf{1}$, namely, α, β from the previous point. But now we also have a possible solution $\sigma(z) = \sigma(y) = \mathbf{0}, \sigma(x) = \mathbf{1}$. Also, since $z \in sinks(\mathcal{G})$, we can freely assign $\delta(z) = \bot$, which can be extended to an admissible assignment $\delta(y) = \mathbf{0}, \sigma(x) = \mathbf{1}$.

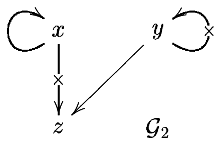
Note that assignment to $sinks(\mathcal{G})$ is unrestricted. Requiring the conditions of Definition 5.1 to hold also for $sinks(\mathcal{G})$ would limit the number of assignments. But it would also go counter their intuitive understanding as contingent facts which can obtain arbitrary semantic values, depending on the external circumstances rather than internal relations.

[9]Note that the definition uses only formulae associated with statements. Consequently, it can be applied unchanged with any logic. We use it, however, only with our propositional logic and the (abbreviated) graphs where nodes may be annotated by disjunctions, implications, etc.

Example 5.3 *Consider two graphs with the same nodes:*



$Equ_1(\_)$	α	β	γ	$Equ_2(\_)$
$x = x \wedge \neg z$	0	1	0	$x = x \wedge \neg z$
$y = \neg y \wedge z$	$\bot$	0	0	$y = \neg y \wedge z$
$z = !z$	1	0	0	$-$



Under truth-valued assignments in α to $\mathcal{G}_1$, y is indeed a paradox. But α is excluded from $adm(\mathcal{G}_1)$ by 5.1.(P). Trying to verify (P) for $\alpha(y) = \bot$, we find another assignment, for instance, β or γ, which assigns truth-values to all nodes.

$\mathcal{G}_2$ is identical to $\mathcal{G}_1$ except that now $z \in sinks(\mathcal{G})$. Consequently, we can freely assign $\alpha(z) = \mathbf{1}$. As a result, not only β, γ but also α will be an admissible assignment for $\mathcal{G}_2$.

The following theorem guarantees the existence of admissible assignments for every system. Thus, our theory does not suffer from semantic incompleteness and revenge problems. Every statement in every system is guaranteed to obtain a semantic value.

Theorem 5.4 *For every $\mathcal{G}$ and $\beta \in \{\mathbf{0}, \mathbf{1}, \bot\}^{sinks(\mathcal{G})}$, there is an $\alpha \in adm(\mathcal{G})$ extending β.*

Proof. Let α_0 extend β assigning $\bot$ to all internal nodes $x \in int(\mathcal{G})$, i.e., $B(\alpha_0) = \varnothing$. If α_0 satisfies the condition (P), we are done. Otherwise, we have its negation, i.e.:

(*) $\exists \alpha_1 \in \{\mathbf{0}, \mathbf{1}, \bot\}^{\mathcal{G}} : B(\alpha_0) \subset B(\alpha_1) \wedge \forall x \in B(\alpha_1) : \alpha_1(x) = \overline{\alpha}_1(Frm(x))$.

Choose such an α_1 and repeat the check. If the condition (P) is satisfied, we are done. Otherwise (*) still holds for α_1, i.e., we can further extend it to an α_2 with $B(\alpha_1) \subset B(\alpha_2)$ satisfying the condition (B). Eventually, we reach an α satisfying the condition (P), either because $B(\alpha) = \mathcal{G}$ or because it can not be further extended. Such an α is admissible. □

Since the condition (3.7) is just (B) for all internal nodes, we have that $sol(\mathcal{G}) \subseteq adm(\mathcal{G})$. But, in general, even when $sol(\mathcal{G}) \neq \varnothing$, there can be more admissible assignments than solutions.

Example 5.5 (i) *The system $x = \neg y \wedge z$, $y = x \wedge z$, has one solution assigning $\mathbf{0}$ to all variables. However, since $sinks(\mathcal{G}) = \{z\}$, we obtain also assignments for $\alpha(z) = \bot$ and $\beta(z) = \mathbf{1}$. The former gives $\alpha(x) = \alpha(y) = \mathbf{0}$, while the latter either $\beta_1(x) = \mathbf{0}, \beta_1(y) = \bot$ or $\beta_2(x) = \bot, \beta_2(y) = \mathbf{0}$.*

(ii) *The system $p = \neg q, q = \neg p$ has two solutions $\alpha(p) = \mathbf{0}, \alpha(q) = \mathbf{1}$ and $\beta(p) = \mathbf{1}, \beta(q) = \mathbf{0}$. These are the only admissible assignments – in particular, it is not possible to assign $\bot$ to any of the two statements.*

The example illustrates the general fact, captured by the following proposition: additional assignments for systems possessing solutions arise only from assignments to sinks which can not be extended to any solution.

Given an $\alpha \in \{\mathbf{0}, \mathbf{1}, \bot\}^{\mathcal{G}}$ and $\varnothing \neq X \subseteq \mathcal{G}$, $\alpha|_X$ denotes the restriction of α to X. If A is a set of αs, $A|_X$ denotes the set of so restricted αs.

Proposition 5.6 *For every $\mathcal{G}$:*

1. $sinks(\mathcal{G}) \neq \varnothing \implies adm(\mathcal{G}) = sol(\mathcal{G}) \cup \{\alpha \in adm(\mathcal{G}) \mid \alpha|_{sinks(\mathcal{G})} \notin sol(\mathcal{G})|_{sinks(\mathcal{G})}\}$.
2. $sinks(\mathcal{G}) = \varnothing \wedge sol(\mathcal{G}) \neq \varnothing \implies adm(\mathcal{G}) = sol(\mathcal{G})$.

Proof. 1. The inclusion $\supseteq$ is trivial, while for $\subseteq$ we have to show that for every $\mathcal{G}$:

$$\forall \alpha \in adm(\mathcal{G}) : \alpha|_{sinks(\mathcal{G})} \in sol(\mathcal{G})|_{sinks(\mathcal{G})} \Rightarrow \alpha \in sol(\mathcal{G}).$$

But this follows easily since the existence of $sol(\mathcal{G})$ extending $\alpha|_{sinks(\mathcal{G})}$ implies that no $B(\alpha) \neq \mathcal{G}$ satisfies condition (P).

2. This follows by the same argument as 1. If there are no sinks but there are some solutions, no assignment α with $B(\alpha) \neq \mathcal{G}$ will satisfy condition (P). □

Thus, every system has an admissible assignment by Theorem 5.4, while by the above proposition, no superfluous assignments will appear. Combining this with the limitations, enforced by the condition (P), on the number of $\bot$ in every admissible assignment, we can conclude thus understood minimality of the proposed approach. Coinciding with the propositional logic on classical propositions and with the boolean equations on non-paradoxical statements, it makes only a minimal extension in order to handle also paradoxical statements. This last extension is elaborated below.

5.3 The logic of statements

Construction of all admissible assignments is trivially a computable function since, in the worst case, one can just try all possibilities, verifying the conditions of Definition 5.1 – Theorem 5.4 guarantees a success. Putting such concerns aside, we rather consider some properties of the arising logic of statements, i.e., the logic with syntax given by the systems of expanded boolean equations, as described in Subsection 5.1, and the semantics of a statement $\mathcal{A}$ being all its admissible assignments.

First, we notice the lack of (any obvious) compositionality. Solutions for several equations can be obtained intersecting the sets of solutions for each equation. Consequently, when extending a system possessing solutions with new equations, $\mathcal{G} \subseteq \mathcal{H}$, we narrow the set of solutions, $sol(\mathcal{H}) \subseteq sol(\mathcal{G})$. No similar relations obtain for the sets $adm(\mathcal{H})$ and $adm(\mathcal{G})$. Assignments for a system are not obtained by composing assignments for subsystems. For instance, there are three assignments for $\mathcal{A} : x \longleftarrow y$, namely, $adm(\mathcal{A}) = \{\langle \mathbf{0}, \mathbf{0}\rangle, \langle \mathbf{1}, \mathbf{1}\rangle, \langle \bot, \mathbf{0}\rangle\}$ (writing $\langle a, b\rangle$ for the assignment $\alpha(x) = a, \alpha(y) = b$) and, likewise, three assignments for $\mathcal{B} : x \multimap\!\!\!\times\!\!\!\rightarrowtail y$, namely, $adm(\mathcal{B}) = \{\langle \mathbf{1}, \mathbf{0}\rangle, \langle \mathbf{0}, \mathbf{1}\rangle, \langle \mathbf{0}, \bot\rangle\}$. Union of two systems might be taken as a kind of conjunction but, in this case, it becomes the system from Example 5.2.(ii). It has only two assignments $\{\langle \mathbf{0}, \bot\rangle, \langle \bot, \mathbf{0}\rangle\}$ which do not result from $adm(\mathcal{A})$ and $adm(\mathcal{B})$ by any obvious set operations. One might look for ways of expressing the dependence of the set of assignments for a system in terms of the assignments for its subsystems. But examples like this illustrate, if not the non-compositionality, then the lack of straightforward compositionality inherent in the approach.

This lack reflects the similar feature of the natural discourse, where subsequent statements lead often to non-monotonic revisions of the earlier assumed truth-values and require their new redistribution. Representing statements by systems of equations, one can envisage various operations on them. The most obvious one would be taking the union of all equations from the arguments. However, the lack of compositionality suggests that the semantics of such operations might be very complicated, if not simply impossible. One always risks facing a new statement, which either resolves previous cataphoras or introduces new anaphoric references in a way requiring re-evaluation of the totality of the discourse. The presented approach allows modeling such complex phenomena by means of viewing statements as sets of admissible assignments, which are narrowed or adjusted as new statements enter the discourse. This might provide a tool for linguistic applications but, due to non-compositionality, we do not expect it to yield any systematic and clean theory.

Instead, we limit attention to particular ways of combining statements, namely, to propositional combinations. This reflects again the picture of propositional logic drawn at the beginning of Section 4. On the one hand, it is only a part of natural reasoning involving only particular statements obtained from others by propositional combinations. On the other hand, it can be seen as a meta-logic of natural reasoning which turns all statements into propositions and considers their truth-functional dependencies. Section 4 formalized this later function in isolation, with actual statements abstracted away and appearing only as the possible value $\bot$ of variables. Now, it will interact with the structure of statements, allowing to identify dependencies between various, possibly paradoxical, substatements. Even if this restriction to propositional treatment of statements is motivated by technical reasons, it seems to reflect the natural procedure. Given a possibly paradoxical statement, its analysis is carried from "outside", by means of truth-valued claims and arguments addressing its structure and possible truth-values of its components.

Thus, we consider now arbitrary statements which enter into propositional combinations forming statements with designated roots, which do not occur elsewhere in the system. We use uppercase script letters $\mathcal{A}, \mathcal{B}, ...$ for such statements, with the implicit convention that the corresponding lower case letters $a, b, ...$ denote their roots, not occurring elsewhere in the system. Any propositional connective applied to statements represents then a statement obtained by applying the connective to the roots of the argument statements. E.g., $\neg\mathcal{A}$ denotes a statement $\mathcal{B} : b = \neg a, a = ...$, where b is a fresh variable and "..." is the formula for a, followed possibly by the rest of the system of equations of $\mathcal{A}$, having a as the only root (in $\mathcal{B}$, a occurs only at the two specified places.)

In analogy to classical tautologies, we consider a statement to be *valid*, or necessarily true, iff all its admissible assignments make the root true, i.e., $\models \mathcal{A}$ iff $\forall \alpha \in adm(\mathcal{A}) : \alpha(a) = \mathbf{1}$, where a is the root of $\mathcal{A}$. We have the following equivalences for every $\alpha \in adm(\mathcal{A})$:

$$\begin{array}{lcl} \alpha(a) = \mathbf{1} & \Leftrightarrow & \alpha \models \mathcal{A} \quad \Leftrightarrow \quad \alpha \models \mathbf{!}\mathcal{A} \\ \alpha(a) = \mathbf{0} & \Leftrightarrow & \alpha \models \neg\mathcal{A} \\ \alpha(a) = \bot & \Leftrightarrow & \alpha \models \neg\neg\mathcal{A} \wedge \neg\mathbf{!}\mathcal{A} \end{array} \tag{5.7}$$

We abbreviate the last formula as $\bot\mathcal{A} \overset{\text{def}}{=} \neg\neg\mathcal{A} \wedge \neg\mathbf{!}\mathcal{A}$. Thus, the predicates of being $True(\_)$, $False(\_)$ or $Paradox(\_)$ are internally expressible in the language of statements. Negation interacts with such predicates according to the propositional laws. E.g., letting α range over $adm(\mathcal{A})$, we have

$$\alpha \models \text{not-}True(\mathcal{A}) \text{ iff } \alpha \models \neg\mathbf{!}\mathcal{A} \Leftrightarrow \overline{\alpha}(\mathbf{!}a) = \mathbf{0} \Leftrightarrow (\alpha(a) = \mathbf{0} \text{ or } \alpha(a) = \bot),$$

and

$$\alpha \models \text{not-}True(\neg\mathcal{A}) \text{ iff } \alpha \models \neg\mathbf{!}\neg\mathcal{A} \Leftrightarrow \overline{\alpha}(\neg a) = \mathbf{0} \Leftrightarrow (\alpha(a) = \mathbf{1} \text{ or } \alpha(a) = \bot).$$

Both these predicates are true when, and only when, $\mathcal{A}$ is a paradox. In propositional logic we have the corresponding tautology $\bot x \leftrightarrow (\neg\mathbf{!}x \wedge \neg\mathbf{!}\neg x)$. Now, such general propositions can be used with particular statements. E.g., validity of $\mathcal{B}\colon \models x = \mathbf{!}x, b = x \vee \neg x$, means that the statement $x = \mathbf{!}x$ is never a paradox, while validity of $\mathcal{A}\colon \models x = \neg x, a = \bot x$, that $x = \neg x$ always is.[10]

Since validity of statements depends on their referential structure, one has to pay special attention to the names of the statements. For instance, the statement $\mathcal{A}\colon\ a = \mathbf{!}x \vee \mathbf{!}\neg x$ is not valid, since the admissible assignment $\alpha(x) = \bot$ makes $\overline{\alpha}(a) = \mathbf{0}$. However, the statement $\mathcal{X}\colon\ x = \mathbf{!}x \vee \mathbf{!}\neg x$ is valid since the only admissible assignment to x is $\alpha(x) = \mathbf{1}$.

Now, among the tautologies of the logic we find the tautologies of our propositional logic. Whenever B is a tautology of the latter, the statement $a = B$ (with a not occurring in B) is valid. But we obtain a lot of new valid statements which are not expressible in the propositional language. These are of particular interest and we give some examples:[11]

$$\begin{array}{rcll} \text{(i)} & \models & x = \neg x, & a = \bot x \\ \text{(ii)} & \models & x = \neg y, y = \mathbf{!}x, & a = \bot x \vee \bot y \\ \text{(iii)} & \models & x = \neg x \wedge y, & a = x \to y \\ \text{(iv)} & \models & x = \neg x \wedge y, & a = \mathbf{!}y \to \bot x \\ \text{(v)} & \models & x = \neg\neg x \wedge \neg\mathbf{!}x, & a = \neg x \\ \text{(vi)} & \models & x = \mathbf{!}x \to y, & a = \neg\mathbf{!}y \to \bot x \\ \text{(vii)} & \models & x = \mathbf{!}x \to y, & a = \mathbf{!}y \to \mathbf{!}x \\ \text{(viii)} & \models & x = \mathbf{!}x, & a = x \vee \neg x \end{array} \tag{5.8}$$

In all cases, a is the root and its validity means that

(i) the only admissible assignment to the Liar makes it $\bot$;

(ii) every assignment for the Strengthened Liar, Example 5.2.(ii), makes one of its nodes $\bot$.

(iii) gives a good example of the more detailed analysis enabled by this logic. The statement x is a Contingent Liar: when $y \in \{\mathbf{0}, \bot\}$ then also $x = \mathbf{0}$, but when $y = \mathbf{1}$ then $x = \bot$. In the former cases, $a = \mathbf{1}$ since the antecedent x of the implication is $\mathbf{0}$, while in the latter one, $a = \mathbf{1}$ since the consequent $y = \mathbf{1}$.

(iv) is another valid statement about the Contingent Liar: if y is true then x is a paradox. Note that replacing $\mathbf{!}y$ by y in a would not give a valid statement $y \to \bot x$: the assignment $y = \bot, x = \mathbf{0}$ is admissible but makes the implication $y \to \bot x$ false.

(v) The statement x, claiming to be a paradox, is false.

(vi) x represents Curry/Löb paradox ($x =$ "If x is true then y"). Validity of a means that it indeed is a paradox whenever y is not true (i.e., is either false or itself a paradox). However,

(vii) when y is true, x is not a paradox but true.

(viii) The Truth-teller is either true or false.

[10]Analogous facts hold in Kripke-Feferman theory, [4], which proves, e.g., not-true(L), not-true(not-L). Many tautologies have their counterparts in that system, which can be seen already from our propositional logic. E.g., validity of **t7**. $\mathbf{!}A \to A$ but not of the opposite implication, has the counterpart in the validity of the downward, but not the upward, T-inference. The implication $\mathbf{!}A \to \mathbf{!!}A$ in **t9** corresponds to the validity of a weakened version of the upward T-inference.

[11]To ease readability, we use freely derived connectives which were defined in Subsection 4.1.

Thus the mere analysis of statements' validity unveils features which may be intuitively obvious but which have always presented a challenge for formal modeling. It allows not only to conclude that a statement is or may be paradoxical, but also analyze precise circumstances which make is so.

A natural next question would concern an inference system for valid statements. As suggested by the discussion of non-compositionality, we would be rather sceptical about any such general system, i.e., one treating arbitrary combinations of arbitrary statements in a compositional way. A calculus for a subset of statements, like the propositional subset described above, might merit further investigation which, however, must be left for the future work.

Let us conclude this section by observing only an example of an unsound inference, i.e., one leading from a valid assumption to an invalid conclusion. Such inferences may easily arise from the misinterpretation of the equational notation. The extended boolean equations, representing also statements possessing possibly no solutions, make this notation a bit misleading. As long as we consider systems possessing solutions, everything is fine because then all equations are indeed satisfied. But when some (sub)statements are paradoxical, we no longer ask about solutions to the (equational) system, but about its admissible assignments. Some equations do not hold and, consequently, substitution of "equals by equals" is no longer sound. For instance, starting with the valid statement $\mathcal{A}$, (i) in (5.8):
$x = \neg x, a = \neg\neg x \wedge \neg!x$
and substituting naively $\neg x$ for x, we obtain $\mathcal{B}$:
$x = \neg\neg x, b = \neg\neg\neg x \wedge \neg!\neg x$.
The conclusion is equivalent to $x = \neg\neg x, b = \neg x \wedge \neg\neg x$ which is a contradiction.

The above examples should give a sufficient gist of the expressivity and possibilities of the logic. In the following section, it is used for a more detailed reflection over some paradoxes and is compared to other proposals.

6 Examples and alternative approaches

6.1 The Liar

The liar statement is a paradox but "This sentence is false" can also have a truth-value, depending on the reference of "this". The statement z

$$\begin{tikzcd} z \arrow[r, "\times" description] & y \end{tikzcd} \tag{6.1}$$

can be read as "This sentence is false" where "this" refers to y. Identical sentences do not make the two statements the same. Unfortunately, it is not always clear which more specific intention may underlie a particular pronouncement of a sentence. Even more unfortunately, in the course of analysis one easily identifies different statements having the same linguistic expression. Such identification enters the common-sense analysis of the Liar. Starting with

$$\begin{tikzcd} L \arrow[loop right, "\times"] \end{tikzcd} \tag{6.2}$$

one observes that if it is false then it is true. But such an observation involves a change of the referential structure. The claim that the Liar is false is the statement x_1 :

$$\begin{tikzcd} x_1 \arrow[r, "\times" description] & L \arrow[loop right, "\times"] \end{tikzcd} \tag{6.3}$$

which, indeed, is false. Just like L, so also x_1 is expressed by the sentence "This sentence is false", where "this" refers also to L. It is not, however, the Liar statement, even though both are not only expressed by identical sentences but even refer, by "this", to the same L. This identity of both – the two sentences and their referents – seems to justify the possibility of further analysis. For if "This sentence is false" is false, than what it claims is true. At this moment it is no longer clear whether the just quoted statement, whose falsity is assumed, is L or x_1. Consequently, it is equally unclear whether we obtain a statement x_1' claiming truth of L or x_1'' claiming truth of x_1:

$$(i)\ \begin{tikzcd} x_1' \arrow[r] & L \arrow[loop right, "\times"] \end{tikzcd} \qquad (ii)\ \begin{tikzcd} x_1'' \arrow[r] & x_1 \arrow[r, "\times" description] & L \arrow[loop right, "\times"] \end{tikzcd} \tag{6.4}$$

In either case the conclusion can be justified, though for very different reasons. In case of x_1', by the mere validity of the implication $x_1 \to x_1'$, i.e., of $\neg L \to !L$, which holds since $\neg L \leftrightarrow !L$

whenever L is paradoxical. This does not make x_1' true but the truth of the implication seems incontestable. In the case of x_1'', on the other hand, the implication holds because x_1'' is, in fact, true. The third justification, common to both cases, could be that the implication holds since x_1 is false. But this is not the kind of argument encountered in a common-sense analysis of the Liar.

A more formal analysis is based on a yet different procedure. Starting with the "equation" $L = \neg L$, one feels justified in performing a self-substitution, yielding a chain $L = \neg L = \neg\neg L = \neg\neg\neg L = ...$ The law of the excluded middle spooks from the background but the "equalities" make one iterate indefinitely in search for a fixed-point.

$$\begin{tikzcd} \cdots \arrow[r, "\times"] & x_3 \arrow[r, "\times"] & x_2 \arrow[r, "\times"] & x_1 \arrow[r, "\times"] & L \arrow[loop right, "\times"] \end{tikzcd} \quad (6.5)$$

This involves yet another referential structure than x_1, x_1' or x_1''. It has nothing to do with L but is only a repeated application of z from (6.1) starting with L for y. It is based on the natural interpretation of the equation $L = \neg L$ which does not notice that such "equations" require special treatment. Substitution of equals for equals in solving equations is based on the preservation of some crucial aspects – in the case of algebraic equations, of the identities and, consequently, of the solutions. As we saw at the end of Section 5, this is not the case for the "equations" involved in the representation of statements. Substitution of equals for equals in this case is sound only when the equals have a truth-value. The chain (6.5) of alternatingly false and true statements is an unfolding of the Liar only in the trivial sense of a syntactic substitution. Although, semantically, it is not sound, one nevertheless tries to apply semantic – and bivalent – intuitions to this unfolding. The distinction between sentences and statements, and assignment of the semantic values to the latter and not to the former, clarifies such equivocations. As we hopefully have shown, it offers also much simpler, finitary means to determine paradoxical nature of statements than infinite or even transfinite iterations of such unsound substitutions.

6.2 Strengthened liar

The statement $x =$ "This sentence is not true", i.e., $x = \neg!x$ or

$$\begin{tikzcd} x \arrow[r, bend right, "\times"'] & y \arrow[l, bend right] \end{tikzcd} \quad (6.6)$$

has no solutions and only two assignments $x = \bot, y = \mathbf{0}$ and $x = \mathbf{0}, y = \bot$, as expressed partially in (ii) of (5.8). The sentence seems to be true and, indeed, it is but only when taken as expressing another statement, namely, the following z with x as above:

$$\begin{tikzcd} z \arrow[r, "\times"] & z_1 \arrow[r] & x. \end{tikzcd} \quad (6.7)$$

This z is true since both $x = \bot$ and $x = \mathbf{0}$ make $z_1 = \mathbf{0}$ – we have the valid statement $\mathcal{Z} : x = \neg!x, z = \neg!x$. So it is true that x is not true, but this does not make x true, even though, due to the same expression and referent, x seems to claim exactly that. "This sentence is not true" seems to claim something true about itself, namely, that it is not true. This claim, however, is exactly the paradox (6.6), while true is not this paradoxical statement but another, albeit expressed by identical sentence, (6.7). (Gaifman calls $\mathcal{Z}$ "the two lines puzzle". A similar, though informal, analysis of the Strengthened Liar can be found in [9].)

Consider now the sentence $y =$ "It is true that this sentence is false". Ignoring possible alternative references of "this", the one pointing to the whole sentence gives the statement $y = !\neg y$ with the graph (6.6). The statement y is different from x, yet, both give rise to the same system which has no solutions.

The examples illustrate indirectly the difference between "is false" and "is not true". Although both Liar and Strengthened Liar are paradoxical, when their "this"s refer to other statements, we have the difference between (6.1) $z \xrightarrow{\times} x$ and (6.7) $z \xrightarrow{\times} z_1 \longrightarrow x$. In the context of bivalent logic, this difference disappears since assertion **!** amounts then to identity. But when x has no truth-value, the two do not coincide. In our propositional logic, $\neg x \to \neg!x$ is a tautology, **t6** of (4.2), but the opposite implication is not.

6.3 Buridan and the Truth-teller

Jean Buridan's solution to the liar paradox (at least, the early one), amounted to saying that every statement claims, in addition to whatever it claims, also its own truth. In the equational representation, it means that every equation has the form $x = ... \wedge x$, while in the graphical one, that every node has a positive loop $\begin{tikzcd} x \arrow[loop right] \end{tikzcd}$. Such a positive loop, taken in isolation, is the Truth-teller which, having undetermined but not paradoxical value, does not seem to offer anything of significance. However, used in conjunction with other statements, it is exactly Buridan's proposal and has significant consequences.

Given a system $\mathcal{G}$, let $\mathsf{Bur}(\mathcal{G})$ denote its *Buridan closure*, obtained by adding such a positive self-loop to every node.

Fact 6.8 *1. If α is a solution for $\mathcal{G}$, then it is also for $\mathsf{Bur}(\mathcal{G})$. (If $\alpha(x) = \overline{\alpha}(A_1 \wedge ... \wedge A_n)$ then also $\alpha(x) = \overline{\alpha}(A_1 \wedge ... \wedge A_n \wedge x)$.)*
2. Every $\mathsf{Bur}(\mathcal{G})$ has a solution, for all $x : \alpha(x) = \mathbf{0}$. (Trivially, for any A, $\mathbf{0} = \overline{\alpha}(A) \wedge \mathbf{0}$.)

The second point signals possible objections to this solution of the Liar and other paradoxes. Adding this, apparently innocuous, self-affirmation to every statement, changes quite a lot. In particular, it makes it possible for every statement to be false. As a value of logic is, to some extent, the value of its tautologies, the logic resulting from Buridan closure would be pretty limited. Consider the following example:

$$\begin{tikzcd} \mathcal{G} : & x \arrow[r, "\times" description] & L \arrow[loop right, "\times" description] \end{tikzcd} \qquad \begin{tikzcd} \mathsf{Bur}(\mathcal{G}) : & x \arrow[r, "\times" description] \arrow[loop below] & L \arrow[loop above, "\times" description] \arrow[loop below] \end{tikzcd}$$

There is no solution for $\mathcal{G}$, and only one assignment $\alpha(x) = \mathbf{0}, \alpha(L) = \bot$. For $\mathsf{Bur}(\mathcal{G})$ we have two solutions with $\alpha(L) = \mathbf{0}$ and then either $\alpha(x) = \mathbf{1}$ or $\alpha(x) = \mathbf{0}$. As x represents the statement "The Liar (L) is false", one may feel uneasy that it can be true, although this is consistent with the basic claim that the Liar is false. The problem with this solution is that x can also be false! In fact, *every composite statement can be false irrespectively of the values of its arguments.* A proposition $x = a \wedge b$, with the values of a and b being $\mathbf{1}$, can be $\mathbf{1}$ just as well as $\mathbf{0}$, since the implicit self-assertion (which turns x into $x = a \wedge b \wedge x$) makes always the assignment of $\mathbf{0}$ possible.

The first point of Fact 6.8 gives that substitution of $!x$ for x preserves the truth-valued solutions. The second point ensures that every Buridanised statement has a truth-value. However, the value of this truth-value is very doubtful. The price, in the form of allowing every composite statement to be false irrespectively of the values of its arguments, seems too high almost no matter what one could purchase for it.

6.4 Kripke's Nixon

In [14], Kripke introduced the following example, arguing for the significance of empirical facts for obtaining possibly paradoxical statements in natural discourse.

j = "Most (more than half) of n's statements are false"
n = $\{a, b, j\}$

To model j's assertion, let us associate with the node n, $Frm(n)$ which evaluates to $\mathbf{1}$ iff at least 1/2+1 of n's neighbours are $\mathbf{1}$, and to $\mathbf{0}$ otherwise:

$$\begin{tikzcd} j \arrow[r, bend left, "\times" description] & n \arrow[l, bend left] \arrow[r] \arrow[d] & a = \mathbf{1} \\ & b = \mathbf{0} & \end{tikzcd}$$

The "riskiness" of the concept of truth which (truth, not the concept) depends on the empirical evidence, boils here down to the observation that j/n may, but need not, be paradoxical. If we insist on the (sink) nodes having fixed truth-values $a = \mathbf{1}$ and $b = \mathbf{0}$, the graph has no solution. Thus, the essence of this example is the Contingent Liar analyzed in Example 5.2.(iii):

$$\begin{tikzcd} j \arrow[r, bend left, "\times" description] & n \arrow[l, bend left] \arrow[r] & a \end{tikzcd} \tag{6.9}$$

We accept fully Kripke's contention that many ordinary statements, in particular those about truth and falsity, can become paradoxical depending on entirely contingent, empirical facts. This is but another way of saying that natural discourse allows paradoxes. But as a contribution to a (formal) theory of paradoxes, it reduces to the observation that some extraneous requirements on the truth-values of some (sub)statements may result in the impossibility of assigning truth-values to others.

Our approach, admitting this contingency without any problems, is significantly simpler than fixed-point constructions originating from [14]. It is also more definite. Kripke gives an example, (12) in [14], of the statement $x =$ "Either x or its negation is true" which (i) has no truth-value in the minimal fixed-point, (ii) has value true in some fixed-points and (iii) never has value false. Such a generous supply of alternatives of fixed-points and possible values does not seem to be any advantage. The equation $x = x \vee \neg x$ (or $x = \mathbf{!}x \vee \mathbf{!}\neg x$, or $x = \mathbf{!}(x \vee \neg x)$) has a unique solution, $x = \mathbf{1}$, which exhausts its semantics according to Proposition 5.6.2. It is doubtful that one could elucidate it more by supplying any alternatives.

6.4.1 Fixed-points and other logics

Solutions of systems of equations and admissible assignments can be formulated as fixed-points. Although this exhausts the analogy, a few more words on the relation to Kripke's approach may be in place. We first formulate the general problem as follows (with a propositional logic for expressing $Frm(x)$ taken as a parameter):

> Given a set of variables X and of equations $\mathcal{G} = \{x = Frm(x) \mid x \in Y \subseteq X\}$, determine a condition (C) for assignments $\alpha \in \{\mathbf{0}, \mathbf{1}, \bot\}^X$ respecting (B), i.e., such that $\forall x \in Y : \alpha(x) \in \{\mathbf{0}, \mathbf{1}\} \Rightarrow \alpha(x) = \overline{\alpha}(Frm(x))$.

It is usually assumed, and sometimes stated, that (C) should minimize the number of $\bot$. Our (P) from Definition 5.1 says that admissible α is maximal with respect to the quasi-ordering $\alpha \preceq \beta \Leftrightarrow B(\alpha) \subseteq B(\beta)$, that is, it assigns a maximal possible number of truth-values satisfying (B). Denoting by $|C(\mathcal{G})|$ the cardinality of a minimal possible subset of X assigned $\bot$ by any fixed-point satisfying a condition (C), this amounts to the simple fact:

Fact 6.10 *For any condition (C) and any* $\mathcal{G} : |P(\mathcal{G})| \leq |C(\mathcal{G})|$.

Kripke's minimal fixed-point condition gives constant assignment $\alpha(x) = \bot$ for every $\mathcal{G}$. Other choices of fixed-points still introduce $\bot$ in cases which should not require any, e.g., [10, 11]. This is often consequence of using Kleene logic. Since it is commonly associated with Kripke's approach, we comment briefly this combination.

The above fact quantifies over conditions relatively to a fixed parameter logic, so it still holds when we replace our logic by Kleene (strong or weak). The (C) condition can be then taken to simply require α to be a solution, since every system has one, e.g., $\alpha(x) = \bot$. This corresponds to the existence of minimal fixed-points, for any solution to a system in Kleene logic gives a fixed-point for a Kripkean truth-predicate. When the solution is truth-valued, it coincides also with ours. Consider, for instance, the following system:

$x = \neg y \vee \neg z$	: y is false or z is false
$y = \mathbf{!}x$	: x is true
$z = \neg x$	: x is false

Our logic gives its only solution: $x = \mathbf{1} = y$, $z = \mathbf{0}$. Using Kleene logic (and dropping $\mathbf{!}$) yields the same solution, giving also a maximal Kripkean fixed-point. But such fixed-points may still involve $\bot$, excluded by our logic. For instance, in Kleene logic $\alpha(x) = \bot$ is the solution of the Liar system $\mathcal{L} : x = \neg x$ and this may be mathematically pleasing. However, extending $\mathcal{L}$ with the equation $y = x \vee \neg x$ gives the solution $\alpha(y) = \bot$. Strengthening the condition (C) with our (P) does not help since, in Kleene logic, this α is the only fixed-point. Intuitions can be disputed but, unless one turns paraconsistent, the claim that the Liar is either true or false appears false. Our logic gives one assignment for this system: $\alpha(x) = \bot$ and $\alpha(y) = \mathbf{0}$.

Thus, when the system has truth-valued solutions our assignments corrrespond to Kripke's maximal fixed-points. But when such solutions do not exist, our proposal yields, in general, fewer paradoxes. Our setting allows also combinations with other logics but their investigation should probably start by identifying some undesirable consequences of our proposal. The above fact, the properties of our logic, as well as the examples in this and previous section, suggest that such consequences, if any, will not concern unnecessary assignment of paradoxes.

6.5 Pointer structures

Our graphical representation is very similar to Gaifman's pointer structures, [6, 7, 8]. Consequently, the basic intuitions, the arising notion of reference and its significance not only for the understanding of paradox but also of truth, are close to the same. The following points reflect only Gaifman's claims in our language:

(i) Truth is assigned to statements (tokens) and not to sentences (types). The latter obtain specific truth-values only under special circumstances, when the statements they express are identified.

(ii) Truth is determined in a holistic fashion. In general, it is not compositional.

(iii) Consequently, although the aspect of correspondence is still present as sinks, i.e., the statements independent from the rest of the system (discourse), the whole picture combines it with the elements of coherence which are often decisive whenever intra-linguistic reference is involved.

(iv) Truth-predicate belongs to the same level of (natural) language as other predicates, simply because it has only one level. It is a network of references, from which one can extract various hierarchical substructures. When formalized, this predicate does not require coding of the syntax of the language but only an adequate representation of the intra-linguistic reference.

The differences between our and Gaifman's work concern the formal expression of these basic intuitions and their technical elaboration.

1. We give a static condition determining the properties of the admissible assignments, while Gaifman uses exclusively operational rules for their iterative construction. An operational semantics is, however, always in need of a denotational one, in particular, when one wants to reason about the results produced by the operational rules or verify their correctness. The lack of denotational semantics seems to us the main shortcoming of Gaifman's framework. In particular, it can be expected to create problems for design of a corresponding logic. In our case, the (proof of the existence of the) denotational semantics yields a trivially correct algorithm for constructing assignments. We have also given a sound and complete reasoning system for the truth-predicate and a logic for (a significant subset of) statements.

2. Gaifman does not offer a specific theory but a framework. A framework enabling experiments with alternative rules for assigning semantic values is certainly valuable. But just like the goal of experimentation is a definite result, the goal of a framework seems a specific theory (or theories) which, making justified choices, enables improvements of specific results. Our experiments led to the presented formulation, in particular, of the condition (P) in Definition 5.1, and its improvements would be welcome.

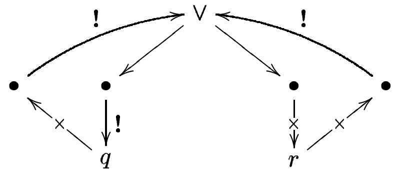
3. As a consequence of the last point, the operational rules vary from paper to paper. Even though some basic items remain unchanged, the results of evaluating particular cases vary depending on the used rules. To make a comparison, we chose the ones from [8] which is the most recent (to our knowledge) and systematic presentation.[12] On p. 23 of [8], the analysis of the case (ii) from our Example 5.5, i.e., $x = \neg y, y = \neg x$, yields, in addition to the two solutions, two other assignments: $\gamma(x) = \bot, \gamma(y) = \mathbf{0}$ and $\delta(x) = \mathbf{0}, \delta(y) = \bot$. Such supply of alternative assignments does not seem any advantage for a system which, intuitively, has nothing paradoxical about itself. If x claims falsity of y while y claims falsity of x then no paradox ensues and the only impossibility is the equivalence of x and y. Additional (im)possibilities might be, perhaps, defended by defending the operational rules giving rise to them. But without denotational semantics, which sets the limits to the soundness of the operational rules, arguments for or against them have little chance of terminating.

An exception to the last claim may be rules which lead to hardly acceptable results and we would consider the above example to be such. In the same paper, Example 1 on p. 19

[12] We use our notation and, in particular, identify both Gaifman's $\neg$ and Fa with our $\neg$, cf. footnote 4.

analyses the following system with the (abbreviated) graph representation:

$$p = !q \vee \neg r$$
$$q = \neg !p$$
$$r = \neg !p$$



The rules force assignment of $\perp$ to all nodes. But $p = \mathbf{1}, q = \mathbf{0} = r$ is the only solution to this system and in our semantics there are no other assignments. The above graph represents also a direct definition of $p = !\neg!p \vee \neg\neg!p$. Its right-hand side is equivalent to $\neg!p \vee !p$ which is a tautology, **t8** of (4.2). This reflects the informal reading of p as claiming that either it is true that ($q =$ it is not true that p) or it is false that ($r =$ it is not true that p).

4. Arguing for our solutions to the above examples, we merely refer to our denotational semantics. It can be criticized but, once accepted, there remains only the question about correctness of the operational evaluation. The assignments to our graphs give the same truth-valued results as solutions of the corresponding systems of boolean equations – provided that the latter exist. Such an equivalence with the accepted theory is, typically, an argument for soundness of a novel formalism.

5. Gaifman's framework internalises the truth-predicate and our truth-predicate shares its general properties. But we have offered its fully formal treatment, including axiomatisation, which gave also a new perspective on the place of propositional logic in the wider context of natural reasoning. The predicate plays distinct role depending on the context of its use.

(i) In the context of classical propositional logic, it is simply identity.

(ii) For propositional logic as meta-logic, i.e., relating to, but not interacting with, possibly paradoxical statements, it is the means of endowing statements with truth-value.

(iii) In the statement logic, it is likewise a truth-assertion enabling extraction of propositional logic, but also a means of making particular statements, like the Truth-teller.

Just like the first point can be used in support of the deflationary theories, the other two suggest their limitations. Our propositional logic allows simple study of the properties of this predicate and identifies precisely the limitations of Tarski's definitions and T-schemata.

6. One final difference is that we did not address infinite graphs, while Gaifman considers both such graphs and first-order logic. With respect to the latter, taking the universal quantification as an infinite conjunction of ground instances, extensions of our results to first-order logic should not present any difficulties, though this remains to be done. Such an extension involves graphs with infinite branching. The more significant from the point of view of paradoxes are, however, graphs with infinite acyclic paths which give rise to Yablo's paradoxes, [19, 20].

Such cases are handled in [8] by the Groundless Pointers Rule which assigns $\perp$ to all involved variables. Thus, for instance, not only in the Yablo's chains but also in the infinite chain $p_1 \longrightarrow p_2 \longrightarrow p_3 \longrightarrow p_4 \longrightarrow \ldots$ all p_i's are assigned $\perp$. Informally, one does not introduce paradoxes without any forcing reasons and here might rather view all p_i's as either **0** or **1**. The same rule applied to the Truth-teller $\circlearrowright p$ assigns $\perp$ to p although, again informally, one would not identify it with the Liar and consider rather only the possibilities of p being **1** or **0**, as does our theory.

The lack of denotational semantics makes such rules appear a bit arbitrary, even if some arguments may be given for their plausibility. An elegant, if only partial, remedy is presented in [3]. Partiality of the remedy consists in that it addresses only the distinction between propositions and paradoxes, without considering possible distribution of $\perp$ among substatements in the later cases.[13] But it gives truth-functional conditions for the existence of solutions for some graphs with infinite paths. It shows that Yablo's chains can be obtained as unwindings of finite cyclic graphs. Its central result is that (truth-valued) solutions for such an unwinding of a finite graph $\mathcal{G}$ can be obtained from the solutions for $\mathcal{G}$. It brings thus the problem of (a large class of) infinite non-circular paradoxes back to the problem of the finite circular ones.

Even if this does not settle the issue of infinite non-circular paradoxes, it strongly suggests that the finite circular cases may be of central importance also for their treatment. We

[13]Another limitation is that [3] does not handle contingent paradoxes like (6.9).

have provided a denotational approach for handling all finite cases. It coincides with the classical logic on classical propositions, with systems of boolean equations on non-paradoxical statements and minimizes the number of paradoxes when these cannot be avoided; it does not leave any semantic gaps, determines semantics of every statement in a unique way, does not involve any infinite constructions and, consequently, is computable. There seems to be no elements in our approach, especially in Definition 5.1, which rely on the involved graphs being finite but applications to the infinite cases require further investigation.

References

[1] Nuel Belnap. Gupta's rule of revision theory of truth. *The Journal of Philosophical Logic*, 11:103–116, 1982.

[2] Dmitri A. Bochvar. On a three-valued logical calculus and its applications to the analysis of paradoxes of the classical extended calculus. *History and Philosophy of Logic*, 2:87–112, 1981. [translation of the Russian original from 1939].

[3] Roy Cook. Patterns of paradox. *The Journal of Symbolic Logic*, 69(3):767–774, 2004.

[4] Solomon Feferman. Reflecting on incompleteness. *The Journal of Symbolic Logic*, 56:1–49, 1991. [publication of the ideas present since 1979].

[5] Solomon Feferman. Axioms for determinateness and truth. *http://math.stanford.edu/~feferman/papers.html*, 2007.

[6] Haim Gaifman. Operational pointer semantics: solution to self-referential puzzles. In Moshe Vardi, editor, *Theoretical Aspects of Reasoning about Knowledge*, pages 43–59. Morgan Kauffman, 1988.

[7] Haim Gaifman. Pointers to truth. *The Journal of Philosophy*, 89(5):223–261, 1992.

[8] Haim Gaifman. Pointers to propositions. In Andre Chapuis and Anil Gupta, editors, *Circularity, Definition and Truth*, pages 79–121. Indian Council of Philosophical Research, 2000.

[9] Laurence Goldstein. 'this statement is not true' is not true. *Analysis*, 52(1):143–153, 1992.

[10] Anil Gupta. Truth and paradox. *The Journal of Philosophical Logic*, 11:1–60, 1982.

[11] Anil Gupta. Truth and paradox. In R. Martin, editor, *Recent Essays on Truth and the Liar Paradox*, pages 175–236. Oxford University Press, 1984.

[12] Anil Gupta and Nuel Belnap. *The Revision Theory of Truth.* MIT Press, Cambridge, 1993.

[13] Hans G. Herzberger. Naive semantics and the liar paradox. *The Journal of Philosophy*, 79:479–497, 1982.

[14] Saul Kripke. Outline of a theory of truth. *The Journal of Philosophy*, 72(19):690–716, 1975.

[15] V. S. Levchenkov. Boolean equations with many unknowns. *Computational Mathematics and Modeling*, 11(2):143–153, 2000.

[16] Sergiu Rudeanu. *Boolean functions and equations.* North-Holland, Amsterdam, 1974.

[17] Helen Skala. The general solution of an arbitrary boolean equation. *The American Mathematical Monthly*, 74(9):1074–1077, 1969.

[18] Lan Wen. Semantic paradoxes as equations. *Mathematical Intelligencer*, 23(1):43–48, 2001.

[19] Stephen Yablo. Paradox without self-reference. *Analysis*, 53(4):251–252, 1993.

[20] Stephen Yablo. Circularity and paradox. In S. A. Pedersen T. Bolander, V. E. Hendricks, editor, *Self-Reference.* CSLI Publications, 2004.