Logic and **L**ogical **P**hilosophy
Volume 33 (2024), 3–26
DOI: 10.12775/LLP.2023.015

Elbert J. Booij

Procedural Semantics and its Relevance to Paradox

Abstract. Two semantic paradoxes, the Liar and Curry's paradox, are analysed using a newly developed conception of procedural semantics (semantics according to which the truth of propositions is determined algorithmically), whose main characteristic is its departure from methodological realism. Rather than determining pre-existing facts, procedures are constitutive of them. Of this semantics, two versions are considered: closed (where the halting of procedures is presumed) and open (without this presumption). To this end, a procedural approach to deductive reasoning is developed, based on the idea of simulation. As is shown, closed semantics supports classical logic, but cannot in any straightforward way accommodate the concept of truth. In open semantics, where paradoxical propositions naturally 'belong', they cease to be paradoxical; yet, it is concluded that the natural choice—for logicians and common people alike—is to stick to closed semantics, pragmatically circumventing problematic utterances.

Keywords: procedural semantics; truth; liar paradox; Curry's paradox

1. Introduction

Semantic antinomies are often addressed against a background of what might be called 'methodological realism'. By this term I want to refer to the presumption, mostly unexpressed yet implicit in the theoretical setup, that objects must be thought of as having their properties *immanently*, i.e., that these properties belong to their bearers, whether or not observed or pondered over, and are as such available for study. This presumption is also compatible with less realistic ontological convictions, hence the qualification 'methodological': for e.g. a nominalist with respect to some domain of entities there is no need ever to run into conflict

Received January 29, 2023. *Accepted* July 13, 2023. *Published online* July 18, 2023

with the predictions of methodological realism, as long as the entities in question do not behave importantly differently than do real objects.

For those who study the logical paradoxes, however, methodological realism is unfortunate, or so I shall argue. For let us have a look at the good old Liar:

1. l = 'l is not true.'

In a methodologically realist understanding of property attribution, it is difficult to escape the conclusion that, for every item x and every property P, the first either instantiates the latter or it does not. However, if P = *being true*, then in both cases 1 immediately reverses this fact, inescapably leading to contradiction. Of course one might hold that having some property is not an all-or-nothing matter. It could be that x is P *to some degree*. But then it would seem that we can divide P into sub-properties P_i, representing all of the degrees to which P can be instantiated, and for any P_i, x either has this property or not. Whether or not x is P *tout court* then depends on which degree P_i is the least that still counts as P, which is likely to be a matter of taste; hence, for every P and every taste there is a fact of the matter: either x is P or not. A more sophisticated approach has it that, apart from being P or not, x has the option of *being neutral* as regards P. But then again, it seems that being neutral as regards P is being fully *positive* as regards *being neutral as regards* P, the alternatives being *being* P and *not being* P. The bottom-line must still be that objects *just are* a certain way, irrespective of who cares to find out.

If so, then the property *truth* is among those ways things can be, and exactly this is what the Liar paradox makes highly problematic to maintain. For l cannot be true, but for that reason it cannot fail to be true either. Any squeezing of a neutral refuge in between truth and falsehood only helps for one round, since it invites 'revenge': on all accounts the neutral territory is an *alternative* to truth and must therefore count as non-truth, only to give the paradox yet another whirl.

A second interesting paradox based on the concept of truth is Curry's paradox. In its most basic form, it is displayed by sentences like c:

2. c = 'If c is true, then the Eiffel Tower is in Rome.'

Assume that c is true. Then the conditional holds, *and* its antecedent is true, so its consequent must be true as well, which means that the Eiffel Tower is in Rome. In summary: if c is true, then the Eiffel Tower is in

Rome. Now, that is just the content of c, hence, c is true. But then the Eiffel Tower must be in Rome—where it is not. Contradiction.

The semantics of choice for methodological realism is *model-theoretic*: the truth of propositions is ultimately based on descriptions (models) of reality in terms of individuals and the properties and relations they instantiate as a matter of primitive fact. My suggestion is that the only way to make sense of paradoxes like the Liar and Curry's paradox is to adopt, at least for properties like truth, a *procedural semantics* instead. If procedures are what eventually determines predication, then the principle of bivalence is false in a special way: the certainty that some object be either P or not P is not taken away by the availability of a third variant, but by the possibility that, since the procedure does not halt, there is no conclusion at all. This is not in itself a new thought: the role of circularity in the emergence of semantic paradoxes has long been appreciated, at least since the dispute between Russell and Poincaré on the matter (Russell, 1906). In Behmann (1931) this point, in relation to Russell's paradox, is put in a slightly more procedural light, when the author speaks of the 'process' of substituting away the 'abbreviation' F (which stands for 'to not apply to oneself'[1]) in the expression $F(F)$, and concludes that 'one gets into an endless regress'.

Here I want to investigate what follows if procedures are viewed, not as ways to find out about meaning, but rather as being constitutive of meaning. In using the term 'procedural semantics', I follow Suppes, who, in relation to issues about the semantics of computational procedures as they were encountered in the 1970's, wrote:

> In finest detail, the meaning of a word, phrase, or utterance is a procedure, or collection of procedures. (Suppes, 1980)

> Thus, if someone tells me that an object in the distance is a cow, I have a perceptual and conceptual procedure for making computations on the input data that reach my peripheral sensory system, and as these data change with the shortening of the distance between the object and me, my computations change and I come to a firm view as to whether the object in question is indeed a cow. (Suppes, 1982, p 29)

Several important ideas leading up to this notion had already been developed in the late 1960's by Tichý (1969, 1971, 2004) as part of his *Transparent Intensional Logic*, a theory in which intensions are defined

[1] I.e. $F(\phi) := \neg\phi(\phi)$

in terms of *constructions* (i.e., algorithmic procedures; see also (Duží et al., 2010)). More recently Horty (2007) and Eder (2019) have identified Frege's concept of *sense* as having procedural affinities. Hints in this direction can also be found in Dummett (1981), in whose work there is a prominent place for the relation between the principle of bivalence and realism (Dummett, 1982, 1991). Eder (*ibid.*) discusses the connection between a procedural view of meaning and the paradoxes, drawing on

> [...] the idea that paradoxical sentences correspond to sense-procedures that, because of their internal structure, fail to determine a truth value.

The type of procedural semantics I have in mind treats the truth of propositions as being determined by the outcome of a procedure: a test, often (but not always) performed on certain physical objects. E.g. to find out whether a certain teacup is blue, the most obvious procedure is using one's eyes to check whether it looks blue. Of course there are other procedures to find out (ask the vendor), but only one of them is *definitional*: that an object is blue is primarily a matter of the way it looks in favourable circumstances. What the vendor says is derivative.

Procedural semantics is not intrinsically at odds with realism. It does, however, provide a way to flesh out anti-realism in such a way that arbitrariness of judgment is avoided. The significance of giving up of methodological realism for dealing with the paradoxes is that we only have to care about actual outcomes: if there is no outcome, then *no conclusion needs to be drawn*. No state of affairs out there for which a blank space must be kept in the books. This is, admittedly, a metaphysical rather than a logical point, but it does have repercussions for the logic as well. One could—indeed, one must—say that there are truth gaps, but as a mere way of speaking: the 'gaps' are no longer represented in the logic itself. This position marks an important difference with the approach of e.g. Tichý (1969), Moschovakis (1994, 2006) and Muskens (2005), by which algorithms, interpreted as Fregean *senses*, are studied as logical objects. According to Tichý and Moschovakis, truth gaps are reflected by truth functions being *partial*. Muskens, who associates propositions with queries in logic programming, explains truth gaps by the divergent behaviour of some of them. Of course all this makes perfect sense, but there is a risk of losing sight of the hard questions about the gaps: there is no mathematical or logical problem with partial functions or diverging queries, but there *is* a conceptual problem with objects being neither P nor $\neg P$ for any predicate, including truth. (A similar point essentially

applies to the ground-breaking work of Kripke (1976), even though there is a clear sense in which the gaps have been theoretically 'dealt with'.)

A second respect in which the present proposal differs from existing approaches is that utter simplicity is pursued with respect to the interpretation of those elements of logic in whose working any complexity appears to be lacking, such as the truth-functional connectives. Surely simplicity can be deceptive, but as more involved treatments of the paradoxes (making use of, e.g. multivalued logic, fixed points, revisions, stratified truth predicates, or different versions of one connective) so far do not seem to have put the issues to rest, it may be worthwhile to see if the procedural approach allows a return to the most basic versions of 'not', 'or', 'every', and so on—even of 'true'. Hence, no predicate is typed in a Russellian sense (Raclavský, 2014). As for the meaning of the quantifiers: the assumption of implicit domain restrictions (e.g., Barwise and Etchemendy, 1987) may be sound in the context of the pragmatics of a conversation, but is hardly convincing as part of the primary meaning of these expressions. Nor shall I at this point be concerned with modality or possible worlds. What is said about simulation, however, (Section 3) does foreshadow a way in which the semantics can be made relevant for counterfactual theorizing in a most natural way.

To see where a procedural approach to semantics leads, in particular in relation to the options for dealing with the Liar paradox and Curry's paradox, I shall first discuss (Section 2) what I want to call *closed* procedural semantics. Here the presumption is that procedures always terminate. In Section 3 the idea of hypothetical reasoning will be expounded in terms of procedures. With the help of these preliminaries it will be demonstrated that closed procedural semantics yields classical logic, but the truth-predicate (to be separately discussed in In Section 4) cannot be made to fit into this scheme. Section 5 will then be dedicated to how open procedural semantics differs from the closed variant. It will be shown that this semantics can accommodate truth in its naive form, without thereby inviting either the Liar or Curry's paradox, but on pain of losing much of classical logic. In the final section I shall argue that the 'natural' solution to deal with this situation is to purge the language from problematic utterances, thereby—in a very pragmatic way—keeping the best of both worlds.

2. Step One: Closed Procedural Semantics

What would the procedure for the truth-predicate be like? Let b be the sentence 'This teacup is blue', and consider the following sentence:

3. b is true.

An intuitive procedure for finding out whether 3 is true would be the following:

1. Find out whether b is true,
2. i.e. find out whether 'This teacup is blue' is true,
3. i.e. find out whether this teacup is blue.

The last step consists of calling the procedure for blueness, and applying it to the teacup. Let us assume that the teacup is indeed blue, then the latter procedure will return a positive outcome, which will feed the procedure for the truth of b, making it return a positive outcome as well. Hereby the conclusion that b is true has been affirmed and the procedure halts.

It is important to distinguish between two levels of reasoning. On the *utterance level* we say things like: 'This teacup is blue,' i.e., we utter some proposition p. On the *procedural level*, where we have a glance 'under the hood of' the cognitive engine, we speak about procedures. Here I shall use the following notation: Π_p is the procedure that determines whether p. The procedure could return a positive outcome (or $+$) in which case, on the utterance level, p is confirmed. It could also return a negative outcome, (or $-$), in which case p is refuted. The crucial point is that it could also, for a variety of reasons, *fail* to return an outcome. If so, then there is nothing to be said on the utterance level.

This third possibility is the reason why procedural semantics is in a better position to deal with Liar sentences and the like. Applying the procedure for truth to the Liar ($\Pi_{T(l)}$), what we do is, informally speaking, the following:

1. Find out whether l is true,
2. i.e. find out whether 'l is not true' is true,
3. i.e. find out whether l is not true,
4. i.e. find out whether l is true and reverse the outcome,
5. ...

However, step 4 includes running the procedure for truth with respect to l. Since this is done as part of this very procedure, the algorithm is

caught in a loop, so it will not terminate. It will not return an outcome. Hence, we are, at the utterance level, at a loss to say whether or not l is true (this, in my opinion, squares tolerably well with philosophical experience!). The thing to keep in mind is that this condition *cannot* be recast as an outcome in its own right, for this would alter the procedure and with it (by the very logic of procedural semantics) the predicate that is being evaluated. I shall return to this point below.

It is useful to draw the following distinction. In *closed procedural semantics* (hereafter, for short, 'closed semantics') the additional presumption is that every (semantic) procedure will always return an outcome. This will be called *the closure presumption*. The closure presumption is an assumption on the procedural level. It will be shown that the logic belonging to closed semantics is classical logic. In *open procedural semantics* the closure presumption is absent.

Although the general idea can be extended to richer languages, we shall, in the interest of simplicity, in this text stick to (non-modal) first-order logic. Every predicate will have its procedure, and every logical constant as well. We shall assume that all semantic procedures can be meaningfully represented by some *propositional* content p, and write Π_p for the defining procedure for p (there may be many procedures computing the same result, but only one of them has this status). A procedure as understood here will always produce *the same* outcome when run on the same input; this will be called *the stability presumption*. As said above, $+$ and $-$ are the only possible outcomes. Procedures will often be combined to produce compound procedures.

In a procedural analysis of expressions of first-order logic, the terminal sub-procedures are those belonging to *atomic* sentences. Defining these in procedural terms is comparatively demanding: clearly, the outcome of, say, $\Pi_{p\wedge q}$ depends on that of Π_p and Π_q in an extremely straightforward way. Nothing comparable is available for the procedure (let us write $\Pi_{B(c)}$) by which to decide whether teacup c is blue (B). As our present interest is almost exclusively in procedures of the former type, my discussion of atomic sentences will be brief and somewhat sketchy. The bottom-line is that $\Pi_{B(c)}$ will be the defining procedure that returns $+$ if teacup c is blue and $-$ if it is not. This is what Definition 2.1 says.

Definition 2.1 (Defining procedure for an atomic sentence). Let Φ be a procedure with set of outcomes $\{+, -\}$, which has access to input from n

objects, and let $\Phi(a_1 \dots a_n)$ be Φ operating on the objects $a_1 \dots a_n$. Let furthermore P be a predicate such that, by definition, for any choice of a_1 through a_n, $P(a_1 \dots a_n)$ iff $\Phi(a_1 \dots a_n)$ returns $+$ and $\neg P(a_1 \dots a_n)$ iff $\Phi(a_1 \dots a_n)$ returns $-$. Then $\Phi(a_1 \dots a_n)$ is the defining procedure for the atomic sentence $P(a_1 \dots a_n)$, i.e., $\Phi(a_1 \dots a_n) = \Pi_{P(a_1 \dots a_n)}$.

The phrase 'access to input from ... objects' in Definition 2.1 is broadly interpretable. 'Access' suggests a causal efficacy on the process from the part of the objects, but they can be ignored as well. The objects in question, furthermore, can be anything from which information can be extracted, ranging from teacups and bosons to prime numbers. I believe this is as it should be: in the interesting cases Φ is a nontrivial procedure which gathers significant information from the objects it operates on, thereby making P a meaningful predicate. An example of such a Φ is the procedure of using one's eyes to see if some object is blue. A lot more could be said about the procedures that 'find' the objects *themselves* (belonging to e.g. descriptions, proper names, and functions), but that would be too much a digression from the core issues to be discussed here.

Definition 2.2 provides a list of the defining procedures of most the basic logical symbols. As there are alternative definitions possible, these are presented as named rules.

DEFINITION 2.2 (Defining procedures for the logical constants).

CONJ	$\Pi_{p \wedge q}$: Run Π_p and Π_q simultaneously. Return $+$ if both procedures do. Return $-$ if one of them does.
DISJ	$\Pi_{p \vee q}$: Run Π_p and Π_q simultaneously. Return $+$ if one of them does. Return $-$ if both procedures do.
NEG	$\Pi_{\neg p}$: Run Π_p and return the opposite of its outcome.
IMP	$\Pi_{p \rightarrow q}$: Run Π_p. If the outcome is $+$, run Π_q and return its outcome. If the outcome is $-$, return $+$.
FALS	$\Pi_{\perp}$: Return $-$.
EXIST	$\Pi_{\exists x p(x)}$: Run $\Pi_{p(x)}$ with respect to all objects x in the relevant domain simultaneously. Return $+$ if one of them does. Return $-$ if all of them do.
ALL	$\Pi_{\forall x p(x)}$: Run $\Pi_{p(x)}$ with respect to all objects x in the relevant domain simultaneously. Return $+$ if all of them do. Return $-$ if one of them does.

It may seem as though, by the present proposal, finding out the truth of some statement p always proceeds by running Π_p, but this is not so. To find out that p is to find out that, *should* Π_p *be run*, it will return a positive outcome. This can be done by performing the test immediately, but that is not the only way to obtain this knowledge—not even the typical way. Most things that are known, are not known by immediate perception, but by *reasoning*. If the semantics is procedural, then it is natural to take a procedural view on reasoning as well.

Reasoning often starts from existing knowledge. Suppose someone knows, for certain p and q, that $p \wedge q$. By procedural semantics, to know that $p \wedge q$ means that one has access to the information that running $\Pi_{p \wedge q}$ will yield a positive outcome. (This should not be mistaken for *knowing that* $\Pi_{p \wedge q}$ will have that outcome—that is knowledge on the procedural level, which need not in any way reach the awareness of the knower: all she knows is that $p \wedge q$.) Knowledge on the procedural level has explanatory value for what else the knower can know without further information. Hence, if what is known is $p \wedge q$, we can infer, on a meta-level, that, as the knower has access to the fact that $\Pi_{p \wedge q}$ will return $+$, she must, given the description of $\Pi_{p \wedge q}$ according to CONJ, also have access to the fact that Π_p will return $+$.

This, by the present approach, is what we express by writing down a deduction rule like $p \wedge q \vDash p$, the elimination rule for conjunction. The introduction rule ($p, q \vDash p \wedge q$) can be motivated in a similar way. Observe that only the procedural definition CONJ is needed to obtain both rules, which means that, if CONJ belongs in some way effectively to the cognitive repertoire of the knower, she has implicit knowledge of both rules. The fact that CONJ is the only defining procedure for conjunction secures that they hang together in the right way. If this is true for all logical concepts, then *Tonk*-like phenomena[2] are impossible in the procedural semantics.

The proofs for all the rules in Lemma 2.1 proceed similarly. Most of them are very immediate.

Lemma 2.1. *The following deduction rules are valid for arbitrary p and q:*

$\wedge$-intro	$p, q \vDash p \wedge q$	
$\wedge$-elim	$p \wedge q \vDash p$	$p \wedge q \vDash q$

[2] The 'perverse' connective *tonk* combines the introduction rule of the disjunction with the elimination rule of conjunction (Prior, 1960), thereby making the logic to which it belongs trivial.

$\vee$-intro	$p \vDash p \vee q \qquad q \vDash p \vee q$
$\vee$-elim	$p \vee q, p \to r, q \to r \vDash r$
dissyll	$p \vee q, \neg p \vDash q$
$\to$-elim	$p, p \to q \vDash q$
dblneg	$\neg\neg p \vDash p$
negexist	$\neg\exists x p(x) \vDash \forall x \neg p(x)$
negall	$\neg\forall x p(x) \vDash \exists x \neg p(x)$

PROOF. (of the less obvious cases)

$\vee$-elim: If $\Pi_{p\vee q}$ returns $+$, then either Π_p or Π_q must have done so (DISJ). Reasoning by cases, using $\to$-*elim*, yields the result.

dissyll: The disjunctive syllogism can be derived without recourse to $\vee$-elim. Given DISJ, if $\Pi_{p\vee q}$ returns $+$ and Π_p returns $-$, then it must have been due to Π_q returning $+$.

dblneg: To return the opposite of the opposite of $+$ (by twice applying $\Pi_{\neg p}$) is again to return $+$. The usual introduction rule for negation (*reductio*, see below) plays no part here. ⊣

Clearly, Lemma 2.1 is incomplete, which is because some of the missing rules can only be elucidated after having discussed hypothetical reasoning.

3. Hypotheses and Simulation

In the previous section, we saw some examples of reasoning from what is known, e.g. that $p \wedge q$. As a matter of fact, however, the procedural machinery works just as well in total absence of empirical (or any) knowledge. It is a remarkable fact about the human type of intelligence that we may know that *if* $p \wedge q$, *then* surely p, even though we do not have the slightest clue as to whether $p \wedge q$ in the first place. This ability is called *hypothetical* reasoning. Compare Ramsey's famous quote on the matter (2000):

> If two people are arguing 'If p, then q?' and are both in doubt as to p, they are adding p hypothetically to their stock of knowledge and arguing on that basis about q; so that in a sense 'If p, q' and 'If p, $\overline{q}$' are contradictories. We can say that they are fixing their degree of belief in q given p.

What does it mean to 'add p hypothetically to (one's) stock of knowledge'? I would suggest that assuming some state of affairs p hypothetically is running a *simulation* of the procedure Π_p, *making it return* $+$. In

most cases, one cannot force Π_p to do so—that would be to manipulate reality itself! What is needed is an *ersatz procedure* (hereafter, an *EP*), let us call it Σ_p, which is realistic enough, but open to manipulation, in order to compute different scenarios. Here 'realistic' does not mean real: a good toy model with enough analogy to what really happens will do. In fact, for the type of hypothetical reasoning that logic is interested in, the analogy *must* be limited, because logical deductions abstract away from specific propositions.

The approach to simulating procedures I will advance here is based on the fact that procedures contain *sub-procedures*. As the procedure itself will be counted among its own sub-procedures, they do so by definition; but generally speaking, sub-procedures are fully functional procedures in their own right. Although they may get their input from various sources, to play the theoretical role I have in mind for them, they also must have $+$ or $-$ as their only possible outcomes and otherwise be opaque from the viewpoint of the main procedure. In the causal structure of a procedure we can furthermore distinguish between *proximal* and *distal* processes. As regards 'calling' (the activating of a procedure), proximal processes causally precede distal ones, whereas the causal flow of the outcomes goes in the opposite direction. E.g. $\Pi_{p\wedge q}$ calls its sub-procedure Π_p, which then produces an outcome that has an effect on $\Pi_{p\wedge q}$ in turn.

An EP $\Sigma_{p\wedge q}$ for $\Pi_{p\wedge q}$ will be a procedure that is *proximally identical* to $\Pi_{p\wedge q}$, i.e., it is an exact copy of it, but for some sub-procedures, in this case Π_p and Π_q, which have been replaced by others, say, Σ_p and Σ_q. Let us call these the *test procedures*. Test procedures are 'dummies': all they do is return an outcome of choice, thanks to which the behaviour of the EPs can be studied, and therefore, by proxy, that of the real procedures. A *test case* is a combination of outcomes from the test procedures. Of course, this also requires that all the EPs involved in the process are connected in the same way that the original procedures are. Let us call EPs meeting this requirement *coordinated* EPs. As the above example suggests, exactly which of the sub-procedures Π_{p_1}, Π_{p_2}, ... of the original procedure will be replaced by EPs is an essential feature of the simulation. I shall call this its *specificity*, which will be represented by the set $\{p_1, p_2, \dots\}$ of their subscripts.

Here is an example (Fig. 1). Suppose we have a procedure Π_c, where $c =$ *This teacup is blue and there is no rhinoceros in this room.* The content of c makes Π_c a very complex procedure. An obvious way to realize it, however, is to use the sub-procedures Π_b and Π_r, testing for

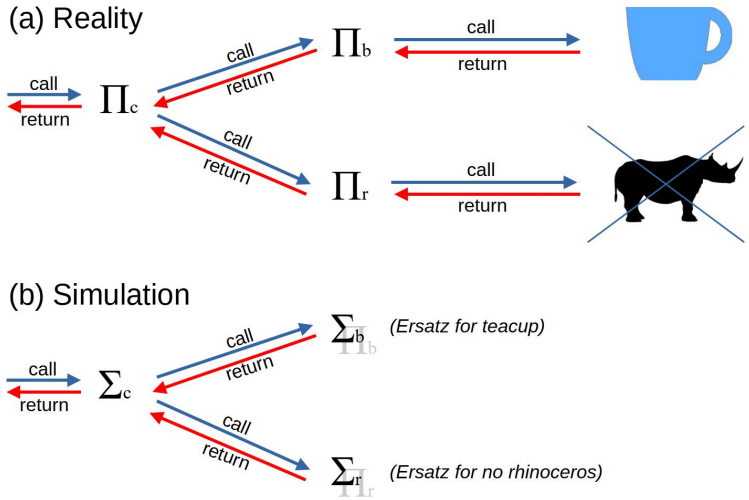

Figure 1. Simulation of a procedure

blueness of the cup and this room being rhino-free, with Π_c (working according to CONJ) being a test for their conjunction. Suppose we are neither interested in the colour of teacups, nor in the content of this room. All we want to know is whether or not b (*This teacup is blue*) follows from c. Such knowledge is *not* commonly obtained (although it would certainly work) by some massive simulation involving a teacup and a room, leading to the observation that, if this is how things stand, the teacup is indeed blue. Thinking in terms of procedures makes it obvious that it is enough to consider the proximal part of Π_c. This is the part that CONJ describes, and to my mind it is no coincidence that procedural shortcuts of this type exactly parallel the sort of semantic abstractions that logic is all about. In this case, to run a simulation of Π_c, the EP Σ_c must be of specificity $\{b, r\}$.

In simulations of, say, the weather or the stock-markets, we want the simulation to match reality as well as possible, which, in the terminology used here, would come to seeking maximal specificity. In logic, however, limiting specificity serves to *abstract away* from specific content. A way to achieve this is to let the specificity of the EPs involved in the simulation be that of the set of propositional terms used in the deduction

(for present purposes, we are interested in propositional, not individual terms[3]). We shall call this *the specificity of the deduction*. In fact, a specificity below that of the deduction may work as well. This can be defined as follows. Let Π_p be a procedure with simulations Σ_p and Σ'_p, then the specificity of the second will be said to be *below* that of the first if Σ'_p is identical to Σ_p, except that some test procedures of Σ'_p replace more proximal sub-procedures of Π_p than do those of Σ_p (this effectively makes Σ'_p a simulation of Σ_p). A simulation of specificity below that of the deduction will never support an invalid deduction, but it may fail to establish a valid one.

With all this in place, we can say what it means for a deduction $p_1, p_2, \cdots \vDash q$ to be *established by simulation*. Once we have coordinated EPs Σ_{p_1}, Σ_{p_2}, ..., and Σ_q, for Π_{p_1}, Π_{p_2}, ..., and Π_q respectively, of the specificity of the deduction or below, to carry out the simulation means to try all possible test cases, to see if, whenever the outputs of Σ_{p_1}, Σ_{p_2}, ..., are $+$, that of Σ_q is $+$ as well. This will establish the deduction. And this will, by Lemma 3.1, allow us to trust its validity:

Lemma 3.1. *If the deduction* $p_1, p_2, \cdots \vDash q$ *has been established by simulation, then, if* Π_{p_1}, Π_{p_2}, ... *all return* $+$, *so does* Π_q.

Proof. For the coordinated EPs Σ_{p_1}, Σ_{p_2}, ... and Σ_q (for Π_{p_1}, Π_{p_2}, ... and Π_q, respectively) there is a test case by which the outcomes of the test procedures exactly match those of the original sub-procedures of Π_{p_1}, Π_{p_2}, ... and Π_q. Suppose that Π_{p_1}, Π_{p_2}, ... all return $+$, then in this test case Σ_{p_1}, Σ_{p_2}, ... all return $+$ as well, for they are proximally identical to them. That the deduction $p_1, p_2, \cdots \vDash q$ has been established means that, whenever Σ_{p_1}, Σ_{p_2}, ... all return $+$, so does Σ_q, and therefore also in this case. Hence, by proximal identity, Π_q returns $+$ as well. ⊣

With this account of hypothetical reasoning we can expand our stock of deduction rules.

Lemma 3.2. *The following deduction rules are valid for arbitrary* p:

falsum $\quad p, \neg p \vDash \bot$
efsq $\quad \bot \vDash p$

[3] Employing procedural semantics for the explanation of the behaviour of *individual* terms in deductions in first-order logic is quite an additional challenge. Fortunately, this is inessential to the issues discussed here.

PROOF. falsum: In no test case will the EPs Σ_p and $\Sigma_{\neg p}$ both return $+$, thus establishing the deduction vacuously.

efsq: For an EP $\Sigma_\perp$ for $\Pi_\perp$ we can take one identical to the original. This will in all 'test cases' return $-$, thus establishing the deduction vacuously. ⊣

For the next couple of rules we need the closure presumption. Apart from this, two of them ($\rightarrow$-*intro* and *reductio*) are special in that they are not established by simulation, but by considering what would follow *if* the antecedent of the statement would be established by simulation.

LEMMA 3.3. *The following deduction rules are valid for arbitrary p and q in closed semantics*:

bival	$\vDash p \vee \neg p$
$\rightarrow$-intro	$(p \vDash q) \vDash p \rightarrow q$
reductio	$(p \vDash \perp) \vDash \neg p$

PROOF. bival: Thanks to the closure presumption, Π_p will return either $+$ or $-$, in which latter case $\Pi_{\neg p}$ returns $+$ (NEG). The result follows from the definition of $\vee$ by DISJ.

$\rightarrow$-intro: Suppose Π_p returns $+$, then, thanks to Lemma 3.1, so does Π_q and, by IMP, $\Pi_{p\rightarrow q}$ as well. If not, thanks to closure, Π_p must return $-$ and, by IMP, $\Pi_{p\rightarrow q}$ again returns $+$.

reductio: By Lemma 3.1, if Π_p returns $+$, then so does $\Pi_\perp$. As the latter is not the case, the former cannot be either, so, by closure, Π_p returns $-$ and, by NEG, $\Pi_{\neg p}$ returns $+$. ⊣

THEOREM 3.1. *Closed procedural semantics provides a full implementation of classical logic.*

PROOF. Lemmas 2.1, 3.2, and 3.3 provide a complete inventory of the rules of classical logic. ⊣

4. Truth

Definition 2.1 can be read as a very general template for how to handle predication in procedural semantics. One of the predicates is the truth-predicate T. In the interesting cases, the object a for which $T(a)$ has to be tested is something of a propositional nature. I take this to be an easy test, and one that will always reach a verdict: its outcome should depend on properties of a which can be determined unconditionally (e.g.,

being a well-formed formula). We could define the procedure for T so as to make it produce a negative outcome if a is not propositional, or no outcome. Let us choose the first option; nothing depends on this decision. If a is propositional, then it is the name of a proposition, say p, which we can also name 'p', so that we can say things like $a =$ 'p'. To retrieve the content of a we shall write $[a]$, so $p = [a]$.

There is little doubt that the *intuitive* understanding of truth is such that anyone who understands the concept will assent to $T(\text{`}p\text{'})$ just in case they assent to p. At least since the work of Tarski, it has been known that this notion of truth is in conflict with classical logic; here I nevertheless intend to stick to this utterly simple idea (also known as *naïve*, or *transparent truth*). The procedural definition will be:

Definition 4.1 (Truth).

TRUE $\Pi_{T(a)}$: If a is propositional, run $\Pi_{[a]}$ and return its outcome; else, return $-$.

Given this, the rules for the truth-predicate T are very straightforward:

Lemma 4.1. *For any proposition p, the following deduction rules with respect to the predicate T are valid*:

T-intro $p \vDash T(\text{`}p\text{'})$
T-elim $T(\text{`}p\text{'}) \vDash p$

When studying the behaviour of the paradoxes, we can describe procedures as it is done in Definition 2.2; however, to see how a procedure works out in practice, it is often helpful to make use of what I shall call an *unfolding scheme* (two of these were given in an informal way in the Introduction). This is a step-wise representation, not of what has to be done in succession; it is a re-wording of the action to be taken, unpacking every step in the next, guided by definitions or information. For the procedure of finding out if l is true, the unfolding scheme would look like this:

1. Run $\Pi_{T(l)}$ and return its outcome.
2. i.e. run $\Pi_{T(\text{`}\neg T(l)\text{'})}$ and return its outcome. (=)
3. i.e. run $\Pi_{\neg T(l)}$ and return its outcome. (TRUE)
4. i.e. run $\Pi_{T(l)}$ and return the opposite of its outcome. (NEG)

...

As we saw before, the procedure gets into a loop (step 4), preventing there to be an outcome.

Confronted with this fact, it is tempting to try to handle this no-outcome condition and turn it into an outcome in its own right, so it is worth pursuing this idea to see where it leads. Let us try to extend the procedure, i.e., to insert a second procedure which will return an outcome just in case the old procedure would not. As this is another alternative for the outcome +, it seems reasonable to redirect this new outcome to −. By way of hypothesis, assume that we have this 'improved' version of TRUE, call it TRUE′, at our disposal. In this context we temporarily rename the old predicate (according to TRUE) and call it T^0, the result of which is that the rules of Lemma 4.1 are valid for T^0. So for the predicate T we now have:

DEFINITION 4.2. Truth, alternative definition

TRUE′ $\Pi_{T(a)}$: Run $\Pi_{T^0(a)}$ and return its outcome. If there is no outcome, return −.

If we apply this procedure to the truth of l, we must consider the following cases.

Suppose that $\Pi_{T(l)}$

- returns +. Then $\Pi_{T^0(l)}$ must have returned +, i.e. $\Pi_{T^0(\text{`}\neg T(l)\text{'})}$ must have. Therefore $\Pi_{\neg T(l)}$ must have returned +, and $\Pi_{T(l)}$ must have returned −. Contradiction.
- returns −. Then:
  - either $\Pi_{T^0(l)}$ must have returned −, i.e. $\Pi_{T^0(\text{`}\neg T(l)\text{'})}$ must have. Therefore $\Pi_{\neg T(l)}$ must have returned −, and $\Pi_{T(l)}$ must have returned +. Contradiction.
  - or $\Pi_{T^0(l)}$ must have returned no outcome, likewise for $\Pi_{T^0(\text{`}\neg T(l)\text{'})}$. Therefore $\Pi_{\neg T(l)}$ has returned no outcome, and neither has $\Pi_{T(l)}$. Contradiction.
- returns no outcome. This is impossible, given the definition of TRUE′.

The conclusion must be that TRUE′ does *not* define a feasible procedure.

This should not come as a surprise: it is well known from Turing's proof that there is no general solution to the halting problem. But a procedure as per TRUE′ must be prepared to decide about halting for *any* $\Pi_{T^0(a)}$, thus, *any* $\Pi_{[a]}$, belonging to any statement a whatsoever.

Just waiting to see if $\Pi_{T^0(a)}$ stops will not work! Exactly this is what belies the seemingly innocent phrase '*If there is no outcome*' in the definition TRUE′. The point is quite general: there is no way to improve on TRUE in such a way that there will always be an outcome. Only if we *could* be certain of the halting behaviour of a procedure, could we view the hanging of a procedure as an outcome. That this cannot be done is quite fundamental.

Yet, what is most worrisome about the predicate T according to TRUE is not its 'gappiness'. The real problem is that, at least in classical logic, we can use seemingly legitimate deductions to derive blatant falsehoods. Let us first take the Liar paradox. Just as above, $l =$ '$\neg T(l)$':

DEDUCTION 4.1. *The Liar paradox*

1	[$T(l)$]	(ass.)
2	$T(\text{`}\neg T(l)\text{'})$	(=)
3	$\neg T(l)$	(T-elim)
4	$\bot$	(falsum)
5	$\neg T(l)$	(reductio)
6	$T(\text{`}\neg T(l)\text{'})$	(T-intro)
7	$T(l)$	(=)
8	$\bot$	(falsum)

It seems that we can deduce a contradiction out of nothing—i.e., with no potentially fallacious presumptions.

Curry's paradox is brought forth by a sentence that is also self-referential, but not by saying of itself that it is false, but by saying of itself that *if* it is true, some claim—freely to be chosen—is true. Here we shall take $r =$ *The Eiffel Tower is in Rome*. The Curry sentence will be c, with $c =$ '$T(c) \rightarrow r$'. Now we can make the following deduction:

DEDUCTION 4.2. *Curry's paradox*

1	[$T(c)$]	(ass.)
2	$T(\text{`}T(c) \rightarrow r\text{'})$	(=)
3	$T(c) \rightarrow r$	(T-elim)
4	r	($\rightarrow$-elim)
5	$T(c) \rightarrow r$	($\rightarrow$-intro)
6	$T(\text{`}T(c) \rightarrow r\text{'})$	(T-intro)
7	$T(c)$	(=)
8	r	($\rightarrow$-elim)

leading to the very obvious (if contingent) misstatement that the Eiffel Tower is in Rome (r).

Having to choose between truth and classical logic is for many a logician quite conflict of loyalties. It has, however, long been appreciated that the conflict must be faced (Burgess, 1986; Gupta, 1982; Leitgeb, 2007; McGee, 1985). Some choose bending truth, whereas others bend classical logic. Let us first try the second horn.

5. Step two: Open Procedural Semantics

There is nothing illegitimate in questioning whether it is really worth the effort to modify the beautiful edifice of classical logic, *mainly* to make room for a single predicate to have its way. But truth is not just any predicate. It is a profoundly logical concept—arguably *the* central concept of logic. It is annoyingly simple and unmistakably fundamental. We use the concept even before we reflect on it: to say that some statement is true is often just a way to make that statement. No difference in intention whatsoever.

To throw out truth seems inconceivable. Yet, if we stick to TRUE as the defining procedure, there is, for arbitrary a, no general rule to the effect that $\vDash T(a) \vee \neg T(a)$. To preserve truth on these terms, therefore, it seems that we must look beyond closed semantics. This operation comes with revisions to classical logic. Procedural semantics offers a way of doing this, one in which the passage from classical logic into a logic that handles predicates like T is entirely natural. The departure from our original assumptions is minimal: it is only dropping the closure presumption. Consequently, all the definitions in Definition 2.1, 2.2, and 4.1 will be kept as they are, and, as a result, all the deduction rules in Lemma 2.1, 3.1, and 3.2 remain valid. But this modification, however modest, deprives us from the rules in Lemma 3.3.

The good news is that, as is easily shown, it is enough to dismantle both paradoxes under discussion. As for the Liar, Deduction 4.1 is blocked at step 5, owing to the failure of *reductio*. Deduction 4.2 (Curry's paradox) is blocked at step 5 for want of $\rightarrow$-*intro*. Thus, there will be no explosion (even though the bomb itself, *efsq*, is still present in open semantics (Lemma 3.2)!). And the Eiffel Tower stays where it is.

Reassuring though all this may be, one might still want to learn in more detail how it is possible at all for these rules to fail. Take

reductio: how could the fact that the assumption that p has impossible consequences *not* lead to $\neg p$? The answer is that, in open semantics, Π_p might not halt at all: that it is demonstrably unable to return $+$ does not mean that it will return $-$. It may also be that Π_p returns *no* outcome—as is the case with $T(l)$.

In Deduction 4.2 (Curry's paradox), the main character on stage is not *reductio*, but $\rightarrow$*-intro*. In this case, the question is even more pressing: whence its failure? How is it that we can have a valid deduction $p \vDash q$ and yet no implication $p \rightarrow q$? This time, the answer appears to be that deductions, in the present interpretation, come with the assumption that the procedures involved reach a conclusion. The deduction $p \vDash q$ is established, once it is demonstrated that *if* Π_p returns $+$, *then* so does Π_q. But only if the first procedure halts, will $\Pi_{p \rightarrow q}$ return $+$. If Π_p does not return an outcome, then neither will $\Pi_{p \rightarrow q}$. And we can indeed prove directly that $\Pi_{T(c)}$ will not halt. To know if c is true, we must proceed as follows:

1. Run $\Pi_{T(c)}$ and return its outcome.
2. i.e. run $\Pi_{T(\ulcorner T(c) \rightarrow r \urcorner)}$ and return its outcome. (=)
3. i.e. run $\Pi_{T(c) \rightarrow r}$ and return its outcome. (TRUE)
4. i.e. run $\Pi_{T(c)}$. If it returns $+$, run Π_r and return its outcome; if it returns $-$, return $+$. (IMP)

...

which is circular.

Now, although this certainly appears to bring some clarity into the status of the Curry sentence itself, it may still seem somewhat strange that Deduction 4.2 is valid up to step 4—in open as well as in closed semantics. Or, more generally, how could a deduction $p \vDash q$ be established in the first place, if we know that Π_p does not return an outcome?

The answer to this riddle is *specificity*. A simulation, after all, remains a surrogate: it is ascertained to behave as the original procedure would have, provided that the latter terminates. But a less specific simulation may reach a conclusion where the original procedure would not have reached one. Since most deductions (including Deduction 4.2) are built up from smaller ones, which can be established by simulations of a specificity below that of the whole deduction, it may occur that, even though all the building blocks of the argument can be soundly established, the argument as a whole is a misrepresentation of what actually obtains. This is possible if the closure presumption fails.

For a very simple example of just this discrepancy between simulation and original, consider the deduction $T(l) \vDash T(l)$. It is doubtlessly valid (it is an instance of $p \vDash p$), thanks to the simulation of specificity $\{T(l)\}$ (in which a test-procedure Σ_p replaces the *entire* procedure $\Pi_{T(l)}$). Yet, as is easily checked using IMP, it is not possible to confirm $T(l) \rightarrow T(l)$.

6. Discussion

The motivation behind procedural semantics as defended in this text has much in common with that behind Kripke's (1976) theory of truth, or the Revision Theory of Truth (Gupta and Belnap, 1993; Herzberger, 1982). But where both these approaches start out from realistic assumptions, subsequently trying to see how—and to which extent—the realm of truth can be charted by studying multiple iterations of its application (using stabilization or fixed points), no such conception pertains to the present proposal. In procedural semantics, there is no convergence to any sort of verdict with respect to the truth of paradoxical sentences. In relation to this, neither the concept of truth, nor the logic in which it is embedded, are in any way enriched with extra notions, like neutral values (Kleene, 1938), operators modifying the truth-predicate, or non-classical alternatives for implication (Field, 2002, 2003). Therefore the spirit of procedural semantics may be closer to that of logical *intuitionism*, where truth is also found by executing a procedure (searching for a proof). Here, like in open procedural semantics, the point of departure from classical logic is giving up bivalence, where the 'gap' is not filled with some neutral value, but really left open. Also, the constructivist character of intuitionism shows clear kinship with the anti-realist treatment of truth presented here.

Thanks to this anti-realist element, procedural semantics provides full legitimation for the contingency that the truth of some proposition p, no matter how well-formed and meaningful, cannot be resolved. On the utterance level, we can say that the teacup is blue, but not that the Liar is true (or false)—of course we can *say* so, but without any solid ground. There is nothing to say, and even this *nothing to say* is more like an exclamation in the Wittgensteinian sense, than the pointing at a state of affairs. Very much in Wittgenstein's spirit, we had better keep silent about such propositions. We may gladly accept this limitation, for on the procedural level we *are* able to speak about such things. But

common discourse is not like that, and, on pain of talking nonsense, we must try to find out which utterances to eschew.

That is one problem. The second problem is which semantics should be preferred: closed or open? The utterly central concept of truth appears to demand open semantics, but, as we have seen, in this system deductive reasoning is seriously impoverished; downright embarking on open semantics would leave us with a sorry residue of the logic we know and love. But *is* there a way out, if we want to keep truth? Strangely enough, there appears to be one. The point is that it is hard to see what benefits, apart from accommodating truth, open semantics has in store anyway, given that the accommodation does not go beyond preventing the paradoxes to cause explosion. Indeed, if it held other promises, it might be given a try; but despite its positive ring, 'open' semantics is a barren land, where no logician should want to stay longer than necessary. It is its existence which is interesting, not its content.

So how can we stay out of it? The fact is that we do so on a daily basis. We just circumvent propositions which are problematic in the way that the Liar and Curry's sentence are. There is, in Yablo's words

> no ground for claiming that our procedures are *vitiated* by the paradoxes, unless one is prepared to accept that we find their vitiation very little handicap in practice. (Yablo, 1985)

Because, on the utterance level, we stay within the segment of language where the closure presumption holds, we have all of classical logic at our disposal, *and* truth! And so, in the dilemma mentioned above—bending truth or bending classical logic—we lean towards the first horn after all. There remains a significant challenge, though, which is that no intrinsic feature of propositions is fully discriminating. Propositions whose procedures will terminate cannot be told apart by being non-well-formed, nor can they be picked out by the fact that they contain the truth-predicate (see Russell's paradox), or by their being self-referential. Self-referring sentences need not be paradoxical (e.g., 'This sentence contains no adverb.'), and there are paradoxes without self-reference (Yablo, 1993).

As far as the use of the truth-predicate is concerned, whether or not its use gives rise to circular reference is dependent on *global* properties of the entire domain of discourse. Kripke (1976) cites an amusing dialogue between Russell and Moore, which must have gone more or less like this:

Russell: George, surely you always tell the truth, don't you?
Moore: Well, that's not quite correct. Not always!
Russell: Man, that must have been the only falsehood you ever uttered.

Should Moore, when he was five years old, once have misled his mother about taking a candy, Moore would have been right, and Russell, wrong. But if Moore was really as truthful as Russell took him to be (i.e., in all his other utterances) then everything said in the above dialogue is paradoxical.

It would seem, then, that, after having purged our conversations from problematic claims, we make our utterances in good faith—trusting them to make sense, which, on the procedural level, means 'trusting' the procedures to be effective. It appears to be a fruitful strategy. In everyday life, the concept of truth thrives in large swathes of classical discourse. There is always a risk of hitting a snag, and making claims that work out paradoxically. But these are only words; nothing will explode in the real world. Surely this is an astonishingly *pragmatic* way of settling the stand-off between classical logic and truth, but it seems to me that procedural semantics demonstrates that there are few other options.

References

Barwise, J., and J. Etchemendy, 1987, *The Liar: An Essay on Truth and Circularity*, Oxford University Press.

Behmann, H., 1931, "Zu den Widersprüchen der Logik und der Mengenlehre", *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 40: 37–48.

Burgess, J. P., 1986, "The truth is never simple", *The Journal of Symbolic Logic*, 51 (3): 663–681. DOI: 10.2307/2274021

Dummett, M., 1981, *Frege: Philosophy of language*, Harvard University Press.

Dummett, M., 1982, "Realism", *Synthese*, 52 (1): 55–112. DOI: 10.1007/BF00485255

Dummett, M., 1991, *The Logical Basis of Metaphysics*, Harvard University Press.

Duží, M., B. Jespersen and P. Materna, 2010, *Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic*, volume 17, Springer Science & Business Media.

Eder, G., 2019, "Truth, paradox, and the procedural conception of Fregean sense", pages 153–168 in *Philosophy of Logic and Mathematics*, De Gruyter.

Field, H., "Saving the truth schema from paradox", 2002, *Journal of Philosophical Logic*, 31 (1): 1–27. DOI: 10.1023/A:1015063620612

Field, H., 2003, "A revenge-immune solution to the semantic paradoxes", *Journal of Philosophical Logic*, 32 (2): 139–177. DOI: 10.1023/A:1023027808400

Gupta, A., 1982, "Truth and paradox", *Journal of Philosophical Logic*, 11 (1): 1–60. DOI: 10.1007/BF00302338

Gupta, A., and N. Belnap, 1993, *The Revision Theory of Truth*, Mit Press. DOI: 10.7551/mitpress/5938.001.0001

Herzberger, H., 1982, "Naive semantics and the liar paradox", *The Journal of Philosophy*, 79 (9): 479–497. DOI: 10.2307/2026380

Horty, J., 2007, *Frege on Definitions: A Case Study of Semantic Content*, Oxford University Press.

Kleene, S. C., 1938, "On notation for ordinal numbers", *The Journal of Symbolic Logic*, 3 (4): 150–155. DOI: 10.2307/2267778

Kripke, S., 1976, "Outline of a theory of truth", *The Journal of Philosophy*, 72 (19): 690–716. DOI: 10.2307/2024634

Leitgeb, H., 2007, "What theories of truth should be like (but cannot be)", *Philosophy Compass*, 2 (2): 276–290.

McGee, V., 1985, "How truthlike can a predicate be? A negative result", *Journal of Philosophical Logic*, 14 (4): 399–410. DOI: 10.1007/BF00649483

Moschovakis, Y. N., 1994, "Sense and denotation as algorithm and value", *Lecture Notes in Logic*, 2: 210–249.

Moschovakis, Y. N., 2006, "A logical calculus of meaning and synonymy", *Linguistics and Philosophy*, 29 (1): 27–89. DOI: s10988-005-6920-7

Muskens, R., 2005, "Sense and the computation of reference", *Linguistics and Philosophy*, 28 (4): 473–504. DOI: 10.1007/s10988-004-7684-1

Prior, A. N., 1960, "The runabout inference-ticket", *Analysis*, 21 (2): 38–39. DOI: 10.1093/analys/21.2.38

Raclavský, J., 2014, "Explicating the notion of truth within transparent intensional logic", pages 167–177, Chapter 12, in *Recent Trends in Philosophical Logic*, Trends in Logic, Springer. DOI: 10.1007/978-3-319-06080-4_12

Ramsey, F. P., 2000, "General propositions and causality", pages 237–257 in R. B. Braithwaite (ed.), *The Foundations of Mathematics and other Logical Essays*, Routledge.

Russell, B., 1906, "Les paradoxes de la logique" *Revue de métaphysique et de morale*, 14 (5): 627–650.

Suppes, P., 1980, "Procedural semantics", pages 27–35 in *Language, Logic, and Philosophy: Proceedings of the 4th International Wittgenstein Symposium.*

Suppes, P., 1982, "Variable-free semantics with remarks on procedural extensions", in T. W. Simon and R. J. Scholes (eds.), *Language, Mind, and Mrain*, Psychology Press.

Tichý, P., 1969, "Intension in terms of Turing machines", *Studia Logica*, 24 (1): 7–25. DOI: 10.1007/BF02134290

Tichý, P., 1971, "An approach to intensional analysis", *Noûs*, 5 (3): 273–297. DOI: 10.2307/2214668

Tichý, P., 2004, *Pavel Tichý's Collected Papers in Logic and Philosophy*, Otago University Press.

Yablo, S., 1985, "Truth and reflection", *Journal of Philosophical Logic*, 14 (3): 297–349. DOI: 10.1007/BF00249368

Yablo, S., 1993, "Paradox without self-reference", *Analysis*, 53 (4): 251–252. DOI: 10.1093/analys/53.4.251

ELBERT J. BOOIJ
ILLC
University of Amsterdam
Amsterdam, Netherlands
E.J.Booij@uva.nl
https://orcid.org/0000-0001-7632-8094