Self-reference and Saul Kripke's 'Outline of a Theory of Truth'

Damian Drexel
12014594

July 2022

705501 Paradoxes
Summer Semester 2022

1 Introduction

In this essay we want to revisit self-reference, following it's path to computer science to show it's importance and summarize the work of Saul Kripke in [Kri76]. As stated by Kripke "... any treatment of the concept of truth must somehow circumvent ..." the self-reference paradox [Kri76]. Commonly Epimenides paradox is brought up as an example in literature. According to this the Cretan philosopher Epimenides stated "All Cretans are liars". Since Epimenides is a Cretan himself, this sentence includes him. If the sentence is true, Epimenides lies and therefore the sentence cannot be true. If the sentence is false, Epimenides is telling the truth, which leads to a paradox. Another example for this paradox is Russell's paradox, which is presented in [Rus03].

Let R be the set which contains all sets that do not contain themselves.

$$R = \{x | x \notin x\} \tag{1}$$

We can then ask if R contains itself. If R is not part of itself ($R \notin R$) it conforms to the definitions of sets that are included in R ($R \in R$). This on the other hand implies that R is part of itself ($R \in R$) which excludes it from the sets contained in R ($R \notin R$), leading to a contradiction.

Kripke himself gives another example for the problem at hand. A sentence such as the following, which Kripke attributes to Jones, has no intrinsically wrong aspects to it:

> Most (i.e., a majority) of Nixon's assertions about Watergate are false.

We can inspect each sentence by Nixon about Watergate and see if the above sentence is true or false. On the other hand, if we assume all assertions by Nixon are evenly true and false and introduce the following assertion by him,

> Everything Jones says about Watergate is true.

we encounter a problem. Since we assume Nixon's assertions are evenly true and false, if the second quote is true it contradicts with the first quote and if the first quote is true it contradicts with our assumption of the even distribution of truth and falsity in Nixon's assertions.

We will come back to this example when we further discuss [Kri76]. As stated on [sel] self-reference is not only related to Gödels first incompleteness theorem but, furthermore, the halting problem and it's undecidability. First we have to introduce the term Turing machine. A Truing machine is a construct, introduced by Alan Turing for the halting problem, that is able to execute any computation [tur], therefore, also all programs running on computers may be seen as Turing machine. For such Turing machines we are looking for another Turing machine, that can determine if they halt [sel]:

$$H(\langle T \rangle, x) = \begin{cases} "yes", & \text{if Turing machine } \langle T \rangle \text{ halts for input } x \\ "no" & \text{else} \end{cases} \tag{2}$$

With $\langle T \rangle$ being a Gödel coded representation of the Turing machine T to test. To prove that no Turing machine exists that solves the halting problem, we can, in reference to the proof in [sel], do as follows:

A Turing machine T is called heterological if it does not halt for input $\langle T \rangle$

We now define a Turing machine H':

- Input: Gödel code of Turing machine A
- Runs H with input $(\langle A \rangle, \langle A \rangle)$
- Halts if A is heterological
- Loops forever if A is not heterological

If H' is heterological this means that it does not halt for input $\langle H' \rangle$. Therefore, H returns "no" and H' halts. By definition, H' is not heterological if it halts for input $\langle H' \rangle$ and therefore it contradicts the initial statement that H' is heterological.

Up until now we revisited the self-reference paradox with various examples and showed it's importance for the field of computer science. Nevertheless, as initially mentioned, Kripke stated that if one is interested in the concept of truth in any form, the problems posed by the topic of self-reference and the self-reference paradox have to be circumvented [Kri76]. Therefore, the following part of this essay is going to summarize *Outline of a Theory of Truth* [Kri76].

2 Summary 'Outline of a Theory of Truth'

In the previous discussion we introduced the example of Nixon and Jones. This example shows that such sentences do not posses an intrinsic property that allows us to determine whether or not it will result in a paradox. Furthermore, the opposite is true as well, as there is no property that can assure a sentence to be unparadoxical.

To tackle this challenge and develop an appropriate theory Kripke states two core ideas:

1. Sentences must be risky, i.e., there is the possibility for them to be "paradoxical if the empirical facts are extremely ... unfavorable"
2. No syntactic or semantic property allows the distinction between sentences that have the potential to be paradoxical and unparadoxical

$A \vee B$		B		
		true	**false**	**undefined**
	true	true	true	true
A	**false**	true	false	undefined
	undefined	true	undefined	undefined

Table 1: Logical or

$A \wedge B$		B		
		true	**false**	**undefined**
	true	true	false	undefined
A	**false**	false	false	false
	undefined	undefined	false	undefined

Table 2: Logical and

Another property of sentences, that is not intrinsic to them but ”depends on the empirical facts” is groundedness. Assume a class C of sentences. To determine if a sentence that asserts the truth value of sentences in this class are true, the truth value of the sentences in this class C has to be asserted. Furthermore, the truth value of a sentence that mentions truth can only be asserted by asserting other sentences. A sentence is grounded *iff* at the end of cascading through sentences to assert the truth value we end with a sentence that does not mention truth. If that is not the case the sentence is called ungrounded.

Kripke goes on to state problems with the hierarchy proposed by Tarski. This hierarchy is built by introducing a language L_0 that does not contain its own truth predicate. Therefore, another language L_1 is introduced that contains a truth predicate T_1 for language L_0. If we follow along with this approach we arrive at a set of languages $\{L_0, L_1, L_2, ...\}$, where each language L_i defines a truth predicate T_i for the language L_j with $j < i$. With this a sentence in language L_j cannot make a statement about its own truth. Kripke argues that this implies an implicit assignment of level to any sentence mentioning truth, which is hardly possible in practise. Furthermore, he argues that the hierarchy is only defined for finite levels, not for languages with transfinite levels. Because of the problems he identifies for hierarchies that assign fixed levels to sentences, Kripke proposes another model, which, too, is using levels.

The proposed model is meant to

1. provide ”... an area rich in formal structure and mathematical properties”
2. with the ”... properties [capturing] important intuitions”

The model is to allow truth-value gaps, meaning, though a sentence may always be meaningful it does not have to always be true or false. More formally assume a nonempty domain D. Let $P(x)$ be a monadic predicate and (S_1, S_2) a pair by which $P(X)$ is interpreted. With S_1 being the extension of $P(x)$, meaning $S_1 = \{x \in D | P(x) = true\}$ and $S_2 = \{x \in D | P(x) = false\}$. Therefore, S_1 and S_2 are disjoint ($S_1 \cap S_2 = \emptyset$). For all elements of D that are not in S_1 or in S_2 $P(x)$ is undefined. To handle this, Kleene's strong three-valued logic is proposed. In this logic if x is true $\neg x$ is false, if x is false $\neg x$ is true, and in any other case x is undefined. The table 1 shows the logical disjunction and table 2 the logical conjunction in this logic.

A language L, with n-ary predicates that are interpreted over relations on a Domain D, is introduced and extended by a monadic predicate $T(x)$. This predicate is only partially defined and interpreted by a set (S_1, S_2), with, as before, S_1 being the extension of $T_(x)$,

S_2 being the antiextension, and $T(x)$ being undefined for anything that is not an element of this set. The interpretation of (S_1, S_2) is denoted as $L(S_1, S_2)$. Let all true sentences of this interpretation be S_1' and let all sentences from D that are not part of this interpretation or false sentences be S_2'. A pair (S_1, S_2) with $S_1 = S_1'$ and $S_2 = S_2'$, which is the case if $T(x)$ is true for L containing $T(x)$, is called fixed point.

Kripke goes on to create such a fixed point, for which a hierarchy analogous to the one by Tarski is created. The hierarchy starts at a language L_0 for which $T(x)$ is undefined. The hierarchy is then constructed by defining each language $L_{\alpha+1}$ with the interpretation of $T(x)$ in language L_α. This implies if $T(x)$ is defined (true or false) in L_α it is also defined in $L_{\alpha+1}$. Furthermore, he deduces "... that for each α, the interpretation of T(x) in $L_{\alpha+1}$ extends the interpretation of $T(x)$ in L_α". So as α increases also the extension and antiextension of $T(x)$ extend and as soon as a sentence is defined its truth value cannot change.

In the discussion it is further shown that this is not only true for finite levels but also for transfinite, in contrast to the critique on Tarski's hierarchy, given before. Furthermore, it is shown that it is possible to extend any language to contain its own truth predicate.

To return to the terms of grounded and ungrounded statements the following definition is given:

> Let A be a sentence of L. If A has a truth value in the smallest fixed point L_α it is called grounded, otherwise it is called ungrounded.

With a smallest fixed point being a fixed point every fixed point extends. In this formalisation liar sentences or other difficult sentences do not pose a problem since they can be shown to be ungrounded. Nevertheless, a sentence which is ungrounded does not have to be paradoxical. But, a possible definition for paradoxical sentences is given:

> "a sentence is paradoxical if it has no truth value in any fixed point."

This shows how this model is circumventing the liar paradox. For this the allowed truth-value gap is used. The sentence can neither be true or false and therefore, in this model, the sentence is undefined.

The language presented in [Kri76] can define its own predicates, but is still no "universal language". Two problems prevent it from being one:

1. The definition of the minimal fixed point, used for the definition of groundedness, is not defined in the object language.

2. Some statements that can be derived from the object language cannot be expressed with the object language. For example, a paradoxical sentence in the object language is not true, but we cannot say it is not true in the object language.

This section summarized the work described in [Kri76]. The goal of the work was to find a model to circumvent the problems that arise with self-reference paradoxes. This was achieved by introducing a hierarchy analogous to the one proposed by Tarski and building on the concept of the truth-value gap. In the end this lead to a model in which paradoxical sentences, like liar sentences, do not pose a problem, since they are neither true nor false but undefined in this model. Still, this model does not offer a universal language.

Although this method poses other problems, it was the foundation and inspiration for various other works following it and stands as a influential work [sel].

3 Conclusion

In the beginning of this essay we discussed examples of the self-reference paradox, for example, the liar paradox of Epimenides. Without going too much into the details of Gödels first incompleteness theorem we went on to the halting problem. We looked into Turing machines and the halting problem, since it not only utilizes Gödels work but also employs self-reference to prove that there is no Turing machine that can solve the halting problem. The halting problem is seen as fundamental to computer science, even though it preceded the emergence of this field. Therefore, self-referencing paradoxes are not just a philosophical or mathematical problem, but are of importance for a field we all encounter in our day-to-day lifes.

The summary of Saul Kripke's *Outline of a Theory of Truth* [Kri76] shows one attempt to circumvent these kinds of paradoxes. We roughly sketched out the work of Kripke and hinted on the importance of it.

As closing remarks we want to return to the importance of this topic to the field of computer science. The halting problem, fundamental to this field, is a major reason for people interested in this area to discuss the topic of self-reference paradoxes. In addition, we argue that as computer scientists we encounter self-references and related problems in our daily work, though, maybe in some other forms than described before.

For example, a topic introduced to every computer science student is that of self-referencing code, or recursion. Recursion is applied to solve various problems. Recursion requires a sound implementation of the code that refers to itself in order to function correctly and not running into problems.

Furthermore, various fields in computer science not only build on logic and therefore on (the solution for) its problems and challenges. The previous example of Turing machines and the halting problem only being one of the topics of theoretical computer sciences that are related to logic. Another area of logic that found its way into computer science being, for example, first-order logic, which heavily influenced knowledge representation. As with other fields, knowledge representation tries to model the truth about given domains and, as already cited in the beginning of this essay, "... any treatment of the concept of truth must somehow circumvent this paradox" [Kri76].

References

[Kri76] Saul Kripke. Outline of a theory of truth. *The journal of philosophy*, 72(19):690–716, 1976.

[Rus03] Bertrand Russell. *The principles of mathematics.* Cambridge University Press, 1903.

[sel] Self-Reference (Stanford Encyclopedia of Philosophy). `https://plato.stanford.edu/entries/self-reference/`. Accessed: 2022-07-01.

[tur] Turing Machines (Stanford Encyclopedia of Philosophy. `https://plato.stanford.edu/entries/turing-machine/`. Accessed: 2022-07-01.