Two Concepts of Universal Logic

§1 *Two Ways to be 'universal'*

Universal Logic approaches have to face the question how their elucidation of universal logic relates to universal computability. Universal logic – it seems – tries to capture all logical reasoning. Universal computation captures all computable algorithms.

So, should they coincide?

The first – and short – answer is that the question might rest on an ambiguity in 'universal': universal computation is universal in the sense of comprehensively capturing all intuitively algorithmic ('computable') procedures; universal logic is universal in the sense of being applicable in all contexts, where not every context has to involve the full force of classical logic.

So, even if FOL is tied to universal computation that does not imply that universal logic is tied to FOL.

So, universal logic (in the sense of a paradigm system) and universal computation (in the sense of a paradigm model) need not coincide.

The second – and longer – answer looks closer at the ideas behind the *Church-Turing Thesis*, a special argument from Gödelian reasoning for paraconsistent universality, a crucial theorem by Turing, and how all this does not show that the strong Universal Logic programme is in conflict with the *Church-Turing Thesis*.

§2 *The Church-Turing Thesis*

The *Church-Turing Thesis* (CTT) can be expressed in different ways, for example:

(CTT') Everything that is computable is recursive(ly computable).

(CTT'') Everything that is computable is computable by a Turing machine.

The first expression (by Church) refers to the recursive *functions*, the second (by Turing) to Turing *machines*. The *Church-Turing Thesis* is generally seen as outlining an upper limit on computability. Nothing seems to be computable that is not Turing computable. Seen from a naive point of view – the naiveté of which will be explained shortly – the existence of super-computability would falsify the *Church-Turing Thesis*. But (CTT) involves a precise and

definite notion of an effective procedure (an *algorithm*) and a corresponding concept of computability:

(CT) identifies the *intuitive* notion of computability with a *formally explicated* notion (being computable by a TM).

The cornerstone of this is the idea of an *algorithm M* on *discrete* symbols ω:

(i) executing steps each of which is mindless,

(ii) where each computation of $M(\omega)$ ends after finitely many steps (if $M(\omega)$ is defined at all),

(iii) *M* is implementable by different devices.

To avoid misunderstandings CTT therefore can be expressed properly as:

(Church-Turing Thesis)

Everything that is intuitively computable is computable by a DTM.

CTT is not talking about any computability that could be nor about any type of machine that could be – keep that in mind!

There are at least three aspects of (CTT):

(i) *reconstructive*: DTM is said to characterise the *intuitive* notion of computability,

(ii) *conceptual*: by proving equivalences between formal models of algorithms we grasp an *absolute* concept of computation,

(iii) *epistemological*: (CTT) outlines (some) structures and limits of the *mind* (being a bridge to meta-logical results).

We are concerned here mostly with the first aspect, sometimes with the third. Both are related to the *cognitive sciences*. Given (CTT), the arithmetization of formal languages and

Montague's Thesis

Natural language are equivalent to some formal languages.

we get a strong version of the *computational theory of mind*:

(Strong CTM)

Anything that could be computed or said can be computed or generated by a deterministic Turing machine.

(CTT) is phrased in terms of the standard notions of computability (i.e., recursive functions or Turing machines). These define steps of computation which are discrete and effective (e.g. writing a symbol, moving one to the left). But could there not be a different concept of computation altogether? Asking for *the Turing Limit* means asking whether there are other concepts of computability and whether a machine corresponding to such a concept of

computability might compute functions which cannot be computed by a deterministic TM. If there is such a machine, it is *beyond* the Turing limit!

We can ask at least three questions now:

(1) What other types of computability are there, if there are any.

(2) Does the existence of this type of computability put (CT) in doubt or even falsify it?

(3) Can there *really be* such machines?

The last question asks whether some type of notional machine could ever be built. David Deutsch introduced a physical interpretation of the (CTT):

(DCT)

No *realisable physical device* can compute functions that are not DTM-computable.

We should keep in mind:

(i) (CTT) is falsified only if some of the functions which are computed by a new type of computation are *intuitively* computable. That something can be *expressed finitely* does not make it *effective*! (E.g., "Divide this by the largest prime.") Any machine requiring non-finite input does not compute effectively either.

(ii) If, for instance, analog machines are not machines in the sense of (CTT), we could try to express an anlog to (CTT) for this kind of device.

(iii) If analog machines are not machines in the sense of (CT) we have to reconsider mechanism only if we consider what they compute to be part of the mind (in distinction to the brain).

Even if the brain was of a non-standard computability type and further on that type was beyond the Turing Limit, this would *not* falsify (CTT) or narrow mechanism, since the mind need not be the brain. Even the 'mind/brain' is not just the brain, but the brain only in as much it is involved in cognitive computations. Maybe the brain is an analog device that *implements on some level* a TM. If the analog processes are not *cognitively penetrable* they need not belong to a level of the mind that concerns cognition in the intuitive sense.

There are machine models that are beyond the power of the (universal) Turing machine, e.g. *Coupled Turing Machines* or Copeland's *Accumulator Machines*, but they do not have finite input, so *are beyond* what is intuitively computable. These machines are notional machines.

§3 *Turing's Theorem*

In his classical paper on computing machines and the 'Entscheidungsproblem' Turing constructively proves two sides of an equivalence

(i) If a TM M accepts some input α, there is a FOL axiomatized theory T of the machine table of M such that $\vdash_T \alpha$ (for all und only the accepted input α).

(ii) If there is a FOL axiomatized theory T such that $\vdash_T \alpha$, there is some TM M which accepts α (for all und only the derivable theorems α).

The first part means in the light of CTT

(i) Everything that is intuitively computable can be captured by a derivation in FOL theory.

The second part means in the light of CTT

(ii) Every derivation of a FOL theory is intuitively computable.

So, TMs and FOL (theories) are computationally equivalent.

This is the backbone of the idea and claim that everything logical (in the narrow sense of being algorithmically computable) can be captured by FOL. In that sense FOL is the universal logic.

Another thesis, *Hilbert's Thesis*, therefore claims that any cogent reasoning anywhere in mathematics can be given a FOL rendering.

So, FOL *can* be used universally – but should it?

§4 *Gödel Sentences as Paradoxes for Universalists*

Gödel's famous *Incompleteness Theorems* show the presence of gaps of negation incompleteness. [A formal system S being *negation incomplete* in case neither $\vdash_S A$ nor $\vdash_S(\neg A)$ for some sentence A.]

Graham Priest uses the Gödel sentence as another argument for dialetheism: supposing (CTT) the informal reasoning that the Gödel sentence G is true can be captured by a formal system N for naïve proofs. Sentence G for some logical system expressive enough says of itself that it is not provable in that very system. Sentence G for the system N *is* provable in N (as exhibited by the informal argument usually given for its truth), and thus negation incompleteness is avoided, at the cost of inconsistency, as system N meets the conditions not only to express and represent the concept of provability, but to express – and even represent – the concept of truth as well. Dialetheism accepting inconsistency in any case (dealing thus with the semantic antinomies) can reap the benefit of avoiding negation incompleteness. For Priest, thus, Gödel's argument shows a dilemma: either we achieve negation completeness or

consistency. And, as we can reason to the truth of G, achieving negation completeness has to be our priority.

But isn't the contradiction engendered by G too much even for the dialetheist? Having G means having a provable sentence claiming truly its own non-provability, and because G is provable ¬G is true, thus it should be provable as well, as we can informally reason to its truth.

As we want the paraconsistent equivalent to soundness N should not prove sentences that are false only. [Paraconsistent system may prove sentences which are true and false at the same time, but proving a sentence that is only false (i.e. not true at the same time) would be a hypercontradiction putting the system used into doubt.]

Consider the following argument:

> G should not be false only, as it should then be provable given convention (T) for truth
>
> (T) True("p") ≡ p
>
> from right to left (so-called 'T-in') and contraposition. If G is at least true, it is not provable, given (T) from left to right (so-called 'T-out'). As we can informally reason to G's truth, this is reasoning is captured by a proof in N, N thus proving G. So, we have
>
> (1) $\vdash_N (\exists x)\mathrm{Proof}(x,G)$
>
> By paraconsistent soundness and T-out we thus get
>
> (2) $\vdash_N \neg(\exists x)\mathrm{Proof}(x,G)$
>
> This means that there is some number *m* which both codes a proof of G, by (1), and does not, by (2)! This is too strange as 'Proof' is a primitive recursive, algorithmic and deterministic relation. So, could we ever have (1) and (2) together?
>
> End of argument.

Without mystery we cannot give up (CTT), and the existence of N. Representing the presence of a proof by 'Proof' should be available.

One measure might be a paraconsistent account of algorithms: one and the same deterministic algorithm providing different answers or an inconsistent answer on the same input. This is a *desideratum*, however. And it seems to clash with our intuitive notion of algorithm.

It is no option either to stress realism again (claiming provability to be different from truth) as G is available and provably *true* in N.

One may claim that *m* makes (1) true by really coding the proof of G, and (2) is thereby another provable sentence of N resulting in saying of *m* falsely that it does not code the proof of G. Thus, *m* turns out to be an inconsistent object in N. And (2) is another contradiction, its truth being guaranteed by the provability of G. As there are other inconsistent objects in N already, this may not worry a dialetheist. The dialetheist may worry as coding a proof is a primitive recursive relation. So that saying of *m* that it does not code a proof of G contradicts the result of a respective algorithm. The algorithm, however, will in no way be suppressed in its workings: it delivers that *m* codes a proof of G, making (1) true. (2) just falsely contradicts this positive output. It cannot interrupt the algorithm which properly establishes (1).
A dialetheist can, therefore, embrace the truth of G, and so avoid Gödelian gaps of negation incompleteness.

§5 *Paradoxes, Gödel Sentences, Turing's Theorem and Universal Reasoning*

In the reasoning concerning the Liar

(λ) λ is not true.

we make use of intuitive principles of semantic self-ascription, the intuitive *Convention T* (Tarski's scheme for "true") and FOL reasoning to derive at

$\lambda \equiv \neg \lambda$ and $\lambda \wedge \neg \lambda$

Intuitively valid reasoning leads to the contradiction. By CTT it can be completely rendered in FOL.

Question: But, FOL – and computationally equivalent systems like Lambda Calculus – are consistent, so how can this be? Is antinomic/paradoxical reasoning not formalizable?

Answer: These systems are consistent in their bare form, not if one adds axioms which are inconsistent (like those leading to λ). The Liar reasoning can be formalized in FOL, but is, of course, explosive in FOL (leading to a trivial, all sentences encompassing theory).

Question: Can paraconsistent universalist reasoning be formalized in FOL?

Answer: It is possible to have a FOL (classical) meta-theory for a universal logic – but the aspiration of a truly universal logic, of course, must be to be able to express its own meta-theory. By *Tarski's Theorem* a FOL theory cannot contain its own semantics, whereas some paraconsistent logics can formalize semantic closure.

Question: What about CTT in the wider sense employed in the reflection on reasoning about Gödel sentences?

Answer: What is intuitively computable is computable by some formal system S. S need not be *full* FOL to justify the idea behind the reflection on closure given the Gödel sentences (and taking them to be provable contradictions in that universally closed system S). So, system S will be computable, and the paradoxical arguments do not contradict CTT.

Manuel Bremer, 2020