Revision Sequences and Computers with an Infinite Amount of Time

BENEDIKT LÖWE, *Rheinische Friedrich–Wilhelms–Universität Bonn, Mathematisches Institut, Beringstraße 6, D — 53115 Bonn, Germany. E-mail: loewe@math.uni-bonn.de*

Abstract

The author establishes a connection between Revision Theory of Truth and Infinite Time Turing Machines as developed by Hamkins and Kidder.

The ideas from this paper have incited Welch to solve the limit rule problem of revision theory.

Keywords: Revision theory of truth, infinite time Turing machines, definability.

1 Introduction

First-order definitions and even inductive definitions using the second-order induction scheme are neither sufficient for scientific discourse in general nor for the technical applications using computers to check properties which seem to be simple from a heuristic standpoint.[1] Many natural-language notions use some degree of circularity or impredicativity in their definitions. One of these notions is the notion of truth, if you consider TRUE to be a predicate of sentences (i.e., natural numbers via Gödelization): If your language contains a truth predicate you will necessarily run into sentences asserting truth of sentences, so to determine their truth values you have to already know the extension of the truth predicate.

Consequently, building on the theory of inductive definitions and the work of Martin, Woodruff and Kripke,[2] Herzberger developed a theory called 'Naive Semantics' that went beyond inductive definitions.[3] This theory was further refined by Gupta and Belnap in [7], [3], and [8], where they call the system 'Revision Theory of Truth' and develop their semantic systems $\mathbf{S}^*$ and $\mathbf{S}^\#$ that allow an analysis of definitions that would be logically illicit in classical definability theory. In fact, the aim of this enterprise is even more aspiring:

> 'The key to the proper resolution of the problem of truth and paradox lies, in our view, in the theory of definitions. [8, p. 113].'

Their focus on the question of truth leads to a certain disrespect towards the definability aspect of their work, although the definability aspect is a natural feature of the theory (being a generalization of inductive definability with its extensive literature on definability and complexity questions). In this paper we shall stress the question 'Which sets of natural numbers can be defined using a Gupta–Belnap definition?' (as opposed to the question 'Which sentences are true in a Gupta–Belnap semantic system?').

[1] See, e.g. the Ehrenfeucht–Fraïssé proof that the notion of connectedness for finite graphs is not first-order definable, *cf.* [6, §1].

[2] [16] and [15].

[3] [11] and [12], especially [11, p. 485 and 489] where the reader can find pictures of what was to become the Revision Sequences of [8] and Definition 2.3.

Roughly speaking (we shall make this more precise in subsection 2.2) a property P of natural numbers is said to be **revision theoretically definable** if there is a formula Φ (possibly containing a symbol for P) such that a natural number n has the property P if and only if n is in every reasonable approximation for the extension of Φ, where 'reasonable' means that the approximation h reoccurs in a transfinite sequence of iterated applications of Φ starting from h.

We will show that the Gupta–Belnap revision sequences are deeply connected with the Infinite Time Turing Machines defined by Hamkins and Kidder.[4] This fact is intuitively plausible: with Turing machines consisting of a finite program (corresponding to the formula Φ in the description of revision theoretic definability) being applied iteratively, their generalization to transfinite computation sequences of ordinal length looks just like the 'transfinite sequence of iterated applications of Φ' in Revision Theory. We will add some substance to this intuition in Section 4.

This connection and the fact that there is a nice structure theory of Infinite Time Turing Computability give rise to possible applications:

The profound analysis of the lengths of Infinite Time Turing Computation gives hopes that the lengths of (the relevant part of) revision sequences can also be understood.

Even more so, Welch's theorem that Infinite Time Turing machine architecture is somewhat robust with respect to changing the limit rule (Theorem 3.7) seems to point into the direction of the limit rule problem of Revision Theory: there has been a discussion[5] whether changing the limit rule for revision sequences might change the semantic content of the definability notions in a relevant manner.

When sections 1 to 5 of this paper were prepared (Spring 1999), the author already suspected that a refinement of the methods of Section 4 might lead to a solution of the limit rule problem. When the paper was presented as a talk at the workshop "Nicht-klassische Formen der Logik" at the XVIII Deutscher Kongreßfür Philosophie in Konstanz (October 1999), the problem had been solved by Philip Welch. The newest results are briefly mentioned in Section 6.

For the paper to be understandable to readers from both Revision Theory and Computability Theory we try to give a rather long introduction to both areas omitting almost all proofs.

The main part of the paper is the construction of the Turing machine $M_{\Phi,\alpha}$ in Section 4 in which we tried to suppress most of the tedious details of storing and copying information.

2 The Gupta–Belnap systems

2.1 Revision sequences

The Revision Theory of Truth as laid out in [8] is very general as it allows definitions of the extensions of arbitrary predicates over arbitrary models (of arbitrary languages), and uses arbitrary many-valued logics. To simplify notation (and to stress the similarity to Turing machines which are normally used to compute subsets of the natural numbers), we shall restrict ourselves to the most basic case throughout this paper; the ground model will be $\mathbb{N}$, the set of natural numbers, our predicate $\dot{x}$ will be unary, and the logic will be classical two-valued logic. Thus the extension we want to define will be a subset of $\mathbb{N}$. As usual in set

[4]Cf. [9].

[5]Cf. [4] and [22].

theory, we call these objects 'real numbers', although they are not exactly real numbers in the sense of real analysis (but rather elements of the Cantor space).

We fix a base language $\mathcal{L}$, and let $\mathcal{L}^*$ be the language $\mathcal{L}$ augmented by the additional unary predicate symbol $\dot{x}$ whose extension we want to define.

Gupta–Belnap definitions can be seen as a generalization of inductive definitions; in an inductive definition[6] we construct a monotone operator $M : \mathbb{R} \to \mathbb{R}$ derived from a formula in the augmented language $\mathcal{L}^*$ and define the extension of $\dot{x}$ to be the least fixed point of M (starting with the empty extension) which must exist since M was monotone.

In the Gupta–Belnap approach we discard the assumption of monotonicity and have to replace it by a closer analysis of the starting extensions.

As in the inductive case, let $\Phi(v_0)$ be a formula of $\mathcal{L}^*$. Then we shall say that a function $\delta_\Phi : \mathbb{R} \to \mathbb{R}$ is the **revision operation determined by** Φ if

$$k \in \delta_\Phi(z) \iff \langle \mathbb{N}, z \rangle \models \Phi[k]$$

for all real numbers z. (Again, note that elements of $\mathbb{R}$ are identified with subsets of $\mathbb{N}$.)

For revision operations $\delta : \mathbb{R} \to \mathbb{R}$ we use the obvious notation for the iterated application of δ. In the following we will consider sequences of real numbers $\vec{s} = \langle s_\alpha ; \alpha \in \eta \rangle$ where η is either a limit ordinal or the class of all ordinals.[7] These sequences are interpreted as approximations to our final interpretation of the predicate $\dot{x}$.

DEFINITION 2.1
Let $\vec{s}$ be a sequence of real numbers of length η (η might be the class of all ordinals), and let d be a natural number. We shall say that '$d \in \dot{x}$' **is $\vec{s}$-stably true** if there is a β such that for all $\alpha \geq \beta$ we have $d \in s_\alpha$.
Likewise, we shall say that '$d \in \dot{x}$' **is $\vec{s}$-stably false** if there is a β such that for all $\alpha \geq \beta$ we have $d \notin s_\alpha$.

DEFINITION 2.2
Let $\vec{s}$ be a sequence of reals. Then a real h is said to be **$\vec{s}$-coherent** (in symbols: $\mathrm{Coh}(h, \vec{s})$) if for all natural numbers d the following hold:

1. If '$d \in \dot{x}$' is $\vec{s}$-stably true, then $d \in h$.
2. If '$d \in \dot{x}$' is $\vec{s}$-stably false, then $d \notin h$.

DEFINITION 2.3
Let $\vec{s}$ be a sequence of reals and Φ a formula of $\mathcal{L}^*$. We shall call $\vec{s}$ a **Φ-revision sequence** if

1. For all ordinals α, we have $s_{\alpha+1} = \delta_\Phi(s_\alpha)$.
2. For every limit ordinal λ, we have that s_λ is $\vec{s}{\restriction}\lambda$-coherent.

In light of the question of the limit rule (discussed later), a revision sequence of the defined kind might be called an **unrestricted revision sequence**. To prepare for the limit rule problem, we introduce restricted revision sequences.

Following [4, p. 400], we shall call a function γ assigning to a sequence of reals $\vec{s}$ of limit length λ a real $\gamma(\vec{s})$ that is $\vec{s}$-coherent a **bootstrapping policy**. If Γ is a class of bootstrapping policies, then we shall say that $\vec{s}$ is a **$\langle \Phi, \Gamma \rangle$-revision sequence** if it's a Φ-revision sequence and there is a $\gamma \in \Gamma$ such that for each limit ordinal λ we have $s_\lambda = \gamma(\vec{s}{\restriction}\lambda)$.

[6] Cf. [17].

[7] Of course, revision sequences of successor length can be easily extended to revision sequences of limit length, so these are the only ones we have to think about.

The most interesting cases are the class Γ_∞ of all bootstrapping policies (being a Φ-revision sequence and being a $\langle\Phi,\Gamma_\infty\rangle$-revision sequence are the same), and the one-element classes where we have one particular limit rule applied in all limit stages.

At this point we single out the class $\Gamma_0 := \{\gamma_0\}$ where γ_0 is the liminf rule (i.e., exactly the stably true formulae hold in the limit stage). The limit rule γ_0 is the minimal bootstrapping policy in the sense that all others let more formulae be true in the limit. Γ_0 will play a decisive rôle in the construction of Section 4.[8]

DEFINITION 2.4
Let Φ be a formula of $\mathcal{L}^*$.

1. We shall call a real h $\langle\Phi,\Gamma\rangle$-**recurring** if h occurs cofinally often in some $\langle\Phi,\Gamma\rangle$-revision sequence $\vec{s}$ of length Ord. The set of $\langle\Phi,\Gamma\rangle$-recurring reals will be denoted by $\mathrm{Rec}_\Gamma(\Phi)$.
2. A real h is called $\langle\Phi,\Gamma\rangle$-**reflexive** if there is some ordinal $\alpha > 0$ and some $\langle\Phi,\Gamma\rangle$-revision sequence $\vec{s}$ such that $s_0 = s_\alpha = h$. We shall call the least such α the $\langle\Phi,\Gamma\rangle$-**reflexivity** of h, in symbols $\mathrm{refl}_{\Phi,\Gamma}(h)$.

In the following we state a couple of simple consequences of the basic definitions. Proofs of these easy facts can be found in [8]:

PROPOSITION 2.5
Let $\vec{s}$ be a sequence of real numbers. Then the following are equivalent:

1. '$d \in \dot{x}$' is $\vec{s}$-stably true,
2. for all h occurring cofinally often in $\vec{s}$ we have $d \in h$.

PROPOSITION 2.6
For any real h the following are equivalent:

1. h is $\langle\Phi,\Gamma_\infty\rangle$-recurring,
2. h is $\langle\Phi,\Gamma_\infty\rangle$-reflexive.

The use of the Belnap rule Γ_∞ in Proposition 2.6 is important. The conclusion is provably false for some other rules, including the Herzberger rule Γ_0.[9]

PROPOSITION 2.7
If h is $\langle\Phi,\Gamma\rangle$-reflexive then $\mathrm{refl}_{\Phi,\Gamma}(h) < \omega_1$.

2.2 *The systems* $\mathbf{S}^\#$ *and* $\mathbf{S}^*$

Now we define the semantic relation for the Gupta–Belnap systems $\mathbf{S}^\#$ and $\mathbf{S}^*$:

$$\mathbb{N} \models^{\mathbf{S}^\#}_{\Phi,\Gamma} \varphi \iff \forall h \in \mathrm{Rec}_\Gamma(\Phi) \exists n \forall p \geq n(\langle \mathbb{N}, \delta^p_\Phi(h)\rangle \models \varphi),$$

and

$$\mathbb{N} \models^{\mathbf{S}^*}_{\Phi,\Gamma} \varphi \iff \forall h \in \mathrm{Rec}_\Gamma(\Phi)(\langle \mathbb{N}, h\rangle \models \varphi).$$

[8] The rule Γ_∞ was the original rule used in [8], and it is called the **Belnap rule**. Herzberger in [11] used the rule Γ_0 whence we call it the **Herzberger rule**. In [11, p. 487], Herzberger compares his $\liminf$ rule to Kripke's $\limsup$ rule of inductive definitions from [15].

[9] [8, p. 175].

We shall say that a real number z is $\mathbf{S}^{\#}_{\Gamma}$-definable ($\mathbf{S}^{*}_{\Gamma}$-definable) if there is a formula of $\mathcal{L}^*$ such that for all natural numbers n the following holds:

$$n \in z \iff \mathbb{N} \models^{\mathbf{S}^{\#}}_{\Phi,\Gamma} \text{`}n \in \dot{x}\text{'}$$

$$\left(n \in z \iff \mathbb{N} \models^{\mathbf{S}^{*}}_{\Phi,\Gamma} \text{`}n \in \dot{x}\text{'}\right).$$

In the following, we shall omit Γ in the notation if $\Gamma = \Gamma_\infty$.

Of course, if φ is a sentence not containing $\dot{x}$ (i.e., a sentence of $\mathcal{L}$), then $\mathbb{N} \models^{\mathbf{S}^{\#}}_{\Phi,\Gamma} \varphi$ if and only if $\mathbb{N} \models \varphi$, so $\{\varphi\,;\, \mathbb{N} \models^{\mathbf{S}^{\#}}_{\Phi,\Gamma} \varphi\}$ does contain the true sentences of arithmetic. Thus, by Gödel's Incompleteness Theorem it can't be first-order definable.

Kremer and Antonelli in [14], [1] and [2] progressed further along this line and showed that $\{\varphi\,;\, \mathbb{N} \models^{\mathbf{S}^{\#}}_{\Phi} \varphi\}$ is Π^1_2-complete.[10]

From our definability standpoint, the most interesting open question about the complexity of revision theory is: 'What reals are $\mathbf{S}^{\#}_{\Gamma}$- ($\mathbf{S}^{*}_{\Gamma}$-)definable?' More precisely, this innocuous question is a vast array of questions. Not only can we try to compute the set of $\mathbf{S}^{\#}_{\Gamma}$-definable reals for all sorts of Γ, but we can ask the question

> Under what circumstances (i.e., what conditions do we have to impose on Γ) is $\mathbf{S}^{\#}_{\Gamma}$-definability equivalent to $\mathbf{S}^{\#}$-definability?

This question keeps us very close to the limit rule question: there have been various proposals for the most natural choice of Γ, and it is still under discussion for which Γ the system $\mathbf{S}^{\#}_{\Gamma}$ is most natural.[11]

If we fix $\Gamma := \Gamma_\infty$, we can (just by counting quantifiers) make the following immediate observation:

PROPOSITION 2.8
The sets $\{\varphi\,;\, \mathbb{N} \models^{\mathbf{S}^{\#}}_{\Phi,\Gamma_\infty} \varphi\}$ and $\{\varphi\,;\, \mathbb{N} \models^{\mathbf{S}^{*}}_{\Phi,\Gamma_\infty} \varphi\}$ are Π^1_2.

PROOF. Using Proposition 2.7 and Proposition 2.6 we can transform the definition equivalently to the form

$$\begin{array}{lll}\forall h\big((\exists\alpha<\omega_1(\exists\vec{s}\in(\mathbb{R})^{\alpha+1} & (& \forall\beta<\alpha(s_{\beta+1}=\delta_\Phi(s_\beta)) \\ & \wedge & \forall\lambda<\alpha(\mathrm{Lim}(\lambda)\to\mathrm{Coh}(s_\lambda,\vec{s}\restriction\lambda)) \\ & \wedge & s_0=s_\alpha=h)) \qquad) \\ & \to & (\exists n\forall p\geq n(\langle\mathbb{N},\delta^p_\Phi(h)\rangle\models\varphi))\big)\end{array}$$

for $\mathbf{S}^{\#}$ and

$$\begin{array}{lll}\forall h\big((\exists\alpha<\omega_1(\exists\vec{s}\in(\mathbb{R})^{\alpha+1} & (& \forall\beta<\alpha(s_{\beta+1}=\delta_\Phi(s_\beta)) \\ & \wedge & \forall\lambda<\alpha(\mathrm{Lim}(\lambda)\to\mathrm{Coh}(s_\lambda,\vec{s}\restriction\lambda)) \\ & \wedge & s_0=s_\alpha=h)) \qquad) \\ & \to & \langle\mathbb{N},h\rangle\models\varphi\big)\end{array}$$

[10]For a weaker result with proof, cf. Proposition 2.8. The Kremer–Antonelli result is not only stronger but also much more general since they do not restrict their attention to the standard model $\mathbb{N}$. Furthermore, Kremer's completeness result extends to a broader class of revision theories he calls **plausible revision theories**, cf. [14, p.590*sqq*]. $\mathbb{N} \models^{\mathbf{S}^{\#}}_{\Phi,\Gamma}$ is a plausible revision theory for arbitary Γ, hence these relations constitute Π^1_2-complete sets.

[11]Cf. [4].

for $\mathbf{S}^*$.

To compute the complexity of these formulae, first note that the function δ_Φ doesn't add complexity since it is first-order definable (via Φ).

Now both formulae are of the form

$$\forall h \Big(\big(\exists \alpha < \omega_1 \exists \vec{s}\, \Psi(\alpha, \vec{s}, h) \big) \rightarrow \Psi^*(h) \Big),$$

where Ψ is arithmetic in its arguments (already assuming that ordinal quantifiers bounded by α will be natural number quantifiers after coding $\langle \alpha, \in \rangle$ as a real). Note that the satisfaction relation in $\langle \mathbb{N}, h \rangle$ is $\Delta^1_1(h)$, so the Ψ^*-part doesn't add complexity either.

Now we code the countable ordinal by a real and receive

$$\forall h \Big(\big(\exists y (\mathrm{WO}(y) \wedge \exists \vec{s}\, \Psi(\alpha, \vec{s}, h)) \big) \rightarrow \Psi^*(h) \Big).$$

Thus the premiss of the implication is $\exists(\Pi^1_1 \wedge \Sigma^1_1)$ which is Σ^1_2, and hence the whole formula is Π^1_2. ■

This computation gives an upper bound for the complexity of any given $\mathbf{S}^\#$-definable real:

COROLLARY 2.9
Every $\mathbf{S}^\#_{\Gamma_\infty}$-definable real (and every $\mathbf{S}^*_{\Gamma_\infty}$-definable real) is a Π^1_2 real.

PROOF. By definition, a real x is $\mathbf{S}^\#_{\Gamma_\infty}$-definable if and only if there is a formula Φ of $\mathcal{L}^*$ such that for all n

$$n \in x \quad \Longleftrightarrow \quad \mathbb{N} \models^{\mathbf{S}^\#}_{\Phi, \Gamma_\infty} `n \in \dot{x}'.$$

But by Proposition 2.8, the right-hand side is Π^1_2. ■

What is to be said about lower bounds? Using an idea of Gupta, Kremer has proved that every inductively definable real is $\mathbf{S}^\#$-definable.[12] Since inductive definability over $\mathbb{N}$ and being Π^1_1 coincide,[13] we have a lower bound for the complexity of $\mathbf{S}^\#$-definability. It was unknown until very recently[14] whether there were more complicated $\mathbf{S}^\#$-definable sets (see Section 6).

3 Infinite time Turing machines

The notion of an Infinite Time Turing Machine is a natural generalization of the ordinary Turing machines that constitute our basic model of computation. Jeffrey Kidder and Joel Hamkins blended the fields of Recursion Theory (Computability Theory) and Set Theory by allowing their Turing machines to resume their computations after infinitely many steps and start at limit steps in a specified limit stage.[15]

Although probably technically of no importance,[16] the investigation of Infinite Time Turing Machines is motivated by certain philosophical thought-experiments using the Lorentz

[12] Cf. [14, p. 590*sqq.*].
[13] Cf. [17, p. 20 and p. 24*sq.*].
[14] Cf. [21].
[15] Cf. [9].
[16] "Thompson lamps, super π machines, and Platonist computers are playthings of philosophers; they are able to survive only in the hothouse atmosphere of philosophy journals. In the end, [Malament–Hogarth] spacetimes and the supertasks they underwrite may similarly prove to be recreational fictions for general relativists with nothing better to do. [5, p.40]"

contraction of spacetime to fit a countable sequence of time intervals into a finite time interval (according to the timeline of a different observer), or, more in the language of general relativity, to fit the entire future time cone of one observer into the past time cone of a second observer.

Readers interested in this basic background are referred to the introduction of [9], and the papers [5] and [13].

In the following, we shall assume that the reader is familiar with the basic theory of ordinary Turing machines.[17]

An Infinite Time Turing Machine works like an ordinary Turing machine—it has a finite program including a specified finite set of states, one of these states is a HALT state telling the program to stop, and in addition to that, it has an infinite tape for output.

The content of the infinite tape of a Turing machine can be seen as a real number. We shall identify the tape with this real number at numerous points in the construction of Section 4. If we have an Infinite Turing Machine Computation of ordinal length α with a tape labeled x, and $\beta < \alpha$ then we shall denote the snapshot real at time β with x^β. We can (and shall) go even further: If we have an Infinite Time Turing Machine with a tape that is split into countably many infinite components $\langle x_i ; i \in I \rangle$, then we shall denote with x_i^β the real that constitutes the snapshot of tape x_i at the time β.

There is no difference between Infinite Time Turing machines and ordinary Turing machines in the part of the Infinite Time Turing machines described up to now. The only difference from ordinary Turing machines is the Infinite Time Turing machine's LIMIT state, and the fact that if the computation doesn't reach the HALT state at any given $\beta < \lambda$ for a limit ordinal λ, then it will go into the LIMIT state and set all cells according to the limsup rule: every cell gets the *limes superior* of the cell values at the stages below λ; in other words: if a cell contains a 1 cofinally often, it has 1 in the limit.

As in the standard case we can define Infinite Time Turing Computability as follows:

DEFINITION 3.1
A real x is **Infinite Time Turing Computable** if there is an Infinite Time Turing Machine that produces, starting from the empty input, the real x on the tape at the time it reaches the HALT state.

This definition immediately leads to Relative Infinite Time Turing Computability (denoted by $\leq_T^\infty$) and a degree structure as for the ordinary Turing degrees.

At present, the structure theory of Infinite Time Turing degrees is only of marginal importance to the results of Section 4. But since the main aim of this paper is to open up a possible area of applications of the field of Infinite Time Computability theory to Revision Theory, it seems reasonable to give the reader a faint idea about what is known in the former area. Thus we briefly describe what is known without any proofs.

Hamkins and Lewis[18] and (following a visit of Joel Hamkins to Kobe in the year 1998) Philip Welch provided us with an abundance of structure theoretic results about Infinite Time Turing Computability and the derived degree structure.

It is easily seen that arithmetic truth is Infinite Time Turing Computable, since you can just

[17] As is to be found in any good textbook of Recursion Theory; e.g., [18].
[18] [9].

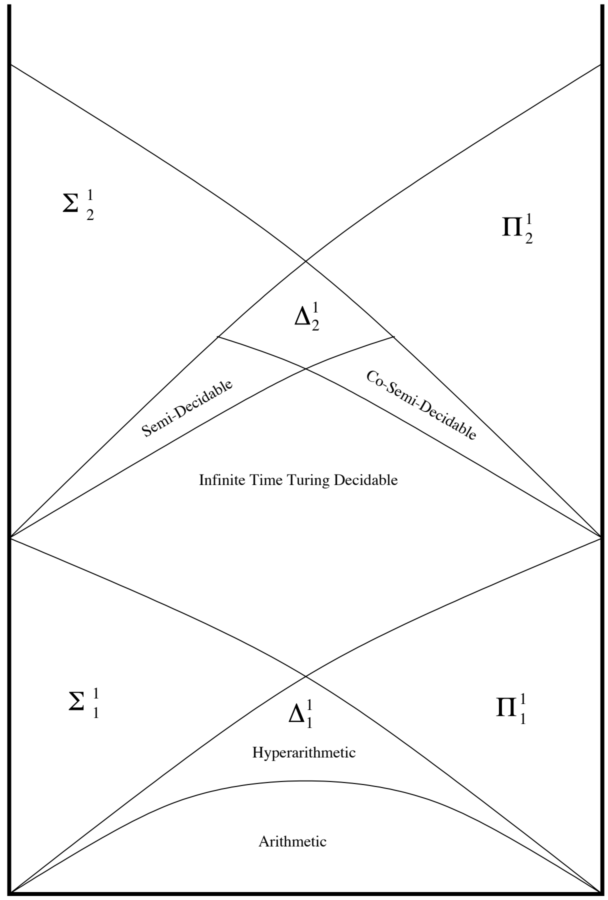

FIGURE 1. The extent of Infinite Time Turing Computation

use ω steps to check every possible witness for an existential quantifier.[19] The class of Infinite Time Turing computable reals forms a subclass of the Δ^1_2-reals as can be seen in Figure 1.

The analysis of the Infinite Time Halting problem is an important component of the structure theory of Infinite Time Turing Machines: The Infinite Time Halting Problem (i.e., the set of machines halting given the empty input) is not Infinite Time Turing Computable. The halting problem and its boldface version (the set of reals coding a machine M and a real x such that M halts given the input x) yield Infinite Time Turing jump operations $x \mapsto x^{\triangledown}$ and $x \mapsto x^{\blacktriangledown}$ of which the following can be shown [9, Theorems 5.1 and 5.6]:

THEOREM 3.2
1. $x <^{\infty}_{T} x^{\triangledown} <^{\infty}_{T} x^{\blacktriangledown}$.
2. Δ^1_2 is closed under $x \mapsto x^{\triangledown}$ and $x \mapsto x^{\blacktriangledown}$.

Hamkins and Lewis[20] have looked much closer at the connections between these jump

[19] [9, Theorem 2.1 and Theorem 2.6].
[20] Cf. [10].

operations and the Infinite Time degree structure, finding situations both symmetrical and asymmetrical with respect to classical Turing degree theory.

Infinite Time Turing Machines are connected to a couple of interesting classes of ordinals. For the following, we fix a coding of countable ordinals as real numbers.

DEFINITION 3.3
Let α be an ordinal.

1. α is called **writable** if there is a real x coding α such that x is Infinite Time Turing Computable,
2. α is called **clockable** if there is an Infinite Time Turing Machine which reaches its HALT state after exactly α steps of computation, and
3. α is called **eventually writable** if there is an Infinite Time Turing Machine and an ordinal η such that after step η the machine just has a fixed code for α on the tape.

We shall write Λ_∞ for the supremum of the writable ordinals, Υ_∞ for the supremum of the clockable ordinals, and Z_∞ for the supremum of the eventually writable ordinals. These ordinals reach quite far into the hierarchy of countable ordinals [9, Corollary 8.2 and Corollary 8.6]:

THEOREM 3.4
Λ_∞ and Z_∞ are admissible ordinals.

Philip Welch [20, Theorem 1.1] was able to prove the following connection between the length of Infinite Time Turing Computations and the possible outputs and then to actually compute Z_∞.

THEOREM 3.5
1. $\Lambda_\infty = \Upsilon_\infty$, and
2. $Z_\infty = \min\{\,\xi\,;\ \mathbf{L}_\xi$ has a transitive Σ_2-end extension$\}$.

The computation of Theorem 3.5 gives a very interesting result about the actual power of the Infinite Time Turing Machine architecture [19, Theorem 2.6]. At limit stages the original Hamkins–Kidder machine used the limsup rule: if a cell contains the value 1 cofinally often, the limit value is also 1. If we look at machine architectures that use different rules, then the computational power does not change (provided the different rule is simple enough). To make this more precise, let us fix the notation: let $z^\beta_{p,y}$ be the real containing the snapshot of the computation of the Infinite Time Turing Machine with the code p on input y at time β.

DEFINITION 3.6
Let $\Gamma : (2^{\mathbb{N}})^{<\mathrm{Ord}} \to 2^{\mathbb{N}}$ be a function. Then Γ is called a **suitable** Σ_2 **operator** if for every machine using Γ in the limit stages there is a Σ_2 formula $\Psi(v_0, v_1, v_2)$ such that

$$\mathbf{L}_\nu[y] \models z^\nu_{p,y}(i) = 0 \leftrightarrow \Psi[i, p, y].$$

THEOREM 3.7
Let γ be a suitable Σ_2 operator. Then a real is computable by a Hamkins–Kidder machine (an Infinite Time Turing Machine as described above) if it is computable by a machine using the limit rule γ.

4 Revision sequences modelled by Infinite Time Turing Machines

By now, it should be clear to the reader that there is some connection between Infinite Time Turing Machines and revision sequences: both concepts move stepwise, taking the information given in the previous step and computing the next one with a finite amount of programming; both concepts are about moving on throughout the ordinals and in both cases we can use the fact that we can run for an infinite amount of time to check arithmetic truth. The following two quotes highlight this similarity:

> A central idea in our approach is to view the function $\delta_{D,M}$ that is supplied by a circular definition D as a rule of revision. Its application to a hypothetical extension X results in a set $\delta_{D,M}(X)$ that is a better candidate for the extension of G, according to the definition, than X. [8, p. 121]

> The Infinite Time Turing Machines have something in the nature of a non-monotonic inductive operator: a set of integers may be input, and at various stages there is a set of integers present on the output tape. [19]

In this section, we shall assume that we have a $\langle \Phi, \Gamma_0 \rangle$-revision sequence of writable ordinal length α, and construct an Infinite Time Turing Machine that in its course of computation constructs the revision sequence on its tape. Our Infinite Time Turing Machine will not use the limsup rule as the standard Hamkins–Kidder machines did, but it will use the liminf rule. This doesn't change the computational power of the machine by Theorem 3.7.

For $\eta < \alpha$ we will denote by ϱ_η the stage of the computation of our machine where s_η appears on the tape for the first time. Our machine will be constructed such that for limit ordinals λ we have $\varrho_\lambda = \bigcup_{\beta<\lambda} \varrho_\beta$.

Now we fix a first-order formula Φ and a writable ordinal α. The Infinite Time Turing Machine we shall construct will have a couple of states that are needed for counting and checking arithmetic truth (we will try and suppress all this detail in the description of the Turing Machine architecture), the START, the HALT and the LIMIT state, and three distinguished states COMPUTE, PRED and NOPRED.

We now describe the action of our machine.

Case 1 : If the machine is in the state START, it splits its tape into six countable parts x_0, x_1, x_2 x_3, x_4 and x_5. It copies the previous content of the tape into x_3, then it initializes x_0, x_1, x_4 and x_5 with an infinite sequence of 0s, and x_2 with an infinite sequence of 1s.

The machine will fill x_0 with a code for the ordinal α, x_2 will be an array telling the machine what natural numbers it has already checked, x_3 will be the storage of initial segments of the new real (i.e., the next real in the revision sequence), and x_1 will be the list of ordinals already handled.

The part x_5 will be used by the machine to do its computations and other details like storing the natural number and the ordinal $\beta < \alpha$ it is working on at the moment. This is the part of the action of the machine that we will suppress almost completely in our account. The bits $x_5(0)$ and $x_5(1)$ will have a special significance:

$x_5(0)$ will be called the SEQUENCE bit, it contains the information whether we are done writing the next element of $\vec{s}$ onto x_3.

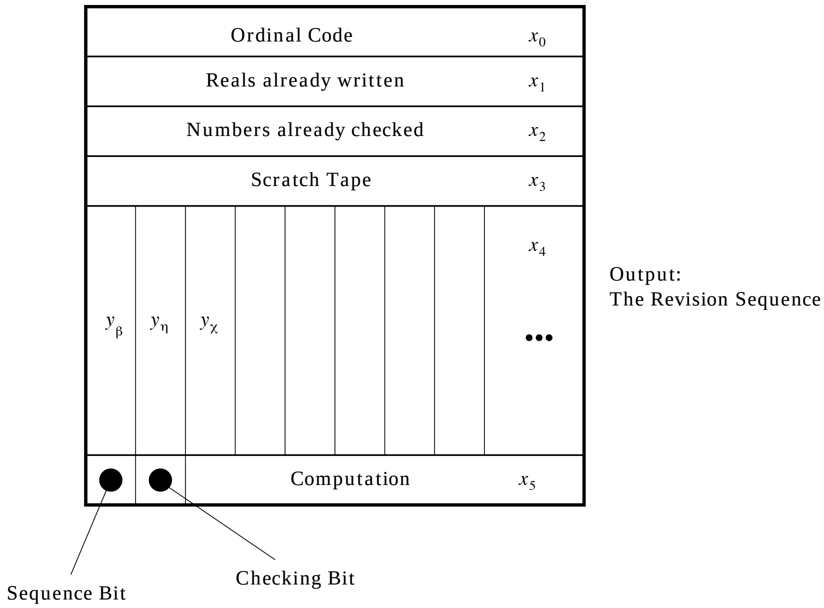

FIGURE 2. The Infinite Time Turing Machine $M_{\Phi,\alpha}$ and its partitioned tape

$x_5(1)$ will be called the CHECKING bit, and it contains the information whether we are finished with checking an arithmetic truth for some fixed natural number.
x_4 will be split up into countably many pieces and will contain the whole revision sequence in the end. The partitioning of the tape can be seen in Figure 2.

After the initialization, the machine writes the ordinal α (i.e., a code for it) onto the tape x_1. Then it splits up the tape x_4 into a countable sequence of tapes $\langle y_\beta; \beta < \alpha\rangle$.[21] The content of tape x_1 tells us the order of the countable sequence of tapes. Then it copies the content of x_3 to every tape y_β and goes to the COMPUTE state.

Case 2 : If the machine is in the COMPUTE state, it browses through x_1 in the order given by the code x_0 to find the (in the order of x_0) least β such that $x_1(\beta) = 0$.
If this β has a predecessor ζ, the machine moves to the PRED state.
If this β has no predecessor, it moves to the NOPRED state.

Case 3 : If the machine is in the NOPRED state, it just sets $x_1(\beta) := 1$ and starts the COMPUTE state case all over again.

Case 4 : If the machine is in the PRED state, the machine sets the CHECKING bit to 1, the SEQUENCE bit to 0, initializes x_3 with an infinite sequence of 0s and starts checking Φ with input y_ζ at the natural number 0.

Case 5 : If the machine is in the LIMIT state, its head moves to the SEQUENCE bit. If this bit contains a 1, the machine writes the content of x_3 to all y_η such that $x_1(\eta) = 0$, afterwards

[21] From this point onwards, we shall confuse α and the ordering of $\mathbb{N}$ isomorphic to α given by x_0. To be exact, we should speak of '$\langle y_n; n \in \mathbb{N}\rangle$ ordered by $y_n \prec y_m$ if and only if $\pi_{x_0}(n) < \pi_{x_0}(m)$ where π_{x_0} is the bijection between $\mathbb{N}$ and α coded by x_0' instead of '$\langle y_\beta; \beta < \alpha\rangle$'. But we don't.

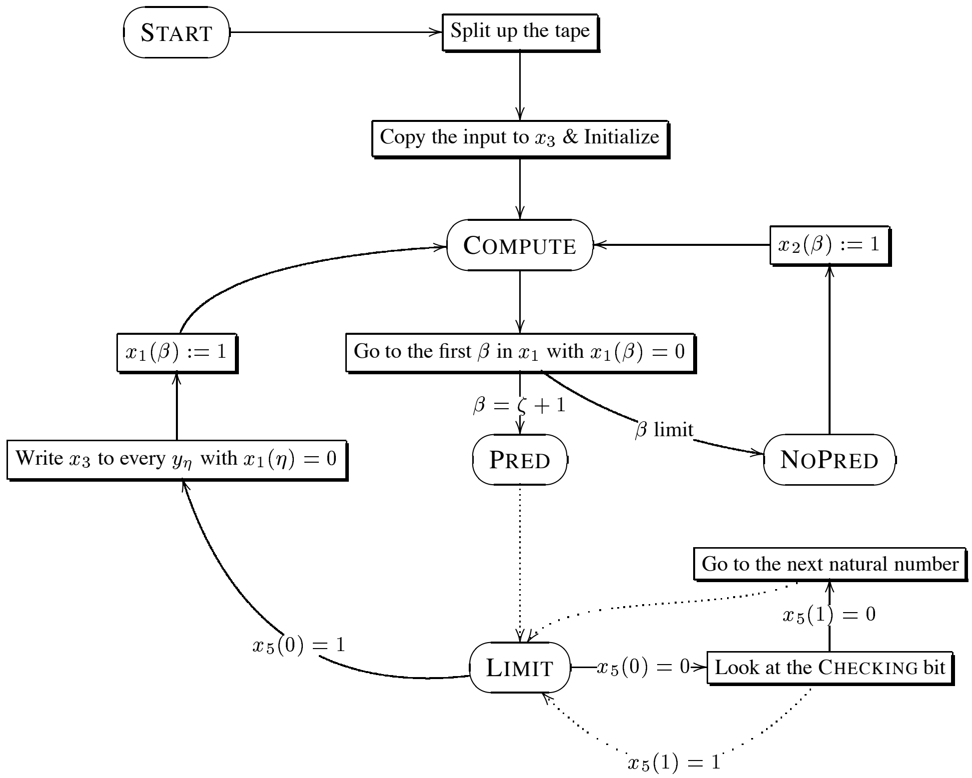

FIGURE 3. Flow Diagram for $M_{\Phi,\alpha}$

sets $x_1(\beta) := 1$ and moves into the state COMPUTE.

If the SEQUENCE bit contains a 0, the head moves on to the CHECKING bit. If the CHECKING bit is 1, the machine continues with its checking job. If the CHECKING bit is 0, the machine writes 1 into the SEQUENCE bit, runs along x_2 until it meets the first 1 in x_2. At that point it overwrites the SEQUENCE bit with 0 and the CHECKING bit with 1, and starts the checking of Φ at the next natural number (with the same input).

As soon as the machine determines whether Φ holds at that natural number, the machine writes the outcome of the check into x_3 and sets the appropriate bit of x_2 and the CHECKING bit to 0.

This completes the definition of our Infinite Time Turing Machine. We shall denote it by $M_{\Phi,\alpha}$. The working schematics of the machine can be seen in the flow diagram, Figure 3.

Note that the construction is somewhat flexible (remember that we are standing on the shoulders of Theorem 3.7). If $\Gamma = \{\gamma\}$ is a singleton with a sufficiently simply definable γ and $\vec{s}$ is a $\langle\Phi,\Gamma\rangle$-revision sequence, we could be a little bit more careful in our construction and receive a machine that computes $\vec{s}$.

THEOREM 4.1

Let α be a writable ordinal, Φ be any given formula of $\mathcal{L}^*$, and $\vec{s} = \langle s_\beta; \beta < \alpha\rangle$ a sequence of reals. Then the following are equivalent:

1. $\vec{s}$ is a $\langle\Phi,\Gamma_0\rangle$-revision sequence, and

2. $M_{\Phi,\alpha}$ eventually computes $\vec{s}$ on the x_4 tape, given s_0 as the input.

PROOF. We describe what the machine does to see that it does exactly what it's supposed to do.

We feed s_0 to the machine in the START state. The machine creates six copies of its tape, writing s_0 on x_3 and erasing all information from the other copies.

The machine then writes the ordinal α onto x_0 (which is possible since α was writable). Then it copies the content of x_3 (which is the original input s_0) to every component of y_η of x_4.

In the COMPUTE state, it finds that we have $x_1(\eta) = 0$ for all ordinals η, thus 0 is the least such ordinal. But 0 has no predecessor, so the machine moves to the NOPRED state, writes 1 to $x_1(0)$ and goes back into the COMPUTE state. Now the least η such that $x_1(\eta) = 0$ is 1 which is the successor of 0.

The machine takes y_0 as the input and starts checking whether $\Phi(y_0, 0)$ is true. If it meets a limit stage during this check, the CHECKING bit is 1, so it resumes its checking until it manages to determine whether $\Phi(y_0, 0)$ holds. If $\Phi(y_0, 0)$ holds, then the machine sets $x_3(0) := 1$, otherwise it writes 0. Afterwards the machine defines $x_2(0)$ to be 0, in order to say that it has checked the number 0. Now the CHECKING bit is set to 0, and the machine can move on: it runs along x_2, meets the first 1 at the first cell and starts to check $\Phi(y_0, 1)$ (after reinitializing the CHECKING bit to 1).

After the machine has checked all natural numbers with the same input, it finds itself in the LIMIT state. At this moment, the SEQUENCE bit is still 0, but every cell on x_2 is filled with a 0. So when the head runs along x_2 if never encounters a 1 and thus never changes the SEQUENCE bit; by the liminf rule, the SEQUENCE bit contains a 1 in the next limit step and hence the machine writes the just computed real (the content of x_3, which is by construction $\delta_\Phi(s_0) = s_1$) into every y_η with $\eta > 0$ (note that 0 is still the only ordinal such that $x_1(0) = 1$). Then it sets $x_1(1) := 1$ and moves in the COMPUTE state. Now 2 is the least ordinal η such that $x_1(\eta) = 0$. As the predecessor of 2 is 1, the computation is started all over again with y_1 as the input.

This describes in fact how that machine computes all successor steps of the revision sequence $\vec{s}$.

What about the limit stages in the sequence $\vec{s}$?

Let λ be a limit ordinal and $\varrho := \bigcup_{\eta<\lambda} \varrho_\eta$.

Then ϱ is a limit, since $\langle \varrho_\eta; \eta < \lambda \rangle$ is a strictly increasing sequence of limit length. But if we look at y_λ in the course of computation up to ϱ, y_λ contains every element of $\vec{s}{\restriction}\lambda$. So the following holds for all natural numbers n:

$$\exists \zeta < \varrho \, \forall \xi \geq \zeta \, (y^\xi_\lambda(n) = 1) \iff \exists \zeta < \lambda \, \forall \xi \geq \zeta (n \in s_\xi).$$

Thus by the liminf rule and by the fact that $s_\lambda = \gamma_0(\vec{s}{\restriction}\lambda)$, we have $y^\varrho_\lambda = s_\lambda$. In fact, this is true for all y^ϱ_χ with $\chi \geq \lambda$.

Now below ϱ the machine has run through the COMPUTE case cofinally often. Thus, the SEQUENCE bit has been reset to 0 cofinally often and by the liminf rule this means that it is 0 at ϱ.

The CHECKING bit is also 0 since the machine has successfully checked a given natural number cofinally often below ϱ.

With the same argument, x_2 at the stage ϱ is constantly 0 (every natural number has been checked cofinally often), so the machine runs along x_2 and in the step $\varrho + \omega$, the SEQUENCE bit is at 1.

Consequently, the machine moves to the COMPUTE state, realizes that λ is the least ordinal such that $x_1(\lambda) = 0$. Since λ does not have a predecessor, it writes $x_1(\lambda) = 1$ and moves on to compute $y_{\lambda+1}$. ∎

5 The limit rule and other applications

Using the correspondence between revision sequences and Infinite Time Turing Machines we head towards possible applications (and point out possible complications and obstacles).

We have mentioned the limit problem of Revision Theory a couple of times before:

'If we let Γ vary, what happens to the notion of $\mathbf{S}_\Gamma^\#$-definability?[22]

The apparent freedom of choosing the limit rule for Infinite Time Turing Machines should give us some way of tackling the limit problem for Revision Theory. In order to use that, we would have to get a uniform version of Theorem 4.1: suppose that h is $\mathbf{S}_\Gamma^*$-definable. Then for every natural number n such that $n \notin h$, we have a revision sequence $\vec{s^n}$ witnessing this (i.e., $n \notin s_0^n$ and s_0^n recurs in $\vec{s^n}$). One of the problems in applying Theorem 4.1 is that we don't know anything about the lengths of the witnessing sequences $\vec{s^n}$, and our theorem works only for sequences of writable ordinal length.

This problem naturally leads us into a second possible field of applications:

The fact that we have a good understanding what ordinals are writable and where the boundaries of writability lie, poses several interesting related questions about Revision Theory:

> What can we say about the length of relevant parts of revision sequences? (i.e., how large can $\mathrm{refl}_{\Phi,\Gamma}(h)$ become?)

A useful solution of this question would also give us the needed uniform version of Theorem 4.1.

In solving the limit rule problem, there are some additional (somewhat deeper) problems with unrestricted revision sequences (i.e., if the class Γ is large), though, since the mere existence of a revision sequence doesn't say anything about the complexity of choices from Γ made on the way to $\mathrm{refl}_{\Phi,\Gamma}(s_0)$.

A third area of possible applications concerns the extent of revision theoretic definability. It might be possible to use the stratification of Δ_2^1 by the relation $\leq_T^\infty$ to get results for the complexity of revision theoretically definable reals.

Concluding, there is a clear connection between Revision Theory and Infinite Time Turing machines, and it seems very promising, but there is still a lot of technical work to be done to get to a point where we can exploit this connection.

---

[22] [4, p.403]: 'What category a proposition is going to belong to depends on the kinds of bootstrapping policies admitted.'

6 Aftermath

In the months between the preparation of the first version and the submission of the final version of this article, Philip Welch has used the methods of this paper in his [21] to solve the questions mentioned in Section 5.

THEOREM 6.1 (Welch)
Every Π^1_2 real is $\mathbf{S}^\#_\Gamma$-definable and $\mathbf{S}^*_\Gamma$-definable for a class of Γ including Γ_0 and Γ_∞.

Theorem 6.1 together with Corollary 2.9 gives an exact computation of the class of $\mathbf{S}^\#_{\Gamma_\infty}$-definable reals. They are exactly the Π^1_2 reals.

THEOREM 6.2 (Welch)
For arithmetic formulae Φ, simple classes Γ of bootstrapping policies and any $\langle\Phi,\Gamma\rangle$-reflexive real number h, we have $\mathrm{refl}_{\Phi,\Gamma}(h) < \Upsilon_\infty(h)$, where $\Upsilon_\infty(h)$ is the supremum of the ordinals clockable relative to h.

Acknowledgements

The author is indebted to Kai-Uwe Kühnberger (Bloomington IN & Tübingen) who introduced him to the problem discussed in this paper, to Philip Welch (Kobe, Bristol & Berkeley CA) for discussions and advice, and to the anonymous referee. Part of this work was completed while the author held a grant of the Studienstiftung des deutschen Volkes. The author wishes to extend his special gratitude towards the Association of Symbolic Logic for a travel grant to Logic Colloquium 1998 in Prague.

References

[1] G. A. Antonelli. The complexity of revision, *Notre Dame Journal of Formal Logic*, **35**, 67–72, 1994.

[2] G. A. Antonelli. The complexity of revision, revised, unpublished notes, 1998.

[3] N. D. Belnap, Jr. Gupta's rule of revision theory of truth, *Journal of Philosophical Logic*, **11**, 103–116, 1982.

[4] A. Chapuis. Alternative revision theories of truth, *Journal of Philosophical Logic*, **25**, 399–423, 1996.

[5] J. Earman and J. D. Norton. Forever is a day: supertasks in Pitowsky and Malament–Hogarth Spacetimes, *Philosophy of Science*, **60**, 22–42, 1993.

[6] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*, Perspectives in Mathematical Logic, Springer Verlag Berlin, 1995.

[7] A. Gupta. Truth and paradox, *Journal of Philosophical Logic*, **11**, 1–60, 1982.

[8] A. Gupta and N. D. Belnap, Jr. *The Revision Theory of Truth*, MIT Press, Cambridge, MA, 1993.

[9] J. D. Hamkins and A. Lewis. Infinite time Turing machines, *Journal of Symbolic Logic*, to appear.

[10] J. D. Hamkins and A. Lewis. Post's problem for supertasks has both positive and negative solutions, *Archive for Mathematical Logic*, submitted.

[11] H. G. Herzberger. Naive semantics and the liar paradox, *Journal of Philosophy*, **79**, 479–497, 1982.

[12] H. G. Herzberger. Notes on naive semantics, *Journal of Philosophical Logic*, **11**, 61–102, 1982.

[13] M. L. Hogarth. Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, **5**, 173–181, 1992.

[14] P. Kremer. The Gupta–Belnap systems $\mathbf{S}^\#$ and $\mathbf{S}^*$ are not axiomatisable, *Notre Dame Journal of Formal Logic*, **34**, 583–596, 1993.

[15] S. Kripke. Outline of a theory of truth, *Journal of Philosophy*, **72**, 690–716, 1975.

[16] R. L. Martin and P. W. Woodruff. On representing 'true-in-L' in L, *Philosophia*, **5**, 217–221, 1975.

[17] Y. N. Moschovakis. *Elementary Induction on Abstract Structures*. Studies in Logic and the Foundations of Mathematics 77, North Holland, Amsterdam, 1977.

[18] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*, McGraw-Hill Publishing Company, Maidenhad (Berkshire), 1967.

[19] P. D. Welch. Eventually infinite time Turing machine degrees: infinite time decidable reals, *Journal of Symbolic Logic*, **65**, 2000.

[20] P. D. Welch. The length of infinite time Turing machine computation, *Bulletin of the London Mathematical Society*, **32**, 129–136, 2000.

[21] P. D. Welch. On Gupta–Belnap revision theories of truth, unpublished notes.

[22] A. M. Yaqūb, *The Liar Speaks the Truth. A Defense of the Revision Theory of Truth*, Oxford University Press, Oxford, 1993.

Received June 1999