Minds & Machines (2011) 21:435–448
DOI 10.1007/s11023-011-9245-z

Turing Interrogative Games

Paweł Łupkowski · Andrzej Wiśniewski

Received: 18 November 2010 / Accepted: 9 March 2011 / Published online: 24 March 2011

Abstract The issue of adequacy of the Turing Test (TT) is addressed. The concept of Turing Interrogative Game (TIG) is introduced. We show that if some conditions hold, then each machine, even a thinking one, loses a certain TIG and thus an instance of TT. If, however, the conditions do not hold, the success of a machine need not constitute a convincing argument for the claim that the machine thinks.

Keywords Turing test · Logic of questions · Interrogative games · Recursion theory

The Turing Test

Despite its years, the Turing Test (TT) still brings in new issues, and the dispute over it is far from being closed; see for example the latest collection of papers (Epstein et al. 2009). One of the central topics in this area is the problem of adequacy of TT as a tool for investigating if artificial agents think.[1] Our aim is to address this issue by using a certain formal approach.

Papers devoted to TT most often are philosophical in nature and methods used. Among recent papers notable exceptions are: Turing Machines approach (Sato and Ikegami 2004), exploring the computational complexity of TT setting (Hernandez-Orallo 2000), applying the interactive proof theory in modeling TT (Bradford and

[1] We prefer ‘thinks’ over ‘is intelligent’, since the concept of intelligence is currently used in a way that does not presuppose the presence of mental processes.

P. Łupkowski (✉) · A. Wiśniewski
Chair of Logic and Cognitive Science, Institute of Psychology, Adam Mickiewicz University, Poznan, Poland
e-mail: Pawel.Lupkowski@amu.edu.pl

A. Wiśniewski
e-mail: Andrzej.Wisniewski@amu.edu.pl

Wollowski 1995; Shieber 2006, 2007). In our paper we will make use of some elementary concepts of recursion theory and of a certain recent result in the (metatheory of the) logic of questions (cf. Wiśniewski and Pogonowski 2010). Our approach differs from the mentioned ones not only in the formal method used. We attempt to stay as close to the Turing's original proposal as it is possible. In order to achieve this we, first, focus on the core of Turing's proposal and then, on this basis, introduce the concept of *Turing Interrogative Game* (TIG for short). We do not claim, however, that TT can be performed *only* via TIG(s). What we claim is: once a TIG is performed, an instance of TT takes place. A comparison of basic properties of TIGs (which are characterized in the section "Turing Interrogative Games") with the features of TT described in this section justifies this claim.

TT is conceptualized in many non-equivalent ways, some of them showing only a distant resemblance to the original Turing's idea.[2] So let us specify what we count as the core of Turing's proposal. We rely on the following sources in which Turing writes or speaks about the Test: "Intelligent Machinery" (Turing 1948), "Computing Machinery and Intelligence" (Turing 1950), "Can Digital Computers Think" (Newman et al. 1952), "Intelligent Machinery, a Heretical Theory" (Turing 1951b), "Can Automatic Calculating Machines be Said to Think?" (Newman et al. 1952), and "Digital Computers Applied to Games" (Turing 1953). The above sources give the following image of the basics of TT:

- TT resembles the so-called imitation game, but the analogy is not complete. The imitation game involves three players: a man (A), a woman (B), and an interrogator; the objective of an interrogator is to identify which of the players, A and B, is a man, and which is a woman. The objective of an interrogator in TT is different: he/she aims at an identification of a machine that thinks. Therefore only two parties are enough: an interrogator (sometimes referred to as a judge or a jury)[3] and a tested agent. In his seminal paper "Computing Machinery and Intelligence" Turing uses the term viva voce for the version of a game with player B omitted (cf. Turing 1950, p. 446). In his later works Turing straightforwardly refers to TT as to a two-parties enterprise. For example, in "Digital Computers…" one reads:

> "I am imagining something like a viva-voce examination, but with the questions and answers all typewritten in order that we need not consider irrelevant matters as the faithfulness with which the human voice can be imitated." (Turing 1951a, pp. 4–5)

- The interrogator is not supposed to know the identity of the tested agent. The parties cannot see or hear each other. As Turing puts it:

> "A considerable proportion of jury, who should not be expert about machines, must be taken in by the pretence. They aren't allowed to see the machine itself—that would make it too easy. So the machine is kept in a far away room

[2] For an overview of the discussion on TT rules, see e.g. Copeland and Proudfoot (2009), Saygin et al. (2001).

[3] For an overview of terminology used in the context of TT, see e.g. Harnish (2002, p. 183).

and the jury are allowed to ask it questions, which are transmitted through to it: it sends back a typewritten answer." (Newman et al. 1952, p. 4)

- The game is played by means of questions and answers. This is only the interrogator who asks questions. The second party never asks questions, but is supposed to answer them. This feature plays an important role in our paper, so the claim deserves a wider justification. First, we find some indirect evidence in "Computing Machinery…". TT resembles the imitation game, and in the latter questions are asked only by the interrogator, whereas answers are given by the players A and B–cf. (Turing 1950, p. 433). Second, the examples of dialogues given by Turing in "Computing Machinery…" display that questions are asked only by the interrogator (cf. Turing 1950, p. 434–435 and p. 446). Finally, in "Can Automatic Calculating Machines…" one reads:

"The idea of the test is that the machine has to try and pretend to be a man, by answering questions put to it, and it will only pass if the pretence is reasonably convincing." (Newman et al. 1952, p. 4)

- The aim of an AI agent tested is to mislead the interrogator in such a way that he/she would not be able to make the accurate identification (cf. Turing 1950, p. 434). The agent should attempt to answer questions in a human-like manner, and even can use some tricks to achieve this, e.g. can make mistakes in calculations, or spelling mistakes, etc.

"Likewise the machine would be permitted all sorts of tricks so as to appear more man-like, such as waiting a bit before giving the answer, or making spelling mistakes…" (Newman et al. 1952, p. 5)

- An interrogator is free in his/her choice of questions asked; Turing does not impose any restrictions here. He writes in "Computing Machinnery…":

"The question and answer method seems to be suitable for introducing almost any one of the fields of human endeavor that we wish to include." (Turing 1950, p. 435)

In "Can Automatic Calculating Machines…" one reads:

"BRAITHWAITE: Would the questions have to be sums, or could I ask it what it had had for breakfast?
TURING: Oh yes, anything. And the questions don't really have to be questions, any more than questions in a law court are really questions. You know the sort of thing. "I put it to you that you are only pretending to be a man" would be quite in order." (Newman et al. 1952, p. 5)

- An agent should be tested long enough to gain more reliable results. As it is clearly stated in "Can Digital Computers Think":

"We had better suppose that each jury has to judge quite a number of times, and that sometimes they really are dealing with a man and not a machine. That will prevent them saying "It must be a machine" every time without proper consideration." (Newman et al. 1952, p. 5)

- The main objective of TT is to differentiate between AI agents that think and AI agents that do not. In this paper we, purposely, do not address the issue of what concept of thinking, if any, is involved in the TT's setting. Discussions on this issue are still open (cf. e.g. Copeland 2000; Moor 1976). Turing himself writes:

"I don't want to give a definition of thinking, but if I had to I should probably by unable to say anything more about it than that it was a sort of buzzing that went on inside my head. But I don't really see that we need to agree on a definition at all. The important thing is to try draw a line between the properties of a brain, or of a man, that we want to discuss, and those that we don't." (Newman et al. 1952, p. 3–4)

Generally speaking, what matters for an AI agent in TT is to evince relevance in a dialogue at the same level as humans do (cf. Ginzburg 2010). It is claimed that the success of an AI agent would provide good reasons to believe that the AI agent thinks (see Stalker 1976).[4] It is presupposed that an AI agent that thinks is able to pass the test. It is assumed that the result of TT is neither predetermined by the structure of the test nor by the fact that the tested party is an AI agent.

Turing Interrogative Games

In this section we introduce the concept of Turing Interrogative Game (TIG). TIGs display the key features of TT (in its viva voce version) specified in the previous section. However, for simplicity we plainly assume that the second party of a TIG is a machine, and we impose some conditions which explicate and/or supplement the TT rules.

TIGs are characterized as follows.

1. There are two parties, **Int** (after 'Interrogator') and **M** (after 'Machine'), which play a TIG.

The role of **Int** is to ask questions, whereas **M** is supposed to answer them. We assume, however, that each question asked by **Int** is accompanied with a *condition* which is set by **Int**. The underlying idea is: **Int** specifies that something is or is supposed to be the case and then asks a question. The role of **M** is to provide an answer that is *correct with respect to the condition* just set by **Int**. By 'correct with respect to a condition' we mean here 'resolving in view of the condition' and nothing more; in particular, truth of the answer is not presupposed (but also not excluded). For simplicity, we assume that conditions are expressed by declarative sentences. Here is an example of a condition-question pair taken from Turing (1950):

[4] Although, it seems, it is not claimed that a failure provides an argument for saying that a tested agent, either a machine or a man, does not think. The (abductive!) reasoning goes from 'an agent passes the test' to 'the agent thinks'.

"I have K at my K1, and no other pieces. You have only K at K6 and R at R1. It is your move. What do you play?" (Turing 1950, p. 435)

One can use more than one sentence to express a condition. In such a case the condition is a conjunction of the sentences used.

Specifying a condition is not mandatory in the original TT setting. Yet the following justifies us in stipulating that each question asked in a TIG is to be accompanied with a condition. First, one can argue that questions asked in TT but not accompanied with explicit conditions are, by default, questions about the actual state of the world or the domain just investigated. In TIGs, we make this feature visible by formulating the relevant condition. Second, TT is a kind of dialogue, and a question of a dialogue that pertains to a given topic sometimes arises from previous claims about the topic, answers to previous questions (if any) included. This can be made explicit in TIGs; the condition accompanied with a raised question is made up of the relevant items of information.

2. As in the case of TT, we assume that **Int** is free in the choice of questions and conditions. This mirrors the active role of an interrogator in TT. Moreover, **M** is not permitted to ask any question.

3. It is assumed that for each question Q and each condition γ there exist(s) some *correct answer(s) to* Q *with respect to* γ. However, we do not claim that, given a question Q and a condition γ, there exists exactly one correct answer to Q with respect to γ. In some cases there may be more such answers. Consider e.g.:

Take arithmetic. Which prime is greater than 7?

Moreover, we assume that any answer to Q that is correct with respect to some condition(s) is either a direct answer to Q or a sentence which says 'there is no correct answer'; we express the latter by $\odot$. By a *direct answer* to a question we mean an expression which is a possible and just-sufficient answer to the question, i.e. gives neither more nor less information than it is called for by the question itself.[5] We use dQ for the set of *all the* direct answers to question Q.

The introduction of $\odot$ seems inevitable in the context of TIGs. Recall that, on our account, 'correctness' is always relative to the corresponding condition. Consider the following question which might be asked by **Int**:

Who was the last king of the United States of America?

Suppose that the question is asked and accompanied with a condition to the effect that it pertains to the real world. Now it becomes a contextually 'tricky' question, and the answer of the form 'there is no correct answer' seems to be the correct one in the context and *within* a TIG. In an everyday conversation the above question would probably be responded to with a comment or a clarification request, but since in TT and TIGs a tested agent is not permitted to ask questions, contextually 'tricky' questions are to be correctly answered by 'there is no correct answer'. The second reason for which we need $\odot$ in TIGs is the following. According to (2) above and (5) below, **Int** is free in the choice of questions and conditions. So, in principle, **Int**

[5] Cf. Belnap and Steel (1976), cf. also Wiśniewski (1995) and Harrah (2002).

can ask a question which is not relevant with regard to the condition just set, or can set a condition which is not relevant to the question just asked. In both cases the answer which is correct with respect to the condition is 'there is no correct answer' and for this reason we need $\odot$. Here is an illustration:

Take arithmetic. Does Pegasus exist?

Finally, we do not forejudge that for each question Q, the set dQ of direct answers to Q exists and is non-empty. Another application of $\odot$ is: we assume that if dQ is empty or no direct answer to Q is defined/exists, then $\odot$ is the unique correct answer to Q with respect to any condition.[6]

4. We also assume that the following hold:

(♠) if β is a direct answer to Q, then there exists a condition γ_β such that β is the unique correct answer to Q with respect to γ_β,
(♣) there exists a condition ϑ such that $\odot$ is the only correct answer to Q with respect to ϑ.

Clause (♠) does not require any comments. As for (♣), observe that if 'there is no correct answer' is the *only* information on condition of which a question is asked, then 'there is no correct answer' is the correct answer.

5. We assume that a TIG is played in a language such that: (*i*) the set of expressions of the language includes declarative sentences (sentences for short) and questions, (*ii*) expressions of the language can be coded by natural numbers, i.e. there exists a coding method according to which each sentence is coded by an unique natural number, and similarly for questions, (*iii*) the set of sentences of the language is denumerable and at least recursively enumerable (r.e. for short), and (*iv*) the set of questions of the language is r.e. (By 'denumerable' we mean, here and below, 'countably infinite'.) Moreover, $\odot$ is supposed to occur in the language.

Clause (*ii*) allows us to use tools taken from (classical) recursion theory.

Each question of the language can occur in TIGs, and each sentence of the language can be used to set a condition in some TIG(s). We also assume that each sentence of the language is a direct answer to some question(s) of the language.

Since sentences and questions have unique numerical codes, in what follows we will sometimes disregard the distinction between the relevant expressions and (their) numerical codes. In particular, this pertains to the considerations below.

6. **M** is a machine and thus proceeds only algorithmically. On the other hand, **M** is supposed to provide those answers to questions asked by **Int** which are correct with respect to conditions that are set by **Int**. Without loss of generality we may assume that the correct answers (if any) provided by **M** are outputs of question-resolving

[6] At first sight this may seem non-intuitive. However, as we will see in "The Trap" (cf. Remark 1), this blocks a possibility of **Int**'s unfair success, i.e. a success achieved by asking a question which cannot be answered with a direct answer due to the lack of such answer(s).

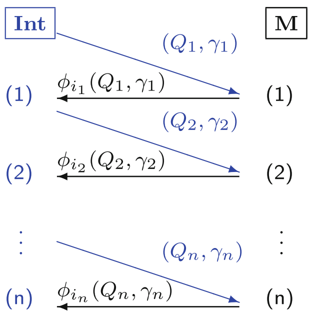
Int
M
(Q_1, γ_1)
(1)
$\phi_{i_1}(Q_1, \gamma_1)$
(1)
(Q_2, γ_2)
(2)
$\phi_{i_2}(Q_2, \gamma_2)$
(2)
(Q_n, γ_n)
(n)
$\phi_{i_n}(Q_n, \gamma_n)$
(n)

Fig. 1 The Turing interrogative game

algorithms which are accessible to **M**; the outputs are sent to **Int**.[7] For our purposes it is convenient to view algorithms as partial recursive functions.

Let Φ be the set of sentences of the language in which a TIG is played, and let Ψ be the set of questions of the language. Domain and range of a function ϕ will be designated by $dom(\phi)$ and $rng(\phi)$, respectively.

By a *question-resolving algorithm* (qr-algorithm for short) we mean a partial recursive function ϕ such that $dom(\phi) \subseteq \Psi \times \Phi$, and for any $(Q, \gamma) \in dom(\phi) : \phi(Q, \gamma)$ is a correct answer to Q with respect to γ.[8]

Let us stress that qr-algorithms do not *define* what is the correct answer in a given context; this is already preestablished.

We say that a qr-algorithm ϕ *pertains* to question Q if $(Q, \beta) \in dom(\phi)$ for some $\beta \in \Phi$. Similarly, a qr-algorithm ϕ *pertains* to sentence γ if $(Q, \gamma) \in dom(\phi)$ for some $Q \in \Psi$.

A qr-algorithm ϕ is *accessible* to **M** if **M** is able to compute the values of ϕ for the whole domain of ϕ. We assume that the class of qr-algorithms that are accessible to **M** is non-empty and finite.[9] We use $\Sigma_{\mathbf{M}}$ for the set of all the qr-algorithms accessible to **M**, and $Rng(\Sigma_{\mathbf{M}})$ for the union of ranges of all the functions in $\Sigma_{\mathbf{M}}$. Clearly, $Rng(\Sigma_{\mathbf{M}})$ is r.e., as an union of a finite number of r.e. sets.

7. A TIG is played in *rounds*. In a round, **Int** sets a condition γ and asks a question Q, and **M** is supposed to provide an answer to Q that is correct with respect to γ. The number of rounds of a TIG is always finite. Yet there is no upper limit on the number of rounds that pertains to all TIGs: the games are unrestricted. It is up to **Int** to decide when the game ends (Fig. 1).

M wins a round iff **M** provides a certain expression that is a correct answer to the question asked with regard to the condition set; otherwise **M** loses the round and **Int**

[7] In order to get a more realistic picture one can assume that there is some intermediate device which "translates" the conditions and questions provided by **Int** into their numerical codes and "translates" the outputs send by **M** into expressions of the language. However, this assumption is inessential for the claims of this paper.

[8] Recall that there may exist many answers which are correct with respect to a given condition. In such a case there may exist many qr-algorithms pertaining to the question and the condition; of course, qr-algorithms pertaining to a given question agree at the "point" referred to by (♠). Let us stress that we do not presuppose that direct answers to a question are always mutually exclusive.

[9] Note that this does not imply that **M** can cope only with finitely many questions, or that questions with infinite sets of direct answers cannot be dealt with!

wins the round. **Int** loses a round if **M** wins the round. Observe that **M** can lose a round in two ways: (a) the expression provided is not an answer of the required kind, or (b) no output is provided.

Int wins the game iff there is a round of the game which is lost by **M**; otherwise **Int** loses the game and **M** wins the game. **M** loses the game if **M** loses a round of the game.

We say that a TIG τ' *extends* a TIG τ if τ' comprises, besides the rounds already played in τ, also some new round(s).

A general comment is in order now. TIGs preserve the key features of TT specified in the first section of this paper. But we do not claim that TT can be performed *only* via TIG(s).[10] We only claim that once a TIG is performed, an instance of TT takes place. This, however, is sufficient for our purposes.

The Trap

One may expect that the success of a machine in a TIG depends only on how "strong" the machine is, that is, what qr-algorithms are accessible to the machine. However, in this section we will show there are systematic reasons for which *each* machine loses some TIG(s).

We need some auxiliary notions; they will be introduced step by step.

A question is called *effective* iff the set of (all the) direct answers to the question is non-empty and r.e. By an *ω-question* we mean a question whose set of direct answers is a denumerable (i.e. countably infinite) set of sentences.

Proposition 1 *Each TIG won by a machine, but played in a language in which there occurs at least one non-effective ω-question, has an extension which is won by the interrogator.*

Proof Let Q be a non-effective ω-question. Hence dQ (i.e. the set of direct answers to Q) is non-empty (actually, denumerable), but not r.e. Thus the set $dQ \cup \{\odot\}$ is not r.e.

For each qr-algorithm ϕ that pertains to Q there exists a one-argument partial recursive function ϕ_Q defined by:

$$\phi_Q(\gamma) = \phi(Q, \gamma)$$

such that $dom(\phi_Q) = \{\gamma : (Q, \gamma) \in dom(\phi)\}$. Observe that $rng(\phi_Q) \subset dQ \cup \{\odot\}$, for any ϕ that pertains to Q. Of course, each $rng(\phi_Q)$ is r.e.

Let Δ_Q be the set of all the functions satisfying the above conditions and defined by means of the qr-algorithms that pertain to Q and are accessible to **M**. Let $Rng(\Delta_Q)$ be the union of ranges of all the functions in Δ_Q. Thus $Rng(\Delta_Q)$ equals,

[10] For example, an analysis of Turing's writings devoted to TT shows that he uses the term 'question' somewhat loosely. In particular, requests for a manifestation of a complex linguistic activity ("Write a poem!", or "I put it to you that you are only pretending to be a man.", etc.) can be issued in TT by an interrogator. However, we do not model this in TIGs. We also simplified the answerhood issue. One can argue that answers allowed in TT include replies of kinds which are not taken into account in TIGs. Moreover, the second party of a TIG is plainly supposed to be a machine.

intuitively speaking, the set of all the possible outputs with respect to Q of the qr-algorithms pertaining to Q which are accessible to **M**. Clearly, $Rng(\Delta_Q) \subset dQ \cup \{\odot\}$. Yet $Rng(\Delta_Q)$ is r.e., as an union of a finite number of r.e. sets (recall that only finitely many qr-algorithms are accessible to **M**). On the other hand, $dQ \cup \{\odot\}$ is not r.e. Hence there exists an element, β, of $dQ \cup \{\odot\}$ which does not belong to $Rng(\Delta_Q)$.

A remark is in order. The fact that β does not belong to $Rng(\Delta_Q)$ only means that there is no qr-algorithm accessible to **M** that gives β on an input consisting of Q and (any!) condition. Of course, β can be an output of a certain qr-algorithm on an input involving a question different from Q, but this is unimportant to the argument to follow.

There are two possibilities: (i) $\beta \in dQ$ or (ii) $\beta = \odot$.

Let τ be a TIG won by **M**.

Suppose that (i) holds. By (♠), there exists condition γ_β such that β is the only correct answer to Q with respect to γ_β. But **Int** is free in the choice of questions and conditions, and any question of the language may occur in a TIG. Recall that although each game is finite, there is no upper limit on the number of rounds that pertains to all games. So **Int** is permitted to extend τ step by step, and each extension will be a TIG which is an extension of the initial game τ. Hence there exists a TIG τ' which is an extension of τ and whose last round involves γ_β and Q. But there is no qr-algorithm accessible to **M** which generates β out of a condition and Q. On the other hand, β is the correct answer to Q with respect to γ_β. So **M** loses the round and thus **Int** wins the game τ'.

Now suppose that (ii) holds. The situation is analogous, due to (♣). □

Remark 1 Observe that the effect pointed at by Proposition 1 comes, in a sense, automatically, due to the lack of an appropriate qr-algorithm accessible to **M**. This, in turn, is not a mere coincidence, but an inevitable phenomenon induced by the properties of the language under consideration: it is always the case that some qr-algorithm needed is lacking. Moreover:

- **Int** always wins in a long run, but in order to achieve this **Int** need not know in advance which condition and/or which question will trigger the effect.
- What is crucial is the presence of a non-effective ω-question. Recall that a non-effective ω-question *has* direct answers (actually, denumerably many of them), but the trouble is that the set of direct answers to the question is not r.e. One may expect that **Int**'s asking a question "obscure" enough so that no direct (i.e. possible and just-sufficient) answer to it is defined/exists automatically results in **M**'s failure. But we have blocked such easy wins. According to what had been said in the section "Turing Interrogative Games", footnote 6, the correct answer to an "obscure" question, with respect to any condition, is $\odot$, and qr-algorithms can mimic this.
- **M** loses the last round of the extension by providing no output. By the way, this raises an interesting question: how long should an interrogator wait in order to conclude that there will be no answer at all?

- Of course, the continuity of conversation can be retained by some tricks, e.g. by allowing the machine to respond with questions and/or clarification requests. This, however, would not change the general picture: a round is lost anyway.

Remark 2 So far we have considered unrestricted TIGs. The situation would be different if we imposed an upper limit on the number of rounds in TIGs. Now for games of maximal permitted length won by **M** there are no extensions won by **Int**. But the question arises: what is the limit? Moreover, it is obvious that, due to the fact that **M** proceeds only algorithmically (with the consequences of this fact, see (6) above) and the properties of the language, **M** would not be able to win *each* game of a maximal permitted length.

By Ω-*sentence* we mean a sentence falling under the schema:

'Q' is an ω-question.

where Q is a question.

One can reasonably claim that if Γ_Υ is the set of Ω-sentences each of which involves a question that belongs to a set of questions Υ, i.e.:

$$\Gamma_\Upsilon = \{\text{`}Q\text{' } \textit{is an}\, \omega\text{–question} : \mathrm{Q} \in \Upsilon\}$$

then Γ_Υ is r.e. iff Υ is r.e. In what follows we assume that the coding method used preserves this effect.

As before, Ψ stands for the set of questions of the language of a TIG. Let Ψ_ω be the set of ω-questions of the language. It seems natural to assume that the following holds:

$$(\heartsuit)\, Rng(\Sigma_{\mathbf{M}}) \cap \Gamma_\Psi \subset \Gamma_{\Psi_\omega}.$$

The underlying idea is: once we have an output of a qr-algorithm saying that Q is an ω-question, then Q is, in fact, an ω-question.

A language *permits* Ω-sentences if for each question of the language, the corresponding Ω-sentence belongs to the language.

Proposition 2 *Each TIG won by a machine, but played in a language that permits Ω-sentences and whose set of ω-questions is not r.e., has an extension which is won by the interrogator.*

Proof The set $Rng(\Sigma_{\mathbf{M}}) \cap \Gamma_\Psi$ is r.e., as an intersection of r.e. sets. However, the set Γ_{Ψ_ω} is not r.e., for if it were, Ψ_ω would be r.e. Hence $Rng(\Sigma_{\mathbf{M}}) \cap \Gamma_\Psi \neq \Gamma_{\Psi_\omega}$. Since $(\heartsuit)$ holds, it follows that there exists an Ω-sentence belonging to Γ_{Ψ_ω} which is not an output of any qr-algorithm accessible to **M** that gives Ω-sentences as outputs. But each sentence, the relevant Ω-sentence included, is a direct answer to some question, and, by $(\spadesuit)$, for each direct answer to a question there exists a condition such that the answer is the unique correct answer to the question with respect to the condition. So the relevant Ω-sentence is the unique correct answer to some question with respect to a certain condition. On the other hand, there is no qr-algorithm accessible to **M** which generates the Ω-sentence out of the question and the condition. Now we reason similarly as in the proof of Proposition 1. □

Therefore we get:

Proposition 3 *If a TIG is played in a language which fulfils at least one of the following conditions*:

(1) *the language includes some non-effective ω-question(s),*
(2) *the language permits Ω-sentences, but the set of ω-questions of the language is not r.e.*

then the machine either loses the TIG or loses some extension of the TIG.

Thus, generally speaking, if the game is played in a language that fulfils any of the above conditions, a machine always loses to an interrogator stubborn enough. This is the good news. But there is also the bad news: each machine loses, regardless of whether the machine thinks or not. This is The Trap.

What is the Price of Getting Out From the Trap

At first sight a way out from The Trap seems simple: perform TIG(s) in a language that does not have the properties (1) and (2) specified above. Yet this is a kind of *ad hoc* strategy, and such a strategy is always no good. Moreover, and this is more important, adopting it gives rise to some new difficulties which we are going to point at in this section.

Suppose that a TIG and its extensions are played in a language L^* which, besides the conditions imposed in the section "Turing Interrogative Games" (cf. (5) above), satisfies the following conditions as well: the set of sentences of L^* is recursive, and L^* permits Ω-sentences. L^* can be viewed as a (formalization of a) fragment of a natural language or as a formalized language.

One can prove the following:

Theorem 1 (Wiśniewski and Pogonowski 2010) *Let L be a language such that*: (a) *among expressions of the language there are sentences and interrogatives,* (b) *both sentences and interrogatives of L can be coded by natural numbers, and* (c) *the set of sentences of L is denumerable and recursive. If the following condition holds*:

($\dagger$) *each infinite recursive set of sentences of L is the set of direct answers to some interrogative of L*

then either the set of ω-interrogatives of L is not r.e. or there exists at least one ω-interrogative of L which is not effective.

The above theorem speaks about interrogatives. Yet, in view of the assumptions of the paper we are just referring to, it is perfectly legitimate either to identify questions of L^* with interrogatives of L^* or to claim that there is one-to-one correspondence between questions and interrogatives of the language. Given any of these, Theorem 1 pertains to language L^*. It follows that if language L^* does not have the properties (1) and (2) specified in the antecedent of Proposition 3, it has the following feature:

(††) *At least one infinite recursive set of sentences of L* is not the set of direct answers to any question of L*.*

As a matter of fact, one can prove that there are denumerably many such sets; Theorem 1 is a consequence of a more general result.

Theorem 2 (Wiśniewski and Pogonowski 2010) *Let L be a language such that*: (a) *among expressions of the language there are sentences and interrogatives*, (b) *both sentences and interrogatives of L can be coded by natural numbers*, (c) *the set of sentences of L is denumerable and recursive*, *and* (d) *the set of all effective ω-interrogatives of L is r.e. There exists an infinite family of infinite recursive sets of sentences of L which are not sets of direct answers to any interrogative of L.*

Thus if each ω-question of L^* is effective, and the set of ω-questions of L^* is r.e., the set of all effective ω-questions of L^* is r.e. and therefore there are denumerably many infinite recursive sets of sentences of L^* which are not sets of direct answers to questions of L^*.

A short recur to natural languages is in order now. There are questions whose contents involve plain references to sets of sentences, and whose direct answers are exactly the elements of the sets referred to. For example, consider:

(*) Which formula of the language of Peano Arithmetic is undecidable?
(**) Which English sentence has the same meaning as the Polish sentence 'Śnieg jest biały'?

It seems that a possible and just-sufficient (i.e. direct) answer to (*) is simply a formula of the language of PA; similarly, a direct answer to (**) is an English sentence. In general, constructions to the effect:

(∇) Which ξ is such that ς?

where ξ is a general name referring to a set of sentences, occur, and, in some cases, it seems natural to count as direct answers to a question falling under the schema (∇) exactly the sentences that belong to the set referred to in the question. Moreover, there is no reason for which (general) names referring to infinite recursive sets of sentences should be forbidden to occur in questions of the analyzed kind. A general name referring to a recursive set of sentences can be a complex expression, involving a reference to the coding method used, but this does not change the picture.

TT is supposed to be run in a natural language. If the above analysis is correct, we are justified in saying that the analogue of condition (†) of Theorem 1 holds for a natural language. Clearly natural languages permit Ω-sentences. Thus a natural language (or a part of it) which, in addition, satisfies the assumptions of Theorem 1 displays at least one of the undesired properties, (1) or (2), specified in the antecedent of Proposition 3. The initial assumptions of Theorem 1, in turn, express desired properties of a language of TT. Therefore The Trap is inescapable.

Yet applying logical concepts and theorems to natural languages is always somewhat risky. So let us be more cautious.

We face the following dilemma: either the set of ω-questions of L^* is r.e. and comprises effective questions only, or the opposite holds. In the latter case we are, again, in the situation described in the previous section: each machine has to lose due to the properties of the language and thus we are in The Trap. In the former case none of the conditions, (1) or (2), of Proposition 3, holds, so, in principle (assuming that there are no other obstacles of a similar kind), a machine does not have to lose a TT performed in the form of TIGs. But we are still in a trouble, though a different one. There are denumerably many decidable/recursive sets of sentences which are not sets of direct answers to any question of the language. So far so good: if there is no question, it cannot be used in a game. But the other side of the coin is: there are denumerably many *well-defined issues* that are *a priori* banned from the game. These issues are expressible in some language(s) distinct from L^* by means of questions (of the language(s)) falling under the scheme:

(∇) Which ξ is such that ς?

such that ξ is a "name" of a recursive set of sentences which does not constitute the set of direct answers to any question of L^*, and the corresponding question in the "new" language has ξ as the set of direct answers. A success of a machine is, in principle, possible. But one can reasonably argue that the success does not count, because there are infinitely many well-defined issues which are *a priori* banned from the game and therefore an abductive reasoning from the success in the test to the claim that the tested machine thinks is, at best, premature. On the other hand, an attempt of changing a language of the test so that all the relevant issues could have been addressed would put us into The Trap again.

Conclusions

Let us conclude. The main objective of TT is to differentiate between machines that think and machines that do not: it is presupposed that a thinking machine is able to pass the test, and if a machine passes TT, there are good reasons to believe that it thinks. TT can (although need not) be performed in the form of TIGs. We have shown that, given some conditions, each machine loses a certain TIG, and thus also each thinking machine (in any rational sense of the word). But once a TIG is performed, an instance of TT takes place. So TT does not reach its main objective as long as the conditions hold. The conditions characterize some properties of a language in which a TIG/TT is played/performed. If, however, they do not hold, one can argue that the success of a machine does not constitute a convincing argument for the claim that the machine thinks, because there exist denumerably many well-defined issues which are *a priori* banned from the game.

References

Belnap, N. D., & Steel, T. B. (1976). *The logic of questions and answers*. New Heaven: Yale University Press.

Bradford, P. G., & Wollowski, M. (1995). A formalization of the Turing test. *ACM SIGART Bulletin, 6*(4), 3–10.

Copeland, B. J. (2000). The Turing test. *Mind and Machines, 10*, 519–539.

Copeland, J., & Proudfoot, D. (2009). Turing's test: A philosophical and historical guide. In R. Epstein, G. Roberts, & G. Beber (Eds.), *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer, Chap. 9* (pp. 119–138). New York: Springer.

Epstein, R., Roberts, G., & Beber, G. (Eds.). (2009). *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*. New York: Springer.

Ginzburg, J. (2010). Relevance for dialogue. In P. Łupkowski, & M. Purver (Eds.), *Aspects of semantics and pragmatics of dialogue. SemDial 2010, 14th workshop on the semantics and pragmatics of dialogue* (pp. 121–129). Poznań: Polish Society for Cognitive Science.

Harnish, R. M. (2002). *Minds, brains, computers. An historical introduction to the foundations of cognitive science*. Oxford: Blackwell Publishers.

Harrah, D. (2002). The logic of questions. In D. Gabbay & F. Guenthner (Eds.), *Handbook of philosophical logic* (2nd ed., pp. 1–60). Dodrecht, Boston, London: Kluwer.

Hernandez-Orallo, J. (2000). Beyond the Turing test. *Journal of Logic, Language, and Information, 9*, 447–466.

Moor, J. (1976). An Analysis of the Turing test. *Philosophical Studies, 30*, 249–257.

Newman, A. H., Turing, A. M., Jefferson, G., & Braithwaite, R. B. (1952). Can automatic calculating machines be said to think? Broadcast discussion transmited on BBC (14, 23 Jan 1952). The Turing digital archive (http://www.turingarchive.org), contents of AMT/B/6.

Sato, Y., & Ikegami, T. (2004). Undecidability in the imitation game. *Mind and Machines, 14*, 133–143.

Saygin, A. P., Cicekli, I., & Akman, V. (2001). Turing test: 50 years later. *Mind and Machines, 10*, 463–518.

Shieber, S. M. (2006). Does the Turing test demonstrate intelligence or not? In *Proceedings of the twenty-first national conference on artificial intelligence (AAAI-06)* (pp. 1539–1542).

Shieber, S. M. (2007). The Turing test as interactive proof. *Nous, 4*(41), 686–713.

Stalker, D. F. (1976). Why machines can't think: A reply to James Moor. *Philosophical Studies, 34*, 317–320.

Turing, A. M. (1948). Intelligent machinery. The Turing digital archive (http://www.turingarchive.org), contents of AMT/C/11.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, LIX*(236), 443–455.

Turing, A. M. (1951a) Can digital computers think? The Turing digital archive (http://www.turingarchive.org), contents of AMT/B/5.

Turing, A. M. (1951b) Intelligent machinery, a heretical theory. The Turing digital archive (http://www.turingarchive.org), contents of AMT/B/4.

Turing, A. M. (1953) Digital computers aplied to games. The Turing digital archive (http://www.turingarchive.org), contents of AMT/B/7.

Wiśniewski, A. (1995). *The posing of questions: Logical foundations of erotetic inferences*. Dodrecht, Boston, London: Kluwer.

Wiśniewski, A., & Pogonowski, J. (2010). Interrogatives, recursion, and incompleteness. *Journal of Logic and Computation, 20*(6), 1187–1199.