Το παράδοξο του Jules Richard και οι συνέπειές του

Γιάννης Κιουστελίδης
jqstel@math.ntua.gr

Το πιο παλιό παράδοξο της Λογικής πιθανόν να οφείλεται στον Κρητικό Επιμενίδη, έναν Έλληνα σοφό των αρχών του έκτου αιώνα π.Χ., ο οποίος υποτίθεται ότι είπε τη φράση: «Οι Κρητικοί είναι πάντα ψεύτες» (Αυτή η φράση περιλαμβάνεται στην προς Τίτον επιστολή του Αποστόλου Παύλου (1,12) και πιστεύεται ότι αναφέρεται σε μία ρήση του Επιμενίδη). Εφ' όσον αυτή η δήλωση γίνεται από έναν Κρητικό, ισχυρίζεται την ίδια της τη διάψευση. Έτσι, δεν μπορεί να είναι ούτε αληθινή ούτε εσφαλμένη.

Αυτό σημαίνει ότι η Λογική έχει αποτύχει;

Δεν χρειάζεται να φτάσουμε ως εκεί. Πρέπει απλά να είμαστε πολύ προσεκτικοί με τις «αυτοαναφερόμενες» προτάσεις, δηλαδή τις προτάσεις εκείνες που μιλούν έμμεσα ή άμεσα για τον εαυτό τους.

Τέτοιες προτάσεις εξαιρούνται με κοινή συμφωνία από κάθε συστηματοποίηση των Μαθηματικών και της Λογικής (και ειδικότερα από κάθε φορμαλιστικό σύστημα αξιωμάτων), ώστε να αποφευχθούν διλήμματα του παραπάνω τύπου.

Έτσι, οδηγείται κανείς στη διάκριση μεταξύ μίας γλώσσας και μίας *«μεταγλώσσας».* Μία γλώσσα παράγει προτάσεις, που αναφέρονται σε κάποια περιοχή δραστηριότητας ή σε κάποια περιοχή έρευνας. Μία μεταγλώσσα παράγει προτάσεις σχετικά με τις προτάσεις που παράγονται από κάποια γλώσσα. Είναι η «γλωσσολογία» εκείνης της γλώσσας.

Κατά τον ίδιο τρόπο, πρέπει να διαχωρίσουμε την Αριθμοθεωρία, η οποία είναι μία προσεκτική αξιωματοποίηση και τυποποίηση της Αριθμητικής, από την «μεταθεωρία» της, που έχει ως αντικείμενο όχι τους ίδιους τους ακέραιους αλλά τους τύπους της Αριθμοθεωρίας.

Το τι μπορεί να συμβεί αν ανακατέψουμε θεωρία και μεταθεωρία, φαίνεται καθαρά στην περίπτωση μιας παραλλαγής του φημισμένου παράδοξου του Jules Richard (1905) βλ. [Kac-Ulam, p.137].

Ας εξετάσουμε τις συναρτήσεις μίας μεταβλητής f(n), που είναι ορισμένες πάνω στους θετικούς ακέραιους, δηλαδή για n = 1, 2, 3, … και έχουν για τιμές θετικούς ακέραιους, π. χ. τις n^2, ή 2n+1. Για συντομία θα τις ονομάσουμε «συναρτήσεις ακεραίων». Θα λέμε ότι μία τέτοια συνάρτηση είναι «υπολογίσιμη» αν υπάρχει ένας αλγόριθμος, μία «συνταγή» που περιέχει έναν πεπερασμένο αριθμό αριθμητικών και λογικών πράξεων (προσθέσεων, αφαιρέσεων, πολλαπλασιασμών, διαιρέσεων και π. χ. συναγωγών τύπου «αν … τότε …»), που να επιτρέπει τον υπολογισμό του f(m), της τιμής της f για οποιονδήποτε θετικό ακέραιο m.

Το σύνολο όλων αυτών των αλγορίθμων μπορεί να διαταχθεί σύμφωνα με το μήκος τους και μεταξύ αλγορίθμων ίδιου μήκους, σύμφωνα με κάποια λεξικογραφική ταξινόμηση των γραμμάτων και συμβόλων που χρησιμοποιούνται στον αλγόριθμο.

Έτσι έχουμε την ακολουθία αλγορίθμων $A_1,A_2,A_3,\ldots$ που αντιστοιχεί προς τις συναρτήσεις $f_1,f_2,f_3,\ldots$

Ας εξετάσουμε τώρα την συνάρτηση g που δίνεται από τον τύπο

$g(n) = f_n(n)+1$.

Πρόκειται προφανώς για μία υπολογίσιμη συνάρτηση του παραπάνω τύπου: Για κάθε ειδική τιμή του n προσδιορίζουμε πρώτα τον αλγόριθμο A_n, στην παραπάνω ακολουθία αλγορίθμων, και μετά τον εφαρμόζουμε στον αριθμό n, παίρνοντας έτσι την τιμή $f_n(n)$. Αυξάνοντας αυτήν την τιμή κατά ένα, παίρνουμε την τιμή g(n).[1]

Ανήκει αυτός ο αλγόριθμος στην παραπάνω ακολουθία;

Σύμφωνα με την επιχειρηματολογία μας όφειλε να ανήκει, γιατί όλοι οι αλγόριθμοι που περιγράφουν συναρτήσεις ακεραίων ανήκουν σε αυτήν την ακολουθία. Ας πούμε λοιπόν ότι έχει τη θέση m μέσα σε αυτόν τον κατάλογο, δηλαδή ότι είναι ο αλγόριθμος A_m.

Η συνάρτηση g(n) είναι, επομένως, απλά η συνάρτηση $f_m(n)$ και η τιμή της, όταν το n παίρνει την τιμή m, όφειλε να είναι $f_m(m)$. Σύμφωνα όμως με τον παραπάνω ορισμό:

$g(m) = f_m(m) + 1$.

Αυτή είναι μία προφανής αντίφαση.[2]

Τι λάθος έγινε;

Μετά από μία προσεκτικότερη παρατήρηση, αποκαλύπτεται ότι η επιχειρηματολογία μας στηρίζεται βασικά στην παραπάνω διάταξη όλων των αλγορίθμων, που περιγράφουν συναρτήσεις ακεραίων. Όμως, παρά το ότι οι συναρτήσεις περιγράφονται μέσα στο σύστημα της Αριθμοθεωρίας, η διάταξή τους είναι μία «μεταμαθηματική» διαδικασία.

Εξετάζοντας βαθύτερα, βλέπουμε ότι όλα εξαρτώνται από την υπόθεση ότι υπάρχει μία γενική μέθοδος, ένας αλγόριθμος, με τον οποίο μπορούμε να αποφασίσουμε ποιο κείμενο είναι ένας αλγόριθμος, για τον υπολογισμό συναρτήσεων του παραπάνω τύπου, και ποιο δεν είναι ένας τέτοιος αλγόριθμος.

---

[1] Εδώ μπορεί να αναγνωρίσει κανείς το ίδιο τέχνασμα με το οποίο ο Georg Cantor απέδειξε το 1874 ότι οι πραγματικοί αριθμοί έχουν μη αριθμήσιμο πλήθος, δηλαδή δεν μπορούν να διαταχθούν σαν μία ακολουθία.

[2] Αυτή η απόδειξη μοιάζει πολύ με τον τρόπο με τον οποίο ο Cantor απέδειξε ότι το δυναμοσύνολο ενός απείρου συνόλου A (το σύνολο των υποσυνόλων του) είναι ακόμα πολυπληθέστερο:

Για κάθε άπειρο σύνολο αριθμών, A, το σύνολο των χαρακτηριστικών του συναρτήσεων:

$\chi A=\{f_a(x), a\in A, f_a(a)=1, f_a(x)=0 \text{ για } x\neq a\}$

είναι, προφανώς, ισοπληθές με το A. Εδώ $f_a(x)$, είναι η εκάστοτε συνάρτηση, που για x=a έχει την τιμή 1 και για όλα τα άλλα x έχει την τιμή 0.

Όμως, το σύνολο: $B=\{\forall f: f:A\rightarrow\{0,1\}\}$ *όλων* των συναρτήσεων πάνω στο A που παίρνουν μόνο τιμές 0 ή 1 είναι πολυπληθέστερο από το σύνολο χA των χαρακτηριστικών συναρτήσεων του A, άρα και από το A. Για παράδειγμα, η συνάρτηση $\varphi(x)=1-f_x(x)$ ανήκει στο B αλλά *δεν* μπορεί να ανήκει στο χA. Αν η φ(x) ανήκε στο χA, τότε θα αντιστοιχούσε σε κάποια χαρακτηριστική συνάρτηση $f_b(x)$, δηλαδή θα ήταν $\varphi(x)=f_b(x)$. Άρα, θα είχαμε: $f_b(x)=1-f_x(x)$, και για x=b θα ήταν $f_b(b)=1-f_b(b)$, δηλαδή $f_b(b)=1/2$. Αυτό, όμως, είναι άτοπο. Ως χαρακτηριστική συνάρτηση η $f_b(x)$ θα έπρεπε να έχει την τιμή 1 για x=b.

Αν το B είναι το σύνολο όλων των λογικών προτάσεων που αναφέρονται στο A τότε η f θα μπορούσε να είναι ένα πρόγραμμα αξιολόγησης με απαντήσεις {F,T} (False or True) αντί {0, 1}.

Έτσι, οδηγούμαστε σ' ένα πρώτο αποτέλεσμα μη αποφανσιμότητας (βλ. [Martin Davis 1982, pp.xvi-xviii] ή [Hermes, 1971, pp.143-144]):

«Δεν υπάρχει αλγόριθμος που επιτρέπει σε κάποιον να αποφασίσει πάντα αν ένας υποτιθέμενος αλγόριθμος, π. χ. ένα τυχαίο κομμάτι κειμένου, είναι αλγόριθμος για τον υπολογισμό μίας συνάρτησης ακεραίων.»

Πρέπει εντούτοις να προσέξουμε ότι για ειδικούς αλγόριθμους αυτή η ερώτηση μπορεί να είναι αποφάνσιμη. Χρησιμοποιώντας για κάποιον εξεταζόμενο αλγόριθμο κατάλληλα επιχειρήματα, που δεν μπορούν να εφαρμοστούν σε άλλες περιπτώσεις, μπορεί να είμαστε σε θέση να δείξουμε ότι ο αλγόριθμος αυτός υπολογίζει μία συγκεκριμένη συνάρτηση.

Κατά παρόμοιο τρόπο, μπορεί κανείς να αποδείξει ότι δεν υπάρχει γενική μέθοδος, δεν υπάρχει αλγόριθμος που να μπορεί να μας πει πότε θα σταματήσει ένας υπολογιστής (μία Μηχανή του Turing), που λειτουργεί σύμφωνα μ' ένα τυχαίο «πρόγραμμα» [Hermes, 1971, pp.145-146].

Για να κάνουμε ευκολονόητη αυτή την επιχειρηματολογία, ακολουθήσαμε μία σειρά επιχειρημάτων, χρησιμοποιώντας γενικούς όρους της καθομιλουμένης γλώσσας αντί για «αυστηρές» μαθηματικές έννοιες. Μία πιο προσεκτική συμπερασματολογία θα όριζε με ακρίβεια την έννοια του αλγορίθμου. Εντούτοις, η ίδια επιχειρηματολογία μπορεί να εφαρμοσθεί και σε πιο αυστηρά καθορισμένες έννοιες.

ΠΑΡΑΠΟΜΠΕΣ

Davis, Martin: Computability and Unsolvability, Dover, New York, 1982.

Hermes, Hans: Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit, Springer Verlag, Berlin, Heidelberg, New York, 1971.

Kac, Mark and Ulam, Stanislaw: Mathematics and Logik, Dover, New York, 1992.

Τι είναι αλγόριθμος και πως κωδικοποιείται αριθμητικά

«Αλγόριθμο» ονομάζουμε μία διαδικασία, μία σειρά από αριθμητικές πράξεις και στοιχειώδεις λογικές κρίσεις που την εφαρμόζουμε σε κάποια αρχικά δεδομένα. Αν ο αλγόριθμος είναι καλά σχεδιασμένος τότε το αποτέλεσμα της εκτέλεσής του είναι η λύση κάποιου λογικού ή αριθμητικού προβλήματος.

Οι αριθμητικές πράξεις είναι οι γνωστές μας πρόσθεση, αφαίρεση, πολλαπλασιασμός και διαίρεση, που συμβολίζονται αντίστοιχα με +, -, x, /.

Οι στοιχειώδεις λογικές κρίσεις είναι του τύπου:

Αν ισχύει το Α τότε κάνε το Β. Συμβολικά, Α→Β (το Α συνεπάγεται το Β). Στην συμβολική καταγραφή τέτοιων εντολών χρησιμοποιούνται τα ακόλουθα σύμβολα:

~ (άρνηση). Ο τύπος '~Α' σημαίνει 'δεν ισχύει το Α'.

∨ (αντιδιαστολή). Ο τύπος 'Α∨Β' σημαίνει 'Α ή Β'.

∧ ή & (σύνθεση). Ο τύπος 'Α∧Β' ή 'Α&Β' σημαίνει '(ταυτοχρόνως) Α και Β'.

→ (συνεπαγωγή). Ο τύπος 'Α→Β' σημαίνει 'το Α συνεπάγεται το Β'.

∃ (υπάρχει). Ο τύπος '∃Α' σημαίνει 'υπάρχει Α (τέτοιο ώστε)'.

∀ (για όλα). Ο τύπος '∀Χ' σημαίνει 'για όλα τα Χ'.

Αυτά τα σύμβολα μαζί με τα αριθμητικά σύμβολα 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 αρκούν για να διατυπωθεί συμβολικά κάθε αριθμητική μέθοδος, κάθε αλγόριθμος που θέλουμε να εκτελέσουμε σε έναν ηλεκτρονικό υπολογιστή (Η/Υ).

Επειδή μία συμβολοσειρά όπως η '~Α∨Β→Γ' είναι όμως δυσνόητη, γι' αυτό τον αλγόριθμο τον περιγράφουμε συνήθως με απλές αγγλικές λέξεις. Π.χ. «If not A or B, then Γ». Διατυπώσεις αυτού του είδους τις ονομάζουμε «προγράμματα», δεν είναι όμως παρά αναδιατυπώσεις ενός αλγορίθμου σε άλλη γλωσσική μορφή. Γι' αυτό χρησιμοποιούμε εδώ εναλλακτικά τους όρους «αλγόριθμος» ή «πρόγραμμα». Αντιστοίχως, «αλγοριθμική επίλυση» σημαίνει «επίλυση με σαφώς διατυπωμένη σειρά στοιχειωδών αριθμητικών και λογικών διαδικασιών», διαδικασιών όπως αυτές που περιγράψαμε.

Όσο και αν φαίνεται πολύπλοκος και πολύπλευρός, ένας Η/Υ δεν είναι παρά μία αυτοματοποιημένη αριθμομηχανή. Γι' αυτό όλα τα προγράμματα εισάγονται σε αυτόν κωδικοποιημένα αριθμητικώς. Σε κάθε ένα από τα παραπάνω σύμβολα αντιστοιχεί ένας αριθμός και σε μία συμβολοσειρά αντιστοιχεί μία σειρά αριθμητικών συμβόλων. Έτσι όλοι οι κώδικες δεν είναι παρά μεγάλοι (κατάλληλα κατασκευασμένοι) ακέραιοι αριθμοί.

Πώς συνδυάζονται στοιχειώδεις κωδικοί αριθμοί σε μεγαλύτερες συμβολοσειρές;

Θα δώσουμε εδώ ένα παράδειγμα κωδικοποίησης που δείχνει ότι οι βασικές αρχές κατασκευής κωδίκων είναι πολύ απλές. Το παράδειγμά μας δεν έχει όμως άμεση σχέση με τον τρόπο που κωδικοποιούνται προγράμματα και αριθμητικά δεδομένα μέσα σε έναν Η/Υ, δεδομένου ότι εκεί χρησιμοποιείται η δυαδική παράσταση αριθμών.

Σε κάθε ένα σύμβολο από αυτά με τα οποία περιγράφουμε τα δεδομένα μας αντιστοιχούμε εδώ έναν φυσικό αριθμό που δεν περιέχει το μηδέν (0).

Για παράδειγμα, μπορούμε να ορίσουμε την εξής αντιστοιχία των συμβόλων που αναφέραμε παραπάνω με κωδικούς:

Σύμβολα:	1	2 … 8	9	→	~	∨	&	∃	∀	Α	Β	Γ
Κωδικοί:	2	3 … 9	11	12	13	14	15	16	17	18	19	21

Έτσι λοιπόν στα σύμβολα Α, →, Β αντιστοιχούν οι κωδικοί 18, 12, 19. Για να κωδικοποιήσουμε τώρα μία σειρά συμβόλων γράφουμε κατά σειράν τους κωδικούς τους διαχωρίζοντάς τους με 0 ώστε να μπορούμε να κάνουμε ευχερώς και μονοσήμαντη αποκωδικοποίηση. Έτσι, στην συμβολοσειρά 'Α→Β' αντιστοιχούμε τον κωδικό 18012019.

Γενικά, τον κωδικό μίας παράστασης Γ, τον παριστάνουμε με <Γ>. Έτσι την παραπάνω περιγραφή μπορούμε να την εκφράσουμε συντομότερα γράφοντας:

Αν <Α>=18 , <→>=12, <Β>=19 τότε <Α→Β>=18012019.

Παρομοίως, με βάση την παραπάνω κωδικοποίηση είναι :

<Α∨Β→Γ>= 18014019012021.

Έτσι έχουμε για κάθε συμβολοσειρά, όπως αυτές που περιγράφουν αλγορίθμους (συστηματικές μεθόδους υπολογισμού) απλά τεράστιους ακέραιους θετικούς αριθμούς όπως ο 18012019 και μπορούμε να μιλήσουμε για τις ιδιότητες αλγορίθμων (τις ιδιότητες συμβολοσειρών) εξετάζοντας τις ιδιότητες τέτοιων μεγάλων ακεραίων.

Αυτοί οι αριθμητικοί κωδικοί αριθμοί καταγράφονται στην μνήμη του Η/Υ και υφίστανται επεξεργασία σε δυαδική μορφή, δηλαδή με χρήση μόνο των αριθμητικών συμβόλων 0 και 1, όπως στον κώδικα τηλεγραφίας του Morse χρησιμοποιούνται μόνο τελείες και παύλες (που αντιστοιχούσαν σε βραχέα και μακρά ηλεκτρικά σήματα του τηλέγραφου) για να καταγράψουμε οποιοδήποτε μήνυμα.

ΠΩΣ ΔΙΑΤΥΠΩΣΕ ΤΟ ΠΑΡΑΔΟΞΟ ΤΟΥ Ο RICHARD

Η παραπάνω παρουσίαση οφείλεται στους Kac και Ulam που είναι επηρεασμένοι από τον Turing. Ο Richard δεν αναφέρεται σε αλγορίθμους αλλά σε απλές φράσεις που ορίζουν όχι ακέραιους αλλά πραγματικούς αριθμούς. Βλέπε:

http://www-history.mcs.st-and.ac.uk/Biographies/Richard_Jules.html

Το παράδοξο εμφανίστηκε για πρώτη φορά σε μια επιστολή του στον Louis Olivier, διευθυντή της *Revue générale des sciences pures et appliquées*.

Ο Jules Richard (Ζυλ Ρισάρ) αναφέρεται στο σύνολο εκείνων των πραγματικών αριθμών που μπορούν να οριστούν με έναν πεπερασμένο αριθμό των λέξεων.

Παραδείγματα λεκτικής περιγραφής πραγματικών αριθμών είναι «ένα τρίτο τρίτο», «η βάση των φυσικών λογαρίθμων» και «η αναλογία της περιφέρειας ενός κύκλου προς τη διάμετρό», κλπ. Όσο πολλές και αν είναι αυτές οι εκφράσεις μπορούν να γραφούν ημι-λεξικογραφικά με τον εξής τρόπο: Πρώτα τις κατατάσσουμε ανάλογα με το μήκος τους (τον αριθμό συμβόλων που τις αποτελούν) και αυτές που έχουν ίδιο μήκος τις καταχωρούμε στην λίστα μας λεξικογραφικά, όπως καταχωρούνται οι λέξεις σε ένα λεξικό.

Για ευκολία, ο Richard, εξετάζει τώρα τον κατάλογο όλων των πραγματικών αριθμών που περιγράφονται με αυτές τις εκφράσεις, που βρίσκονται μεταξύ 0 και 1. Δηλαδή μελετά μίαν ακολουθία πραγματικών αριθμών: r_1, r_2,.... με $0<r_k<1$. Ορίζει, τώρα ένα νέο πραγματικό αριθμό r ως εξής: Το ακέραιο μέρος του r είναι 0, το νιοστό δεκαδικό ψηφίο του r είναι 1 αν το νιοστό δεκαδικό ψηφίο του r_v δεν είναι 1, και το νιοστό δεκαδικό ψηφίο του r είναι 2 εάν το νιοστό δεκαδικό ψηφίο του r_v είναι 1.

Οι προηγούμενες δύο παράγραφοι είναι όμως μια λεκτική έκφραση που ορίζει μονοσήμαντα έναν πραγματικό αριθμό r. Έτσι ο r πρέπει να είναι ένας από τους αριθμούς r_v. Ωστόσο, ο r κατασκευάστηκε έτσι ώστε δεν ισούται με κανέναν από τους r_v. (*r* is an undefinable number). Αυτό είναι μία παράδοξη αντίφαση.

Το παράδοξο δημοσιεύτηκε στο περιοδικό *Les Principes des mathématiques et le problème des ensembles* (1905).

Οι Αριθμοί του Richard ή Ριχαρδιανοί αριθμοί

Μια παραλλαγή του παραδόξου εξετάζει ακέραιους αριθμούς αντί για πραγματικό-αριθμούς διατηρώντας τον αυτοαναφορικό χαρακτήρα:

Καταχωρούμε όλες τις εκφράσεις που αναφέρονται σε φυσικούς αριθμούς σε έναν κατάλογο σύμφωνα με το μήκος τους και για ισομήκεις αλφαβητικά. Στην συνέχεια τους αριθμούμε έτσι ώστε στην πρώτη έκφραση του καταλόγου να αντιστοιχεί ο αριθμός 1, στην επόμενη να αντιστοιχεί ο 2, και ούτω καθεξής. Δεδομένου ότι κάθε λεκτική έκφραση αντιστοιχεί σε έναν μοναδικό ακέραιο αριθμό είναι δυνατόν σε μερικές περιπτώσεις ο αριθμός που αντιστοιχεί σε μία έκφραση να έχει την ιδιότητα στην οποία αναφέρεται η έκφραση αυτή.

Αν, για παράδειγμα, η έκφραση «πρώτος αριθμός» (δηλαδή αριθμός διαιρετός μόνο από τον εαυτό του και το 1) έτυχε να είναι 43η, τότε ο 43 έχει την ιδιότητα στην οποία αναφέρεται η έκφραση. Αντιθέτως, αν η έκφραση «άρτιος αριθμός» (αριθμός διαιρετός δια 2) είναι 23η στον κατάλογο προφανώς δεν έχει την ιδιότητα για την οποία μιλά. Τότε λέμε ότι ο 23 είναι «Ριχαρδιανός αριθμός».

Ονομάζουμε έναν θετικό ακέραιο αριθμό, «αριθμό τύπου Richard ή Ριχαρδιανό», εάν η λεκτική έκφραση που καταλογογραφείται με αυτόν τον αριθμό αναφέρεται σε ιδιότητα που ο αριθμός αυτός ΔΕΝ έχει.

Πιο σωστά, η φράση «ο x είναι τύπου Richard» σημαίνει « ο x ΔΕΝ έχει την ιδιότητα που ορίζεται από την έκφραση στην θέση x του καταλόγου μας». Έτσι, στα παραπάνω παραδείγματα, ο 43 ΔΕΝ είναι τύπου Richard ενώ ο 23 είναι.

Επειδή η ιδιότητα να είναι ένας αριθμός τύπου Richard είναι μια αριθμητική ιδιότητα των ακεραίων, ανήκει στη λίστα όλων των ορισμών ιδιοτήτων.

Συνεπώς, στην ιδιότητα να είναι κάποιος ακέραιος τύπου Richard αντιστοιχεί κάποιος ακέραιος, v. Για παράδειγμα, ο χαρακτηρισμός «είναι τύπου Richard» μπορεί να αντιστοιχεί στον αριθμό m. Τότε τίθεται το ερώτημα: Είναι ο m τύπου Richard;

Ας υποθέσουμε ότι ο m είναι τύπου Richard. Τότε με βάση τον ορισμό <u>δεν</u> πρέπει να έχει την ιδιότητα που ορίζεται από την έκφραση με αριθμό m, άρα δεν είναι τύπου Richard. Οδηγούμαστε, δηλαδή σε αντίφαση. Ας υποθέσουμε, λοιπόν, ότι ο m δεν είναι τύπου Richard, τότε, με βάση τον ορισμό, ο m πρέπει να έχει την ιδιότητα στην οποία αναφέρεται ο ορισμός με αριθμό m. Αυτό σημαίνει ότι ο m είναι τύπου Richard, συμπέρασμα που είναι επίσης αντίφαση. Έτσι, η δήλωση «ο m είναι τύπου Richard» δεν είναι δυνατό να χαρακτηρισθεί με συνέπεια ούτε ως αληθής ούτε ως ψευδής.

Η μη επιλυσιμότητα του προβλήματος «Τερματισμού».

Μία απόδειξη με τον τρόπο του Jules Richard μέσω κωδικοποίησης τύπου Gödel.

Στη θεωρία υπολογισμού, το πρόβλημα τερματισμού μπορεί να οριστεί ως εξής: "*Δοθέντος ενός τυχαίου υπολογιστικού προγράμματος να διαπιστωθεί αν το πρόγραμμα θα σταματήσει να τρέχει ή αν θα συνεχίσει να τρέχει για πάντα*".

Ο Alan Turing απέδειξε το 1936 ότι δεν μπορεί να υπάρξει ένας γενικός αλγόριθμος για την λύση του προβλήματος τερματισμού για όλα τα δυνατά ζεύγη (p,x) όπου p ένα πρόγραμμα και x τα δεδομένα εισόδου σε έναν Η/Υ. Ένα βασικό στοιχείο της απόδειξης ήταν η λεπτομερής περιγραφή μίας ανύπαρκτης τότε κατάλληλα προγραμματιζόμενης αριθμομηχανής που ονομάσθηκε αργότερα «μηχανή του Turing». Μία τέτοια μηχανή είναι κάθε ηλεκτρονικός Υπολογιστής (Η/Υ).

Το πρόβλημα αυτό φαίνεται αρχικά όχι πολύ ενδιαφέρον έχει όμως ισοδύναμες μορφές με ευρεία εφαρμογή. Ισοδύναμο με αυτό το πρόβλημα μπορούμε να θεωρήσουμε το ερώτημα αν οποιοδήποτε πρόβλημα επιλύεται αλγοριθμικά ή όχι. Αν υπάρχει τέτοιος αλγόριθμος, μπορούμε να προγραμματίσουμε τον υπολογιστή να τον εκτελεί και αφού εκτυπώσει την απάντηση να σταματά. Η πρόβλεψη του σταματήματος είναι έτσι ισοδύναμη με την πρόβλεψη του ότι ένα πρόβλημα είναι αλγοριθμικά επιλύσιμο.

Άλλο ισοδύναμο πρόβλημα είναι το αν υπάρχει αλγόριθμος (πρόγραμμα) που να διαπιστώνει αυτομάτως αν σε ένα δοσμένο πρόγραμμα υπάρχουν προγραμματιστικά λάθη. Πάλι, μπορούμε να κατασκευάσουμε τον αλγόριθμο έτσι ώστε η εκτέλεσή του να σταματά όταν διαπιστωθεί η ύπαρξη λάθους στο εξεταζόμενο πρόγραμμα.

Θα δούμε εδώ ότι, ενώ σε ειδικές περιπτώσεις αυτά τα ερωτήματα μπορούν να απαντηθούν, δεν υπάρχει γενικός αλγόριθμος, δηλαδή γενική μέθοδος (συστηματικός τρόπος), που να μπορεί να τα απαντήσει πάντοτε:

Έστω ότι p(x) είναι η έξοδος του προγράμματος p όταν εφαρμόζεται σε κάποια αριθμητική είσοδο x. Η είσοδος x μπορεί όμως να είναι επίσης ο δυαδικός κώδικας που χρησιμοποιεί ο Η/Υ για οποιασδήποτε συμβολοσειρά με δεδομένα εισόδου, δηλαδή η αριθμοποίησή τους. Οποιαδήποτε μη αριθμητικά δεδομένα αντικαθίστανται έτσι από έναν κωδικό αριθμό που είναι θετικός ακέραιος δοσμένος σε δυαδική μορφή.

Σε κάθε πρόγραμμα p μπορούμε παρομοίως να αντιστοιχήσουμε κάποιαν αριθμητική κωδικοποίησή του, π.χ. τον δυαδικό κώδικα #p με τον οποίο παριστάνεται στον Η/Υ.[3]

Τότε μπορούμε να ταξινομήσουμε όλα τα δυνατά προγράμματα με αύξουσα τάξη του μεγέθους αυτών των κωδικών αριθμών που είναι θετικοί ακέραιοι. Έτσι έχουμε την ακολουθία $P = \{p_n, n \in G\}$, όπου G είναι το σύνολο των κωδικών αριθμών όλων των προγραμμάτων.

[3] Τον κωδικό αριθμό αυτόν θα τον ονομάζουμε «αριθμό Gödel του προγράμματος» προς τιμήν του Gödel που πρώτος χρησιμοποίησε αριθμητική κωδικοποίηση για λογικούς τύπους, δηλαδή φορμαλιστικές εκφράσεις των Μαθηματικών (της Αριθμοθεωρίας). Η κωδικοποίηση που χρησιμοποιούμε εδώ δεν είναι όμως ο δικός του τρόπο κωδικοποίησης μαθηματικών εκφράσεων. Εξάλλου δεν αλλάζει τίποτα στους συλλογισμούς μας αν χρησιμοποιηθεί οποιοδήποτε αριθμητική κωδικοποίηση αρκεί να μπορεί να αποκωδικοποιείται μονοσήμαντα.

Το ερώτημα είναι έτσι: Υπάρχει γενικό πρόγραμμα (γενικός αλγόριθμος) που να προβλέπει για κάθε πρόγραμμα p_n και οποιαδήποτε δεδομένα εισόδου x το αποτέλεσμα $p_n(x)$;

Έστω ότι υπάρχει ένα τέτοιο πρόγραμμα, R, που σταματάει μόνο όταν έχει υπολογίσει τον αριθμό $p_n(x)$.

Εξετάζουμε τώρα την αριθμητική ακολουθία $p_n(n)$, δηλαδή $p_{\#p}(\#p)$ και ορίζουμε ως g το πρόγραμμα το οποίο δίνει αποτέλεσμα $p_x(x) + 1$ όταν έχει ως δεδομένο εισόδου τον αριθμό x: $g(x)=p_x(x)+1$

Αυτό είναι ένα αποδεκτό, δηλαδή γνήσιο, πρόγραμμα, δεδομένου ότι ακολουθεί πολύ απλούς κανόνες: Αν δοθεί ένας δυαδικός κώδικας n, και δεδομένα εισόδου x, πρώτα προσδιόρισε το πρόγραμμα p με αυτόν τον κωδικό αριθμό (x = #p) και στη συνέχεια υπολόγισε τον αριθμό $p_x(x) + 1$. Επομένως το g ανήκει στην παραπάνω ακολουθία P και έχει έναν κωδικό αριθμό, έναν αριθμό Gödel m = #g.

Αλλά τι συμβαίνει εάν εισαγάγουμε σε αυτό τον δικό του κωδικό αριθμό; Το αποτέλεσμα στην έξοδο πρέπει να είναι g(#g). Αλλά σύμφωνα με τον ορισμό του g θα πρέπει επίσης να είναι g (#g) + 1. Καταλήξαμε έτσι σε μιαν αντίφαση.

Ο μόνο τρόπος για να αποφευχθεί τέτοια αντίφαση είναι να συμπεράνουμε ότι τέτοιο πρόγραμμα, g, δεν μπορεί να υπάρξει, δηλαδή, ότι δεν υπάρχει κανένας γενικός αλγόριθμος ο οποίος να προβλέπει την έξοδο ενός προγράμματος.

Το πρόβλημα Τερματισμού είναι αλγοριθμικά μη-αποκρίσιμο {αναποκρίσιμο} ή μη-αποφάνσιμο!

Έμμεσα υποτίθεται στην επιχειρηματολογία αυτήν ότι υπάρχει ένα πρόγραμμα, g, (μια «καθολική» μηχανή Turing) που μπορεί να προσομοιώσει οποιοδήποτε άλλο πρόγραμμα p. Ένα τέτοιο πρόγραμμα υπάρχει πράγματι, επειδή ένα πρόγραμμα είναι απλώς ένας αλγόριθμος, μια ακολουθία από στοιχειώδεις αριθμητικές και λογικές πράξεις, δηλαδή, εκείνες της Αριθμητικής και της Προτασιακής λογικής: +, -, *, /, ∧, ∨, →. Οποιοδήποτε προγραμματιζόμενος υπολογιστής μπορεί να ακολουθήσει τέτοιες οδηγίες: Με δεδομένα εισόδου (#p, x) που αποτελούνται από έναν αριθμό Gödel #p και έναν τυχαίο φυσικό αριθμό x, πρώτα αποκωδικοποίησε αυτόν τον αριθμό Gödel για να μάθεις ποιες είναι οι οδηγίες του προγράμματος p. Στη συνέχεια εφάρμοσε αυτές τις οδηγίες στον x για να πάρεις τον p(x).

Η δυσκολία φαίνεται λοιπόν να προκύπτει και πάλι από το ότι δεν ξέρουμε τι ακριβώς είναι ένα εύλογο, ένα σωστά διατυπωμένο πρόγραμμα. Η συντακτικά ορθή διατύπωσή του δεν αρκεί. Μπορεί, για παράδειγμα, να περιέχει φαύλους κύκλους. Δεν μπορούμε να ορίσουμε με ακρίβεια τι είναι ένα πρόγραμμα. Έτσι αποτυγχάνει και η προσπάθεια ταξινόμησης των εύλογων προγραμμάτων.

Σημειώστε επίσης ότι για την απόδειξη αυτή δεν έχει σημασία τι είναι η έξοδος p(x) του προγράμματος p για κάθε τυχαία είσοδο x. Μας ενδιαφέρει μόνο η p(#p).

Τα θεωρήματα του Τούρινγκ

Ο Turing θέλησε να μελετήσει συστηματικά τις ιδιότητες αλγορίθμων (τυποποιημένες αριθμητικές- μαθηματικές διαδικασίες). Για να το κάνει αυτό

φαντάστηκε μία μηχανή που θα εκτελεί κάθε είδους αλγορίθμους. Η μηχανή αυτή επεξεργάζεται σύμβολα και τα καταγράφει σε μίαν ατέρμονα ταινία τηλέτυπου ή τηλέγραφου που κινείται κάθε φορά κατά μία θέση εμπρός ή πίσω. Αυτή είναι η μνήμη της μηχανής. Η επεξεργασία γίνεται με εσωτερικές εντολές-οδηγίες γραμμένες και αυτές συμβολικά. Η μηχανή που φαντάστηκε είναι ουσιαστικά αυτό που ονομάζουμε σήμερα ηλεκτρονικό υπολογιστή, ενώ η ταινία εγγραφών αντικαταστάθηκε σήμερα από μαγνητικές μνήμες.

Πρώτο αποτέλεσμα της μελέτης του Turing είναι ότι μπορεί να υπάρξει μία «καθολικής μηχανή»: μία μηχανή που προσομοιώσει οποιοδήποτε άλλη τέτοια μηχανή αν της δοθούν οι εσωτερικές εντολές και τα εξωτερικά εισαγόμενα δεδομένα της μηχανής που ζητούμε να απομιμηθούμε. Με άλλα λόγια, *μπορούν να υπάρξουν Οικουμενικοί προγραμματιζόμενοι υπολογιστές (Η/Υ)*.

Σημειώστε ότι μέσα σε έναν Η/Υ τόσο τα εισαγόμενα δεδομένα (γράμματα και οι αριθμοί) όσο και το ίδιο το πρόγραμμα είναι κωδικοποιημένα συνήθως ως δυαδικοί αριθμοί δηλαδή με χρήση μόνο των συμβόλων 0 και 1, όπως τα σήματα Morse κωδικοποιούνται με τελείες και παύλες. Έτσι τα πάντα είναι αριθμοποιημένα και εκτελούνται ως αριθμητικές πράξεις.

Ο Η/Υ (και ισοδύναμα μία μηχανή του Turing) είναι απλά μια μεγάλη αυτοματοποιημένη αριθμομηχανή.

Αυτή η αριθμητική κωδικοποίηση όλων των προγραμμάτων σημαίνει όμως ότι μπορούμε να τα διατάξουμε σε μίαν ακολουθία με βάση το μέγεθος του κωδικού τους αριθμού (μεταφρασμένου π.χ. σε δεκαδική μορφή). Παρομοίως, αφού τα δεδομένα εισόδου κωδικοποιούνται επίσης αριθμητικά, μπορούμε να καταγράψουμε τα αποτελέσματα εφαρμογής όλων των αλγορίθμων σε τυχόντα δεδομένα εισόδου σε έναν πίνακα T(j,k) j,k=1,2,3,… όπου j ο κωδικός εκάστου αλγορίθμου και k ο κωδικός κάποιων δεδομένων εισόδου.

Αυτά τα αποτελέσματα της εκτέλεσης αλγορίθμων μπορούν να γραφούν και στην μορφή: $T_j(k)$, όπου T_j είναι ο κατά σειράν j αλγόριθμος και k τα εισαγόμενα δεδομένα (σε αριθμητική μορφή- κωδικοποίηση).

Τα συμπεράσματα των Turing και Gödel προκύπτουν (όπως και του Jules Richard) με εξέταση των στοιχείων της διαγωνίου του πίνακα T(j,k) ή $T_j(k)$ όπου j,k=1,2,3,…

Κεντρικό αποτέλεσμα της μελέτης προγραμματιζόμενων αριθμομηχανών από τον Turing ήταν ότι το ερώτημα πότε σταματά μία τέτοια μηχανή (πότε τερματίζει την εκτέλεση του προγράμματός της) δεν είναι αλγοριθμικά αποκρίσιμο.

Ακριβέστερα, το πρόβλημα τερματισμού (σε σύγχρονη παράφραση) είναι το εξής:

Υπάρχει άραγε γενική μέθοδος , γενικός αλγόριθμος, για να διαπιστώσουμε αν ένα τυχαίο πρόγραμμα Η/Υ, με τυχόντα δεδομένα εισόδου θα τερματιστεί ή όχι;

Παρά την κοινοτυπία του, το πρόβλημα είναι σημαντικό γιατί ισοδυναμεί με γενική μέθοδο επίλυσης προβλημάτων. Αν είχαμε ένα τέτοιο πρόγραμμα ελέγχου του σταματήματος τότε για κάθε αριθμητικό πρόβλημα θα μπορούσαμε να αναζητήσουμε συστηματικά τον αλγόριθμο που τερματίζει αφού βρει την λύση του προβλήματος.

Ο Turing απέδειξε λοιπόν ότι δεν υπάρχει κανένα πρόγραμμα κανένας αλγόριθμος (καμία συστηματική μέθοδος) που να λύνει το πρόβλημα τερματισμού.

Για να το αποδείξει αυτό δέχεται κατ' αρχήν υποθετικά ότι υπάρχει ένα τέτοιο πρόγραμμα Η. Το πρόγραμμα Η μπορεί να τροποποιηθεί έτσι ώστε να προκύψει ένα νέο πρόγραμμα G που[4] *κάνει τα αντίθετα από ότι το Η (απαντά με αντίθετο τρόπο από ότι το Η)*. Έτσι το G κάνει τα εξής:

Όταν εισαχθεί στο G ο κωδικός <Q> ενός τυχαίου άλλου προγράμματος, Q, και ζητήσουμε από το G να βρει το αποτέλεσμα της εφαρμογής του Q πάνω στον κωδικό του αριθμό, <Q>, τότε το G:

1. *Δεν* σταματά αν το Q σταματά όταν εισαγάγουμε τον δικό του κωδικό <Q> ως είσοδο.
   Q(<Q>) σταματά ⇔ G(<Q>) ΔΕΝ σταματά.
2. *Σταματά* (τερματίζει) εάν το Q δεν σταματά όταν του δοθεί ο δικός του κωδικός ως είσοδος.
   Q(<Q>) δεν σταματά ⇔ G(<Q>) σταματά.
   Όπου εδώ το σύμβολο '⇔' σημαίνει ισοδυναμία.

Τώρα ο Turing παρατηρεί ότι οδηγούμαστε σε λογικό αδιέξοδο αν δώσουμε στο G τον δικό του κωδικό ως είσοδο. Από τις παραπάνω παραδοχές το G θα τρέχει για πάντα αν σταματά, ή σταματά αν τρέχει για πάντα. Έτσι το G (και συνεπακόλουθα το Η) δεν μπορεί να υπάρξουν.

Αν βάλουμε στις παραπάνω αντιστοιχίες Q=G τότε προκύπτει:

G(<G>) σταματά ⇔ G(<G>) δεν σταματά.

πράγμα που είναι άτοπο. Άρα μία τέτοια μηχανή, G, και μία αντίστοιχη, Η, δεν μπορούν να υπάρξουν. Αυτό σημαίνει ότι δεν υπάρχει *γενικά εφαρμόσιμη* συστηματική μέθοδος (αλγόριθμος) που να απαντά πάντοτε το πρόβλημα τερματισμού. Για ειδικές περιπτώσεις όμως ίσως μπορούμε να το απαντήσουμε μελετώντας την ειδική δομή του προγράμματος.

Από αυτό το αποτέλεσμα του Turing προκύπτει με εις άτοπον απαγωγή και η μη πληρότητα της Αριθμητικής Peano που ανακάλυψε ο Gödel:

Ας υποθέσουμε ότι το θεώρημα μη-πληρότητας ήταν ψευδές, δηλαδή, ότι υπήρχε ένα συνεπές και υπολογίσιμο αποδεικτικό σύστημα F με βάση το οποίο οποιαδήποτε πρόταση σχετικά με ακέραιους αριθμούς θα μπορούσε είτε να αποδειχθεί ή να διαψευστεί. Αν η Αριθμητική ήταν πλήρης τότε θα μπορούσαμε απλά να εξετάσουμε κάθε δυνατή απόδειξη στο F, μέχρι να βρούμε μια απόδειξη ότι ένα τυχόν πρόγραμμα είτε δεν θα σταματήσει είτε θα σταματήσει. Αυτό είναι δυνατό επειδή τη δήλωση ότι ένα συγκεκριμένο πρόγραμμα υπολογιστή σταματά τελικά είναι απλώς μια δήλωση σχετικά με ακέραιους. Αλλά αυτό θα μας δώσει έναν αλγόριθμο για να λύσει το πρόβλημα τερματισμού που ήδη γνωρίζουμε ότι είναι αδύνατο. Επομένως το F δεν μπορεί να υπάρξει.

Με προσεκτικότερη εξέταση, μπορούμε να πάρουμε ένα ισχυρότερο αποτέλεσμα. Όπως στην απόδειξη του Τούρινγκ, υποθέτουμε ότι τροποποιούμε το Η για να δημιουργήσουμε ένα νέο πρόγραμμα G που

[4] (όπως οι έφηβοι κάνουν συχνά το αντίθετο από ότι λένε οι γονείς τους)

1. Τρέχει για πάντα αν αποδεικνύεται ότι το Q σταματά όταν του δοθεί ως είσοδος ο δικός του κωδικός ή
2. Σταματά αν αποδεικνύεται ότι το Q τρέχει για πάντα. όταν του δοθεί ως είσοδος ο δικός του κωδικός.

Τώρα καταλήγουμε στο συμπέρασμα ότι το G(<G>) σταματά όταν βρει μίαν απόδειξη του ότι δεν σταματά, και δεν σταματά όταν βρει μίαν απόδειξη του ότι σταματά.

Μία υποτιθέμενη απόδειξη του ότι το G(<G>) τρέχει για πάντα, γίνεται βέβαια με μια κρυφή παραδοχή: ότι το σύστημα αξιωμάτων F μέσα στο οποίο διεξάγουμε τις αποδείξεις είναι συνεπές. Αν το F είναι ασυνεπές τότε, σύμφωνα με μία πρόταση της συμβολικής λογικής, μπορούμε να αποδείξουμε μέσα σε αυτό οτιδήποτε[5].

Αλλά αυτό σημαίνει ότι, αν η θεωρία F μπορούσε να αποδείξει ότι η F είναι συνεπής, τότε θα μπορούσε επίσης να αποδείξει ότι το G(<G>) τρέχει για πάντα καταλήγοντας σε αυτοδιάψευση. Το μόνο δυνατό συμπέρασμα είναι ότι αν η F είναι συνεπής, τότε η F δεν μπορεί να αποδείξει τη δική της συνέπεια.

Αυτό το αποτέλεσμα ονομάζεται: Δεύτερο θεώρημα μη-πληρότητας του Gödel.

Το δεύτερο θεώρημα μη-πληρότητας τεκμηριώνει ότι μαθηματικές θεωρίες ικανές να αποδείξουν τη δική τους συνέπεια, είναι μόνο αυτές που δεν έχουν συνέπεια! Αν θέλουμε να αποδείξουμε ότι μια θεωρία F είναι συνεπής, τότε μπορούμε να το κάνουμε μόνο μέσα σε μία πιο ισχυρή θεωρία. Ένα τετριμμένο παράδειγμα είναι η θεωρία F+Con(F) (F συν το αξίωμα ότι η F είναι συνεπής). {Όπου con σημαίνει consistent, δηλαδή συνεπής}

Αλλά τότε πώς ξέρουμε ότι η F+Con(F) είναι το ίδιο συνεπής;

Αυτό μπορούμε να το αποδείξουμε μέσα σε μια ακόμα ισχυρότερη θεωρία: F+Con(F)+Con(F+Con(F)) (F+Con(F) συν το αξίωμα ότι η F+Con(F) είναι συνεπής). Και ούτω καθεξής.

Για να πάρουμε ένα συγκεκριμένο παράδειγμα: το δεύτερο θεώρημα μη-πληρότητας μας λέει ότι το πιο δημοφιλές σύστημα αξιωμάτων για τους ακέραιους αριθμητική του Πεάνο, δεν μπορεί να αποδείξει τη δική του συνέπεια. Ή σε σύμβολα, PA δεν μπορεί να αποδείξει Con(PA). Αν θέλουμε να αποδείξουμε την Con(PA), τότε πρέπει να κινηθούμε σε ένα ισχυρότερο σύστημα αξίωμα, όπως το ZF (τα αξιώματα Ζερμέλο-Φρένκελ για τη θεωρία συνόλων). Στο αξιωματικό σύστημα ZF μπορούμε να αποδείξουμε την πρόταση Con(PA) αρκετά εύκολα, χρησιμοποιώντας το Αξίωμα Απείρου για να πλάσουμε ένα άπειρο σύνολο που χρησιμεύει ως ένα μοντέλο της PA.

Από την άλλη πλευρά, πάλι από το δεύτερο θεώρημα μη-πληρότητας, το ZF δεν μπορεί να αποδείξει τη δική του συνέπεια. Αν θέλουμε να αποδείξουμε την Con(ZF), ο απλούστερος τρόπος για να το κάνουμε είναι να δεχθούμε την ύπαρξη απείρων μεγαλύτερων από ότι μπορεί να οριστεί μέσα στην ZF. Αυτά τα άπειρα ονομάζονται «μεγάλοι πληθάριθμοι.»

Μπορούμε λοιπόν να αποδείξουμε τη συνέπεια της ZF στο ZF + LC (όπου LC είναι το αξίωμα ότι οι μεγάλοι πληθάριθμοι υπάρχουν). Αλλά αν θέλουμε να αποδείξουμε ότι ZF + LC είναι συνεπής θεωρία, τότε χρειαζόμαστε μια ακόμα πιο ισχυρή θεωρία, με ακόμη μεγαλύτερα άπειρα.

[5] Είναι γνωστή η «απόδειξη» του Ράσελ ότι αν 2=1 τότε ο Ράσελ είναι ο πάπας: Ο Ράσελ και ο Πάπας είναι δύο. Αλλά αφού 2=1 προκύπτει ότι είναι ένας. Έτσι ο Ράσελ και ο Πάπας ταυτίζονται.

Σχόλιο για το διαγώνιο επιχείρημα

Δημιουργός του είναι ο Georg Cantor, που το χρησιμοποίησε το 1874 γιά να αποδείξει ότι οι πραγματικοί αριθμοί είναι μη αριθμήσιμοι, δηλαδή ότι δεν μπορούμε να τους γράψουμε σαν μία ακολουθία.

Στη συνέχεια ήρθε Jules Richard, που το χρησιμοποίησε το 1905 προκειμένου να δείξει ότι οι λεκτικές περιγραφές αριθμών δεν μπορούν να ταξινομηθούν. Δεν μπορούμε να τις καταγράψουμε ως ακολουθία.

Στη συνέχεια ήρθε ο Kurt Gödel που αντί για λεκτικές περιγραφές εξέτασε τύπους της φορμαλιστικής λογικής και το χρησιμοποίησε το 1931 για να δείξει ότι η πρωτοβάθμια Αριθμητική είναι μη πλήρης, κατασκευάζοντας μια αριθμητική-λογική φόρμουλα, η οποία δεν είναι αποδείξιμη ούτε απορρίψιμη μέσω των επιλεγμένων αξιωμάτων.

Τέλος, ο Alan Turing (Τούρινγκ) αντί για λεκτικές περιγραφές εξέτασε αριθμητικούς αλγορίθμους και το χρησιμοποίησε το 1936 για να δείξει ότι το «πρόβλημα σταματήματος /τερματισμού της λειτουργίας» για μια μηχανή Τούρινγκ (μια αλγοριθμική μηχανή), ισοδύναμη με έναν υπολογιστή, είναι αλγοριθμικά μη επιλύσιμο.

Όλα αυτά προέκυψαν από μία αριθμοποίηση (μια αριθμητική κωδικοποίηση) αρχικά λεκτικών περιγραφών και μετά λογικών τύπων ή ισοδύναμα, αλγορίθμων (υπολογιστικών προγραμμάτων). Σημειώστε ότι άρρητοι αριθμοί είναι (πέρα από γεωμετρικούς ορισμούς) προϊόντα επαναληπτικών ή αναδρομικών διαδικασιών, δηλαδή, αλγόριθμοι.

Εγγενής προϋπόθεση σε όλα αυτά είναι ότι υπάρχουν μη αριθμήσιμο πλήθος από αριθμούς ή αριθμητικο-λογικούς τύπους ή αλγορίθμους με απεριόριστη έκταση, δηλαδή, ότι όλες αυτές οι έννοιες αναφέρονται-ανήκουν σε απροσδιόριστο σύνολα.

Οι άνθρωποι τείνουν να θεωρούν τέτοια επιτεύγματα, ως υπεράνθρωπα και να ηρωοποιούν ή και να αποθεώνουν αυτούς τους επιστήμονες.