Κεφάλαιο 1

Υπολογισιμότητα (Computability)

1.1 Ιστορία - Εισαγωγή

Η Συλλογιστική του Αριστοτέλη αποτέλεσε την πρώτη προσπάθεια θεμελίωσης της λογικής και των μαθηματικών. Ο *Leibni(t)z* πρότεινε το εξής πρόγραμμα:

1. Να δημιουργηθεί μια τυπική γλώσσα (*formal language*), με την οποία να μπορούμε να περιγράψουμε όλες τις μαθηματικές έννοιες και προτάσεις.

2. Να δημιουργηθεί μια μαθηματική θεωρία (δηλαδή ένα σύνολο από αξιώματα και συμπερασματικούς κανόνες συνεπαγωγής), με την οποία να μπορούμε να αποδεικνύουμε όλες τις ορθές μαθηματικές προτάσεις.

3. Να αποδειχθεί ότι αυτή η θεωρία είναι συνεπής (*consistent*), (δηλαδή ότι η πρόταση "Α και όχι Α" ($A \wedge \neg A$) δεν είναι δυνατόν να αποδειχθεί σ' αυτή τη θεωρία).

Η πραγμάτωση αυτού του προγράμματος άρχισε πολύ αργότερα, προς το τέλος του 19ου αιώνα. Πολλοί επιστήμονες ασχολήθηκαν με τον ορισμό της ενιαίας γλώσσας της μαθηματικής (ή συμβολικής) λογικής (*Boole, Frege*, κ.α). Άλλοι ασχολήθηκαν με τον ορισμό της ενιαίας θεωρίας των συνόλων (*Cantor*, κ.α.) και άλλοι με την παραγωγή (*derivation*) όλων των αληθών μαθηματικών προτάσεων με χρήση της Συνολοθεωρίας (*Russel, Whitehead*, κ.α.).

Στην αρχή αυτού του αιώνα ο *Hilbert* βάλθηκε να πραγματοποιήσει το 3ο μέρος του προγράμματος του *Leibni(t)z*, δηλαδή να βρει έναν αλγόριθμο που να αποκρίνεται (*decides*) για την ορθότητα κάθε μαθηματικής πρότασης. Τελικά, όμως, το 1931 ο *Gödel* απέδειξε ότι:

- Δεν υπάρχει τέτοιος αλγόριθμος.

- Είναι αδύνατον να αποδειχθεί η συνέπεια της Συνολοθεωρίας.

- Επιπλέον, οποιαδήποτε (δηλαδή όχι μόνο η Συνολοθεωρία) αξιωματική θεωρία των Μαθηματικών, που περιλαμβάνει τουλάχιστον την Αριθμοθεωρία, θα περιλαμβάνει και μη αποκρίσιμες (*undecidable*) προτάσεις.

- Κωδικοποιώντας προτάσεις με φυσικούς αριθμούς (αυτή η κωδικοποίηση λέγεται σήμερα "Γκεντελοποίηση": *Gödelization*) μπόρεσε να παρουσιάσει μια συγκεκριμένη πρόταση που είναι μη αποκρίσιμη.

Το αποτέλεσμα αυτό του *Gödel* ήταν η αιτία μιας σημαντικής κρίσης στα κλασσικά μαθηματικά, μα συγχρόνως και η απαρχή των μοντέρνων δυναμικών μαθηματικών. Το κεντρικό ερώτημα δεν είναι πια απλά αν μια πρόταση είναι αληθής η ψευδής, αλλά αν είναι "αποκρίσιμη ή μη αποκρίσιμη", δηλαδή αν είναι "υπολογίσιμη (*computable*) ή όχι". Αυτό ακριβώς είναι και το αντικείμενο της **Θεωρίας της Υπολογισιμότητας** (*computability*). Αν δοθεί ότι μια συνάρτηση f είναι υπολογίσιμη, ποιο είναι το κόστος ή τα αγαθά (*resources*) που χρειάζονται για να υπολογίσουμε την f; Αυτό είναι το βασικό ερώτημα της **Θεωρίας της Πολυπλοκότητας** (complexity)[1].

Διάφοροι επιστήμονες (*Turing, Church, Kleene, Post, Markov*, κ.α.) βάλθηκαν να ξεκαθαρίσουν τις έννοιες: υπολογίσιμο ή επιλύσιμο (*solvable*) με αλγόριθμο, υπολογίσιμη συνάρτηση και αποκρίσιμο πρόβλημα. Κατέληξαν, λοιπόν, σε διαφορετικά υπολογιστικά μοντέλα, τα οποία όμως αποδείχθηκαν όλα ισοδύναμα μεταξύ τους.

Η περίφημη **Θέση (thesis) Church-Turing** λέει λοιπόν απλουστευμένα: "Όλα τα γνωστά και τα "άγνωστα" μοντέλα της έννοιας "υπολογίσιμος" είναι μηχανιστικά ισοδύναμα (*effectively equivalent*)". Δηλαδή δοθέντος ενός αλγορίθμου σε ένα μοντέλο για μια συγκεκριμένη συνάρτηση f, μπορούμε μηχανιστικά (με τη βοήθεια μηχανής) να κατασκευάσουμε αλγόριθμο σε ένα άλλο μοντέλο για την ίδια συνάρτηση f.

Ας χρησιμοποιήσουμε εδώ ένα γνωστό μοντέλο υπολογισμού, μια γλώσσα προγραμματισμού υψηλού επιπέδου. Μια συνάρτηση[2] τότε, θα λέγεται υπολογίσιμη, αν υπάρχει πρόγραμμα που υπολογίζει την τιμή της για κάθε όρισμα. Είναι προφανές ότι υπάρχουν συναρτήσεις μη υπολογίσιμες, γιατί:

- Υπάρχουν άπειρα μεν, αλλά μόνο αριθμήσιμα (*countable*) διαφορετικά προγράμματα. Εκτός αυτού μπορούμε χρησιμοποιώντας κωδικοποίηση

[1] Κάποιες φορές χρησιμοποιείται και ο όρος υπολογιστός αντί για υπολογίσιμος

[2] θα πρέπει εδώ να διευκρινίσουμε ότι αναφερόμαστε σε συναρτήσεις $f : \mathbb{N}^n \to \mathbb{N}$, αφού τα δεδομένα σε ένα πραγματικό υπολογιστή κωδικοποιούνται με φυσικούς αριθμούς (ακολουθίες από 0 και 1). Γενικότερα, αναφερόμαστε σε συναρτήσεις από αριθμήσιμα σύνολα σε αριθμήσιμα σύνολα

να τα απαριθμήσουμε μηχανιστικά (*effectively enumerate*)[3]

- Από την άλλη μεριά όμως, ξέρουμε ότι υπάρχουν μη αριθμήσιμες άπειρες (*uncountable*) διαφορετικές συναρτήσεις. Αυτό αποδεικνύεται με διαγωνιοποίηση (*diagonalization*), ανάλογη με αυτή που χρησιμοποιούμε για να δείξουμε ότι το σύνολο $\mathbb{R}$ είναι μη αριθμήσιμο[4]

Ας στραφούμε σε ένα συγκεκριμένο πρόβλημα που είναι μη αποκρίσιμο. Όπως ξέρουμε ο μεταγλωττιστής (*compiler*) μιας γλώσσας προγραμματισμού μπορεί να ελέγξει αν ένα πρόγραμμα είναι συντακτικά ορθό ή όχι. Τι γίνεται όμως με τα λάθη χρόνου εκτέλεσης (*run time errors*); Θα ήταν ωραίο να είχαμε ένα πρόγραμμα που θα μπορούσε να ελέγχει αν ένα συντακτικά ορθό πρόγραμμα θα σταματήσει κάποτε ή αν θα τρέχει για πάντα. Δυστυχώς τέτοιο πρόγραμμα δεν υπάρχει.

Θεώρημα 1.1.1. *Το* **halting problem (HP)** *είναι μη αποκρίσιμο.*

Απόδειξη. Έστω ότι $\pi_0, \pi_1, \pi_2, \ldots$ είναι μια μηχανιστική απαρίθμηση (*effective enumeration*) όλων των προγραμμάτων. Ας υποθέσουμε ότι το **HP** είναι επιλύσιμο. Τότε κατασκευάζουμε ένα πρόγραμμα π, που ελέγχει αν το πρόγραμμα π_n με είσοδο n σταματάει ή όχι και ανάλογα με την απάντηση σε αυτόν τον έλεγχο, το πρόγραμμα π σταματάει αν το $\pi_n(n)$ δεν σταματάει, και αντιστρόφως:

$$\pi\colon \text{read}(n); \textbf{if } \pi_n(n) \text{ terminates } \textbf{then } \text{loop\_forever } \textbf{else } \text{halt}$$

[3]Απόδειξη: Κάθε πρόγραμμα μιας γλώσσας προγραμματισμού είναι στοιχείο του Σ^*, όπου $\Sigma = \{a_1, a_2, \ldots, a_m\}$ το αλφάβητο της γλώσσας. Το Σ^* όμως αποτελεί την ένωση $\bigcup_{n=0}^{\infty} \Sigma_n$, όπου Σ_n το σύνολο των συμβολοσειρών του αλφάβητου Σ που έχουν μήκος n. Κάθε σύνολο Σ_n είναι πεπερασμένο και έτσι αν διατάξουμε τα στοιχεία του αλφαβητικά μπορούμε να θεωρήσουμε την ακόλουθη αρίθμηση για το Σ^*:

$$\begin{aligned}
\Sigma_0 &: \ \{\varepsilon\} \\
\Sigma_1 &: \ \{a_1, a_2, \ldots, a_m\} \\
\Sigma_2 &: \ \{a_1a_1, a_1a_2, \ldots, a_1a_m, \ldots, a_ma_m\} \\
&\vdots
\end{aligned}$$

Η παραπάνω αρίθμηση του Σ^*είναι μηχανιστική, δηλαδή μπορεί να γίνει με πρόγραμμα. Επομένως, με κατάλληλη χρήση compiler για τον έλεγχο ορθότητας μπορούμε να κατασκευάσουμε μηχανιστική αρίθμηση των συντακτικά ορθών n προγραμμάτων

[4]Απόδειξη: Ας θεωρήσουμε το σύνολο των ολικών συναρτήσεων $\varphi\colon \mathbb{N} \to \mathbb{N}$ και έστω $\varphi_0, \varphi_1, \varphi_2, \ldots$ μια αρίθμηση τους (ολικές ονομάζονται οι συναρτήσεις που ορίζονται για κάθε $x \in \mathbb{N}$). Ορίζουμε μια συνάρτηση f ως εξής: $f(x) = \varphi_x(x) + 1, \forall x \in \mathbb{N}$. Η f είναι προφανώς ολική συνάρτηση και επομένως θα αντιστοιχίζεται σε κάποιο δείκτη y στην παραπάνω αρίθμηση μας, δηλαδή $f = \varphi_y$. Τότε όμως θα ισχύει ότι $\varphi_y(y) = f(y) = \varphi_y(y)+1$ που είναι άτοπο. Επομένως το σύνολο των ολικών συναρτήσεων δεν είναι αριθμήσιμο.

Φυσικά αυτό το πρόγραμμα π κάπου θα εμφανίζεται στην παραπάνω αρίθμηση. Ας πούμε ότι ο δείκτης για το π είναι i, δηλαδή $\pi = \pi_i$. Η ιδέα της διαγωνιοποίησης είναι να δώσουμε το δείκτη i για input στο π_i. Τότε το $\pi_i(i)$ σταματάει αν και μόνο αν το $\pi(i)$ σταματάει και αυτό συμβαίνει αν και μόνο αν το $\pi_i(i)$ δεν σταματάει. Αντίφαση. □

Τελικά πολλά άλλα προβλήματα είναι επίσης μη επιλύσιμα. Αν και το HP δεν είναι επιλύσιμο, μπορούμε να κατασκευάσουμε με μηχανιστικό τρόπο μια άπειρη λίστα όλων των προγραμμάτων, με την αντίστοιχη είσοδο για την οποία σταματούν. Αυτό δεν σημαίνει φυσικά ότι μπορούμε να επιλύσουμε το HP, γιατί αν για παράδειγμα το $\pi_k(n)$ δεν έχει εμφανισθεί στη λίστα μας, δεν ξέρουμε αν θα προστεθεί στη λίστα αργότερα ή αν δε θα εμφανισθεί ποτέ στη λίστα. Για να ακριβολογούμε λίγο περισσότερο δίνουμε τους παρακάτω ορισμούς.

Ορισμός 1.1.2. Ένα σύνολο S λέγεται αποκρίσιμο ή υπολογίσιμο ή επιλύσιμο (*decidable, computable, solvable*) αν και μόνο αν υπάρχει ένας αλγόριθμος που σταματάει ή μια υπολογιστική μηχανή που δίνει έξοδο "ναι" για κάθε είσοδο $a \in S$ και έξοδο "όχι" για κάθε είσοδο $a \notin S$.

Ορισμός 1.1.3. Ένα σύνολο S λέγεται καταγράψιμο (με μηχανιστική γεννήτρια) (*listable, effectively generatable*) αν και μόνο αν υπάρχει μια γεννήτρια διαδικασία ή μηχανή που καταγράφει όλα τα στοιχεία του S. Στην, πιθανώς άπειρη, λίστα εξόδου επιτρέπονται οι επαναλήψεις και δεν υπάρχει περιορισμός για την διάταξη των στοιχείων.

Μερικές απλές ιδιότητες:

1. Αν το S είναι αποκρίσιμο τότε και το $\overline{S}$ είναι αποκρίσιμο.
2. Αν το S είναι αποκρίσιμο τότε το S είναι και καταγράψιμο.
3. Αν το S και το $\overline{S}$ είναι καταγράψιμα τότε το S είναι αποκρίσιμο.
4. Αν το S είναι καταγράψιμο με γνησίως αύξουσα διάταξη τότε το S είναι αποκρίσιμο.

Απόδειξη. 1. Πρέπει να δείξουμε ότι υπάρχει πρόγραμμα που αποκρίνεται για το $\overline{S}$. Αρκεί να προσθέσουμε στο πρόγραμμα που αποκρίνεται για το S μία εντολή που αντιστρέφει το "ναι" και το "όχι" πριν την έξοδο.

2. Μπορούμε να χρησιμοποιήσουμε το πρόγραμμα π που αποκρίνεται για το S, διαδοχικά για όλους τους φυσικούς αριθμούς:

$n := 0$;
repeat
  if $\pi(n) =$ ''ναι'' **then** output(n);
  $n := n + 1$
forever

3. Μπορούμε να τρέξουμε παράλληλα τα προγράμματα π_1, π_2 που καταγράφουν τα S και $\overline{S}$ αντιστοίχως, μέχρι η είσοδος n να εμφανιστεί σε κάποια από τις δύο λίστες εξόδου. Σε αυτό το σημείο η ερώτηση "$n \in S$;" μπορεί να απαντηθεί.

   Παρατήρηση: Υπάρχει μία μέθοδος που μας επιτρέπει να προσομοιώσουμε την παράλληλη εκτέλεση δύο (ή γενικότερα αριθμήσιμων το πλήθος) προγραμμάτων, το λεγόμενο **dovetailing**. Η ιδέα είναι δίκαιος καταμερισμός του χρόνου στα επιμέρους προγράμματα:

   Για δύο προγράμματα, π_1 και π_2, έχουμε:

   repeat
     τρέξε ένα βήμα του π_1 (αν δεν έχει τερματίσει);
     τρέξε ένα βήμα του π_2 (αν δεν έχει τερματίσει)
   forever (ή μέχρι να τερματίσουν και τα δύο)

   Γενικά, για αριθμήσιμα το πλήθος προγράμματα π_1, π_2, ...

   $n := 1$
   repeat
     for $i := 1$ **to** n **do**
       τρέξε ένα βήμα του π_i (αν δεν έχει τερματίσει);
     $n := n + 1$
   forever

4. Μπορούμε να χρησιμοποιήσουμε το π, που καταγράφει το S με γνησίως αύξουσα διάταξη μέχρι η είσοδος n να εμφανιστεί ή να ξεπεραστεί στην λίστα εξόδου. Σε αυτό το σημείο αποφασίζεται αν $n \in S$ ή $n \notin S$ αντίστοιχα.

□

1.2 Αυτοαναφορά - Διαγωνιοποίηση

Πολλά παράδοξα στην λογική και τα μαθηματικά βασίζονται στην τεχνική της αυτοαναφοράς.

Η πρώτη γνωστή χρήση αυτοαναφοράς γίνεται τον τον 6ο αιώνα π.Χ., οπότε ο Επιμενίδης ο Κρης είπε:

Πας Κρης ψεύτης.

Τον 4ο αιώνα π.Χ., ο Ευβουλίδης έκανε ακόμα πιο σαφή την αυτοαναφορά:

Αυτή η πρόταση είναι ψευδής.

Άλλα παράδοξα που έχουν σχέση με την αυτοαναφορά προκάλεσαν αμφισβήτηση για την συνέπεια της Αφελούς Συνολοθεωρίας, με χαρακτηριστικότερο το παράδοξο (ή καλύτερα αντινομία) του Russell: Θεωρούμε το σύνολο A που περιέχει όλα τα σύνολα που δεν είναι μέλη του εαυτού τους, δηλαδή $A = \{x \mid x \notin x\}$. Ανήκει το A στον εαυτό του ή όχι;

Αναφέρουμε ενδεικτικά ακόμα:

- τον κουρέα σε ένα χωριό που ξυρίζει όλους όσους δεν ξυρίζονται μόνοι τους (ποιος ξυρίζει τον κουρέα;)
- το μεταπαιχνίδι G που παίζεται από δύο παίκτες: Ο πρώτος παίκτης διαλέγει ένα πεπερασμένο παιχνίδι. Μετά οι δύο παίκτες παίζουν το παιχνίδι αυτό (την πρώτη κίνηση την κάνει ο δεύτερος παίκτης). Είναι το μεταπαιχνίδι G πεπερασμένο ή άπειρο;

Η αυτοαναφορά έχει σχέση με την τεχνική της διαγωνιοποίησης (Cantor). Παρακάτω δίνουμε ένα απλό παράδειγμα της τεχνικής:

Έστω $n \times n$ boolean πίνακας (δηλαδή τα στοιχεία του παίρνουν τιμές 0 και 1) A. Θεωρούμε το διάνυσμα d που προκύπτει αν πάρουμε τα στοιχεία της διαγωνίου, δηλαδή:

$$d_i = A_{ii}, \text{ για κάθε } i, \text{ με } 1 \leq i \leq n.$$

Αντιστρέφουμε τα στοιχεία του διανύσματος d και παίρνουμε ένα νέο διάνυσμα D, δηλαδή:

$$D_i = 1 - d_i, \text{ για κάθε } i, \text{ με } 1 \leq i \leq n.$$

Τότε μπορεί εύκολα να δειχθεί ότι το διάνυσμα D δεν εμφανίζεται ούτε ως γραμμή ούτε ως στήλη στον πίνακα A.

Η παραπάνω τεχνική, κατάλληλα τροποποιημένη, μπορεί να λειτουργήσει και για άπειρο πίνακα, αλλά και για πίνακες όπου τα στοιχεία δεν είναι κατά ανάγκη boolean. Για παράδειγμα, μπορούμε να δείξουμε ότι το πλήθος των συναρτήσεων από τους φυσικούς στους φυσικούς δεν είναι αριθμήσιμο. Παρομοίως, μπορούμε να δείξουμε ότι το πλήθος των αυξουσών συναρτήσεων από τους φυσικούς στους φυσικούς δεν είναι αριθμήσιμο.

Παρακάτω, θα δούμε πώς η παραπάνω τεχνική έχει εφαρμογές στο πρόβλημα τερματισμού και στην απόδειξη του θεωρήματος μη πληρότητας του Gödel.

1.2.1 Πρόβλημα τερματισμού

Ας θεωρήσουμε μία μηχανιστική απαρίθμηση των προγραμμάτων με μία είσοδο:

$$\pi_0, \pi_1, \pi_2, \ldots$$

Αν το πρόγραμμα π_x τερματίζει με είσοδο $y \in \mathbb{N}$, θα γράφουμε $\pi_x(y) \downarrow$, αλλιώς (αν δεν τερματίζει) $\pi_x(y) \uparrow$. Το *πρόβλημα τερματισμού* είναι: "Δίνονται x, y. Ισχύει $\pi_x(y) \downarrow$;"

Ας θεωρήσουμε ότι υπάρχει πρόγραμμα $\text{halt}(x, y)$ που λύνει το πρόβλημα τερματισμού. Τότε κατασκευάζουμε πρόγραμμα:

$\text{read}(x)$;
if $\text{halt}(x, x)$ **then** loop forever **else** stop

Αυτό το πρόγραμμα θα υπάρχει κάπου στην απαρίθμηση των προγραμμάτων μίας εισόδου. Ας πούμε ότι είναι το π_i. Τότε όμως, θα έχουμε:

$$\pi_i(i) \downarrow \iff \pi_i(i) \uparrow .$$

Επομένως, το πρόβλημα τερματισμού δεν επιδέχεται λύση (με πρόγραμμα).

1.2.2 Θεώρημα μη πληρότητας του Gödel

Ας θεωρήσουμε τύπους στην γλώσσα της αριθμητικής Peano. Υπάρχει μηχανιστική απαρίθμηση όλων των τύπων με μία ελεύθερη μεταβλητή:

$$\varphi_0, \varphi_1, \varphi_2, \ldots$$

Επιπλέον ας θεωρήσουμε συνεπή αξιωματικοποίηση, δηλαδή:

Αν φ είναι αποδείξιμος, τότε φ αληθής.

Θα θέλαμε επίσης το σύστημά μας να είναι και πλήρες, δηλαδή:

Αν φ είναι αληθής, τότε φ αποδείξιμος.

Η γλώσσα της αριθμητικής Peano είναι τόσο ισχυρή που μπορεί να κωδικοποιήσει τύπους της μηχανιστικής απαρίθμησης, αλλά ακόμα και την έννοια της απόδειξης (από τα αξιώματα και με χρήση συμπερασματικών κανόνων). Έτσι, μπορούμε να γράψουμε τον εξής τύπο:

"δεν υπάρχει απόδειξη για $\varphi_x(x)$"

Αυτός ο τύπος θα υπάρχει κάπου στην απαρίθμηση των τύπων με μία ελεύθερη μεταβλητή. Ας πούμε ότι είναι ο φ_i. Αυτό σημαίνει όμως ότι:

$$\varphi_i(i) = \text{"δεν υπάρχει απόδειξη για } \varphi_i(i)\text{"}$$

Η πρόταση $\varphi_i(i)$ δεν μπορεί να είναι ψευδής (γιατί;). Έτσι, η πρόταση $\varphi_i(i)$ είναι αληθής, αλλά όχι αποδείξιμη, επομένως:

Θεώρημα 1.2.1 (μη πληρότητας)**.** *Κάθε συνεπής αξιωματικοποίηση της αριθμητικής Peano είναι μη πλήρης.*

Δηλαδή δεν μπορούμε να έχουμε ταυτόχρονα και την ιδιότητα της πληρότητας που θα θέλαμε.

Σημείωση: Δέν ισχύει το παραπάνω για την αριθμητική Presburger (που δέν έχει την πράξη $*$).