Θέση Church-Turing I

Πολλοί τρόποι περιγραφής αλγορίθμων. Όλοι είναι μηχανιστικά ισοδύναμοι και ειδικά ισοδύναμοι με μερικές αναδρομικές συναρτήσεις

Θέση Church-Turing: *Όλες οι υπολογίσιμες συναρτήσεις είναι μερικές αναδρομικές και όλες οι αποκρίσιμες σχέσεις είναι αναδρομικές.*

Υπενθύμιση: Μια σχέση R λέγεται αναδρομική όταν η χαρακτηριστική της συνάρτησης χ_R είναι αναδρομική, όπου:

$$\chi_R(x,y) = \begin{cases} 1 & \text{αν } R(x,y) \\ 0 & \text{αλλιώς} \end{cases}$$

Συνέπεια της θέσης του Church: Για να δείξουμε οτι μια συνάρτηση ή σχέση είναι (μερική) αναδρομική, αντί μιας ακολουθίας τυπικών συνεπαγωγών (με πρωταρχικές συναρτήσεις και σχήματα) ή αντί ενός πλήρους προγράμματος ή Τ.Μ., αρκεί να δίνουμε ενορατικά (αλλά ορθά) επιχειρήματα για τον τρόπο υπολογισμού τους.

Θέση Church-Turing II

Έτσι φαίνεται τελικά οτι καταλήξαμε στον τυπικό πια ορισμό της εννοιας **υπολογίσιμος**. Φυσικά υπάρχουν πολλές συναρτήσεις που δεν είναι υπολογίσιμες.

Θεώρημα

Υπάρχουν μη υπολογίσιμες συναρτήσεις.

Απόδειξη.

Απαριθμούμε όλες τις μερικές αναδρομικές συναρτήσεις φ_0, φ_1, φ_2, ...

Άσκηση: Χρησιμοποιώντας κωδικοποίηση, απαρίθμησε μηχανιστικά την κλάση $\mathcal{PR}$ ή όλα τα προγράμματα **while**.

Άρα υπάρχουν άπειρες μεν, αλλά **μόνο** αριθμήσιμες το πλήθος μερικές αναδρομικές συναρτήσεις ($\aleph_0$). Από την άλλη μεριά όμως, υπάρχουν μη αριθμήσιμες το πλήθος συναρτήσεις γενικώς ($2^{\aleph_0}$ δείχνεται με διαγωνιοποίηση). □

Θέση Church-Turing III

Παράδειγμα μιας μη υπολογίσιμης συνάρτησης:

Το πρόβλημα τερματισμού *(halting problem).*

$$g(x,y) = \begin{cases} 1, & \text{αν } \varphi_x(y) \downarrow \\ 0, & \text{αλλιώς} \end{cases}$$

Universal Machine I

Καθολικό Πρόγραμμα (universal program): ένα πρόγραμμα που παίρνει για είσοδο οποιοδήποτε **πρόγραμμα** με τα **δεδομένα εισόδου** του και **προσομοιάζει** την εκτέλεσή του. Αυτή η έννοια είναι παρόμοια με τη λειτουργία ενός επόπτη (monitor), μεταφραστή (compiler) ή διερμηνέα (interpreter) σ' ένα πραγματικό υπολογιστή.

Μιλήσαμε για την ισοδυναμία των μοντέλων, άρα μπορούμε να μιλάμε είτε για καθολικό While program, είτε για καθολικό Goto program, είτε για καθολική μηχανή Turing (ιστορικά έτσι πρωτοχρησιμοποιήθηκε η έννοια της καθολικότητας), είτε για καθολική μερική αναδρομική συνάρτηση.

Καθολικό While program $\omega^n(x, y)$ με

- είσοδο x = κωδικός ενός Goto προγράμματος π με n μεταβλητές.
- είσοδο y = κωδικός των n τιμών εισόδου $[a_1, a_2, \ldots, a_n]$ για τις n μεταβλητές του π και
- έξοδο την ίδια με την έξοδο του $\pi[a_1, a_2, ..., a_n]$.

Καθολική αναδρομική συνάρτηση φ_e.
$\forall x, y\ \varphi_e(x, y) \simeq \varphi_x(y)$
όπου το $\simeq$ σημαίνει: είτε είναι και οι δυο πλευρές ορισμένες και ίσες, είτε καμία πλευρά δεν είναι ορισμένη.

Universal Machine II

πρώτος σκελετός για $\omega^n(x, y)$:

label := 0;
while label $\leqslant$ maxlab **do**
  εκτέλεσε εντολή της label;
  βρές επόμενη label
end

Δεύτερη προσέγγιση για $\omega^n(x, y)$:

label := 0; maxlab := decodesearch(x);
while label $\leqslant$ maxlab **do**
  instruction := decode(x at label);
  label := execnext(instruction,y);
  y:= execvalue(instruction,y)
end

Universal Machine III

Εφαρμόζουμε την ακόλουθη κωδικοποίηση:

Κωδικοποίηση για την είσοδο:
$\langle a_1, a_2, \ldots, a_n \rangle \mapsto C_f(a_1, a_2, \ldots, a_n)$

Για τις εντολές του GOTO προγράμματος:

- l_i: assignment **goto** l_j $\mapsto C_f(0, i, C_f(\text{assignment}), j)$ όπου η κωδικοποίηση για τις αναθέσεις, είναι:

Αναθέσεις	*Κωδικοποίηση*
κενή εντολή	$C_f(0,0,0)$
$x_i := 0$	$C_f(1,i,0)$
$x_i := x_j$	$C_f(2,i,j)$
$x_i := x_j + 1$	$C_f(3,i,j)$
$x_i := x_j \dot{-} 1$	$C_f(4,i,j)$

- l_i: **if** $x_m <> 0$ **then goto** l_j **else goto** l_k $\mapsto C_f(1, i, m, j, k)$

Έτσι τα προγράμματα goto θα κωδικοποιούνται ως εξής:

$$\text{goto πρόγραμμα με } m \text{ εντολές} \mapsto C_f(C_f(\text{εντολή}_0), \ldots, C_f(\text{εντολή}_{m-1}))$$

Universal Machine IV

Τρίτη βελτίωση για $\omega^n(x, y)$:

label:=0; maxlab := decodesearch(x);
while label<= maxlab **do**
  $z := D^f_{\text{label}}(x)$;
  if $D^f_0(z) = 0$ **then**
    ass := $D^f_2(z)$;
    label := $D^f_3(z)$
  end
  else
    $m := D^f_2(z)$;
    if $m <> 0$ **then** label := $D^f_3(z)$ **else** label := $D^f_4(z)$
  end
  y := execvalue(ass, y)
end

Η execvalue αποκωδικοποιεί την ass και ανάλογα μεταβάλλει τις τιμές των μεταβλητών του προγράμματος.

Τέταρτη βελτίωση για $\omega^n(x, y)$: κ.ο.κ. 'Ασκηση.

Θεώρημα Κανονικής μορφής Kleene I

Παρατήρηση

Λόγω της μηχανιστικής ισοδυναμίας των προγραμμάτων **while** και **goto**, έπεται η εξής πρόταση:

> *Υπάρχει ένα καθολικό πρόγραμμα* ***while*** ω^n, *που έχει μόνο ένα "πραγματικό"* ***while*** *βρόχο και ίσως πολλούς* ***for*** *βρόχους (ή* ***if*** *εντολές ...), έτσι ώστε για κάθε πρόγραμμα* ***while*** π *και για κάθε είσοδο να ισχύει:*
>
> $$\omega^n(\text{κωδικός}(\pi), \text{είσοδος}) \simeq \pi(\text{είσοδος})$$

Θεώρημα Κανονικής μορφής Kleene II

Αν μεταφράσουμε την προηγούμενη πρόταση σε ορολογία μερικών αναδρομικών συναρτήσεων, καταλήγουμε στο εξής:

Θεώρημα κανονικής μορφής Kleene (Kleene normal form theorem)
$\forall n\ \exists \psi, T \in \mathcal{P}$
$\forall f \in \mathcal{PR}^n$ ($\mathcal{PR}^n$ μερικές αναδρομικές συναρτήσεις με n ορίσματα)
$\exists$ δείκτης x:
$\forall \vec{y}\ f(\vec{y}) \simeq \psi(\mu z[T(x, \vec{y}, z) = 0], x, \vec{y})$

όπου:
f: αντιστοιχεί στο πρόγραμμα π
y: αντιστοιχεί στην είσοδο του π
x: αντιστοιχεί στον κώδικο του π (δείκτης της f)
z: αντιστοιχεί στον αριθμό επαναλήψεων του βρόχου **while** στο ω^n
T: πρωταρχική αναδρομική συνάρτηση που αντιστοιχεί στο σώμα του **while** στο ω^n
ψ: πρωταρχική αναδρομική συνάρτηση που αντιστοιχεί στο τμήμα του ω^n έξω από τον βρόχο **while**

Θεώρημα Κανονικής μορφής Kleene III

Παρατηρήσεις:

1. T λέγεται συνήθως *κατηγόρημα Kleene*.
2. Αρκεί μια μόνο εφαρμογή του τελεστή μ για να περιγραφεί οποιαδήποτε μερική αναδρομική συνάρτηση.
3. Το παραπάνω θεώρημα (KNF) δίνει μια μηχανιστική απαρίθμηση $\{\varphi_x\}$ όλων των μερικών αναδρομικών συναρτήσεων. Σαν εφαρμογή θα δείξουμε το εξής (περίφημο) απότελεσμα αποκρισιμότητας:

Θεώρημα

Το HP (πρόβλημα τερματισμού), είναι μη επιλύσιμο, δηλαδή η συνάρτηση g δεν είναι αναδρομική όπου:

$$g(x,y) = \begin{cases} 1, & \text{αν } \varphi_x(y) \downarrow \\ 0, & \text{αλλιώς} \end{cases}$$

Θεώρημα Κανονικής μορφής Kleene IV

Απόδειξη (διαγωνιοποίηση και εις άτοπον απαγωγή).

Ας υποθέσουμε ότι η g ήταν αναδρομική. Ορίζουμε:

$$h(x) = \begin{cases} \varphi_x(x)+1, & \text{αν } g(x,x)=1 \\ 0, & \text{αλλιώς} \end{cases} = \begin{cases} \varphi_x(x)+1, & \text{αν } \varphi_x(x)\downarrow \\ 0, & \text{αλλιώς} \end{cases}$$

Η συνάρτηση h μπορεί να υπολογιστεί με τη βοήθεια της υπολογίσιμης g. Αρα, λόγω θέσης Church-Turing, η h είναι (ολική) αναδρομική. Αρα, λόγω της κανονικής μορφής Kleene, έχει κάποιο δείκτη (έστω) y, δηλαδή $\varphi_y = h$ (ολική συνάρτηση). Τότε όμως:

$$\underline{\varphi_y(y)} = h(y) = \begin{cases} \varphi_y(y)+1, & \text{αν } \varphi_y(y)\downarrow \\ 0, & \text{αλλιώς} \end{cases} = \underline{\varphi_y(y)+1}, \quad \text{αφού } \forall y : \varphi_y(y)\downarrow$$

Άτοπο. Συνεπώς η g δεν είναι αναδρομική. □

Θεώρημα Κανονικής μορφής Kleene V

Εναλλακτική απόδειξη του ίδιου με προγράμματα:

Έστω μηχανιστική απαρίθμηση προγραμμάτων: $\pi_0, \pi_1, \pi_2, \ldots$. Ας υποθέσουμε ότι υπάρχει πρόγραμμα $\mathsf{halt}(x, y)$ που αποκρίνεται αν το x-οστό πρόγραμμα με είσοδο y σταματάει.

Το εξής πρόγραμμα (με είσοδο x):
l_0: **if** $\mathsf{halt}(x, x)$ **then goto** l_0 *(* loop forever *)* **else goto** l_1 *(* stop *)*

θα εμφανίζεται κάπου στην απαρίθμηση (έστω) με δείκτη i. Τότε όμως:

$\pi_i(i)$ σταματάει **ανν** not $\mathsf{halt}(i, i)$ **ανν** $\pi_i(i)$ δεν σταματάει. **'Ατοπο**. □