

ΕΠΕΞΕΡΓΑΣΙΑ ΦΩΝΗΣ & ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

8ο Εξάμηνο 2022 – 2023

Assignment – Solutions

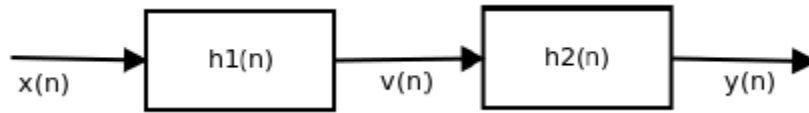
Ζαρίφης Στέλιος – el20435

Email: el20435@mail.ntua.gr

Contents

Άσκηση 1	2
Άσκηση 2	5
Άσκηση 3	7
Άσκηση 4	10
Άσκηση 5	11
Άσκηση 6	13
Άσκηση 7	17
Άσκηση 8	19
Άσκηση 9	22

Άσκηση 1



Ερώτημα 1

Δεδομένων των κρουστικών αποκρίσεων h_1, h_2 , θα δείξουμε ότι το συνολικό σύστημα έχει απόκριση $h = h_1(n) * h_2(n)$, ότι δηλαδή ισχύει το εξής:

$$y(n) = x(n) * h(n)$$

Χριάζεται να αποδείξουμε ότι είναι:

$$y(n) = v(n) * h_2(n) = (x(n) * h_1(n)) * h_2(n) = x(n) * (h_1(n) * h_2(n))$$

Για το $x(n) * h_1(n)$ έχουμε από τον τύπο της συνέλιξης:

$$x(n) * h_1(n) = \sum_{m=-\infty}^{+\infty} x(m)h_1(n-m) \quad (1)$$

Ομοίως, χρησιμοποιώντας τη σχέση (1), για το $(x(n) * h_1(n)) * h_2(n)$ έχουμε:

$$\begin{aligned} (x(n) * h_1(n)) * h_2(n) &= \sum_{k=-\infty}^{+\infty} (x(k) * h_1(k))h_2(n-k) = \\ &= \sum_{k=-\infty}^{+\infty} \left(\sum_{m=-\infty}^{+\infty} x(m)h_1(k-m) \right) h_2(n-k) = \sum_{k=-\infty}^{+\infty} x(m) \left(\sum_{m=-\infty}^{+\infty} h_1(k-m) h_2(n-k) \right) \quad (2) \end{aligned}$$

Αν θέσουμε $l = k - m \Rightarrow k = l + m$, το τελευταίο άθροισμα γράφεται ως εξής:

$$(x(n) * h_1(n)) * h_2(n) = \sum_{k=-\infty}^{+\infty} x(m) \left(\sum_{m=-\infty}^{+\infty} h_1(l) h_2(n-l-m) \right)$$

Από τον ορισμό της συνέλιξης, έχουμε:

$$\sum_{m=-\infty}^{+\infty} h_1(l) h_2(n-l-m) = \sum_{m=-\infty}^{+\infty} h_1(l) h_2((n-m)-l) = h_1(n-m) * h_2(n-m) \quad (3)$$

Συνεπώς, η πρόταση (2) με τη βοήθεια της (3) μας δίνει:

$$(x(n) * h_1(n)) * h_2(n) = \sum_{k=-\infty}^{+\infty} x(m)(h_1(n-m) * h_2(n-m)) \quad (4)$$

Και για μια τελευταία φορά, από τον ορισμό της συνέλιξης, η (4) γίνεται:

$$\begin{aligned}(x(n) * h_1(n)) * h_2(n) &= \sum_{k=-\infty}^{+\infty} x(m)(h_1(n-m) * h_2(n-m)) = \\ &= x(n) * (h_1(n) * h_2(n))\end{aligned}$$

Άρα, αφού $y(n) = x(n) * (h_1(n) * h_2(n))$, η κρουστική απόκριση του συνολικού συστήματος είναι πράγματι $h(n) = h_1(n) * h_2(n)$.

Ερώτημα 2

Θα ξεκινήσουμε από τον ορισμό της της συνέλιξης $h_1(n) * h_2(n)$:

$$h_1(n) * h_2(n) = \sum_{m=-\infty}^{+\infty} h_1(m)h_2(n-m) \quad (1)$$

Θέτοντας $k = n - m \Rightarrow m = n - k$, προκύπτει από τον (1):

$$h_1(n) * h_2(n) = \sum_{m=-\infty}^{+\infty} h_1(m)h_2(n-m) = \sum_{m=-\infty}^{+\infty} h_1(n-k)h_2(k) \quad (2)$$

Και η (2), μετά από αντιμετάθεση όρων και από τον ορισμό της συνέλιξης γράφεται ως:

$$h_1(n) * h_2(n) = \sum_{m=-\infty}^{+\infty} h_1(n-k)h_2(k) = \sum_{m=-\infty}^{+\infty} h_2(k)h_1(n-k) = h_2(n) * h_1(n)$$

Άρα δείξαμε ότι $h_1(n) * h_2(n) = h_2(n) * h_1(n)$, δηλαδή η συνολική κρουστική απόκριση δεν εξαρτάται από τη σειρά με την οποία εμφανίζονται τα συστήματα.

Ερώτημα 3

Θεωρούμε τις εξισώσεις διαφορών των συστημάτων $h_1(n), h_2(n)$:

$$V(z) = H_1(z)X(z) = X(z) \left(\sum_{r=0}^M b_r z^{-r} \right) \xrightarrow{Z^{-1}} v(n) = \sum_{r=0}^M b_r x(n-r) \quad (3)$$

$$\begin{aligned}Y(z) &= H_2(z)V(z) = \frac{1}{1 - \sum_{k=1}^N a_k z^{-k}} V(z) \Rightarrow \\ \Rightarrow Y(z) - Y(z) \left(\sum_{k=1}^N a_k z^{-k} \right) &= V(z) \xrightarrow{Z^{-1}} y(n) - \sum_{k=1}^N a_k y(n-k) = v(n) \quad (4)\end{aligned}$$

Και από τις (3), (4), έχουμε για το ολικό σύστημα:

$$y(n) - \sum_{k=1}^N a_k y(n-k) = \sum_{r=0}^M b_r x(n-r)$$

Ερώτημα 4

Αλλάζοντας τη σειρά των συστημάτων, έχουμε:

$$V(z) = H_2(z)X(z) = \frac{1}{1 - \sum_{k=1}^N a_k z^{-k}} X(z) \Rightarrow$$

$$\Rightarrow V(z) - V(z) \left(\sum_{k=1}^N a_k z^{-k} \right) = X(z) \xrightarrow{z^{-1}} v(n) - \sum_{k=1}^N a_k v(n-k) = x(n) \quad (5)$$

$$Y(z) = H_1(z)V(z) = Y(z) \left(\sum_{r=0}^M b_r z^{-r} \right) \xrightarrow{z^{-1}} y(n) = \sum_{r=0}^M b_r v(n-r) \quad (6)$$

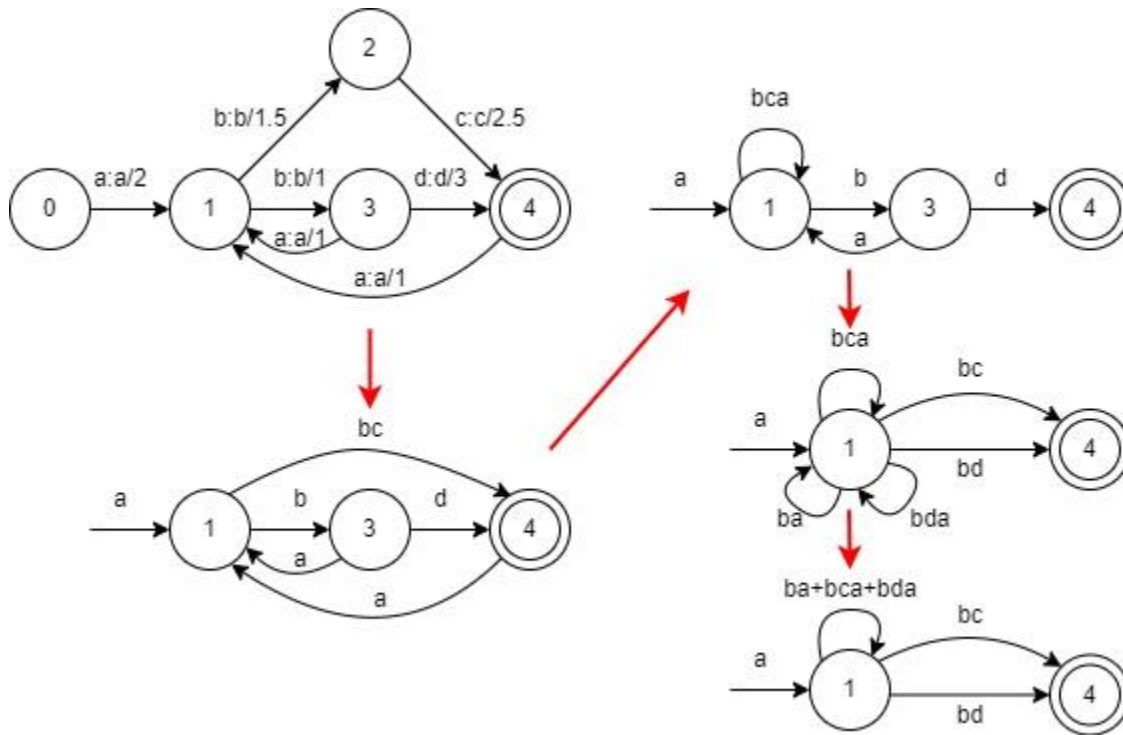
Και από τις (5), (6), έχουμε το σύστημα εξισώσεων διαφορών:

$$\begin{cases} v(n) - \sum_{k=1}^N a_k v(n-k) = x(n) \\ y(n) = \sum_{r=0}^M b_r v(n-r) \end{cases}$$

Άσκηση 2

Ερώτημα 1

Σταδιακά συνενώνουμε states και transitions, όπως φαίνεται στα παρακάτω βήματα:



Από το τελευταίο αυτόματο, βλέπουμε πως:

1. Απαιτείται στην αρχή της συμβολοσειράς να υπάρχει ένα a για να μεταβούμε στην κατάσταση 1.
2. Ύστερα, αν έρθει οσοδήποτε φορές κάποιο από τα strings ba, bca, bda , μεταβαίνουμε από την κατάσταση 1 στον εαυτό της.
3. Με οποιοδήποτε από τα strings bc, bd μεταβαίνουμε στην κατάσταση 4

Για τη 2η πρόταση, μπορούμε να χρησιμοποιήσουμε ένα ε transition για να την εκφράσουμε ως $ba + bca + bda = b(\varepsilon + c + d)a$

Το regular expression της μηχανής είναι: $a(b(\varepsilon + c + d)a)^*(bc + bd)$

Ερώτημα 2

Η πιο πιθανή συμβολοσειρά είναι εκείνη με το ελάχιστο συνολικό κόστος. Από τη στιγμή που έχουμε transition από το 1 στον εαυτό του (και έχουμε μόνο θετικά βάρη), μια διαδρομή που θα επιλέξει αυτό το transition, σίγουρα δε θα είναι βέλτιστη (ελαχίστου κόστους). Οι εναπομείνουσες συμβολοσειρές (διαδρομές που δε χρησιμοποιούν το προαναφερθέν transition) είναι οι abc και abd . Για αυτές τα κόστη είναι:

- $abc: 2 + 1.5 + 2.5 = 6$
- $abd: 2 + 1 + 6 + 3 = 12$

Συνεπώς η abc είναι η πιο πιθανή συμβολοσειρά.

Ερώτημα 3

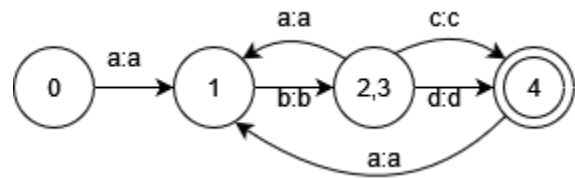
Από tropical semiring, το κόστος της γραμματοσειράς $abdababc$ υπολογίζεται ως:

$$2 + 1 + (6) + 3 + 1 + 1 + (6) + 1 + 1.5 + 2.5 = 25$$

Ερώτημα 4

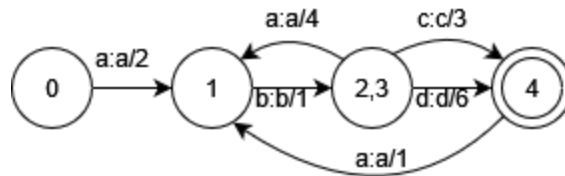
Σχηματίζουμε τον πίνακα και έχουμε την ισοδύναμη ντετερμινιστική μηχανή χωρίς κόστος:

	a	b	c	d
0	{1}	\emptyset	\emptyset	\emptyset
1	\emptyset	{2,3}	\emptyset	\emptyset
2	\emptyset	\emptyset	4	\emptyset
3	1	\emptyset	\emptyset	4
4	1	\emptyset	\emptyset	\emptyset



Ερώτημα 5

Ακόμα, για την ισοδύναμη ντετερμινιστική μηχανή με κόστος έχουμε:



Άσκηση 3

Έχουμε το αλφάβητο $\Sigma = \{A, G, C, T, E, F\}$

Ερώτημα 1

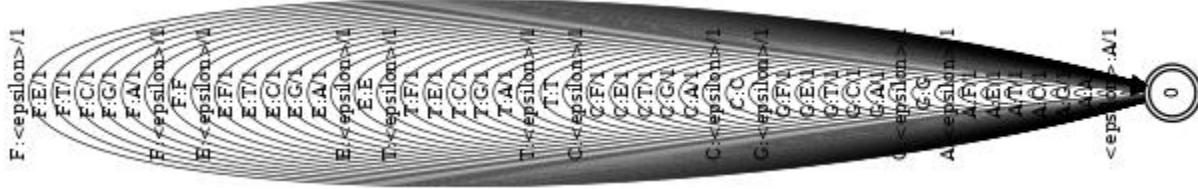
Θέλουμε έναν transducer που να υλοποιεί την απόσταση Levenshtein. Είναι ένα αυτόματο με transitions που παίρνουν input και δίνουν output κάποιο γράμα του Σ , με βάρος 0 αν το input ταυτίζεται με το output και 1 σε άλλη περίπτωση. Επίσης input και output μπορεί να είναι και το ϵ transition, αλλά όχι ταυτόχρονα input και output. Για το σχεδιασμό θα χρησιμοποιήσουμε το εργαλείο openfst. Δημιουργούμε τα αρχεία chars.syms και ex3transducer.txt που περιέχουν πληροφορίες για την αλφάβητο και τις μεταβάσεις του αυτομάτου:

chars.syms	
<epsilon> 0	
A 1	
G 2	
C 3	
T 4	
E 5	
F 6	
ex3transducer.txt	
0 0 <epsilon> A 1	0 0 T T 0
0 0 A A 0	0 0 T <epsilon> 1
0 0 A G 1	0 0 T A 1
0 0 A C 1	0 0 T G 1
0 0 A T 1	0 0 T C 1
0 0 A E 1	0 0 T E 1
0 0 A F 1	0 0 T F 1
0 0 A <epsilon> 1	0 0 T <epsilon> 1
0 0 G G 0	0 0 E E 0
0 0 G <epsilon> 1	0 0 E <epsilon> 1
0 0 G A 1	0 0 E A 1
0 0 G C 1	0 0 E G 1
0 0 G T 1	0 0 E C 1
0 0 G E 1	0 0 E T 1
0 0 G F 1	0 0 E F 1
0 0 G <epsilon> 1	0 0 E <epsilon> 1
0 0 C C 0	0 0 F F 0
0 0 C <epsilon> 1	0 0 F <epsilon> 1
0 0 C A 1	0 0 F A 1
0 0 C G 1	0 0 F G 1
0 0 C T 1	0 0 F C 1
0 0 C E 1	0 0 F T 1
0 0 C F 1	0 0 F E 1
0 0 C <epsilon> 1	0 0 F <epsilon> 1
	0

Και τρέχουμε τις εντολές

```
fstcompile --isymbols=chars.syms --osymbols=chars.syms ex3transducer.txt >
ex3transducer.fst
fstdraw --isymbols=chars.syms --osymbols=chars.syms ex3transducer.fst | dot
-Tpng > ex3transducer.png
```

Και ο transducer φαίνεται εδώ:



Ερώτημα 2

Η βέλτιστη αντιστοίχιση ανάμεσα στις γραμματοσειρές *AECAGEF* και *TETCGA* μπορεί να λυθεί με δυναμικό προγραμματισμό:

	#	A	E	C	A	G	E	F
#	0	1	2	3	4	5	6	7
T	1	1	2	3	4	5	6	7
E	2	2	1	2	3	4	4	5
T	3	3	2	2	3	4	5	6
C	4	4	3	2	3	4	5	6
G	5	5	4	3	3	3	4	5
A	6	5	5	4	3	4	4	5
G	7	6	6	5	4	3	4	5

Οι αντικαταστάσεις που έγιναν, κάνοντας backtracking στον πίνακα που σχηματίσαμε είναι:

1. $F \rightarrow G, cost = 1$
2. $E \rightarrow A, cost = 1$
3. $G \rightarrow G, cost = 0$
4. $A \rightarrow C, cost = 1$
5. $C \rightarrow T, cost = 1$
6. $E \rightarrow E, cost = 0$
7. $A \rightarrow T, cost = 1$

Οπότε η βέλτιστη αντιστοίχιση είναι αυτή που φαίνεται πάνω και έχει κόστος 5.

Ερώτημα 3

Η δεύτερη καλύτερη αντιστοίχιση (με το αμέσως μεγαλύτερο κόστος), χρησιμοποιώντας τον πίνακα του ερωτήματος 2, είναι η εξής:

	#	A	E	C	A	G	E	F
#	0	1	2	3	4	5	6	7
T	1	1	2	3	4	5	6	7
E	2	2	1	2	3	4	4	5
T	3	3	2	2	3	4	5	6
C	4	4	3	2	3	4	5	6
G	5	5	4	3	3	3	4	5
A	6	5	5	4	3	4	4	5
G	7	6	6	5	4	3	4	5

Αυτήν τη φορά, αντί να αντικαταστήσουμε $F \rightarrow G$, αντικαθιστούμε $F \rightarrow \varepsilon$ και $\varepsilon \rightarrow G$. Με αυτόν τον τρόπο, έχουμε εγγυημένα τη δεύτερη καλύτερη αντιστοίχιση, αφού με αυτήν την παράκαμψη έχουμε μία μονάδα κόστος παραπάνω από τη βέλτιστη, δηλαδή κόστος 6.

Άσκηση 4

Έχουμε το all – pole μοντέλο με συνάρτηση μεταφοράς:

$$V(z) = \frac{1}{\prod_{k=1}^q (1 - c_k z^{-1})(1 - c_k^* z^{-1})}, c_k = r_k e^{j\theta_k}$$

Ο λογάριθμος της συνάρτησης μεταφοράς είναι:

$$\ln(V(z)) = - \sum_{k=1}^q [\ln(1 - c_k z^{-1}) + \ln(1 - c_k^* z^{-1})]$$

Αναπτύσσοντας σε σειρά Taylor, έχουμε

$$\begin{aligned} \ln(V(z)) &= \sum_{k=1}^q \left[\sum_{n=1}^{\infty} \frac{(c_k z^{-1})^n}{n} + \sum_{n=1}^{\infty} \frac{(c_k^* z^{-1})^n}{n} \right] = \sum_{k=1}^q \left[\sum_{n=1}^{\infty} \frac{z^{-n} r_k^n e^{j\theta_k n}}{n} + \sum_{n=1}^{\infty} \frac{z^{-n} r_k^n e^{-j\theta_k n}}{n} \right] = \\ &= \sum_{n=1}^{\infty} \sum_{k=1}^q \frac{z^{-n} r_k^n (e^{j\theta_k n} + e^{-j\theta_k n})}{n} = \sum_{n=1}^{\infty} \sum_{k=1}^q \frac{z^{-n} r_k^n 2 \cos(\theta_k n)}{n} \xrightarrow{z^{-1}} 2 \sum_{k=1}^q \frac{r_k^n}{n} \cos(\theta_k n) \end{aligned}$$

Άρα δείξαμε ότι το cepstrum είναι:

$$\hat{v}(n) = 2 \sum_{k=1}^q \frac{r_k^n}{n} \cos(\theta_k n)$$

Άσκηση 5

Θεωρούμε τα σήματα φωνής $x(n), y(n), 0 \leq n \leq N - 1$ και τις αυτοσυσχετίσεις:

$$R_x(k) = \sum_{n=0}^{N-1-k} x(n)x(n+k), R_y(k) = \sum_{n=0}^{N-1-k} y(n)y(n+k)$$

Με βέλτιστους *LPC* συντελεστές από τη μέθοδο Levinson:

$$a_x = (a_{x0}, a_{x1}, \dots, a_{xp}), a_y = (a_{y0}, a_{y1}, \dots, a_{yp}), a_{x0} = a_{y0} = -1$$

Ερώτημα 1

Έχουμε από τον ορισμό της ενέργειας λάθους πρόβλεψης για το σήμα x :

$$\begin{aligned} E_x &= \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1+p} \sum_{k=0}^p (a_{xk}(n-k))^2 = \sum_{n=0}^{N-1+p} \left(\sum_{k=0}^p a_{xk}(n-k) \right) \left(\sum_{l=0}^p a_{xl}(n-l) \right) = \\ &= \sum_{k=0}^p a_{xk} \sum_{n=0}^{N-1+p} (n-k) \left(\sum_{l=0}^p a_{xl}(n-l) \right) = \sum_{k=0}^p a_{xk} \sum_{l=0}^p a_{xl} \sum_{n=0}^{N-1+p} (n-k)(n-l) \quad (1) \end{aligned}$$

Από τον ορισμό της αυτοσυσχέτισης, έχουμε:

$$R_x(k-l) = \sum_{n=0}^{N-1+p} x(n)x(n+k-l) = \sum_{n=0}^{N-1+p} x(m-k)x(m-l)$$

Οπότε η (1) θα γίνει:

$$E_x = \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1+p} \left(\sum_{k=0}^p a_{xk}(n-k) \right)^2 = \sum_{k=0}^p a_{xk} \sum_{l=0}^p a_{xl} R_x(k-l)$$

Το οποίο σε μορφή πινάκων γράφεται:

$$E_x = \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1+p} \left(\sum_{k=0}^p a_{xk}(n-k) \right)^2 = a_x R_x a_x^T$$

Και ο R_x είναι τετραγωνικός με διάσταση $p - 0 + 1 = p + 1$:

$$R_x = \begin{bmatrix} R_x(0) & R_x(1) & \dots & R_x(p) \\ R_x(1) & R_x(0) & \dots & R_x(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(p) & R_x(p-1) & \dots & R_x(0) \end{bmatrix}$$

Ερώτημα 2

Εργαζόμεστε παρόμοια και έχουμε για την ενέργεια του υβριδικού λάθους από τη γραμμική πρόβλεψη του x με τους συντελεστές του y :

$$\begin{aligned}
 E_{xy} &= \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1+p} \sum_{k=0}^p (a_{yk}(n-k))^2 = \sum_{n=0}^{N-1+p} \left(\sum_{k=0}^p a_{yk}(n-k) \right) \left(\sum_{l=0}^p a_{yl}(n-l) \right) = \\
 &= \sum_{k=0}^p a_{yk} \sum_{n=0}^{N-1+p} (n-k) \left(\sum_{l=0}^p a_{yl}(n-l) \right) = \sum_{k=0}^p a_{yk} \sum_{l=0}^p a_{yl} \sum_{n=0}^{N-1+p} (n-k)(n-l) = \\
 &= \sum_{k=0}^p a_{yk} \sum_{l=0}^p a_{yl} R_x(k-l) = a_y R_x a_y^T
 \end{aligned}$$

Ερώτημα 3

Προβλέποντας το σήμα x με τους βέλτιστους LPC συντελεστές του y , η ενέργεια του νέου υβριδικού λάθους πρόβλεψης θα είναι μεγαλύτερη ή ίση από την E_x . Η ισότητα ισχύει μόνο αν

$$(a_{x0}, a_{x1}, \dots, a_{xp}) = (a_{y0}, a_{y1}, \dots, a_{yp})$$

Άρα είναι $E_{xy} \geq E_x \Rightarrow E_{xy}/E_x \geq 1$, οπότε το πεδίο τιμών είναι το $[1, +\infty)$.

Άσκηση 6

Θεωρούμε τη μοντελοποίηση εναλλαγής άφωνων και έμφωνων ήχων με ένα *HMM* μοντέλο 4 καταστάσεων με τις πιθανότητες:

	<i>State 1</i>	<i>State 2</i>	<i>State 3</i>	<i>State 4</i>
$P(V)$	0.5	0.8	0.25	0.2
$P(U)$	0.5	0.2	0.75	0.8

Έχουμε ακόμα τις πιθανότητες μετάβασης καταστάσεων:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 0.25 & 0.2 & 0.3 & 0.25 \\ 0.2 & 0.25 & 0.3 & 0.25 \\ 0.4 & 0.2 & 0.2 & 0.2 \\ 0.25 & 0.3 & 0.2 & 0.25 \end{bmatrix}$$

Και τις πιθανότητες αρχικής κατάστασης $\pi_i = 0.25, i = 1, 2, 3, 4$

Παρατηρούμε την ακολουθία $O_1 O_2 O_3$: $\mathbf{O} = (UVU)$

Ερώτημα 1

Από τα δεδομένα, έχουμε για την αρχικοποίηση:

$$\begin{aligned} v_1(1) &= \pi_1 \cdot P(U|1) = 0.25 \cdot 0.5 = 0.125 \\ v_1(2) &= \pi_2 \cdot P(U|2) = 0.25 \cdot 0.2 = 0.05 \\ v_1(3) &= \pi_3 \cdot P(U|3) = 0.25 \cdot 0.75 = 0.1875 \\ v_1(4) &= \pi_4 \cdot P(U|4) = 0.25 \cdot 0.8 = 0.2 \end{aligned}$$

Ο αλγόριθμος Viterbi μας δίνει: $v_t(j) = \max_{1 \leq i \leq N} v_{t-1}(i) a_{ij} b_j(O_t)$

$$\begin{aligned} v_2(1) &= \max_{1 \leq i \leq N} \{v_1(i) a_{i1} b_1(O_1)\} = \\ &= \max\{v_1(1) a_{11} P(V|1), v_1(2) a_{21} P(V|1), v_1(3) a_{31} P(V|1), v_1(4) a_{41} P(V|1)\} = \\ &= \max\{v_1(1) a_{11} P(V|1), v_1(2) a_{21} P(V|1), v_1(3) a_{31} P(V|1), v_1(4) a_{41} P(V|1)\} = \\ &= \max\{0.015625, 0.005, 0.0375, 0.025\} = 0.0375 \end{aligned}$$

...Τα υπόλοιπα βήματα θα τα υλοποιήσει ο υπολογιστής για εμάς :)

Χρησιμοποιούμε python:

```
def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}]
    path = {}

    for y in states:
        V[0][y] = start_p[y] * emit_p[y][obs[0]]
        path[y] = [y]

    for t in range(1, len(obs)):
        V.append({})
        newpath = {}

        for y in states:
            (prob, state) = max((V[t-1][y0] * trans_p[y0][y] *
emit_p[y][obs[t]], y0) for y0 in states)
            V[t][y] = prob
            newpath[y] = path[state] + [y]

        path = newpath
    n = 0
    if len(obs) != 1:
        n = t
    print_dptable(V)
    (prob, state) = max((V[n][y], y) for y in states)
    return (prob, path[state])

def print_dptable(V):
    s = " " + " ".join("%7d" % i for i in range(len(V))) + "\n"
    for y in V[0]:
        s += "%.5s: " % y
        s += " ".join("%.7s" % ("%f" % v[y]) for v in V)
        s += "\n"
    print(s)
```

Source: <https://notebook.community/superbock/parallel2015/viterbi>

Ορίζουμε τα states, observations, αρχικές πιθανότητες, πιθανότητες μεταβάσεων και τις πιθανότητες του *HMM* και τρέχουμε το πρόγραμμα:

```

states = ('State1', 'State2', 'State3', 'State4')

observations = ('U', 'V', 'U')

start_probability = {'State1' : 0.25,
                     'State2' : 0.25,
                     'State3' : 0.25,
                     'State4' : 0.25}

transition_probability = {
    'State1' : {'State1' : 0.25,
                'State2' : 0.2,
                'State3' : 0.3,
                'State4' : 0.25},
    'State2' : {'State1' : 0.2,
                'State2' : 0.25,
                'State3' : 0.3,
                'State4' : 0.25},
    'State3' : {'State1' : 0.4,
                'State2' : 0.2,
                'State3' : 0.2,
                'State4' : 0.2},
    'State4' : {'State1' : 0.25,
                'State2' : 0.3,
                'State3' : 0.2,
                'State4' : 0.25}}

emission_probability = {
    'State1' : {'V': 0.5, 'U': 0.5},
    'State2' : {'V': 0.8, 'U': 0.2},
    'State3' : {'V': 0.25, 'U': 0.75},
    'State4' : {'V': 0.2, 'U': 0.8}}

mostProbableSequence = viterbi(observations, states, start_probability,
                               transition_probability, emission_probability)

print("Most probable state sequence is:", mostProbableSequence)

```

Τα αποτελέσματα φαίνονται παρακάτω:

```

      0      1      2
State: 0.12500 0.03750 0.00480
State: 0.05000 0.04800 0.00240
State: 0.18750 0.01000 0.01080
State: 0.20000 0.01000 0.00960

Most probable state sequence is: (0.0108, ['State4', 'State2', 'State3'])

```

Οπότε ο πίνακας μετά τον αλγόριθμο Viterbi είναι:

	1: U	2: V	3: U
<i>State 1</i>	$v_1(1) = 0.125$	$v_2(1) = 0.0375$	$v_3(1) = 0.0048$
<i>State 2</i>	$v_1(2) = 0.05$	$v_2(2) = 0.048$	$v_3(2) = 0.0024$
<i>State 3</i>	$v_1(3) = 0.1875$	$v_2(3) = 0.01$	$v_3(3) = 0.0108$
<i>State 4</i>	$v_1(4) = 0.2$	$v_2(4) = 0.01$	$v_3(4) = 0.0096$

Ερώτημα 2

Η πιο πιθανή ακολουθία προκύπτει από τον άνω πίνακα (και το python script):

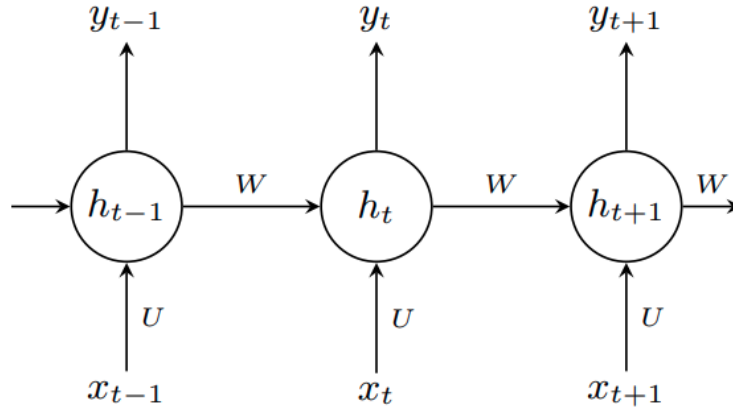
$$\mathbf{Q}^* = [\textit{State 4}, \textit{State 2}, \textit{State 3}]$$

Ερώτημα 3

Για τη ζητούμενη πιθανότητα έχουμε:

$$\mathbf{P}^* = (\mathbf{O}, \mathbf{Q}^* | \lambda) = \max\{v_3(i)\} = 0.108$$

Άσκηση 7



Ερώτημα α

Δεδομένου $y = \sigma(Wx)$, έχουμε για την παράγωγό του ως προς x :

$$\frac{\partial y}{\partial x} = \frac{\partial(\sigma(Wx))}{\partial x} = \frac{\partial\sigma(Wx)}{\partial(Wx)} \frac{\partial(Wx)}{\partial x} = \frac{\partial\sigma(Wx)}{\partial(Wx)} W$$

Θέτοντας $v = Wx$, έχουμε την ιακωβιανή:

$$\frac{\partial\sigma(Wx)}{\partial(Wx)} = \frac{\partial\sigma(v)}{\partial v} = \begin{bmatrix} \frac{\partial\sigma(v_1)}{\partial v_1} & \frac{\partial\sigma(v_1)}{\partial v_2} & \dots & \frac{\partial\sigma(v_1)}{\partial v_n} \\ \frac{\partial\sigma(v_2)}{\partial v_1} & \frac{\partial\sigma(v_2)}{\partial v_2} & \dots & \frac{\partial\sigma(v_2)}{\partial v_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\sigma(v_n)}{\partial v_1} & \frac{\partial\sigma(v_n)}{\partial v_2} & \dots & \frac{\partial\sigma(v_n)}{\partial v_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial\sigma(v_1)}{\partial v_1} & 0 & \dots & 0 \\ 0 & \frac{\partial\sigma(v_2)}{\partial v_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial\sigma(v_n)}{\partial v_n} \end{bmatrix}$$

Συνεπώς, είναι:

$$\frac{\partial y}{\partial x} = \frac{\partial\sigma(Wx)}{\partial(Wx)} W = \text{diag}(\sigma') W$$

Και έχει διαστάσεις $(n \times n) \cdot (n \times d) \rightarrow n \times d$

Ερώτημα b

Έχουμε το δεδομένο $L = \sum_{t=0}^T L_t$. Επίσης, κάθε L_t προκύπτει από την h_t , οπότε είναι:

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

Όμως, ξέρουμε ότι $h_t = \sigma(W h_{t-1} + U x_t)$, άρα έχουμε:

$$\frac{\partial h_t}{\partial W} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W} = \dots$$

Οπότε για κάθε L_t είναι:

$$\frac{\partial L_t}{\partial W} = \sum_{k=1}^t \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

Όμως για κάθε παράγωγο έχουμε:

$$\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots \frac{\partial h_{k+1}}{\partial h_k} = \prod_{i=k}^t \frac{\partial h_{i+1}}{\partial h_i}$$

Και τελικά παίρνουμε:

$$\frac{\partial L}{\partial W} = \sum_{t=0}^T \frac{\partial L_t}{\partial W} = \sum_{t=0}^T \sum_{k=1}^t \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}, \text{ όπου } \frac{\partial h_t}{\partial h_k} = \prod_{i=k}^t \frac{\partial h_{i+1}}{\partial h_i}$$

Άσκηση 8

Ερώτημα α

Forget Gate (f_t)

Η Forget Gate καθορίζει ποιες πληροφορίες από την προηγούμενη κατάσταση του cell (c_{t-1}) θα πρέπει να «ξεχάσει» το *LSTM*. Με εισόδους την προηγούμενη κρυφή κατάσταση (h_{t-1}) και την τρέχουσα είσοδο (x_t) παράγει μια τιμή μεταξύ 0 και 1. Τιμές κοντά στο 1 σημαίνουν ότι θα διατηρηθεί μεγάλο μέρος της πληροφορίας, ενώ η τιμές κοντά στο 0 σημαίνουν ότι θα απορριφθεί μεγάλο μέρος της πληροφορίας.

Input Gate (i_t)

Η Input Gate ελέγχει ποιες νέες πληροφορίες θα πρέπει να προστεθούν στην κατάσταση του cell. Κάνει 2 μετασχηματισμούς: έναν *sigmoid* και ένα *tanh*. Το *sigmoid* αποφασίζει ποια μέρη της τρέχουσας εισόδου (x_t) θα πρέπει να ενημερωθούν, ενώ το *tanh* δημιουργεί ένα διάνυσμα νέων candidate τιμών (\tilde{c}_t). Με λίγα λόγια, καθορίζει τη συμβολή των candidate τιμών στη νέα κατάσταση του cell.

Output Gate (o_t)

Η Output Gate καθορίζει ποιες πληροφορίες από την κατάσταση του cell (c_t) θα πρέπει να περάσουν στην κρυφή κατάσταση (h_t). Κάνει επίσης 2 μετασχηματισμούς: έναν *sigmoid* και ένα *tanh*. Το *sigmoid* αποφασίζει ποια μέρη της κατάστασης του cell θα πρέπει να προχωρούν και το *tanh* κλιμακώνει τις τιμές αυτές στο διάστημα $[-1, 1]$. Με λίγα λόγια, ελέγχει τη ροή των πληροφοριών από την κατάσταση του κυττάρου στην έξοδο.

Συνολικά, οι πύλες επιτρέπουν στα *LSTMs* να θυμούνται ή να ξεχνούν επιλεκτικά πληροφορίες, να ενημερώνουν την κατάσταση των cells με νέες πληροφορίες και να εξάγουν επιθυμητές πληροφορίες. Έτσι, τα *LSTMs* μπορούν να αποτυπώσουν τις εξαρτήσεις στα διαδοχικά δεδομένα.

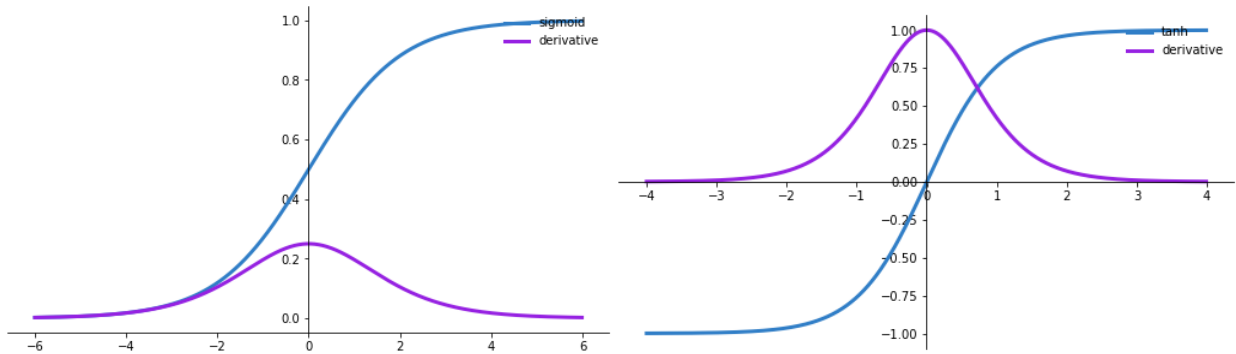
Ερώτημα b

Και οι τρεις πύλες (f_t, i_t, o_t) έχουν μη αρνητικές τιμές, αφού οι τιμές τους είναι απεικονίσεις της σιγμοειδούς συνάρτησης, που έχει πεδίο τιμών το $[0, +\infty)$.

Ερώτημα c

Φυσικά και μας ενδιαφέρει η εξάρτηση του c_t από τους όρους f_t, i_t, o_t . Αν δεν τους αγνοούσαμε, θα είχαμε αθροίσματα γινομένων των συναρτήσεων σ, \tanh και των παραγώγων αυτών. Όπως φαίνεται και στα παρακάτω διαγράμματα, πληροφορία μέσα από την παράγωγο

της \tanh δε χάνεται μόνο αν η είσοδος είναι 0 (αλλιώς κλιμακώνεται σε κάτι μικρότερο του 1), ενώ με τη σιγμοειδή είμαστε «καταδικασμένοι» να χάσουμε τουλάχιστον το 75% (για είσοδο 0). Έτσι, φανερό είναι ότι η αναδρομική εφαρμογή των συναρτήσεων αυτών – και ακόμα χειρότερα το γινόμενό τους – οδηγεί σε απώλεια τεράστιου μέρους της πληροφορίας, δηλαδή στο φαινόμενο vanishing gradient.



Source: <https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/>

Ερώτημα d

Δεδομένων:

$$\frac{\partial C_t}{\partial C_k} = \prod_{i=k+1}^t \frac{\partial C_i}{\partial C_{i-1}}, f_t = 1, i_t = 0$$

Έχουμε:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \xrightarrow{f_t=1, i_t=0} C_t = C_{t-1}, \forall t$$

Χρησιμοποιώντας τις 2 αυτές σχέσεις, παίρνουμε:

$$\frac{\partial C_t}{\partial C_k} = \prod_{i=k+1}^t \frac{\partial C_i}{\partial C_{i-1}} \xrightarrow{\frac{\partial C_i}{\partial C_{i-1}} = 1} \frac{\partial C_t}{\partial C_k} = \prod_{i=k+1}^t 1 = 1 \Rightarrow \frac{\partial C_t}{\partial C_k} = 1$$

Ερώτημα e (Bonus)

Αρχικά είναι:

$$\frac{\partial C_t}{\partial C_{t-1}} = \frac{\partial C_t}{\partial f_t} \frac{\partial f_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial i_t} \frac{\partial i_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial \tilde{C}_t} \frac{\partial \tilde{C}_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial C_{t-1}} \quad (1)$$

Επίσης για τους επιμέρους όρους έχουμε:

$$\frac{\partial C_t}{\partial f_t} = \frac{\partial(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t)}{\partial f_t} = C_{t-1} \quad (2)$$

$$\frac{\partial f_t}{\partial h_{t-1}} = \frac{\partial(\sigma(W_f h_{t-1} + U_f x_t))}{\partial h_{t-1}} = \sigma'(\cdot) W_f \quad (3)$$

$$\begin{aligned} \frac{\partial h_{t-1}}{\partial C_{t-1}} &= \frac{\partial(o_{t-1} \odot \tanh(C_{t-1}))}{\partial C_{t-1}} = \frac{\partial o_{t-1}}{\partial C_{t-1}} \odot \tanh(C_{t-1}) + o_{t-1} \odot \frac{\partial \tanh(C_{t-1})}{\partial C_{t-1}} = \\ &= \frac{\partial(\sigma(W_o h_{t-2} + U_o x_{t-1}))}{\partial C_{t-1}} \tanh(C_{t-1}) + \delta = 0 + \delta = \delta = o_{t-1} \odot \tanh'(C_{t-1}) \quad (4) \end{aligned}$$

$$\frac{\partial C_t}{\partial i_t} = \frac{\partial(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t)}{\partial i_t} = \tilde{C}_t \quad (5)$$

$$\frac{\partial i_t}{\partial h_{t-1}} = \frac{\partial(\sigma(W_i h_{t-1} + U_i x_t))}{\partial h_{t-1}} = \sigma'(\cdot) W_i \quad (6)$$

$$\frac{\partial C_t}{\partial \tilde{C}_t} = \frac{\partial(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t)}{\partial \tilde{C}_t} = i_t \quad (7)$$

$$\frac{\partial \tilde{C}_t}{\partial h_{t-1}} = \frac{\partial(\tanh(W_c h_{t-1} + U_c x_t))}{\partial h_{t-1}} = \tanh'(\cdot) W_c \quad (8)$$

$$\frac{\partial C_t}{\partial C_{t-1}} = \frac{\partial(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t)}{\partial C_{t-1}} = f_t \quad (9)$$

Η (1) με τη βοήθεια των (2, 3, ..., 9) μας δίνει:

$$\begin{aligned} \frac{\partial C_t}{\partial C_{t-1}} &= \frac{\partial C_t}{\partial f_t} \frac{\partial f_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial i_t} \frac{\partial i_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial \tilde{C}_t} \frac{\partial \tilde{C}_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial C_{t-1}} \Rightarrow \\ &\Rightarrow \boxed{\frac{\partial C_t}{\partial C_{t-1}} = \sigma'(\cdot) W_f \delta \odot C_{t-1} + \sigma'(\cdot) W_i \delta \tilde{C}_t + i_t \odot \tanh(\cdot) \delta + f_t} \end{aligned}$$

Έχουμε τις εξής παρατηρήσεις για τα hidden/cell states:

1. Hidden States: Οι παράγωγοι $\frac{\partial h_t}{\partial h_{t-1}}$ από ένα σημείο και ύστερα θα λαμβάνουν τιμές είτε μόνο μεγαλύτερες του 1, είτε μόνο μικρότερες του 1 (θετικές). Θα έχουμε δηλαδή φαινόμενα exploding ή vanishing gradient, αντίστοιχα.
2. Cell States: Οι παράγωγοι $\frac{\partial C_t}{\partial C_{t-1}}$ μπορούν να λάβουν τιμές χωρίς τον προηγούμενο περιορισμό. Έχουμε δηλαδή την ευχέρεια να ορίζουμε το f_t κατά πώς βολεύει ώστε να αποφεύγουμε τα προβλήματα exploding/vanishing gradient.

Άσκηση 9

Έχουμε το τμήμα στατιστικής σημασιολογικής γραμματικής.

Ερώτημα 1

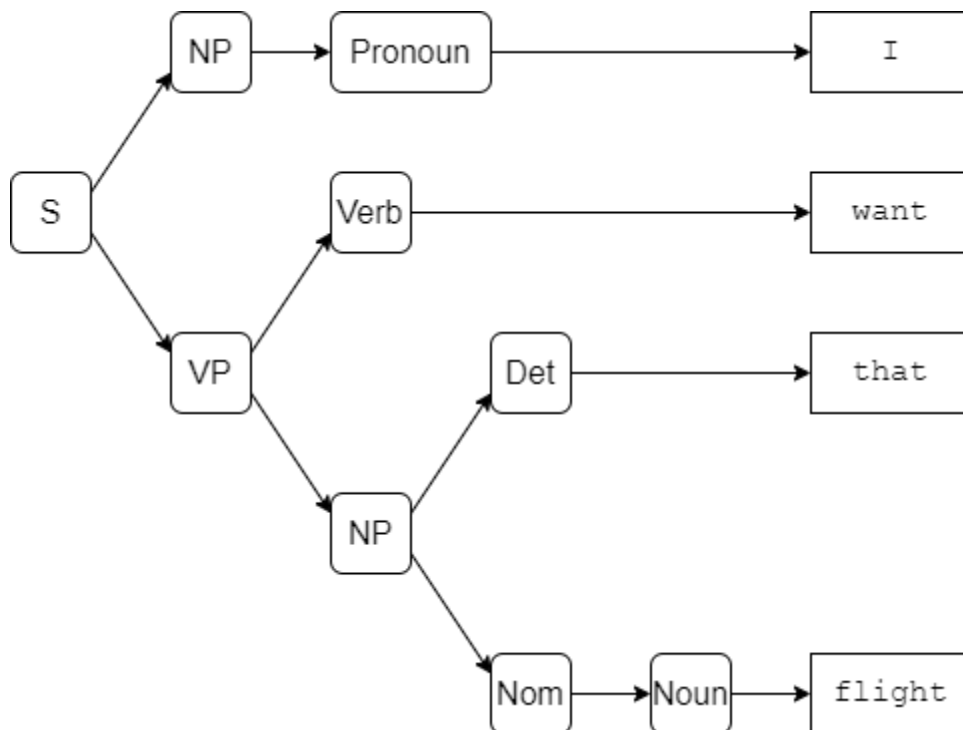
Παρατηρούμε ότι το άθροισμα των πιθανοτήτων των κανόνων με αριστερό σύμβολο το S είναι 0.80 (ένας μοναδικός κανόνας) και όχι 1. Αυτό σημαίνει ότι η γραμματική δεν είναι πλήρης ως προς τους κανόνες με αριστερό σύμβολο το S .

Παρομοίως, το άθροισμα των πιθανοτήτων των κανόνων με αριστερό σύμβολο το NP είναι $0.20 + 0.35 + 0.05 + 0.40 = 1.00$. Άρα η γραμματική είναι πλήρης ως προς τους κανόνες με αριστερό σύμβολο το NP .

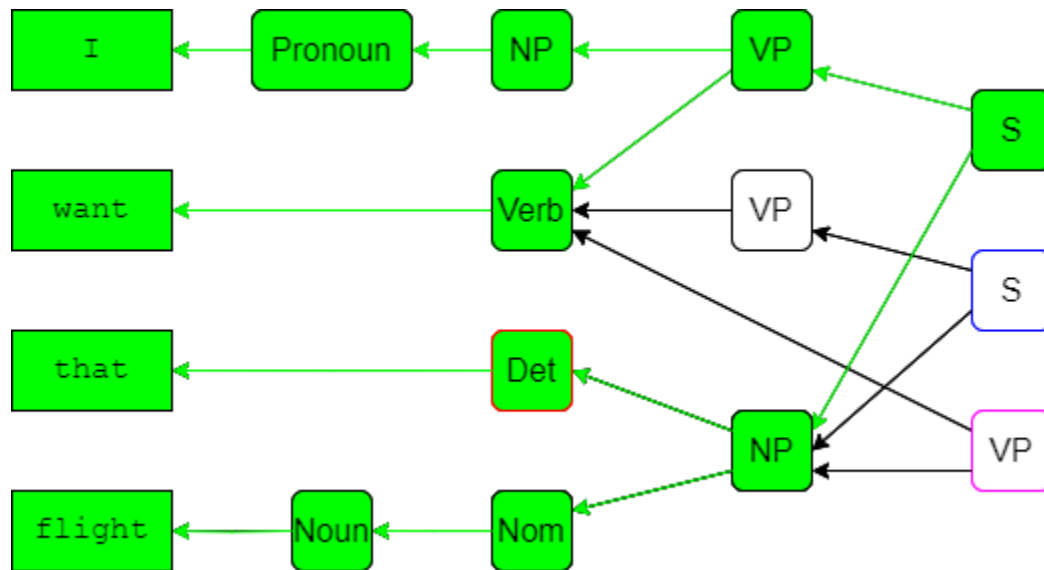
$S \rightarrow NP VP$	[0.80]
$NP \rightarrow Det Nom$	[0.20]
$NP \rightarrow ProperNoun$	[0.35]
$NP \rightarrow Nom$	[0.05]
$NP \rightarrow Pronoun$	[0.40]
$VP \rightarrow Verb$	[0.55]
$VP \rightarrow Verb NP$	[0.40]
$Verb \rightarrow want$	[0.40]
$Nom \rightarrow Noun$	[0.75]
$Pronoun \rightarrow I$	[0.60]
$Pronoun \rightarrow you$	[0.40]
$Det \rightarrow the$	[0.80]
$Det \rightarrow that$	[0.05]
$Noun \rightarrow flight$	[0.50]

Ερώτημα 2

Για κάθε token της πρότασης *I want that flight* (φύλλο) προχωράμε χρησιμοποιώντας αντίστροφα τους κανόνες παραγωγής και κατασκευάζουμε το δέντρο προς τα πάνω.



Για να εξετάσουμε αν είναι μοναδικό, θα κατασκευάζουμε το δέντρο προς τα πάνω (πάλι), αλλά αυτήν τη φορά θα πάρουμε όλα τα δυνατά μονοπάτια και θα ελέγξουμε αν είναι δυνατό να υφίστανται πάνω από 1 μορφές του δέντρου:



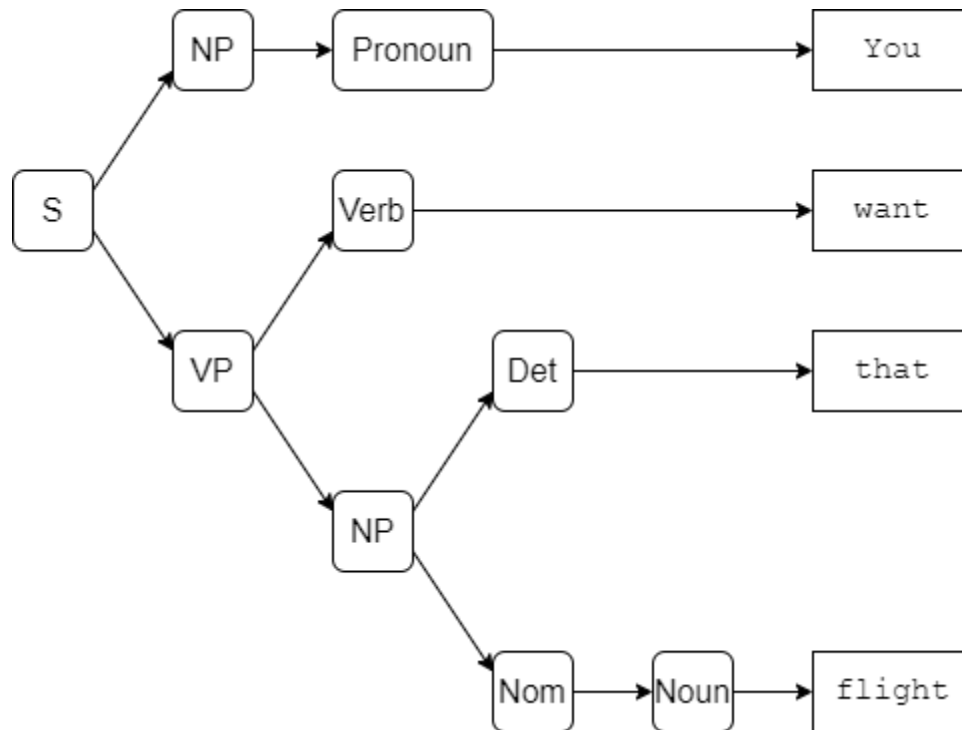
Ξεκινάμε από τα φύλλα. Προχωράμε προς τα δεξιά με τους κανόνες που τα παράγουν. Εξηγούμε τους περιορισμούς:

1. Κόκκινο περίγραμμα: Το *Det* απαιτείται να υπάρχει γιατί αλλιώς δεν μπορεί να παραχθεί το *that*. Αλλά για να υπάρχει το *Det*, απαιτείται ένα *NP* που θα παράξει ταυτόχρονα *Det* και *Nom*.
2. Μπλε περίγραμμα: Το συγκεκριμένο *S* παράγει δέντρο που δεν οδηγεί στην παραγωγή του *I*, άρα δεν ταιριάζει στην πρότασή μας.
3. Μωβ περίγραμμα: Αυτό το *VP* δεν είναι δυνατόν να υπάρχει γιατί απαιτεί ένα *S* που θα παράγει ταυτόχρονα *VP* και *NP*. Όμως η τοπολογία του δέντρου, αφήνει το συγκεκριμένο *VP* χωρίς κάποιο πιθανό *NP* (αφού το παράγει) ώστε να παραχθούν από κάποιο *S*. Άρα, ούτε αυτή η έκδοση υφίσταται.

Έχοντας παράξει όλα τα πιθανά δέντρα, αφού έμεινε μόνο ένα που βγάζει νόημα (το πράσινο), αποφαινόμεστε ότι δεν υπάρχει αμφισημία.

Ερώτημα 3

Σχηματίζουμε και το δέντρο της πρότασης *you want that flight*:



Και υπολογίζουμε τις πιθανότητες για τα δύο δέντρα, ως γινόμενο των επιμέρους πιθανοτήτων, ύστερα από την εφαρμογή κάθε κανόνα:

Για την πρόταση *I want that flight* έχουμε πιθανότητα:

$$P = 1 \cdot 0.8 \cdot 0.4 \cdot 0.6 \cdot 0.4 \cdot 0.4 \cdot 0.2 \cdot 0.05 \cdot 0.75 \cdot 0.5 = 0.0001152$$

Για την πρόταση *You want that flight*

$$P = 1 \cdot 0.8 \cdot 0.4 \cdot 0.4 \cdot 0.4 \cdot 0.4 \cdot 0.2 \cdot 0.05 \cdot 0.75 \cdot 0.5 = 0.0000768$$

Επαληθεύουμε τα αποτελέσματά μας με Python!

```
import nltk
from nltk.tree import Tree

def draw_tree(tree_string):
    tree = Tree.fromstring(tree_string)
    tree.pretty_print()

grammar = nltk.CFG.fromstring("""
S -> NP VP
NP -> Det Nom
NP -> Proper Noun
NP -> Nom
NP -> Pronoun
VP -> Verb
VP -> Verb NP
Verb -> 'want'
Nom -> Noun
Pronoun -> 'I'
Pronoun -> 'you'
Det -> 'the'
Det -> 'that'
Noun -> 'flight'
""")

print("+-----+")
print("| For sentence 'I want that flight' |")
print("+-----+")

sent = ['I', 'want', 'that', 'flight']

parser = nltk.ChartParser(grammar)
parse_trees = list(parser.parse(sent))

if len(parse_trees) == 1:
    print("The sentence has a unique parse tree:")
    draw_tree(str(parse_trees[0]))
else:
    print("The sentence has multiple parse trees:")
    for i, tree in enumerate(parse_trees):
        print(f"Parse tree {i+1}:")
        draw_tree(str(tree))

print("+-----+")
print("| For sentence 'You want that flight' |")
print("+-----+")

sent = ['you', 'want', 'that', 'flight']

parser = nltk.ChartParser(grammar)
parse_trees = list(parser.parse(sent))

if len(parse_trees) == 1:
    print("The sentence has a unique parse tree:")
    draw_tree(str(parse_trees[0]))
else:
    print("The sentence has multiple parse trees:")
    for i, tree in enumerate(parse_trees):
        print(f"Parse tree {i+1}:")
        draw_tree(str(tree))
```

Και έχουμε τα ακόλουθα δέντρα (και την επιβεβαίωση ότι είναι μοναδικά):

