HY437 – Αλγόριθμοι CAD

Διδάσκων: Χ. Σωτηρίου

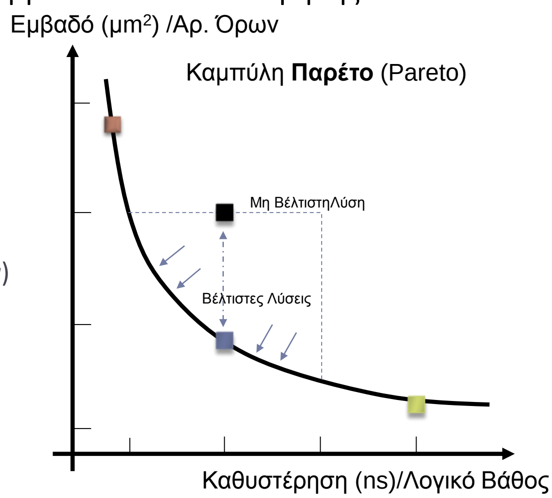
http://inf-server.inf.uth.gr/courses/CE437/

Περιεχόμενα

- Εισαγωγή στην Πολύ-επίπεδη Λογική Σύνθεση
  - Τρόποι αναπαράστασης, Κόστος: Βάθος και Εμβαδό
- Δυαδικό Δίκτυο (Boole) και Μετασχηματισμοί
  - Ορισμός/Παραδείγματα των μετασχηματισμών/πράξεων: `ELIMINATION`, `DECOMPOSITION`, `EXTRACTION`, `SIMPLIFICATION` και `SUBSTITUTION`
- Αλγεβρική και Δυαδική Παραγοντοποίηση
  - Διαίρεση Δυαδικής Συνάρτησης – Αλγεβρική και Δυαδική
  - Πυρήνες και Συν-Πυρήνες – Αλγόριθμοι Υπολογισμού
- Αμοιβαία Παραγοντοποίηση Συναρτήσεων
  - Εξαγωγή κοινών κύβων ή Πυρήνων: Αλγόριθμοι Κάλυψης Ορθογωνίων
- Υλοποίηση των Πράξεων/Μετασχηματισμών

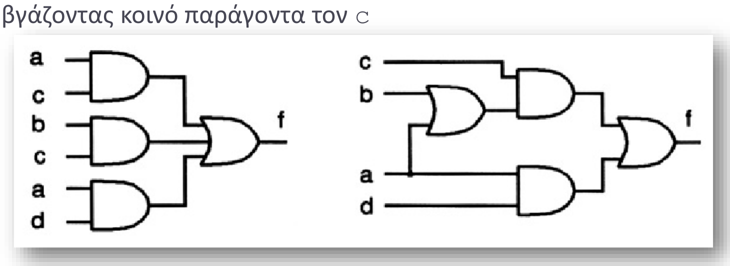
Βασικός Στόχος Πολύ-επίπεδης Λογικής Σύνθεσης

- Συμψηφισμός μεταξύ Εμβαδού και Καθυστέρησης
- Εμβαδό
  - Αριθμός Όρων (Literal Count)
  - μm^2
- Καθυστέρηση
  - Λογικό Βάθος (Αριθμός Παρενθέσεων)
  - ns
- Η επιλογή της λύσης γίνεται βάση περιορισμών: **f/T, A, P**



Αναδόμηση – Παραγοντοποίηση Συνάρτησης

- Παράδειγμα:
  - `f = ac + bc + ad = c(a + b) + ad`
  - βγάζοντας κοινό παράγοντα τον c



- Άλλες Παραγοντοποιήσεις:
  - `f = ac + bc + ad = (a + bc)(c + d)= = (c + ad)(a + b)`

Αλγεβρικές καί Δυαδικές (στο $\mathbb{B}$) Παραγοντοποιήσεις

- Διαφορές μεταξύ ιδιοτήτων βασικής άλγεβρας (στο $\mathbb{R}$), αντί του $\mathbb{B}$

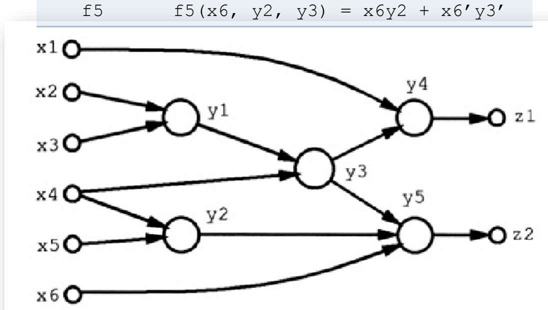
Ιδιότητα	στο R	στο B
Ταυτότητα (Idempotence)	a. a = a^2	a . a = a
Αντίστροφο	–	Για x υπάρχει x'
Επιμερισμός (Distributivity)	–	a + bc = (a + b)(a + c)
Απορρόφηση	–	a + ab = a

- Αν οι ιδιότητες που χρησιμοποιούνται για την παραγοντοποίηση μιας συνάρτησης περιλαμβάνουν *τις παραπάνω ειδικές ιδιότητες του* $\mathbb{B}$ τότε η παραγοντοποίηση ονομάζεται **Δυαδική** (Boolean), αλλιώς **Αλγεβρική**

Παράδειγμα Απλού Δικτύου Boole

- Για κάθε κόμβο y_i ορίζεται συνάρτηση fi:

Κόμβος	Ορισμός
f1	f1(x2, x3) = x2' + x3'
f2	f2(x4, x5) = x4' + x5'
f3	f3(x4, y1) = x4'y1'
f4	f4(x1, y3) = x1 + y3'
f5	f5(x6, y2, y3) = x6y2 + x6'y3'



Δίκτυο Boole και Μετασχηματισμοί

- Αρχική Δομή Δικτύου Boole – Πολύ-επίπεδες Εξισώσεις
  - PI = {a, b, c, d, e}, PO = {w, x, y, z}

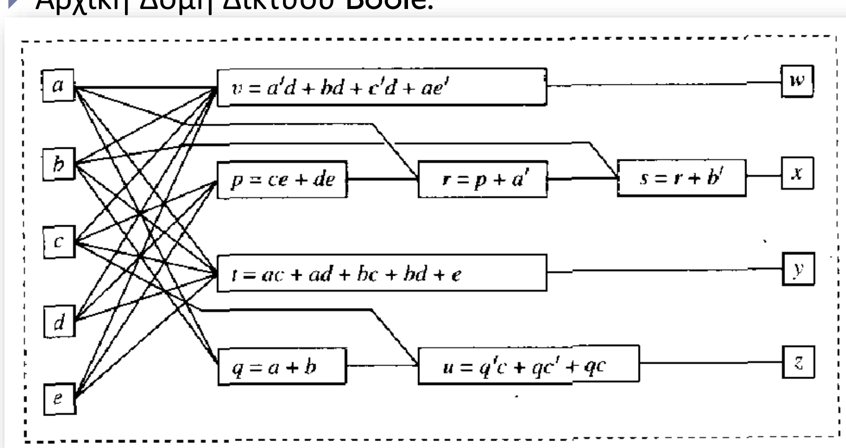
p = ce + de	v = a′d + bd + c′d + ae′
q = a + b	w = v
r = p + a′	x = s
s = r + b′	y = t
t = ac + ad + bc + bd + e	z = u
u = q′c + qc′ + qc	

- Σε επίπεδο Εισόδων/Εξόδων:

	a′d + bd + c′d + ae′
f =	a′ + b′ + ce + de
	ac + ad + bc + bd + e
	a + b + c

Δίκτυο Boole και Μετασχηματισμοί

- Αρχική Δομή Δικτύου Boole:

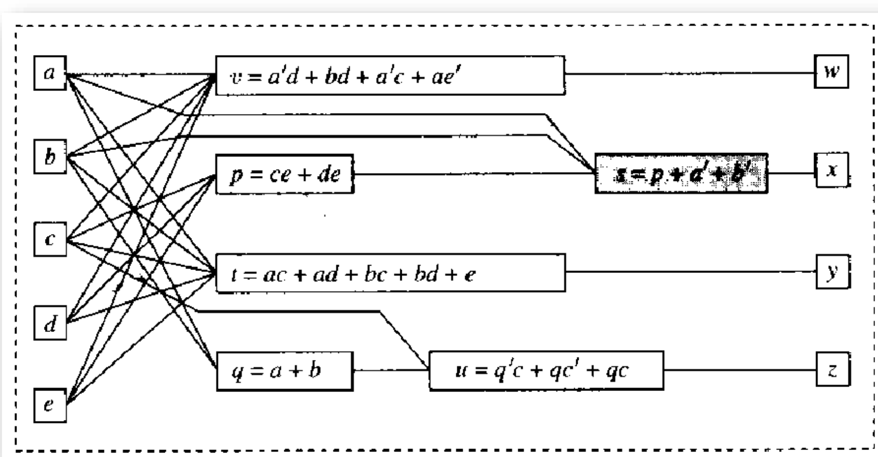


Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `ELIMINATION` – ΑΠΑΛΟΙΦΗ ΚΟΜΒΟΥ
  - Αφαίρεση εσωτερικού κόμβου από το δίκτυο
  - **Αντικατάσταση όλων των εμφανίσεων του κόμβου από τη αντίστοιχη του έκφραση**
- Παράδειγμα:
  - Στο προηγούμενο δίκτυο, έχουμε κόμβους `s`, `r`:
    - …, `s = r + b’`, `r = p + a’`, …
  - Αν αντικαταστήσουμε τον `r`, τότε όπου r θέτουμε τα περιεχόμενα του, και έτσι
    - **`s = p + a’ + b’`**

Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `ELIMINATION` – ΑΠΑΛΟΙΦΗ ΚΟΜΒΟΥ

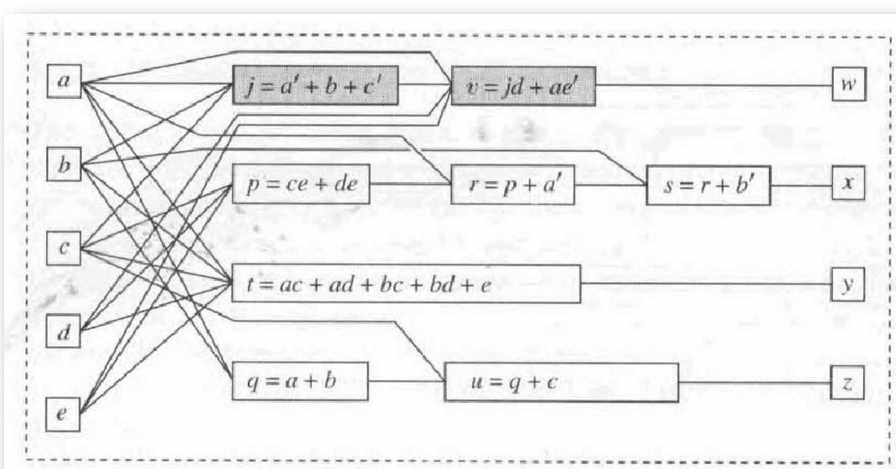


Δίκτυο Boole και Μετασχηματισμοί

- Πράξη DECOMPOSITION – ΑΠΟΣΥΝΘΕΣΗ ΚΟΜΒΟΥ
  - Από-μείωση πολύπλοκης έκφρασης σε απλούστερες
  - **Αποσύνθεση κόμβου ορίζεται ως αντικατάσταση του από 2 ή περισσότερους κόμβους που αποτελούν ένα νέο υπο-δίκτυο με ισοδύναμη συμπεριφορά με τον αρχικό**
- Παράδειγμα
  - Στο προηγούμενο δίκτυο, έχουμε κόμβο v, ως:
    - …, v = a’d + bd + c’d + ae’, …
  - Ο κόμβος v μπορεί να από-μειωθεί για μείωση των όρων του μέσω παραγοντοποίησης ως:
    - v = (a + b + c’)d + ae’
  - Και η υπο-έκφραση a + b + c’ να οριστεί ως νέος κόμβος:
    - **j = a’ + b + c’, v = jd + ae’**

Δίκτυο Boole και Μετασχηματισμοί

- Πράξη DECOMPOSITION - ΑΠΟΣΥΝΘΕΣΗ ΚΟΜΒΟΥ

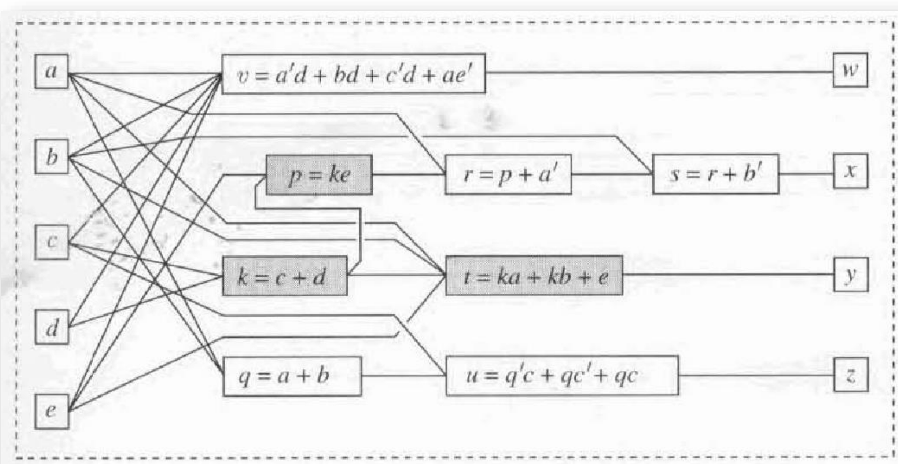


Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `EXTRACTION` – ΕΞΟΡΥΞΗ ΚΟΙΝΟΥ ΠΑΡΑΓΟΝΤΑ
  - Εύρεση/Εξόρυξη κοινού παράγοντα για 2 κόμβους, απλοποιώντας και τις 2 εκφράσεις
  - **Εξόρυξη σε σύνολο κόμβων ορίζεται ως ταυτόχρονη αποσύνθεση προσθέτοντας νέο κόμβο για τον κοινό παράγοντα**
- Παράδειγμα
  - Στο προηγούμενο δίκτυο, έχουμε κόμβους `p`, `t` ως:
    - …, `p = ce + de`, `t = ac + ad + bc + bd + e`, …
  - Παραγοντοποιόντας (από κοινού) τους δυο κόμβους:
    - …, `p = (c + d)e`, `t = (c + d)(a + b) + e`, …
  - Και εισάγουμε τον κοινό παράγοντα (c + d) ως νέο κόμβο:
    - …, `k = c + d`, `p = ke`, `t = ka + kb + e`, …

Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `EXTRACTION` - ΕΞΟΡΥΞΗ ΚΟΙΝΟΥ ΠΑΡΑΓΟΝΤΑ

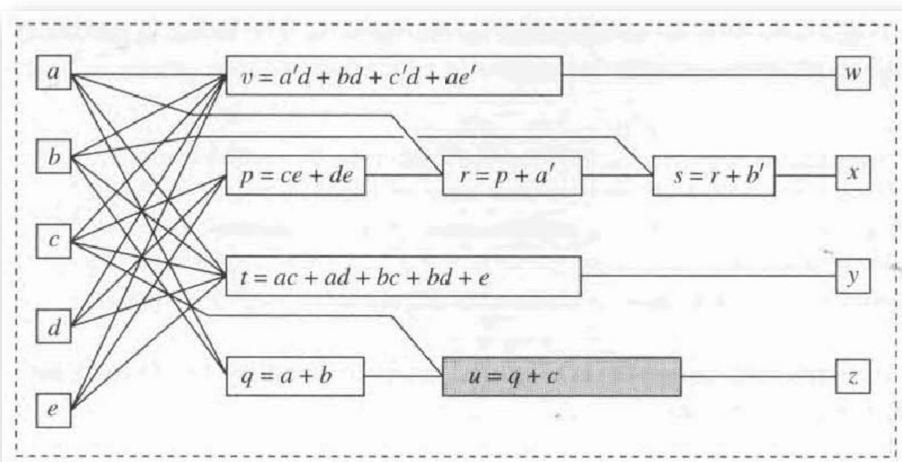


Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `SIMPLIFICATION` – ΑΠΛΟΠΟΙΗΣΗ ΚΟΜΒΟΥ
  - Βελτιστοποίηση δυαδικής έκφρασης αλλάζοντας ή όχι την στήριξη του κόμβου
  - **Η απλοποίηση κόμβου αντιστοιχεί στην ελαχιστοποίηση/βελτιστοποίηση στην δυαδική έκφραση του (π.χ. `ESPRESSO` στην τοπική συνάρτηση ή χρήση DC)**
- Παράδειγμα
  - Στο προηγούμενο δίκτυο, έχουμε κόμβο `u` ως:
    - …, `u = q'c + qc' + qc`, …
  - Μπορεί να απλοποιηθεί ως:
    - …, `u = q + c`, …
  - *Τοπική Απλοποίηση*

Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `SIMPLIFICATION` – ΑΠΛΟΠΟΙΗΣΗ ΚΟΜΒΟΥ

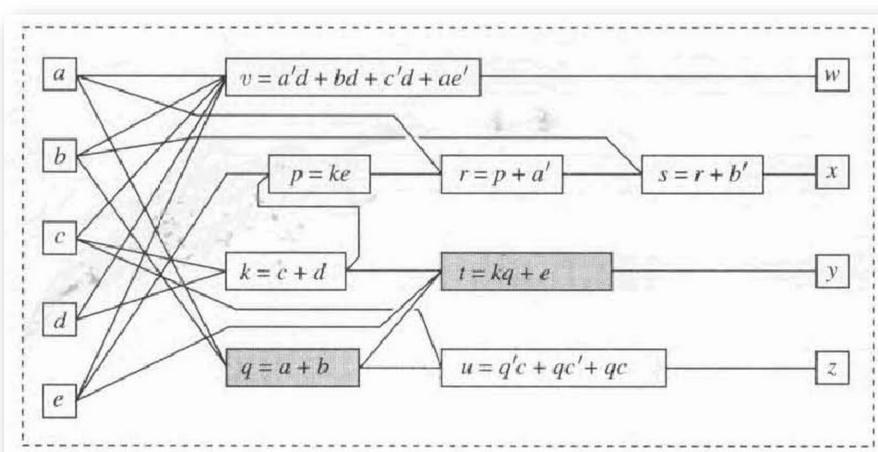


Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `SUBSTITUTION` – ΑΝΤΙΚΑΤΑΣΤΑΣΗ ΚΟΜΒΟΥ
  - ΑΠΟΣΥΝΘΕΣΗ/ΕΞΟΡΥΞΗ βάση τρεχόντων κόμβων
  - **Η αντικατάσταση αλλάζει την στήριξη ενός κόμβου, μέσω της εκμετάλλευσης των αποτελεσμάτων της ΑΠΟΣΥΝΘΕΣΗΣ/ΕΞΟΡΥΞΗΣ, χρησιμοποιώντας τους ήδη υπάρχοντες κόμβους**
- Παράδειγμα
  - Στο προηγούμενο δίκτυο, έχουμε κόμβο `v` ως:
    - `…, t = ka + kb + e, q = a + b, …`
  - Συνεπώς βλέπουμε ότι ο q μπορεί να αντικατασταθεί ως υποκόμβος του t:
    - `…, t = kq + e, …`

Δίκτυο Boole και Μετασχηματισμοί

- Πράξη `SUBSTITUTION` – ΑΝΤΙΚΑΤΑΣΤΑΣΗ ΚΟΜΒΟΥ

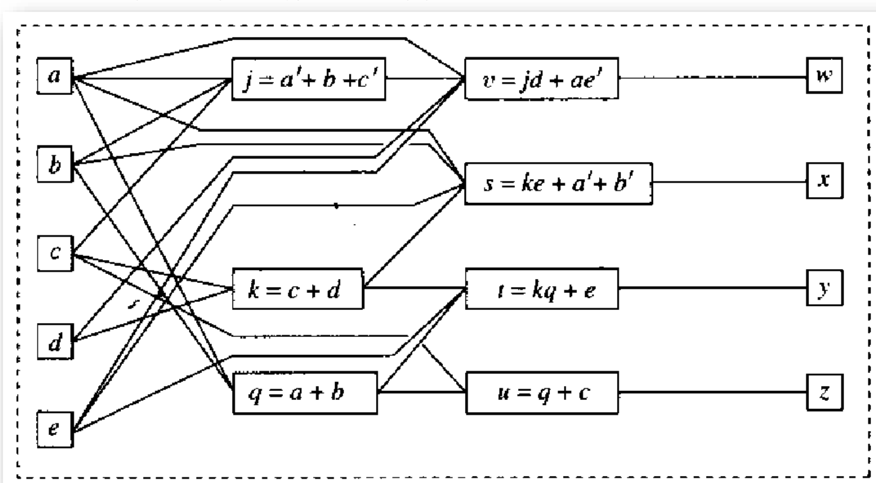


Δίκτυο Boole και Μετασχηματισμοί

- Η ευριστική πολυεπίπεδη βελτιστοποίηση είναι αντίστοιχη της διαδικασίας του `ESPRESSO`:
  - Έχουμε 5 Πράξεις
    - `ELIMINATION`, `DECOMPOSITION`, `EXTRACTION`, `SIMPLIFICATION` και `SUBSTITUTION`
  - Ένα ή περισσότερους Περιορισμούς/Συναρτήσεις Κόστους
    - **Εμβαδό (Αριθμός Όρων), Καθυστέρηση (Βάθος), Ισχύ (Μεταβάσεις)**
  - Μια ή περισσότερες Ευριστικές Στρατηγικές - *Μετα-Αλγόριθμος*
  - Εφαρμόζουμε τις πράξεις, βάση της στρατηγικής μέχρι να φτάσουμε τον στόχο ή μέχρι το κόστος να μην μειώνεται περαιτέρω…

Δίκτυο Boole και Μετασχηματισμοί

- Τελική Λύση Ελάχιστου Εμβαδού:



Δίκτυο Boole και Μετασχηματισμοί

- SIS – Στρατηγική απλοποίησης “rugged”
  - **sweep** = απαλειφή κόμβων μίας εισόδου ή σταθερών
  - **eliminate k** = απαλειφή με μέγιστη αύξηση κόστους k
  - **simplify** = απλοποίηση με χρήση ESPRESSO (nocomp = χωρίς OFF-set)
  - **full_simplify** = απλοποίηση με χρήση ESPRESSO και DCs
  - **resub -a** = αλγεβρική αντικατάσταση για κάθε ζευγάρι κόμβων
  - **fx** = εξόρυξη ενός ή δυο κύβων

script.rugged

```
sweep; eliminate -1
simplify -m nocomp
eliminate -1

sweep; eliminate 5
simplify -m nocomp
resub -a

fx
resub -a; sweep

eliminate -1; sweep
full_simplify -m nocomp
```


Ανάλυση των Πράξεων Δυαδικού Δικτύου

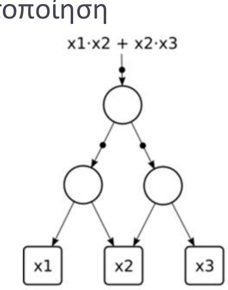
- ELIMINATION
  - Αντικατάσταση του Κόμβου – αφαίρεση επιπέδου στο δίκτυο
  - Χρειαζόμαστε πράξη ELIMINATE
- DECOMPOSITION
  - **Εξεύρεση καλού παράγοντα** για την παραγοντοποίηση του κόμβου
  - Χρειαζόμαστε πράξη *_FACTOR (ALG_FACTOR, BOOL_FACTOR)
- EXTRACTION
  - **Εξεύρεση καλού κοινού παράγοντα** για την παραγοντοποίηση πολλαπλών κόμβων
  - Χρειαζόμαστε πράξη *_COMMON_FACTOR
- SIMPLIFICATION
  - **Δυαδική απλοποίηση κόμβου** – *με εξωτερικά και εσωτερικά DC*
  - Χρειαζόμαστε πράξη ESPRESSO (την έχουμε από την 2-επίπεδη βελτιστοποίηση)
- SUBSTITUTION
  - **Διαίρεση ενός κόμβου με υπάρχοντες** για αντικατάσταση
  - Χρειαζόμαστε πράξη *_DIVISION (ALG_DIVISION, BOOL_DIVISION)

Αναπαραστάσεις Πολύ-επίπεδων Κυκλωμάτων

- SOP (Sum-of-Products)/POS – Άθροισμα/Γινόμενο Κύβων
  - Μη κανονική μορφή
  - Κατάλληλη για 2-επίπεδη βελτιστοποίηση, παραγοντοποίηση, αποσύνθεση
- Παραγοντοποιημένη Μορφή – Factored Form (**Βέλτιστη SOP ή POS**)
  - Μη κανονική επίσης μορφή
  - Ισομορφική με δέντρο που ο κάθε κόμβος είναι AND ή OR και τα φύλλα είναι οι όροι
  - Επιτρέπει καλή ακρίβεια στην προσέγγιση του εμβαδού (τρανζίστορ) από το LC
- NAND ή NOR Δίκτυα
  - Λαμβάνουν υπόψη τους αντιστροφείς, γρηγορότερα στην επεξεργασία – Δυσκολία παραγοντοποίησης/ελαχιστοποίησης

Αναπαραστάσεις Πολύ-επίπεδων Κυκλωμάτων

- OBDDs – Ordered BDDs
  - Κανονική Μορφή
  - Οικονομική Αναπαράσταση και Ευκολία στην Επαναχρησιμοποίηση Κοινών Τμημάτων (Συναρτήσεων)
  - Δυσκολία στην Ελαχιστοποίηση/Βελτιστοποίηση/Παραγοντοποίηση
- AIGs – And-Inverter Graphs
  - Μη κανονική μορφή
  - Σύνθεση = μετασχηματισμοί στα AIG
    - Απλούστεροι αλγόριθμοι
  - Διασύνδεση με την Τεχνολογική Απεικόνιση



Παραγοντοποιημένη Μορφή

- **<u>Ορισμός:</u>** Μια παραγοντοποιημένη μορφή ορίζεται αναδρομικά, σύμφωνα με τους ακόλουθους κανόνες:
  1. Ένα γινόμενο αντιστοιχεί σε μονό όρο ή γινόμενο παραγοντοποιημένων μορφών
  2. Ένα άθροισμα αντιστοιχεί σε μονό όρο ή άθροισμα παραγοντοποιημένων μορφών
  3. Μια παραγοντοποιημένη μορφή είναι άθροισμα ή γινόμενο
- Παραδείγματα:
  - `x, y', abc'`
    `a + b'c`
    `((a' + b)cd +e)(a + b') + e`
- Δεν είναι Π.Μ.: `(a + b)'c`
- Δεν είναι κανονική η αναπαράσταση, π.χ.:
  - `ab + c(a + b) = bc + a(b + c) = ac + b(a +c)`

Αλγεβρικές καί Δυαδικές (στο $\mathbb{B}$) Παραγοντοποιήσεις – Π.Μ.

- Διαφορές μεταξύ ιδιοτήτων βασικής άλγεβρας (στο $\mathbb{R}$), αντί του $\mathbb{B}$

Ιδιότητα	στο R	στο B
Ταυτότητα (Idempotence)	$a.\ a = a^2$	`a . a = a`
Αντίστροφο	–	Για x υπάρχει x'
Επιμερισμός (Distributivity)	–	`a + bc = (a + b)(a + c)`
Απορρόφηση	–	`a + ab = a`

- Αν οι ιδιότητες που χρησιμοποιούνται για την παραγοντοποίηση μιας συνάρτησης περιλαμβάνουν *τις παραπάνω ειδικές ιδιότητες του* $\mathbb{B}$ τότε η παραγοντοποίηση ονομάζεται **Δυαδική** (Boolean), αλλιώς **Αλγεβρική**

Αλγεβρικές καί Δυαδικές (στο $\mathbb{B}$) Παραγοντοποιήσεις – Π.Μ.

- **Ορισμός:** Μια έκφραση ορίζεται ως *αλγεβρική*, όταν κανένας κύβος της δεν εμπεριέχει άλλον, δηλ. $Ci \nleq Cj$, για $i \neq j$. Μια έκφραση που δεν είναι αλγεβρική ονομάζεται Boolean (Δυαδική).
- Παράδειγματα:
  - `a + bc` : αλγεβρική, `a + ab` : boolean (`a` > `ab` στην δυαδική άλγεβρα ή `ab` > `a`, αν το θεωρήσουμε ως σύνολο όρων)
- Οι αλγεβρικές μορφές ***δεν είναι πρώτες ή μη περιττές*** – μπορεί να περιέχουν περιττούς κύβους
  - `a + a'b` ή `xy + x'z + yz`

Παραδείγματα Παραγοντοποιημένων Μορφών

- Έστω η έκφραση:
  - `Q = abg + acg + adf + aef + afg + bd + ce + be + cd`
- Παραδείγματα Παραγοντοποίησεων:
  1. `(b + c)(d + e) + ((d + e + g)f + (b + c)g)a` **- 12 Όροι**
  2. `(b + c)(d + e + ag) + (d + e + g)af` **– 11 Όροι**
  3. `(af + b + c)(ag + d + e)` **– 8 Όροι**
- η Τρίτη Π.Μ. είναι **Boolean**, όχι Αλγεβρική
  - ***Απαιτούνται κανόνες της Δυαδικής Άλγεβρας για να πάρουμε την αρχική αναπαράσταση***

Κόστος Παραγοντοποίησης (Factorisation Value)

- **Ορισμός:** Το κόστος μιας αλγεβρικής παραγοντοποίησης `F = G1G2 + R` ορίζεται (και αναδρομικά) ως:
  - **FACT_VAL(F, G2) = LITS(F)-(LITS(G1) + LITS(G2) + LITS(R))**

Όπου οι `G1`, `G2`, `R` είναι αλγεβρικές εκφράσεις και
LITS(P) = Αριθμός όρων στην SOP μορφή της P

- Παράδειγμα
  - `f = ae + af + ag + bce + bcf + bcg + bde + bdf + bdg`
    - 24 Όροι
  - Μετά από παραγοντοποίηση:
  - `f = (a + b(c + d))(e + f + g)` **- 7 Όροι**

Κόστος Παραγοντοποίησης (Factorisation Value)

- Κέρδος σε όρους = 24 – 7 = **17**

- FACT_VAL(F, `(e + f + g)`) =
  LITS(F) – (LITS(`a + b(c + d)`)+ LITS(`e + f + g`)) =
  24 – (5 + 3) = **16**
- 16 αντί για 17, λόγω του LITS (όροι του SOP) και του γεγονότος ότι η (`a + b(c + d)`) είναι επίσης παραγοντοποιημένη

- Η FACT_VAL μπορεί να χρησιμοποιηθεί αναδρομικά αν η G1 είναι επίσης παραγοντοποιημένη, όπως εδώ…

Μέγιστη Παραγοντοποίηση

- **Ορισμός:** Μια Π.Μ. είναι μέγιστα παραγοντοποιημένη, όταν:
  1. Για κάθε (υποσύνολο της Π.Μ.) SOP, δεν υπάρχουν συντακτικά ισοδύναμοι παράγοντες στα γινόμενα
  2. Για κάθε (υποσύνολο της Π.Μ.) POS, δεν υπάρχουν συντακτικά ισοδύναμοι παράγοντες στα αθροίσματα

- Παράδειγματα
  - `αb + ac`, `(a + b)(a + c)`
    ΔΕΝ είναι μέγιστα παραγοντοποιημένες μορφές μια και παραγοντοποιούνται ως:
    `a(b + c)`,`a + bc`

Αλγεβρική (ή Ασθενής) Διαίρεση

- **Ορισμός:** Ορίζουμε ως Διαίρεση – `*_DIVISION` - την πράξη που για δυο SOP εκφράσεις `F` και `P`, παράγει δυο SOP εκφράσεις `Q` και `R`,
  `(Q, R) = *_DIVISION(F, P)`, έτσι ώστε
  `F = PQ + R`
- Αν `PQ` αλγεβρικό γινόμενο, τότε η διαίρεση ονομάζεται αλγεβρική, αλλιώς ονομάζεται Boolean
- Το υπόλοιπο R, ***πρέπει να έχει το λιγότερο δυνατό αριθμό κύβων***
- Αν το υπόλοιπο `R` είναι κενό, τότε ο `P` είναι παράγοντας της `F`, αλλιώς είναι διαιρέτης

Αλγεβρική (ή Ασθενής) Διαίρεση

Αλγόριθμος WEAK_DIV(F, P)

```
WEAK_DIV(F, P) {
   foreach(cube pi of P) {
       Vpi = NULL
       foreach (cube fj of F) {
          if (fj contains all literals of pi) {
               vij = fj with the literals of pi deleted
               Vpi = Vpi U vij;
          }
       }
   }
   Q = ∩i Vpi // Vpi intersection //
   R = F - PQ
   return (Q, R)
}
```

- Για κάθε κύβο SOP **του P**, p_i ελέγχουμε κάθε κύβο SOP **του F**, f_j
- Παράγουμε κύβο-παράγοντα (γινόμενο όρων) v_{ij}, αφαιρώντας τους όρους του p_i, προσθέτουμε στο V^{pi}
- Η τομή των V^{pi} (κοινό μέρος του κύβου-παράγοντα) είναι το πηλίκο Q

Αλγεβρική (ή Ασθενής) Διαίρεση – Παράδειγμα 1

- F = ad + abc + bcd, P = a + bc
- Ξεκινάμε με p_1 = a,
- f_1 = ad = p_1.d => v_{11} = d
- f_2 = abc = p_1.bc => v_{12} = bc
- f_3 = bcd, δεν έχει κοινούς όρους, συνεπώς:
- V^a = d + bc
- Κατόπιν, p_2 = bc
- f_1 = ad, δεν έχει κοινούς όρους,
- f_2 = abc = a.p_2 => v_{22} = a
- f_3 = bcd = p_2.d => v_{23} = d, συνεπώς:
- V^{bc} = a + d, και:
- **Q = ∩ {d + bc}, {a + d} = d, R = F – PQ = abc**
- Άρα: **F = (a + bc)d + abc**

Αλγεβρική (ή Ασθενής) Διαίρεση – Παραδείγματα 2, 3

- F = ac + ad + bc + bd + e, P = a + b
- Ξεκινάμε, όπως και στο προηγούμενο παράδειγμα, και καταλήγουμε ότι:
- V^a = c + d
- V^b = c + d, και:
- **Q = ∩ {c + d}, {c + d} = c + d, R = F – PQ = e**
- Άρα: **F = (a + b)(c + d) + e**

- F = ad + aef + ab + b'cd + b'cef, P = a + b'c
- Καταλήγουμε στά:
- V^a = d + ef + b
- V^b = d + ef, και:
- **Q = ∩ {d + ef + b}, {d + ef + b} = d + ef, R = F – PQ = ab**
- Άρα: **F = (a + b)(d + ef) + ab**

Αλγεβρική Διαίρεση

Αλγόριθμος ALGEBRAIC_DIVISION(A, B)

```
ALGEBRAIC_DIVISION(A, B) {
   for (i = 1 to n) { /* take one cube Ci^B of divisor at a time */
        D = {Cj^A such that Cj^A ≥ Ci^B};
        if (D == NULL) return (NULL, A);
        Di = D where variables in sup(Ci^B) are dropped;
        if (i == 1)
              Q = Di;       /* initialise quotient to Di */
        else
              Q = Q ∩ Di;  /* quotient is intersection of Dj, j = 1…i */
   }
   R = A – Q x B;
   return (Q, R);
}
```

- $A = \{C_j^A,\ j = 1, 2, \ldots, l\}$ – κύβοι ή μονώνυμα του διαιρετέου
- $B = \{C_i^B,\ i = 1, 2, \ldots, n\}$ – κύβοι ή μονώνυμα του διαιρέτη
- Ισοδύναμος αλγόριθμος με τον προηγούμενο – WEAK_DIV

Αλγεβρική Διαίρεση - Ιδιότητες

- **Θεώρημα:** Για δυο αλγεβρικές εκφράσεις f_i, f_j, η πράξη διαίρεσης $\mathbf{f_i\ /\ f_j}$ δεν παράγει πηλίκο (είναι κενή), αν ισχύει μία η περισσότερες από τις παρακάτω συνθήκες
  - f_j περιέχει μεταβλητή που δεν υπάρχει στην f_i,
  - f_j περιέχει κύβο με στήριξη που δεν εμπεριέχεται σε κανένα κύβο της f_i,
  - f_j περιλαμβάνει περισσότερους όρους από την f_i,
  - Ο αριθμός εμφανίσεων μιας μεταβλητής της f_j είναι μεγαλύτερος από τον αντίστοιχο στην f_i
- **Πόρισμα:** Βάση του Θεωρήματος, αν οι f_i, f_j είναι κόμβοι σε BN, τότε $\mathbf{f_i\ /\ f_j}$ = NULL, αν δεν υπάρχει μονοπάτι από τον κόμβο v_i της f_i στον v_j της f_j
  - Χρήσιμο για την πράξη SUBSTITUTE

Δυαδική (Boolean) Διαίρεση

- Για Δυαδική Διαίρεση: $F = PQ + R$ πρέπει να
  - υπολογίσουμε P, R (ελαχιστοποιημένες) έτσι ώστε $PQ + R$ να είναι κάλυψη της F
  - Κάλυψη της F ➔ $PQ + R \geq F_{ON}$
- **Θεώρημα**: Έστω $f1 = hx + e$ κάλυψη της F. Έστω συνάρτηση g, όπου $g \neq x \leq F_{DC}$, ή αναλυτικά : $gx' + x'g \leq F_{DC}$. Τότε η $f2 = hg + e$ είναι επίσης κάλυψη της F.
- **Απόδειξη**: βάση του ότι διαφορές των $f1$, $f2$ ($f1 \neq f2$) εμπεριέχονται στο DC της F

Δυαδική (Boolean) Διαίρεση – Βασικός Αλγόριθμος

- Μπορούμε να πραγματοποιήσουμε Δυαδική Διαίρεση `F = PQ + R`, ως εξής:
  - Έστω νέα μεταβλητή `x` στην στήριξη της `F` – όπου θέλουμε `xQ + R` κάλυψη της `F`
  - Σχηματίζουμε το DC set $x \neq g$, δηλ. `x'g + xg'`
  - **Προεκτείνουμε τα ON, DC σύμφωνα με το νέο DC:**
  - Θέτουμε $F_{DCnew} = F_{DC} \cup \{x'g + xg'\}$
  - Θέτουμε $F_{ONnew} = F_{ON} \cap F_{DC}'$
  - Θέτουμε $F_{OFFnew} = F_{ON}' \cap F_{DC}'$
  - Ελαχιστοποιούμε την `F` με τα νέα σύνολα ON, DC
    - ***απαλείφουμε το x' και μεταφέρουμε σημεία του DC στο ON*** χρησιμοποιώντας το ότι `x = g` στα σχετικά σημεία του B
  - Επιστρέφουμε (`F/x`, `R`), όπου `R` οι όροι της `F` που **δεν εμπεριέχουν** την μεταβλητή `x`

Δυαδική (Boolean) Διαίρεση – Βασικός Αλγόριθμος

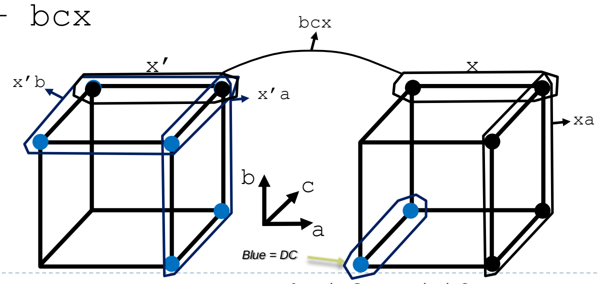
Αλγόριθμος BOOLEAN_DIVISION(f, g)

```
BOOLEAN_DIVISION(f, g) {
    DC = x'g + xg';                                // x (+) g //
    F_DCnew = F_DC U DC;                          // extend DC //
    F_ONnew = F_ON ∩ F_DC';                       // extend ON //
    F_ONSIMPLIFIED = full_simplify(F_ONnew, F_DCnew); // minimize //
    e = F_ONSIMPLIFIED - F_ONSIMPLIFIED/x;     // compute remainder //
    return (F_ONSIMPLIFIED/x, e);
}
```

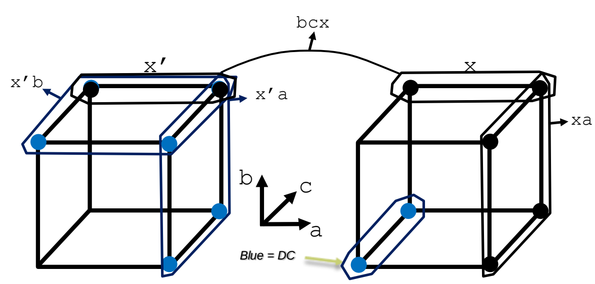
- Στόχος της απλοποίησης είναι να απαλείψει το `x'`
  - Όταν απλοποιούμε το ON set χρησιμοποιούμε το ότι `x` = `g`!!!
- Το βήμα `full_simplify` είναι ο ESPRESSO με DCs
  - **Ουσιαστικά EXPAND με την συνθήκη ότι x = g**
- Γενικά δεν παράγει πάντα καλούς παράγοντες
- Το υπόλοιπο `e` είναι το τμήμα της $F_{ONSIMPLIFIED}$, που δεν εμπεριέχει `x`

Δυαδική (Boolean) Διαίρεση – Παράδειγμα 1

- $F = a + bc, g = a + b$
  - Είναι προφανές ότι αλγεβρικά δεν διαιρείται η F με την g
- DC = $x'g + xg' = x'(a + b) + x(a + b)' = x'a + x'b + xa'b'$
- $F_{ONnew} = (a + bc)(x'a + x'b + xa'b')' =$
  $(a + bc)(xb + xa + x'a'b') =$
  $ax + bcx$



Δυαδική (Boolean) Διαίρεση – Παράδειγμα 1



- Άρα $F_{ONSIMPLIFIED} = ax + cx$
- $F = a + bc = x(a + c) = (a + b)(a + c)$

Δυαδική (Boolean) Διαίρεση – Παράδειγμα 2

- $F = ac + bc + ad, g = a + b$
- DC = $x'g + xg' = x'(a + b) + x(a + b)' =$
  $x'a + x'b + xa'b'$
- $F_{ONnew} = (ac + bc + ad)(x'a + x'b + xa'b')' =$
  $(ac + bc + ad)(xb + xa + x'a'b') =$
  $acx + bcx + adx$
- Προσθέτουμε στο ON set τα σημεία που ικανοποιούν την $x = a + b$, δηλ. $xa'b, xab'$
- Άρα $F_{ONSIMPLIFIED} = adx + cx$

Πράξη SUBSTITUTION – Διαίρεση Κόμβων

Αλγόριθμος SUBSTITUTE(Gn(V,E))

```
SUBSTITUTE(Gn(V, E)) {
   for (i = 1 to |V|) {
         for (j = 1 to |V|) {
              A = set of cubes of node fi;
              B = set of cubes of node fj;
              if (DIVEMPTY(A, B)== FALSE) {  /* check divide conditions */
                    (Q, R) = ALGEBRAIC_DIVISION(A, B);
                    if (Q ≠ NULL) {
                          fQ = sum of cubes of Q; /* create new BN nodes */
                          fR = sim of cubes of R;
                          if (substitution is favourable) /* condition */
                            fi = j . fQ + fR;  /* substitute */
                    }
              }
         }
   }
}
```

- Η συνθήκη έχει να κάνει με το κέρδος σε όρους ή το τελικό, αρχικό βάθος

Πυρήνες και Συν-πυρήνες

- Για τις πράξεις `DECOMPOSE`, `EXTRACT`, ο ***κατάλληλος διαιρέτης δεν είναι γνωστός*** ή υπαρκτός κόμβος στο δίκτυο
- Χρειαζόμαστε μέθοδο, αλγόριθμο για να υπολογίζουμε
  - (α) όλους ή καλούς διαιρέτες μιας έκφρασης/συνάρτησης
    - Είτε ένα κύβο (μονώνυμο), είτε άθροισμα κύβων (πολυώνυμο)
  - (β) όλους τους κοινούς διαιρέτες, ή τους κοινούς καλούς διαιρέτης ενός συνόλου έκφρασεων/συνάρτησεων
- Δημιουργούμε την έννοια του **Πυρήνα** μιας έκφρασης
- **Ορισμός:** μια έκφραση F είναι άνευ-κύβων (cube-free) όταν δεν υπάρχει κύβος που να την διαιρεί χωρίς υπόλοιπο (evenly), δηλ. δεν υπάρχει κύβος C, έτσι ώστε F = QC
- **Ορισμός:** Ονομάζουμε **πυρήνα** της F οποιονδήποτε άνευ-κύβων διαιρέτη της F: (F/c), όπου c κύβος, τον οποίο ονομάζουμε **συν-πυρήνα**

Παράδειγμα Πυρήνων - 1

- Έστω `f = abcd + abce + abde`
- **Μέθοδος 1η** – Επαναληπτική και Bottom-up
  - Τέμνοντας τους 3 κύβους της f, εξάγουμε συν-πυρήνες (Επ. 0)
  - {abcd, abce} → κοινό abc (συν-πυρήνας), πυρήνας: (d + e)
  - {abcd, abde} → κοινό abd (συν-πυρήνας), πυρήνας: (c + e)
  - {abce, abde} → κοινό abe (συν-πυρήνας), πυρήνας: (c + d)
  - Τέμνοντας τώρα επαναληπτικά τους κύβους, συν-πυρήνες τους Επιπέδου n, εξάγουμε τους συν-πυρήνες/πυρήνες του (n + 1)
  - {abc, abd} → κοινό ab (συν-πυρήνας), πυρήνας: (cd + ce + de)
  - … ο (cd + ce + de) είναι επιπέδου 1 και εμπεριέχει τους (c + d), (d + e)
  - Μπορεί να υλοποιηθεί με πίνακα με **$O(|F|^2\log|F|)$** πολυπλοκότητα για πυρήνες επιπέδου 0 (με *binary heap*)

Παράδειγμα Πυρήνων - 2

- Έστω f = abcd + abce + abde
- **Μέθοδος 2η** – Αναδρομική και Top-down
  - Για κάθε μεταβλητή στήριξης, x, της f, εντοπίζουμε τον μεγαλύτερο υποκύβο C ≥ x (εμπεριέχει x), που είναι κοινός σε πολλαπλούς (>2) κύβους της f
  - ο C είναι συν-πυρήνας, ενώ ο f/C πυρήνας
  - ο f/C μπορεί να εμπεριέχει μικρότερους συν-πυρήνες ➔ επαναλαμβάνουμε αναδρομικά
  - x = a, C = ab, f/C = cd + ce + de
  - x = b, C = ab, f/C = cd + ce + de
  - x = c, C = abc, f/C = d + e
  - x = d, C = abd, f/C = c + e
  - x = e, C = abe, f/C = c + d

Παράδειγμα Πυρήνων - 2

- Έστω f = adf + aef + bdf + bef + cdf + cef + bfg + h = (a + b + c)(d + e)f + bfg + h

Πυρήνας	Συν-Πυρήνας	Επίπεδο
d + e	af, cf	0
d + e + g	bf	0
a + b + c	df, ef	0
(a + b +c)(d + e) + bg	f	1
((a + b + c)(d + e) + bg)f + h	1	2

- Ο πυρήνας είναι επιπέδου 0, όταν η έκφραση του δεν εμπεριέχει άλλους πυρήνες, ενώ είναι επιπέδου n, όταν εμπεριέχει τουλάχιστον ένα πυρήνα επιπέδου (n – 1)

Αναδρομικός Υπολογισμός Πυρήνων - 1

Αλγόριθμος R_KERNELS(f)

```
R_KERNELS(f) {
        K = NULL;
        for each variable x in sup(f) {
          if (|CUBES(f, x)| >= 2) {
            /* skip cases where f/x yields one cube */
            C = largest cube containing x s.t. CUBES(f, C) = CUBES(f, x);
            K = K U R_KERNELS(f/C);
          }
        }
        K = K U f;
        return(K);
}
```

- `CUBES(f, x)` – επιστρέφει σύνολο κύβων της f που εμπεριέχουν όρο x
- f : Δυαδική Συνάρτηση, K: σύνολο Πυρήνων
- Ο παραπάνω αλγόριθμος υπολογίζει τον ίδιο πυρήνα πολλαπλές φορές
- Θα μπορούσαμε να εξετάζουμε τις μεταβλητές *μόνο 1 φορά κατά την αναδρομή!*

Αναδρομικός Υπολογισμός Πυρήνων - 1

- Παράδειγμα: f = ace + bce + de + g

- i = 3, c στην στήριξη των {ace, bce}, C = {c, e}
  - ce είναι συν-πυρήνας (co-kernel)
  - (a + b) είναι πυρήνας
- i = 5, e στην στήριξη των {ace, bce, de}, C = {c, e}
  - (ac + bc + d) είναι πυρήνας

- Άρα:
- K = {(a + b), (ac + bc + d)}

Αναδρομικός Υπολογισμός Πυρήνων - 2

Αλγόριθμος KERNELS(f, j)

```
KERNELS(f, j) {
        K = NULL;
        for i = j to n {
          if (|CUBES(f, li)| >= 2) {
            /* skip cases where f/li yields one cube */
            C = largest cube containing li s.t. CUBES(f, C) = CUBES(f, li);
            /* = C = largest cube dividing f/{li} evenly */
            if (lk not in C for all k < i)
               K = K U KERNELS(f/C, i + 1);
          }
        }
        K = K U f;
        return(K);
}
```

- f : Δυαδική Συνάρτηση, j: αριθμός όρου, K: σύνολο Πυρήνων
- Βελτιστοποίηση: Η κάθε μεταβλητή στήριξης της f εξετάζεται 1 φορά!!!
  - Το j μόνο αυξάνεται – όλοι οι όροι < j ΔΕΝ συμμετέχουν στην αναδρομική κλήση
  - Γραμμή → `if (lk not in C for all k < i) {}`
- ***ΠΡΟΣΟΧΗ: Ο `KERNELS` δεν υπολογίζει όλους τους πυρήνες κάθε επιπέδου***

Αναδρομικός Υπολογισμός Πυρήνων - 2

- Έστω `f` = `abcd + abce + abde`
- `KERNELS(f, 1)`
  - x = a, C = ab, f/C = cd + ce + de,
    `(lk not in C for all k < 1) =` **`TRUE`** (για `k < 1`) →
    `KERNELS(cd + ce + de, 2);`
    - x = c, C = c, f/C = d + e,
      `(lk not in C for all k < 3) =` **`TRUE`** →
      `KERNELS(d + e, 4);` → *προσθέτει το* (`d + e`)
    - x = d, C = d, f/C = c + e,
      `(lk not in C for all k < 4) =` **`TRUE`** →
      `KERNELS(c + e, 5);` → *προσθέτει το* (`c + e`)
    - x = e, C = e, f/C = c + d,
      `(lk not in C for all k < 5) =` **`TRUE`** →
      `KERNELS(c + d, 5);` → *προσθέτει το* (`c + d`)

    → *προσθέτει το* (`cd + ce + de`)
  - … (συνεχίζει)

Αναδρομικός Υπολογισμός Πυρήνων - 2

- … (συνέχεια)
- x = b, C = ab, f/C = cd + ce + de
  `(lk not in C for all k < 2) = ` **`FALSE`**
- x = c, C = abc, f/C = d + e,
  `(lk not in C for all k < 3) = ` **`FALSE`**
  Ο συν-πυρήνας `abc` κόβεται από τον αλγόριθμο βάση της συνθήκης – όμως ο `ab` έχει ληφθεί υπόψη, καθώς και ο συν-παράγοντας `c` στον πυρήνα του `ab` – βάση αυτής της παρατήρησης αγνοείται
- x = d, C = abd, f/C = c + e,
  `(lk not in C for all k < 4) = ` **`TRUE`** ➔
  `KERNELS(c + e, 4);` ➔ *προσθέτει το* (`c + e`)
- x = e, C = abe, f/C = c + d,
  `(lk not in C for all k < 5) = ` **`TRUE`** ➔
  `KERNELS(c + d, 6);` ➔ *προσθέτει το* (`c + d`)

- Άρα K = {`(cd + ce + de), (d + e), (c + e),(c + d)`}
- ***Παρά το ευριστικό, κάποιοι πυρήνες υπολογίζονται πολλαπλές φορές, π.χ. (c+e)!***

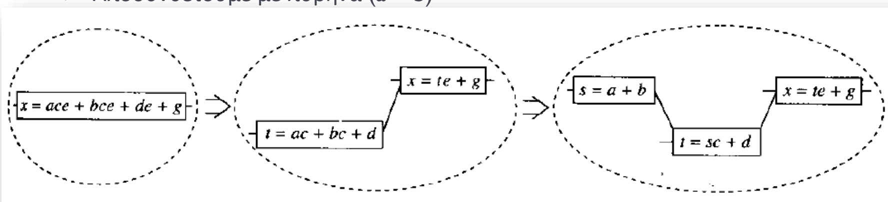
Πράξη DECOMPOSE και Πυρήνες

- Ο στόχος της ΑΠΟΣΥΝΘΕΣΗΣ είναι διττός:
  - (1) Μειώνει
    - Αριθμό εμφανίσεων
    - Πολυπλοκότητα κόμβου

    } διευκολύνει την Τεχνολογική Απεικόνιση
  - (2) Δημιουργεί μικρούς κόμβους
    - Ιδανικότερους, και με μεγαλύτερη πιθανότητα για ΑΝΤΙΚΑΤΑΣΤΑΣΗ από την SUBSTITUTE
- Στρατηγικές Επιλογή Διαιρέτη
  - (1) Ένα LEVEL-0 Πυρήνα (γρήγορο, Low-Effort)
  - (2) Τον Πυρήνα που παράγει τα καλύτερα αποτελέσματα (High-Effort)
- Εναλλακτική Μέθοδος
  - Διαιρούμε με Πυρήνα, κατόπιν βρίσκουμε απλό διαιρέτη (αντί πυρήνα) που παράγει άνευ-κύβων έκφραση

Πράξη DECOMPOSE και Πυρήνες

- Παράδειγμα:
  - Αποσυνθέτουμε με πυρήνα (ac + bc + d)
  - Νέος κόμβος t
  - Αποσυνθέτουμε με πυρήνα (a + b)



- Εναλλακτικά:
  - Αποσυνθέτουμε με πυρήνα (a + b), μένει t = (a + b), f = tce + de + g
  - Βλέπουμε ότι στην αρχική διαίρεση, το πηλίκο είναι ce, άρα μπορούμε να μεταφέρουμε το e στον κόμβο t (η f τώρα είναι άνευ κύβων):
    t = e(a + b), f = tc + de + g

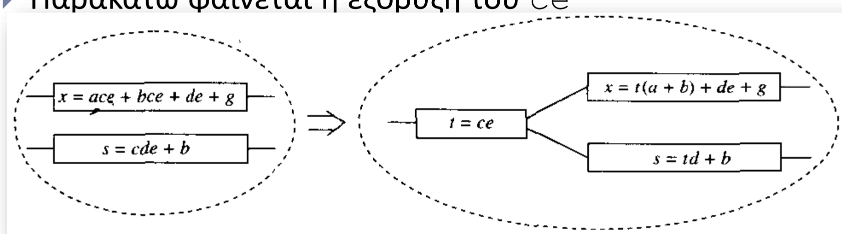
Παραγοντοποίηση από κοινού – Κοινοί κύβοι και Πυρήνες – Πράξη EXTRACTION

- Πράξη EXTRACTION ➔ εξόρυξη είτε κοινών κύβων (συν-πυρήνων) είτε κοινών πυρήνων από πολλαπλές συναρτήσεις (κόμβους του BN)
- **<u>Μέθοδος 1η</u>** – Κοινοί Κύβοι/Συν-πυρήνες
  - Για 2 συναρτήσεις σε SOP μορφή, οι κοινοί Συν-πυρήνες τους αντιστοιχούν σε τομή 2 ή περισσότερων από τους κύβους τους
  - Για 2 η περισσότερες συναρτήσεις, f_i, μπορούμε να σχηματίσουμε μια βοηθητική συνάρτηση f_{aux}:
  - Όπου οι κοινοί συν-πυρήνες των συναρτήσεων θα είναι οι συν-πυρήνες της f_{aux}, που ΔΕΝ έχουν στην στήριξη τους τις νέες μεταβλητές α_i

$$f_{aux} = \sum_i a_i . f_i$$

Παραγοντοποίηση από κοινού – Κοινοί κύβοι και Πυρήνες – Πράξη EXTRACTION

- Έστω f_x = ace + bce + de + g, f_s = cde + b
- Σχηματίζουμε faux = a_x.ace + a_x.bce + a_x.de + a_x.g + a_s.cde + a_s.b (νέες μεταβλητές a_x, a_s)
- Συνολικά, οι συν-πυρήνες της faux είναι: {ce, de, b, a, a_x.ce, a_x.e, a_x, a_s, 1}
- ***Άρα, οι κοινοί συν-πυρήνες είναι οι {ce, de}***
- Παρακάτω φαίνεται η εξόρυξη του ce



Παραγοντοποίηση από κοινού – Κοινοί κύβοι και Πυρήνες – Πράξη EXTRACTION

- Αντικατάσταση κόμβου βάρους (LC) **l**, που εμφανίζεται **n** φορές στο BN, οδηγεί σε κέρδος (LC):
  - **Κέρδος = n . l - n – l**
- Η εξόρυξη γίνεται δεκτή μόνο αν βελτιώνει το κόστος σε Εμβαδό ή Καθυστέρηση
- … Ομοίως για την DECOMPOSE

Παραγοντοποίηση από κοινού – Κοινοί κύβοι και Πυρήνες – Πράξη EXTRACTION

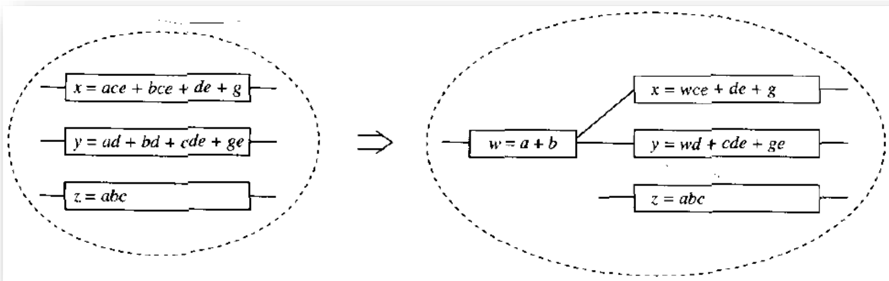
- **Μέθοδος 2η** – Κοινοί Πυρήνες
  - Για 2 συναρτήσεις σε SOP μορφή, οι κοινοί Πυρήνες αντιστοιχούν σε κοινούς κύβους των πυρήνων ➔ ο κάθε κύβος ενός πυρήνα αντιστοιχεί σε μια έκφραση, όρο
  - Για 2 η περισσότερες συναρτήσεις, f_i, μπορούμε να σχηματίσουμε μια βοηθητική συνάρτηση f_{aux}, ***που αντιστοιχεί στους πυρήνες των f_i***, όπου x_i είναι η συνολική στήριξη(sup) των κύβων των πυρήνων $$f_{aux} = \sum_i a_i . k_{xi} . \prod_j^{|k_{xi}|} x_j, \ j \in \sup(k_{xi})$$
  - Οι κοινοί πυρήνες των συναρτήσεων θα είναι οι συν-πυρήνες της f_{aux}, που πάλι ΔΕΝ έχουν στην στήριξη τους τις νέες μεταβλητές $α_i$

Παραγοντοποίηση από κοινού – Κοινοί κύβοι και Πυρήνες – Πράξη EXTRACTION

- Έστω f_x = ace + bce + de + g, K(fx) = {(a + b), (ac + bc + d), (ace + bce + de + g)}, f_y = ad + bd + cde + ge, K(f_y) = {(a + b + ce), (cd + g), (ad + bd + cde + ge)}
- $f_{aux} = a_x.k_{x1}.xa.xb + a_x.k_{x2}.xac.xbc.xd + a_x.k_{x3}.xace.xbce.xde.xg + a_y.k_{y1}.xa.xb.xce + a_y.k_{y2}.xcd.xg + a_y.k_{y3}.xad.xbd.xcde.xge$
- Συνολικά, οι συν-πυρήνες της faux είναι: {xaxb, xg, a_x, a_y, 1}
- ***Άρα, ο κοινός πυρήνας είναι ο {(a + b)}***

Παραγοντοποίηση από κοινού – Κοινοί κύβοι και Πυρήνες – Πράξη EXTRACTION

- Έστω f_x = ace + bce + de + g, f_y = ad + bd + cde + ge
- ***Ο κοινός πυρήνας είναι ο {(a + b)}***
- Παρακάτω φαίνεται η εξόρυξη του (a + b)



Κάλυψη Ορθογωνίων Πίνακα – Εναλλακτική Μέθοδος Εξόρυξης

- <u>**Ορισμός:**</u> Ένα ορθογώνιο (R, C) ενός Πίνακα B, όπου Bij = {0, 1, *} είναι ένα υποσύνολο σειρών, R, και στηλών, C, έτσι ώστε Bij ανήκει στο {1, *} για i στο R, j στο C.
  - * είναι σημείο DC
  - Ένα ορθογώνιο (R1, C1) συμπεριλαμβάνει αυστηρά το ορθογώνιο (R2, C2) αν R2 ≤ R1, C2 ≤ C1
  - Ένα ορθογώνιο ονομάζεται απλοϊκό (trivial) όταν |R| = 1, ή |C| = 1
  - Ένα ορθογώνιο του Πίνακα B, ονομάζεται πρώτο, όταν δεν συμπεριλαμβάνεται αυστηρά σε οποιοδήποτε άλλο ορθογώνιο του B

	1	2	3	4	5
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
4	1	0	1	1	1

Κάλυψη Ορθογωνίων Πίνακα – Εναλλακτική Μέθοδος Εξόρυξης

- **Ορισμός:** Ένα ορθογώνιο (R, C) ενός Πίνακα B, όπου Bij = {0, 1, *} είναι ένα υποσύνολο σειρών, R, και στηλών, C, έτσι ώστε Bij ανήκει στο {1, *} για i στο R, j στο C.
  - * είναι σημείο DC
  - Ένα ορθογώνιο (R1, C1) συμπεριλαμβάνει αυστηρά το ορθογώνιο (R2, C2) αν R2 ≤ R1, C2 ≤ C1
  - Ένα ορθογώνιο ονομάζεται απλοϊκό (trivial) όταν |R| = 1, ή |C| = 1
  - Ένα ορθογώνιο του Πίνακα B, ονομάζεται πρώτο, όταν δεν συμπεριλαμβάνεται αυστηρά σε οποιοδήποτε άλλο ορθογώνιο του B

	1	**2**	**3**	**4**	**5**
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
4	1	0	1	1	1

Κάλυψη Ορθογωνίων Πίνακα – Εναλλακτική Μέθοδος Εξόρυξης

- **Κάλυψη Ορθογωνίων:** Ένα σύνολο ορθογωνίων $\{(R_k, C_k)\}$, σχηματίζουν ***κάλυψη ορθογωνίων*** ενός πίνακα B, αν και μόνο αν,
  $(B_{ij} = 1)$ => i ανήκει στο R_k, j ανήκει στο C_k, για κάποιο k

- Το κάθε ορθογώνιο μπορεί να ενέχει ένα βάρος (ή κόστος), το οποίο να ορίζεται από μια συνάρτηση $w(R_k, C_k)$
- Έτσι, το βάρος μιας κάλυψης ορθογωνίων $\{(R_k, C_k)\}$, ορίζεται ως το άθροισμα: $\sum_k w(R_k, C_k)$

Εξαγωγή Πυρήνων μέσω Κάλυψης Ορθογωνίων

- Έστω x = abcdg + abcdh + abce + abcf + abi
- Σχηματίζω Πίνακα **Κύβων-Όρων** (Σειρές, Στήλες)

		a	b	c	d	e	f	g	h	i
		1	2	3	4	5	6	7	8	9
abcdg	1	1	1	1	1			1		
abcdh	2	1	1	1	1				1	
abce	3	1	1	1		1				
abcf	4	1	1	1			1			
abi	5	1	1							1

- Η κάλυψη των ***Πρώτων Ορθογωνίων*** του πίνακα αντιστοιχεί σε κοινούς όρους των κύβων ➔ συν-πυρήνες (αλλιώς δεν είναι άνευ-κύβων το αποτέλεσμα):
  - {{1,2},{1,2,3,4}} => συν-πυρήνας abcd, πυρήνας (g + h)
  - {{1,2,3,4},{1,2,3}} => συν-πυρήνας abc, πυρήνας (dg + dh + e + f)
  - {{1,2,3,4,5},{1,2}} => συν-πυρήνας ab, πυρήνας (cdg + cdh + ce + cf + i)

Εξαγωγή Πυρήνων μέσω Κάλυψης Ορθογωνίων

- **<u>Θεώρημα:</u>** C είναι συν-πυρήνας της f, αν και μόνο αν είναι κύβος που αντιστοιχεί σε πρώτο ορθογώνιο του Πίνακα Κύβων-Όρων, και συμπεριλαμβάνει τουλάχιστον 2 σειρές
  - Πυρήνες Επιπέδου 0 αντιστοιχούν σε πρώτα ορθογώνια ***με μέγιστο πλάτος στηλών***

		a	b	c	d	e	f	g	h	i
		1	2	3	4	5	6	7	8	9
abcdg	1	1	1	1	1			1		
abcdh	2	1	1	1	1				1	
abce	3	1	1	1		1				
abcf	4	1	1	1			1			
abi	5	1	1							1

L0

Πρόβλημα Κάλυψης Ορθογωνίων

- **<u>Κόστος/Βάρος:</u>** Αν για την συνάρτηση βάρους, WEIGHT, ισχύει η ιδιότητα: RECT(R1) ≥ RECT(R2) ⇔ WEIGHT(R1) > WEIGHT(R2), τότε μια βέλτιστη κάλυψη θα περιλαμβάνει σύνολο από Πρώτα Ορθογώνια
- Ανάλογο του Θεωρήματος Quine στο UCP!!!
- Τα Π.Ο. όμως δεν διασφαλίζουν Ελάχιστο Κόστος (λ.χ. επικαλύπτονται) ➔ μεγαλύτερο βάρος
- **<u>Ευριστικά</u>**:
  1. Βρίσκουμε Ελάχιστη Κάλυψη Βάση Π.Ο.
  2. Ελαχιστοποιούμε το Μέγεθος τους για Απομείωση Κόστους
- **<u>Αριθμός Π.Ο.</u>** Για (n x n) Πίνακα ➔ 2^n – 2 Π.Ο.

Αριθμός Πρώτων Ορθογωνίων - 1

- Για 4x4 πίνακα – Μέγιστος Αριθμός Πρώτων Ορθογώνιων
  - Αριθμός Πρώτων Ορθογώνιων για (n x n) Πίνακα = **2^n - 2**

4

1	0	1	1
1	1	0	1
1	1	1	0
0	1	1	1

1	0	1	1
1	1	0	1
1	1	1	0
0	1	1	1

4

4

1	0	1	1
1	1	0	1
1	1	1	0
0	1	1	1

1	0	1	1
1	1	0	1
1	1	1	0
0	1	1	1

2

Αριθμός Πρώτων Ορθογωνίων – 2

- Για κάθε προσαύξηση σειράς και στήλης, από n σε (n+1), έχουμε:
  - Αριθμός Ορθογωνίων $(n+1)x(n+1) = 2 \times \boldsymbol{K} + 2$
  - Όπου ***K***, ο αριθμός Ορθογωνίων *(nxn)*
- Δηλαδή, έχουμε:
  - Βασική Περίπτωση: ***f(2) = 2, f(3) = 6***
  - Αναδρομική Σχέση: ***f(n+1) = 2xf(n) + 2***
- Έτσι, προκύπτει
  - $f(n) = 2^n - 2$
- Άρα: $\boxed{O(nxn) = 2^n - 2}$

0	1	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Πρόβλημα Κάλυψης Ορθογωνίων

- <u>Εφαρμογές Κάλυψης Ορθογωνίων</u>
  - **Εξαγωγή Συν-πυρήνων** – Πίνακας Κύβων-Όρων
  - **Εξαγωγή Κοινών Κύβων** – Πίνακας Κύβων-Όρων
  - **Εξαγωγή Κοινών Πυρήνων** – Πίνακας Συν-Πυρήνων-Κύβων
  - Η Κάλυψη Ορθογωνίων επιτρέπει την ***μη-αλγεβρική επεξεργασία/παραγοντοποίηση*** μέσω επικαλυπτόμενων Π.Ο.
- Εξαγωγή Κοινών Κύβων
  - Κοινός κύβος σε 2 ή περισσότερες εκφράσεις/κόμβους
  - Για να επιτευχθεί είτε ο ελάχιστος αριθμός όρων, είτε συμψηφισμός όρων/βάθους
  - ΣΕΙΡΑ = Κύβοι, ΣΤΗΛΗ = Όροι

Κάλυψη Ορθογωνίων Πίνακα – Εξαγωγή Κοινών Κύβων

- Έστω F = abc + abd + eg, G = abfg, H = bd + ef
- Το κάθε Π.Ο., με αριθμό σειρών > 2, αντιστοιχεί σε κοινό κύβο
  - π.χ. {{1, 2, 4}, {1, 2}} αντιστοιχεί στον ab
  - Έστω ότι πραγματοποιούμε εξόρυξη του ab:
- F = **X**c + **X**d + eg, G = **X**fg, H = bd + ef, **X = ab**

		a	b	c	d	e	f	g
		1	2	3	4	5	6	7
F1 (abc)	1	1	1	1				
F2 (abd)	2	1	1		1			
F3 (eg)	3					1		1
G1 (abfg)	4	1	1				1	1
H1 (bd)	5		1		1			
H2 (ef)	6					1	1	

Κάλυψη Ορθογωνίων Πίνακα – Εξαγωγή Κοινών Κύβων

- Για επαναληπτική εκτέλεση, σημαδεύουμε τα καλυμμένα Π.Ο., και προσθέτουμε τον παράγοντα σε στήλες και σειρές
- Αν τώρα επιλέξουμε το Π.Ο. {{2,5},{2,4}}
  - Αντιστοιχεί στον συν-παράγοντα bd
  - Επειδή έχει επικάλυψη με το ab, έχει και τομή!
  - Δυαδική παραγοντοποίηση
- F = **X**c + **XY** + eg, G = **X**fg, H = **Y** + ef, **X = ab, Y = bd**

		a	b	c	d	e	f	g	X
		1	2	3	4	5	6	7	8
F1 (abc)	1	*	*	1					1
F2 (abd)	2	*	*		1				1
F3 (eg)	3					1		1	
G1 (abfg)	4	*	*				1	1	1
H1 (bd)	5		1		1				
H2 (ef)	6					1	1		
X1 (ab)	7	1	1						

Κάλυψη Ορθογωνίων Πίνακα – Εξαγωγή Κοινών Πυρήνων

- Εξαγωγή Κοινών Πυρήνων
  - Κοινός πυρήνας σε 2 ή περισσότερες εκφράσεις/κόμβους
  - ΣΕΙΡΑ = Συν-Πυρήνας, ΣΤΗΛΗ = Κύβοι Εκφράσεων και Πυρήνων
- Έστω `F = af + bf + ag + cg + ade + bde + cde`, `G = af + bf + ace + bce`, `H = ade + cde`
- Υπολογίζουμε τους Πυρήνες των `F`, `G`, `H`:

Πυρήνες/Συν-πυρήνες F	Πυρήνες/Συν-πυρήνες G	Πυρήνες/Συν-πυρήνες H
de + f + g (a)	ce + f (a, b)	a + c (de)
de + f (b)	a + b (f, ce)	
a + b + c (de)		
a + b (f)		
de + g (c)		
a + c (g)		

Κάλυψη Ορθογωνίων Πίνακα – Εξαγωγή Κοινών Πυρήνων

- Απαριθμούμε τους κύβους των F, G, H από 1 έως 13 μια και ο κύβος που παράγεται δεν είναι μοναδικός
- Το Π.Ο. {{3,4,9,10},{1,2}} αντιστοιχεί στον πυρήνα `(a + b)`, ο οποίος διαιρεί τις `F`, `G`

		a	b	c	ce	de	f	g
		1	2	3	4	5	6	7
F (a)	1					5	1	3
F (b)	2					6	2	
F (de)	3	5	6	7				
F (f)	4	1	2					
F (c)	5					7		4
F (g)	6	3		4				
G (a)	7				10		8	
G (b)	8				11		9	
G (ce)	9	10	11					
G (f)	10	8	9					
H (de)	11	12		13				

Κάλυψη Ορθογωνίων Πίνακα – Εξαγωγή Κοινών Πυρήνων

- F = deX + fX + ag + cg, G = ceX + fX,
  H = ade + cde,
  X = a + b

- Για επαναληπτική εκτέλεση, σημαδεύουμε τα καλυμμένα Π.Ο. (μοναδικούς κύβους), και προσθέτουμε τον παράγοντα σε στήλες και σειρές

		a	b	c	ce	de	f	g
		1	2	3	4	5	6	7
F (a)	1					*	*	3
F (b)	2					*	*	
F (de)	3	*	*	7				
F (f)	4	*	*					
F (c)	5					7		4
F (g)	6	3		4				
G (a)	7				*		*	
G (b)	8				*		*	
G (ce)	9	*	*					
G (f)	10	*	*					
H (de)	11	12		13				
X	14	14	15					