Αιτιώδεις Σχέσεις και Χρονισμός

Η Σχέση Happens-Before (Συμβαίνει-πριν)

◊ Οι εκτελέσεις, ως ακολουθίες γεγονότων, καθορίζουν μια καθολική διάταξη σε αυτά.

◊ Ωστόσο είναι δυνατό δύο υπολογιστικά γεγονότα από διαφορετικούς επεξεργαστές να μην επηρεάζουν το ένα το άλλο, παρότι η εκτέλεση καθορίζει μια (αυθαίρετη) διάταξη αυτών.

◊ Έτσι, η δομή αιτιότητας μεταξύ των γεγονότων χάνεται.

Έστω a μια αυθαίρετη εκτέλεση.

- Έστω ότι φ_1 και φ_2 είναι 2 γεγονότα της a που εκτελούνται από τον ίδιο επεξεργαστή. Το φ_1 *επηρεάζει αιτιωδώς* (casually influences) το φ_2, αν το φ_1 προηγείται του φ_2 στην a.
- Έστω ότι φ_1 και φ_2 είναι 2 γεγονότα της a που εκτελούνται από διαφορετικές διεργασίες p_i και p_j, αντίστοιχα. Το φ_1 *επηρεάζει αιτιωδώς* το φ_2 αν το φ_1 είναι γεγονός στο οποίο αποστέλλεται μήνυμα m από την p_i στην p_j ενώ το φ_2 είναι το γεγονός παραλαβής του m από την p_j.
- Επίσης, ισχύει η μεταβατική ιδιότητα (αν το φ_1 επηρεάζει αιτιωδώς το φ_2 και το φ_2 επηρεάζει αιτιωδώς το φ_3, τότε το φ_1 επηρεάζει αιτιωδώς το φ_3).

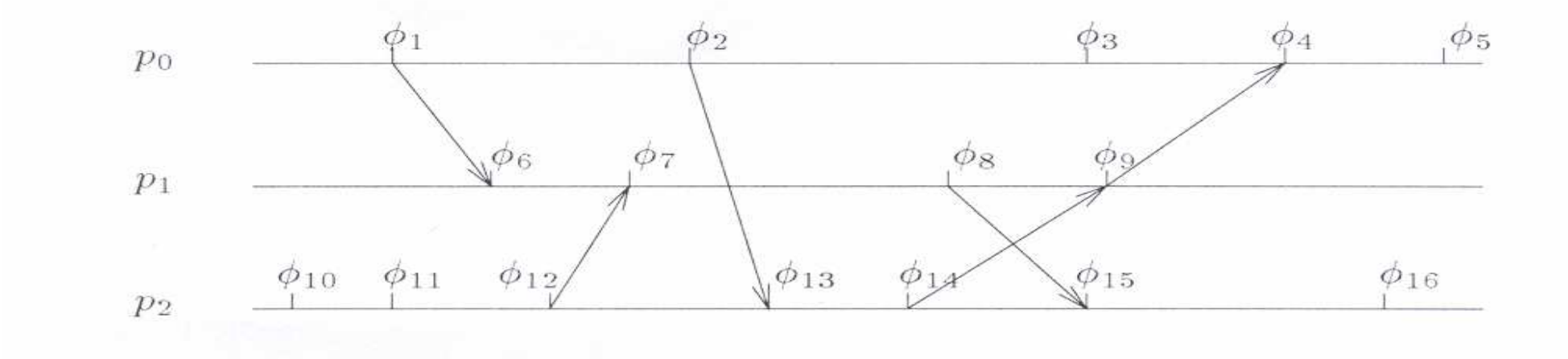
Η σχέση Happens-Before (Προηγείται ή Συμβαίνει-πριν)

Πιο φορμαλιστικά:

Δεδομένων δύο γεγονότων φ_1 και φ_2 στην a, το φ_1 *Συμβαίνει-πριν* (happens before) από το φ_2, το οποίο συμβολίζεται $\varphi_1 \rightarrow \varphi_2$, αν ισχύει μια από τις ακόλουθες συνθήκες:

- τα φ_1, φ_2 είναι γεγονότα που εκτελούνται από την ίδια διεργασία p_i και το φ_1 προηγείται του φ_2 στην ακολουθία a,
- το φ_1 είναι γεγονός στο οποίο αποστέλλεται ένα μήνυμα m από μια διεργασία p_i σε μια άλλη p_j, και το φ_2 είναι το γεγονός παραλαβής του m από την p_j,
- υπάρχει γεγονός φ τέτοιο ώστε $\varphi_1 \rightarrow \varphi$ και $\varphi \rightarrow \varphi_2$.

Παρατηρούμε ότι η → καθορίζει μια μερική διάταξη.



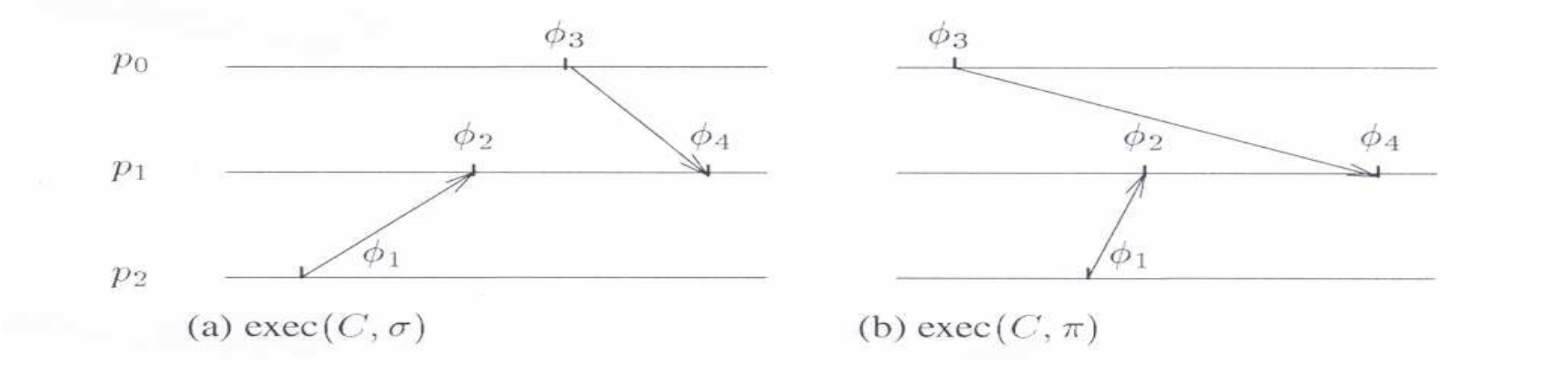
Αιτιώδεις Αναδιατάξεις (Casual Shuffles)

Η σχέση *Συμβαίνει-πριν* χαρακτηρίζει τις αιτιώδεις σχέσεις μιας εκτέλεσης.

Αν τα γεγονότα μιας εκτέλεσης αναδιαταχθούν χωρίς ωστόσο να καταστρατηγούν τη σχέση *Συμβαίνει-πριν*, το αποτέλεσμα εξακολουθεί να είναι έγκυρη εκτέλεση και οι δύο αυτές εκτελέσεις είναι παρόμοιες σε όλες τις διεργασίες.

Ορισμός: Έστω ότι a = exec(C,σ) είναι ένα τμήμα εκτέλεσης. Μια μετάθεση (permutation) π του χρονοδιαγράμματος σ είναι μια *αιτιώδης αναδιάταξη* (casual shuffle) του σ αν:

- για κάθε j, $0 \leq j \leq n-1$, $\sigma|j = \pi|j$, και
- αν ένα μήνυμα m αποστέλλεται από μια διεργασία p_j κατά την εκτέλεση του γεγονότος φ της a, τότε στην π το φ προηγείται του γεγονότος παραλαβής του m.



Αιτιώδεις Αναδιατάξεις (Casual Shuffles)

Λήμμα 1

Έστω ότι a = exec(C,σ) είναι ένα τμήμα εκτέλεσης. Οποιαδήποτε καθολική διάταξη των γεγονότων του σ είναι συνεπής με τη σχέση *Συμβαίνει-πριν* της a, είναι μια αιτιώδης αναδιάταξη (casual shuffle) της σ.

Λήμμα 2

Έστω ότι a = exec(C,σ) είναι ένα τμήμα εκτέλεσης. Έστω ότι π είναι μια αιτιώδης αναδιάταξη (casual shuffle) του σ. Τότε, a' = exec(C,π) είναι ένα έγκυρο τμήμα εκτέλεσης παρόμοιο (indistinguishable) με το a σε όλες τις διεργασίες.

Λογικά Ρολόγια (Logical Clocks)

Πως μπορούν οι διεργασίες να παρατηρήσουν την σχέση Συμβαίνει-πριν σε μια εκτέλεση a;

Με κάθε γεγονός φ, συσχετίζουμε μια χρονοσφραγίδα LT(φ) (Logical Time of φ).

Χρειαζόμαστε μια μερική διάταξη < στις χρονοσφραγίδες, έτσι ώστε για κάθε ζεύγος γεγονότων φ_1 και φ_2:

$$\text{αν } \varphi_1 \rightarrow \varphi_2 \text{ τότε } LT(\varphi_1) < LT(\varphi_2)$$

Απλός Αλγόριθμος για ανάθεση και διατήρηση χρονοσφραγίδων

- Κάθε διεργασία p_j διατηρεί μια τοπική μεταβλητή LT_j, που ονομάζεται λογικό ρολόι (logical clock). Το LT_j αποθηκεύει μη-αρνητικούς ακεραίους και έχει αρχική τιμή 0.
- Ως μέρος κάθε υπολογιστικού γεγονότος φ, η p_j αυξάνει το LT_j ώστε να είναι μεγαλύτερο κατά μία μονάδα από το μέγιστο της τρέχουσας τιμής του LT_j και της μεγαλύτερης χρονοσφραγίδας κάθε μηνύματος που παραλαμβάνεται στο φ.
- Σε κάθε μήνυμα που αποστέλλεται στο φ ανατίθεται ως χρονοσφραγίδα το LT_j.
- Η χρονοσφραγίδα, LT(φ), που συσχετίζεται με το γεγονός φ της διεργασίας p_j, είναι η νέα τιμή LT_j που υπολογίστηκε κατά τη διάρκεια εκτέλεσης του γεγονότος.

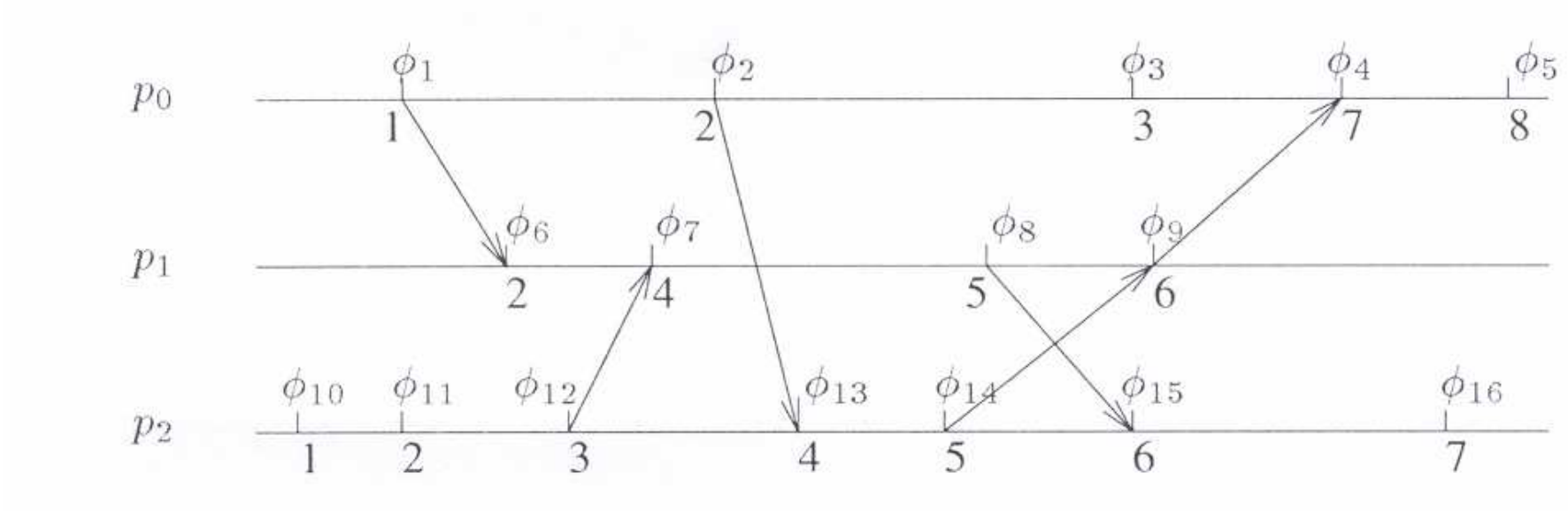
Λογικά Ρολόγια (Logical Clocks)

Η μερική διάταξη των χρονοσφραγίδων καθορίζεται από την γνωστή σχέση < των ακεραίων.

Για κάθε διεργασία p_j, το LT_j είναι γνήσια αύξων.

Τότε, και οι τρεις συνθήκες της σχέσης *Συμβαίνει-πριν* ικανοποιούνται.

Θεώρημα: Έστω ότι a είναι μια αυθαίρετη εκτέλεση και ας υποθέσουμε ότι φ_1 και φ_2 είναι δύο οποιαδήποτε γεγονότα της a. Αν $\varphi_1 \rightarrow \varphi_2$, τότε $LT(\varphi_1) < LT(\varphi_2)$.



Διανυσματικά Ρολόγια (Vector Clocks)

Αρνητικά Λογικών Ρολογιών

Αν $LT(\varphi_1) \geq LT(\varphi_2)$, τότε γνωρίζουμε ότι το φ_1 δεν συμβαίνει πριν το φ_2.

Είναι το αντίστροφο αληθές;

ΟΧΙ! Είναι δυνατό $LT(\varphi_1) < LT(\varphi_2)$ αλλά να μην ισχύει $\varphi_1 \rightarrow \varphi_2$.

Το πρόβλημα είναι ότι η σχέση *Συμβαίνει-πριν* ορίζει μερική διάταξη, ενώ οι λογικές χρονοσφραγίδες είναι ακέραιοι και η < ορίζει καθολική διάταξη.

Διανυσματικά Ρολόγια (Vector Clocks)

Ορισμός: Τα γεγονότα φ_1 και φ_2 *συμβαίνουν ταυτόχρονα* σε μια εκτέλεση a, το οποίο συμβολίζεται $\varphi_1 \parallel_a \varphi_2$, αν δεν ισχύει ούτε ότι $\varphi_1 \rightarrow \varphi_2$, ούτε ότι $\varphi_2 \rightarrow \varphi_1$.

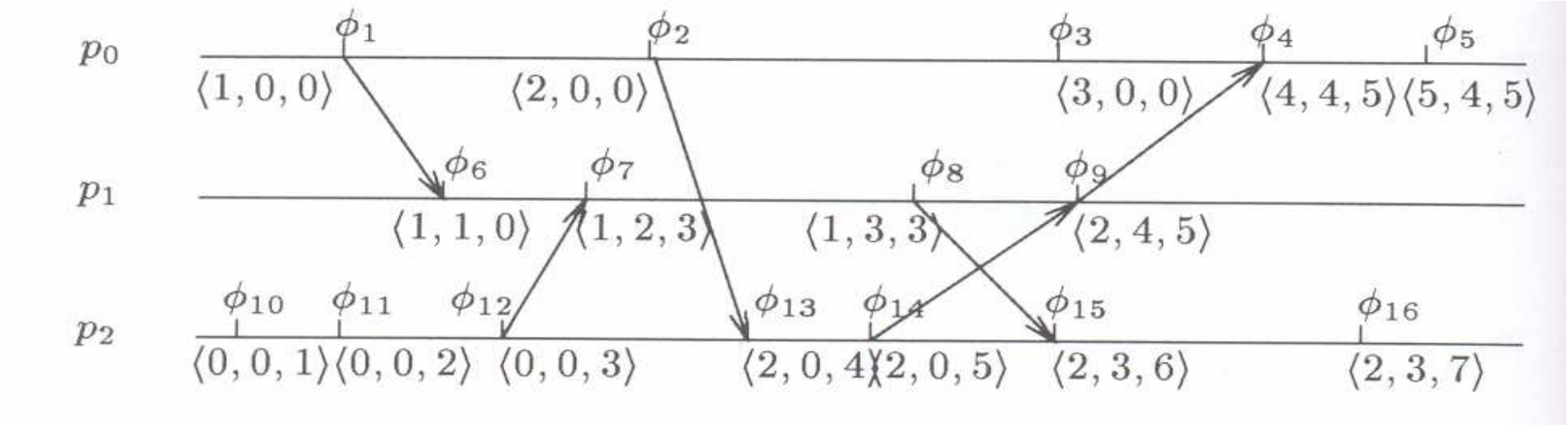
Αν $\varphi_1 \parallel_a \varphi_2$, τότε υπάρχουν 2 εκτελέσεις a_1 και a_2, και οι δύο παρόμοιες (indistinguishable) με την a, τέτοιες ώστε το φ_1 συμβαίνει πριν από το φ_2 στην a_1, και το φ_1 συμβαίνει μετά το φ_2 στην a_2 ή το αντίστροφο.

$\Rightarrow$ οι διεργασίες δεν γνωρίζουν αν το φ_1 συμβαίνει πριν από το φ_2 ή το αντίστροφο στην a, ενώ ότι από τα δύο και αν συμβαίνει οι διεργασίες δεν παρατηρούν κάποια διαφορά.

Διανυσματικά Ρολόγια (Vector Clocks)

- Κάθε διεργασία p_j διατηρεί μια τοπική μεταβλητή VC_j (Vector Clock) που ονομάζεται *διανυσματικό ρολόι* της p_j. Το VC_j είναι ένας πίνακας n στοιχείων, κάθε στοιχείο του οποίου είναι ένας μη-αρνητικός ακέραιος με αρχική τιμή 0.
- Ως μέρος κάθε υπολογιστικού γεγονότος φ μιας διεργασίας p_j, η p_j ενημερώνει το VC_j ως εξής:
  - το $VC_j[j]$ αυξάνεται κατά 1
  - Για κάθε $i \neq j$, το $VC_j[i]$ παίρνει τιμή ίση με το μέγιστο της τρέχουσας τιμής του και της μεγαλύτερης τιμής της εγγραφής i όλων των διανυσματικών χρονοσφραγίδων των μηνυμάτων που ελήφθησαν κατά τη διάρκεια εκτέλεσης του φ.
  - Σε κάθε μήνυμα που αποστέλλεται κατά τη διάρκεια εκτέλεσης του φ ανατίθεται ως χρονοσφραγίδα η νέα τιμή του VC_j.
- Η διανυσματική χρονοσφραγίδα του φ, VC(φ), είναι η τιμή του VC_j στο τέλος της εκτέλεσης του φ.

Διανυσματικά Ρολόγια (Vector Clocks)



Πρόταση: Για κάθε διεργασία p_j, σε κάθε προσβάσιμη καθολική κατάσταση, ισχύει ότι $VC_j[j] \geq VC_i[j]$, για κάθε i, $0 \leq i \leq n-1$.

Μερική Διάταξη Διανυσματικών Ρολογιών

Έστω ότι v_1 και v_2 είναι δύο διανύσματα των n ακεραίων το καθένα. Τότε, $v_1 \leq v_2$ αν και μόνο αν για κάθε j, $0 \leq j \leq n-1$, $v_1[j] \leq v_2[j]$; $v_1 < v_2$ αν και μόνο αν $v_1 \leq v_2$ και $v_1 \neq v_2$.

Τα διανύσματα v_1 και v_2 είναι μη συγκρίσιμα αν δεν ισχύει ούτε ότι $v_1 \leq v_2$, ούτε ότι $v_2 \leq v_1$.

Λέμε ότι οι διανυσματικές χρονοσφραγίδες εκφράζουν την εκτέλεση ταυτόχρονων γεγονότων (δηλαδή την παραλληλία), αν για κάθε ζεύγος γεγονότων φ_1 και φ_2, $\varphi_1 \parallel \varphi_2$ αν και μόνο αν τα $VC(\varphi_1)$ και $VC(\varphi_2)$ είναι μη-συγκρίσιμα.

Διανυσματικά Ρολόγια (Vector Clocks)

Θεώρημα

Έστω ότι a είναι μια αυθαίρετη εκτέλεση και έστω ότι φ_1 και φ_2 είναι δύο οποιαδήποτε γεγονότα της a. Αν $\varphi_1 \rightarrow \varphi_2$ τότε $VC(\varphi_1) < VC(\varphi_2)$.

Απόδειξη

Έστω ότι τα φ_1, φ_2 είναι γεγονότα του ίδιου επεξεργαστή και ας υποθέσουμε ότι το φ_1 προηγείται του φ_2. *Γιατί ισχύει ο ισχυρισμός για τα φ_1 και φ_2 σε αυτή την περίπτωση;*

Έστω ότι το φ_1 είναι γεγονός στ οποίο αποστέλλεται κάποιο μήνυμα m και φ_2 είναι το γεγονός παραλαβής του μηνύματος. *Γιατί ισχύει ο ισχυρισμός για τα φ_1 και φ_2 σε αυτή την περίπτωση;*

Ισχύει η μεταβατική ιδιότητα για τη σχέση ≤ σε διανύσματα.

Διανυσματικά Ρολόγια (Vector Clocks)

Θεώρημα

Έστω ότι a είναι μια αυθαίρετη εκτέλεση και έστω ότι φ_1 και φ_2 είναι δύο οποιαδήποτε γεγονότα της a. Αν $VC(\varphi_1) < VC(\varphi_2)$ τότε $\varphi 1 \rightarrow \varphi_2$.

Απόδειξη

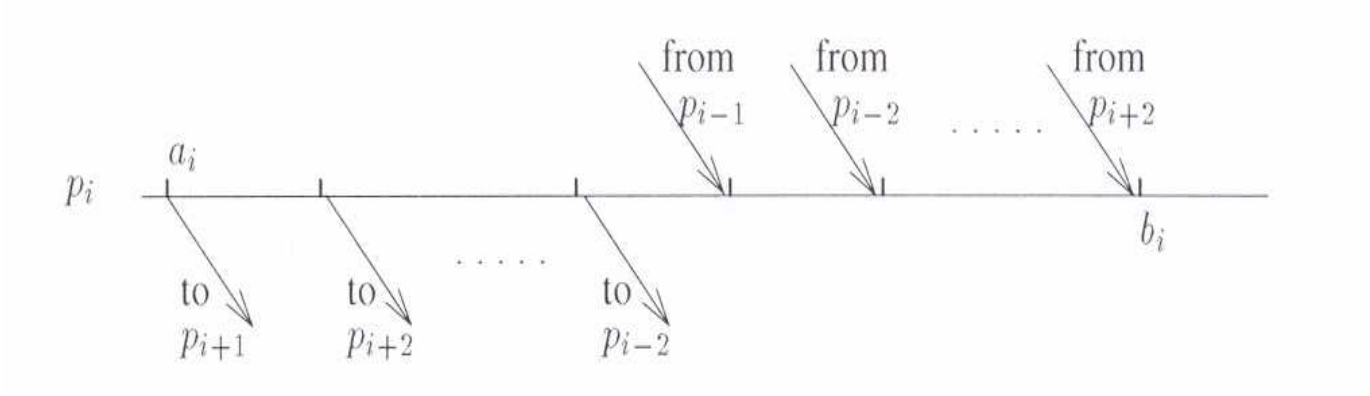
- ◊ Έστω ότι φ_1 και φ_2 είναι δύο γεγονότα που συμβαίνουν ταυτόχρονα, το φ_1 από τη διεργασία p_i και το φ_2 από τη p_j, $p_i \neq p_j$.
- ◊ Ας υποθέσουμε ότι $VC_i[i](\varphi_1) = m$. Ο μόνος τρόπος το i-οστό στοιχείο του διανυσματικού ρολογιού μιας διεργασίας p_j να αποκτήσει τιμή τουλάχιστον m είναι μέσω μιας αλυσίδας μηνυμάτων που ξεκινούν από την p_i (είτε στο γεγονός φ_1 ή σε επόμενο γεγονός).
- ◊ Μια τέτοια αλυσίδα θα συνεπαγόταν ότι τα φ_1, φ_2 δεν συμβαίνουν ταυτόχρονα. Άτοπο!

Άρα, $VC_j[i](\varphi_2) < m \Rightarrow VC_i(\varphi_1)$ δεν μπορεί να είναι μικρότερο από $VC_j(\varphi_2)$.

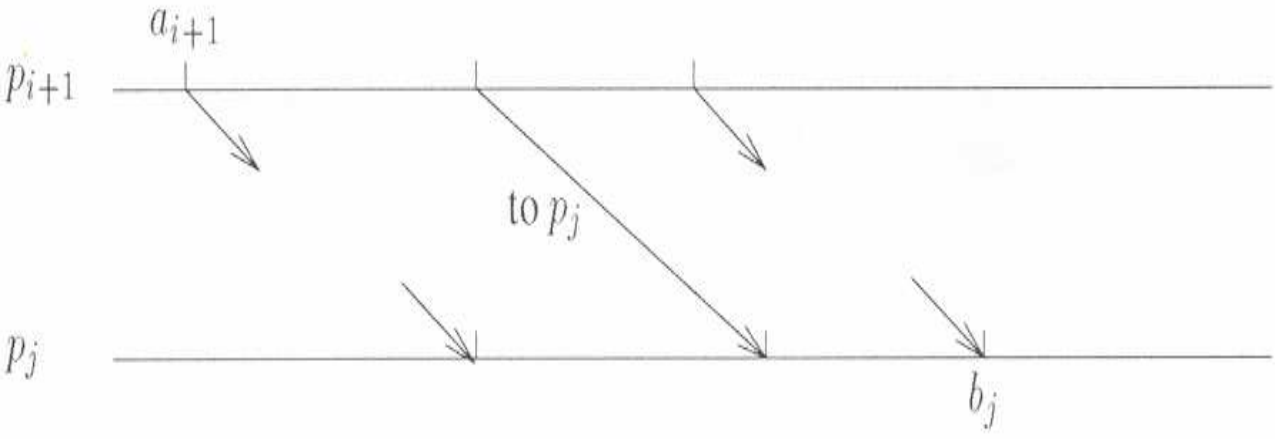
Ένα κάτω όριο στο μέγεθος των διανυσματικών ρολογιών

Ας θεωρήσουμε μια κλίκα και μια εκτέλεση α στην οποία:

- Κάθε διεργασία j στέλνει σειριακά ένα μήνυμα στις διεργασίες j+1, j+2, ..., n-1, 0, 1, ..., j-2.
- Αφού σταλούν όλα αυτά τα μηνύματα, η j λαμβάνει τα μηνύματα που της έχουν αποσταλεί με την εξής σειρά: j-1, j-2, ..., 0, n-1, n-2, ..., j+2.



Τα γεγονότα της διεργασίας p_i στην α



p_{i+1} και p_j στην α

Για κάθε j, έστω ότι a_j είναι το γεγονός αποστολής του $1^{ου}$ μηνύματος της j και b_j το γεγονός παραλαβής του τελευταίου.

Ένα κάτω όριο στο μέγεθος των διανυσματικών ρολογιών

Λήμμα 1

Για κάθε i, $a_{i+1} \parallel b_i$.

Απόδειξη

Οι αιτιώδεις σχέσεις είναι απλές και δεν υπάρχουν σχέσεις μεταξύ των γεγονότων αποστολής που να προκύπτουν λόγω μεταβατικής ιδιότητας.

Η p_{j+1} δεν στέλνει μήνυμα στην p_j.

Λήμμα 2

Για κάθε i και j, $0 \leq i \neq j \leq n-1$, $a_{i+1} \rightarrow b_j$.

Απόδειξη

Γιατί αυτό είναι αληθές;

Ένα κάτω όριο στο μέγεθος των διανυσματικών ρολογιών

Θεώρημα: Αν VC είναι μια συνάρτηση που απεικονίζει κάθε γεγονός της α σε ένα διάνυσμα στο χώρο R^k με τρόπο που να εκφράζει τη Σχέση Συμβαίνει πριν (και την ταυτόχρονη εκτέλεση γεγονότων), τότε $k \geq n$.

Απόδειξη: Έστω i οποιοσδήποτε ακέραιος, $0 \leq i \leq n-1$. Από το Λήμμα 1 $\Rightarrow$ $a_{i+1} \parallel b_i$.

Αφού το VC εκφράζει παραλληλία $\Rightarrow$ τα $VC(a_{i+1})$ και $VC(b_j)$ είναι μη-συγκρίσιμα

$\Rightarrow$ υπάρχει r τ.ω. $VC[r](b_i) < VC[r](a_{i+1})$. Έστω f(i) ένας τέτοιος ακέραιος.

Με τον τρόπο αυτό ορίζεται συνάρτηση f: {0, ..., n-1} -> {0, ..., k-1}.

Αποδεικνύουμε ότι $k \geq n$, αποδεικνύοντας ότι η f είναι 1-1.

Ας υποθέσουμε (δια της μεθόδου της εις άτοπο απαγωγής) το αντίθετο. Τότε, ∃ δείκτες i και j, $i \neq j$, τ.ω. $f(i) = f(j)$. Έστω $r = f(i) = f(j)$.

Εξ ορισμού της f, $VC[f(i)](b_i) < VC[f(i)](a_{i+1})$ και $VC[f(j)](b_j) < VC[f(j)](a_{j+1}) \Rightarrow$

$VC[r](b_i) < VC[r](a_{i+1})$ και $VC[r](b_j) < VC[r](a_{j+1})$

Από Λήμμα 2, $a_{i+1} \rightarrow b_j \Rightarrow VC(a_{i+1}) < VC(b_j)$.

Από όλα τα παραπάνω έχω: $VC[r](b_i) < VC[r](a_{i+1}) \leq VC[r](b_j) < VC[r](\alpha_{j+1})$.

Άτοπο (αντιτίθεται στο Λήμμα 2)!

Συστήματα Διαμοιραζόμενης Μνήμης

Δεδομένων δύο γεγονότων e_1 και e_2 μιας εκτέλεσης α, το e_1 Συμβαίνει-πριν το e_2, το οποίο συμβολίζεται $e_1 \rightarrow e_2$, αν ισχύει μια από τις ακόλουθες συνθήκες:

1. τα e_1 και e_2 είναι γεγονότα του ίδιου επεξεργαστή και το e_1 εκτελείται πριν από το e_2 στην α,
2. τα e_1 και e_2 είναι αλληλεπιδρώτα γεγονότα (δηλαδή και τα δύο προσβαίνουν την ίδια μεταβλητή και το ένα εκ των δύο επιτελεί write σε αυτή) και το e_1 προηγείται του e_2 στην α,
3. υπάρχει ένα γεγονός e τ.ω. $e_1 \rightarrow e$ και $e \rightarrow e_2$.

Παραδείγματα Χρησιμότητας της Αιτιότητας

Συνεπείς Τομές (consistent cuts)

Σε ένα κατανεμημένο σύστημα δεν υπάρχει «παρατηρητής» που να γνωρίζει την κατάσταση όλων των διεργασιών (omniscient observer) έτσι ώστε να μπορεί να καταγράφει συνεπή στιγμιότυπα του συστήματος.

Αυτό θα ήταν ωστόσο επιθυμητό π.χ., για τους ακόλουθους λόγους:

- επαναφορά του συστήματος μετά από κατάρρευση
- ανίχνευση αδιεξόδων
- ανίχνευση τερματισμού, κλπ.

Υποθέτουμε ότι σε οποιοδήποτε γεγονός, κάθε διεργασία λαμβάνει το πολύ ένα μήνυμα.

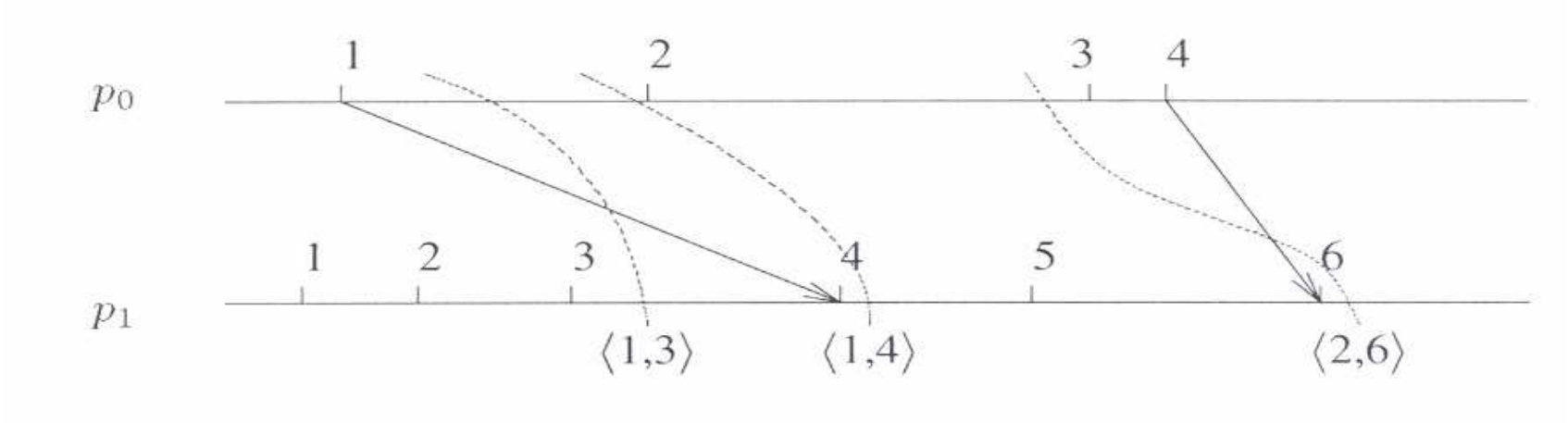
Δοθέντος μιας εκτέλεσης α, αριθμούμε τα βήματα κάθε διεργασίας με τους ακεραίους 1,2,3, ...

Η *τομή* μιας εκτέλεσης είναι ένα διάνυσμα $\mathbf{k} = <k_1, ..., k_{n-1}>$ n θετικών ακεραίων.

Δεδομένης μιας τομής **k** μιας εκτέλεσης, είναι δυνατό να κατασκευαστεί ένα σύνολο από καταστάσεις μια για κάθε διεργασία τ.ω. η κατάσταση της διεργασίας p_j είναι η κατάσταση της p_j αμέσως μετά την εκτέλεση του k_j-οστού βήματός της στην α.

Παραδείγματα Χρησιμότητας της Αιτιότητας

Συνεπείς Τομές



Εύρεση της μέγιστης συνεπούς τομής

Δοθέντος μιας τομής **k** της εκτέλεσης, ζητείται να βρεθεί η πιο πρόσφατη **συνεπής** τομή που προηγείται της **k** (ή τουλάχιστον δεν έπεται αυτής).

Εύρεση Κατανεμημένων Στιγμιοτύπων

Αντί να δίνεται μια τομή, οι διεργασίες λαμβάνουν κάποια ένδειξη έναρξης υπολογισμού μιας συνεπούς τομής.

Μοντελοποίηση Φυσικών Ρολογιών

Υποθέτουμε ότι υπάρχει ένας μηχανισμός, που ονομάζεται *ρολόι υλικού* (hardware clock), ο οποίος παρέχει κάποιου είδους χρονισμό στις διεργασίες.

Πιο φορμαλιστικά:

Σε μια χρονισμένη εκτέλεση, έχει συσχετισθεί μια κάθε διεργασία p_i, μια γνήσια αύξουσα συνάρτηση HC_i με πεδίο ορισμού και πεδίο τιμών το σύνολο των μη-αρνητικών πραγματικών αριθμών. Αν η p_i εκτελέσει ένα βήμα τη χρονική στιγμή t (πραγματικός χρόνος), η τιμή $HC_i(t)$ είναι διαθέσιμη ως είσοδος στη συνάρτηση μετάβασης της p_i. Ωστόσο, η συνάρτηση μετάβασης της p_i δεν μπορεί να αλλάξει την HC_i.

Υποθέτουμε ότι $HC_i(t) = t + c_i$, όπου c_i είναι μια σταθερά.

Μοντελοποίηση Φυσικών Ρολογιών

Μπορούμε να διασπάσουμε μια εκτέλεση σε n υποακολουθίες, όπου κάθε υποακολουθία αναπαραστά την «όψη εκτέλεσης» μιας διεργασίας (δηλαδή εκείνα τα γεγονότα της εκτελέσης που έχουν εκτελεστεί από τη διεργασία).

Ορισμός 1

Μια *όψη εκτέλεσης με τιμές ρολογιού* μιας διεργασίας p_i αποτελείται από μια αρχική κατάσταση για την p_i, μια ακολουθία από γεγονότα που εκτελούνται από την p_i, και μια τιμή του ρολογιού υλικού για κάθε τέτοιο γεγονός. Οι τιμές του ρολογιού υλικού πρέπει να είναι γνήσια αύξουσες, και αν μια ακολουθία γεγονότων είναι μη-πεπερασμένη, οι τιμές αυτές θα πρέπει να αυξάνονται χωρίς πάνω όριο.

Ορισμός 2

Μια *χρονισμένη όψη εκτέλεσης με τιμές ρολογιού* μιας διεργασίας p_i είναι μια όψη εκτέλεσης με τιμές ρολογιού της p_i με κάθε γεγονός της οποίας έχει συσχετισθεί και ο πραγματικός χρόνος στον οποίο συνέβει το γεγονός. Η συσχέτιση αυτή πρέπει να είναι συνεπής με τις τιμές του ρολογιού υλικού έχοντας τη μορφή $HC_i(t) = t + c_i$.

Χρονισμένες Εκτελέσεις

Δοθέντος μιας χρονισμένης εκτέλεσης με τιμές ρολογιού α, είναι δυνατό να κατασκευαστεί για κάθε διεργασία p_i, μια χρονισμένη όψη με τιμές ρολογιού, η οποία θα συμβολίζεται α | i.

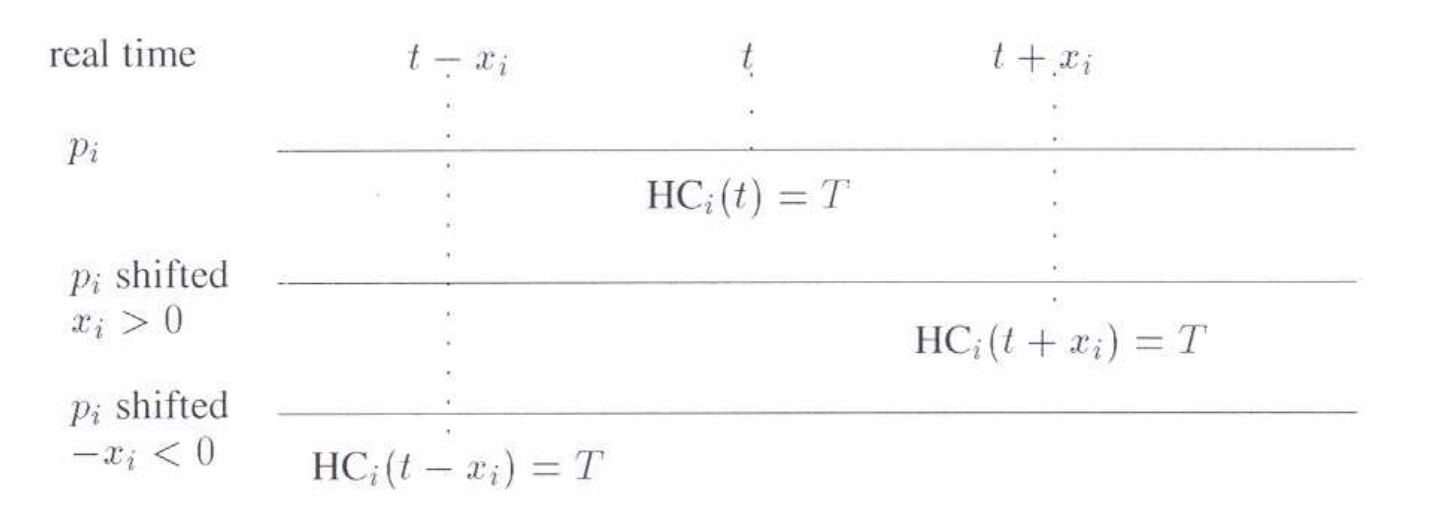
Ένα σύνολο από n χρονισμένες όψεις με τιμές ρολογιού $\{\eta_0, ..., \eta_{n-1}\}$, μια για κάθε διεργασία p_i, μπορούν να *συγχωνευτούν* ως ακολούθως:

- Η αρχική κατάσταση προκύπτει με συνδυασμό των αρχικών καταστάσεων όλων των χρονισμένων όψεων. Η ακολουθία γεγονότων προκύπτει με πολύπλεξη (interleaving) των γεγονότων των χρονισμένων όψεων με τέτοιο τρόπο ώστε αυτή να σέβεται τον πραγματικό χρόνο εκτέλεσης του κάθε γεγονότος.
- Οι συγκρούσεις επιλύονται με διάταξη των γεγονότων deliver κάθε χρονικής στιγμής t πριν από τα υπολογιστικά γεγονότα της t. Άλλες επιπρόσθετες συγκρούσεις επιλύονται βάσει των δεικτών των διεργασιών.

Το αποτέλεσμα συμβολίζεται με $\text{merge}(\eta_0, ..., \eta_{n-1})$.

Η $\text{merge}(\eta_0, ..., \eta_{n-1})$ είναι έγκυρη εκτέλεση αν οι χρονισμένες όψεις είναι συνεπείς: αν ένα μήνυμα της p_j παραλαμβάνεται από την p_i τη χρονική στιγμή t στην η_i, αλλά η p_j δεν το έχει στείλει μέχρι την t στην η_j, η $\text{merge}(\eta_0, ..., \eta_{n-1})$ δεν είναι έγκυρη εκτέλεση.

Μετακινώντας στο χρόνο την εκτέλεση μιας διεργασίας



Ορισμός: Έστω ότι α είναι μια χρονισμένη εκτέλεση με τιμές ρολογιών υλικού και έστω ότι x είναι ένα διάνυσμα n πραγματικών αριθμών. Ορίζουμε τη shift(α, x) να είναι η εκτέλεση merge(η_0, ..., η_{n-1}), όπου ηi είναι η χρονισμένη όψη προκύπτει αν προσθέσουμε την τιμή xi στον πραγματικό χρόνο που αντιστοιχεί σε κάθε γεγονός της α | i.

Το αποτέλεσμα της μετακίνησης μιας εκτέλεσης στο χρόνο δεν είναι απαραίτητα έγκυρη εκτέλεση (αφού στην εκτέλεση που προκύπτει από τη μετακίνηση θα μπορούσε κάποιο μήνυμα να παραλαμβάνεται πριν αποσταλεί).

Λήμμα: Έστω ότι α είναι μια χρονισμένη εκτέλεση με τιμές ρολογιών υλικού HC_i, και έστω ότι x είναι ένα διάνυσμα n πραγματικών αριθμών. Στη shift(α,x):

1. το ρολόι υλικού της p_i είναι $\mathrm{HC}_i' = \mathrm{HC}_i - x_i$,
2. κάθε μήνυμα από την p_i στην p_j έχει καθυστέρηση $\delta - x_i + x_j$, όπου δ είναι η καθυστέρηση του μηνύματος στην α.

Το Πρόβλημα Συγχρονισμού Ρολογιών

Οι διεργασίες θα πρέπει να επικοινωνήσουν προκειμένου να συγχρονίσουν τα ρολόγια τους.

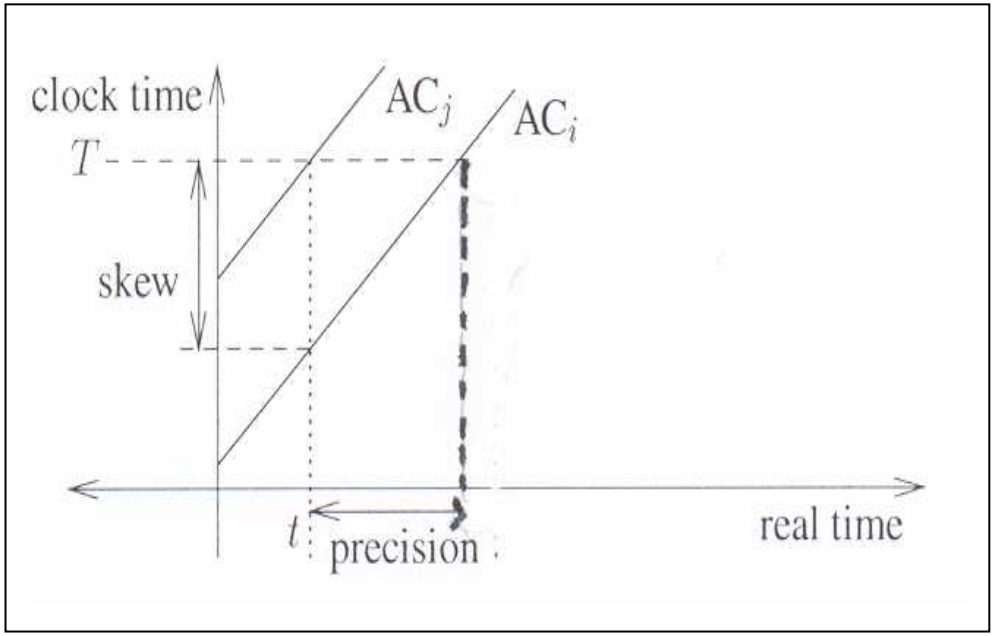
Κάθε διεργασία p_i χειρίζεται μια μεταβλητή adj_i. Το προσαρμοσμένο ρολόι της διεργασίας p_i ορίζεται να είναι $AC_i(t) = HC_i(t) + adj_i(t)$.

Ορισμός (επίτευξη ε-συγχρονισμού ρολογιών): Σε μια επιτρεπτή χρονισμένη εκτέλεση, θα πρέπει να υπάρχει ένας πραγματικός αριθμός t_f τ.ω. ο αλγόριθμος έχει τερματίσει μέχρι την χρονική στιγμή t_f και για όλες τις διεργασίες p_i και p_j και όλες τις χρονικές στιγμές $t \geq t_f$, $|AC_i(t) - AC_j(t)| \leq \varepsilon$. Το ε ονομάζεται *απόκλιση (skew)*.

Υποθέσεις

Υπάρχουν μη αρνητικοί ακέραιοι (σταθερές) d και u, $d \geq u$, τ.ω. σε κάθε επιτρεπτή χρονισμένη εκτέλεση, η καθυστέρηση κάθε μηνύματος βρίσκεται στο διάστημα [d-u, d].

Το u εκφράζει την *αβεβαιότητα* στην καθυστέρηση του μηνύματος.



Το Πρόβλημα Συγχρονισμού Ρολογιών – Η απλή περίπτωση 2 διεργασιών

Απλός Αλγόριθμος

Η διεργασία p_0 θέτει την adj_0 στην τιμή 0 και στέλνει την τρέχουσα τιμή του ρολογιού υλικού της στη διεργασία p_1.

Όταν η p_1 λάβει Τ από την p_0, θέτει την $adj_1 = T + (d-u) – HC_1$. (έτσι $AC_1=T+(d-u)$)

Στην καλύτερη περίπτωση: skew = 0.

Στη χειρότερη περίπτωση: skew = u.

Τι θα συμβεί αν η adj_1 πάρει τιμή $T + d – HC_1$;

Το καλύτερο είναι η καθυστέρηση να εκτιμηθεί σε $d – u/2$.

Ποια είναι η απόδοση του αλγορίθμου στην καλύτερη και στη χειρότερη περίπτωση τότε;

Το Πρόβλημα Συγχρονισμού Ρολογιών – Η απλή περίπτωση 2 διεργασιών

Λήμμα: Η καλύτερη απόκλιση που μπορεί να επιτευχθεί στη χειρότερη περίπτωση από έναν αλγόριθμο συγχρονισμού ρολογιών Α για 2 διεργασίες p_0 και p_1 είναι u/2.

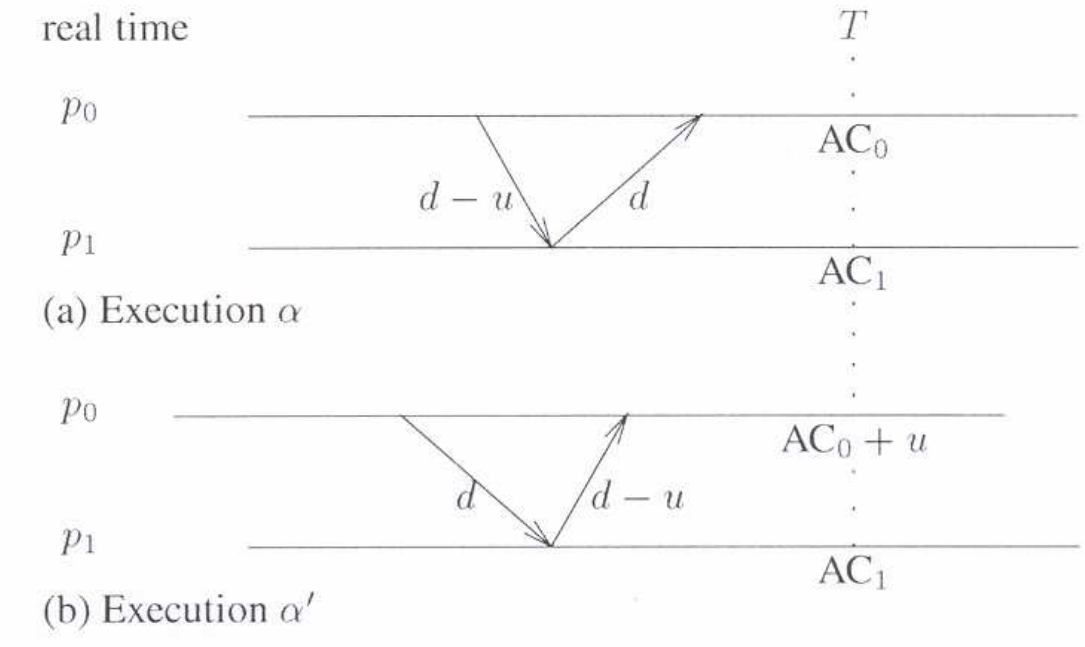
Απόδειξη: Έστω ότι α είναι μια εκτέλεση του α στην οποία η καθυστέρηση των μηνυμάτων από την p_0 στην p_1 είναι d-u ενώ από την p_1 στην p_0 είναι d.

Έστω ότι AC_0 και AC_1 είναι οι τιμές των προσαρμοσμένων ρολογιών των διεργασιών κάποια χρονική στιγμή Τ μετά τον τερματισμό. Εφόσον ο Α έχει απόκλιση ε, $AC_0 \geq AC_1 - \varepsilon$.

Έστω ότι α' = shift(α, <-u, 0>). Η α' είναι έγκυρη εκτέλεση αφού όλες οι καθυστερήσεις μηνυμάτων είναι μεταξύ d-u και d.

Τη χρονική στιγμή Τ, στην α', το προσαρμοσμένο ρολόι της p0 έχει τιμή AC_0 + u, ενώ το προσαρμοσμένο ρολόι της p1 είναι AC_1. Αφού η απόκλιση του αλγορίθμου είναι ε, $AC_1 \geq (AC_0 + u) - \varepsilon$.

Άρα, $AC_0 \geq AC_0 + u - 2\varepsilon \Rightarrow \varepsilon \geq u/2$.



Ένα Άνω Φράγμα

Ένας Απλός Αλγόριθμος για n Διεργασίες

- Επιλογή μιας διεργασίας ως κεντρικής.
- Εφαρμογή του αλγορίθμου για 2 διεργασίες μεταξύ οποιασδήποτε διεργασίας και της κεντρικής.

Ποια είναι η απόκλιση αυτού του αλγορίθμου;

Υπάρχει ένας ελαφρώς καλύτερος αλγόριθμος!

Algorithm 20 A clock synchronization algorithm for n processors:
code for processor p_i, $0 \leq i \leq n-1$.

initially $diff[i] = 0$

1: at first computation step:
2: send HC (current hardware clock value) to all other processors

3: upon receiving message T from some p_j:
4: $diff[j] := T + d - u/2 - HC$
5: if a message has been received from every other processor then
6: $adj := \frac{1}{n} \sum_{k=0}^{n-1} diff[k]$

Ένα Άνω Φράγμα

Θεώρημα: Ο παραπάνω αλγόριθμος συγχρονισμού ρολογιών επιτυγχάνει απόκλιση $u(1 - 1/n)$ για n διεργασίες.

Απόδειξη: Έστω α μια οποιαδήποτε εκτέλεση του αλγορίθμου.

Αφότου η p_i λάβει τιμή Τ από την p_j, η $diff_i[j]$ αποθηκεύει την εκτίμηση της p_i για τη διαφορά των HC_j και HC_i. Εξαιτίας του τρόπου υπολογισμού του $diff_i[j]$, το σφάλμα στην εκτίμηση είναι το πολύ $+u/2$ ή $-u/2$. Άρα ισχύει ότι:

Λήμμα: Για κάθε χρονική στιγμή t που έπεται της χρονικής στιγμής στην οποία η pi θέτει την $diff_i[j]$, $j \neq i$, ισχύει ότι $diff_i[j] = HC_j(t) - HC_i(t) + err_{ji}$, όπου err_{ji} είναι μια σταθερά με $-u/2 \leq err_{ji} \leq u/2$.

Από τον ορισμό των προσαρμοσμένων ρολογιών, για κάθε χρονική στιγμή t μετά τον τερματισμό του αλγορίθμου:

$$|ACi(t) - ACj(t)| = | HC_i(t) + 1/n \sum_{k=0}^{n-1} diff_i[k] - HC_j(t) - 1/n \sum_{k=0}^{n-1} diff_j[k] | \quad (1)$$

Μετά από αλγεβρικούς υπολογισμούς, έχουμε:

$$1/n \mid HC_i(t) - HC_j(t) + diff_i[i] - diff_j[i] + HC_i(t) - HC_j(t) + diff_i[j] - diff_j[j] + \sum_{k=0,k\neq i,j}^{n-1} (HC_i(t) - HC_j(t) + diff_i[k] - diff_j[k]) |$$

Ένα Άνω Φράγμα

Απόδειξη (συνέχεια)

Από ιδιότητες της απόλυτης τιμής και επειδή diffi[i] = diffj[j] = 0:

$(1) \leq 1/n(|HC_j(t) - HC_i(t) + diff_j[i]|^{(*)} + |HC_i(t) - HC_j(t) + diff_i[j]|^{(\sim)} + \sum_{k=0,k\neq i,j}^{n-1} |HC_i(t) - HC_j(t) + diff_i[k] - diff_j[k]|^{(\#)})$

$^{(*)}$ -> διαφορά μεταξύ της γνώσης της p_i για το δικό της ρολόι και της εκτίμησης της p_j για το ρολόι της $p_i = |err_{ij}| \leq u/2$

$^{(\sim)}$ -> διαφορά μεταξύ της γνώσης της pj για το δικό της ρολόι και της εκτίμησης της p_i για το ρολόι της $p_j = |err_{ji}| \leq u/2$

$^{(\#)}$ -> διαφορά μεταξύ της εκτίμησης της p_i για το ρολόι της p_k και της εκτίμησης της p_j για το ρολόι της p_k:

$$|HC_i(t) - HC_j(t) + HCk(t) - HC_i(t) + err_{ki} - HCk(t) + HC_j(t) - err_{kj}| \leq u.$$

$\Rightarrow$ η συνολική έκφραση είναι $\leq 1/n(u/2 + u/2 + (n-2)\ u) = 1/n(n-1)u = (1 - 1/n)u$, όπως απαιτείται!

Ένα Κάτω Φράγμα

Θεώρημα: Για κάθε αλγόριθμο συγχρονισμού ρολογιών για n διεργασίες που επιτυγχάνει απόκλιση ε, το ε είναι τουλάχιστον $u(1 - 1/n)$.

Απόδειξη: Έστω A ένας οποιοσδήποτε αλγόριθμος ε-συγχρονισμού ρολογιών.

Έστω α μια χρονισμένη εκτέλεση του A με τιμές ρολογιού HC_i τ.ω. για όλες τις διεργασίες p_i και p_j, $i < j$:

- η καθυστέρηση οποιουδήποτε μηνύματος από την p_i στην p_j είναι ακριβώς d-u, ενώ
- η καθυστέρηση οποιουδήποτε μηνύματος από την p_j στην p_i είναι ακριβώς d.

Έστω AC_i η τιμή του προσαρμοσμένου ρολογιού της p_i μετά τον τερματισμό.

Λήμμα: Για κάθε k, $1 \leq k \leq n-1$, $AC_{k-1}(t) \leq AC_k(t) - u + \varepsilon$.

Απόδειξη: Έστω k οποιαδήποτε τιμή, $1 \leq k \leq n-1$.

Ορίζουμε την a' = shift(α, **x**), όπου $x_i = -u$ αν $0 \leq i \leq k-1$ και $x_i = 0$ αν $k \leq i \leq n-1$.

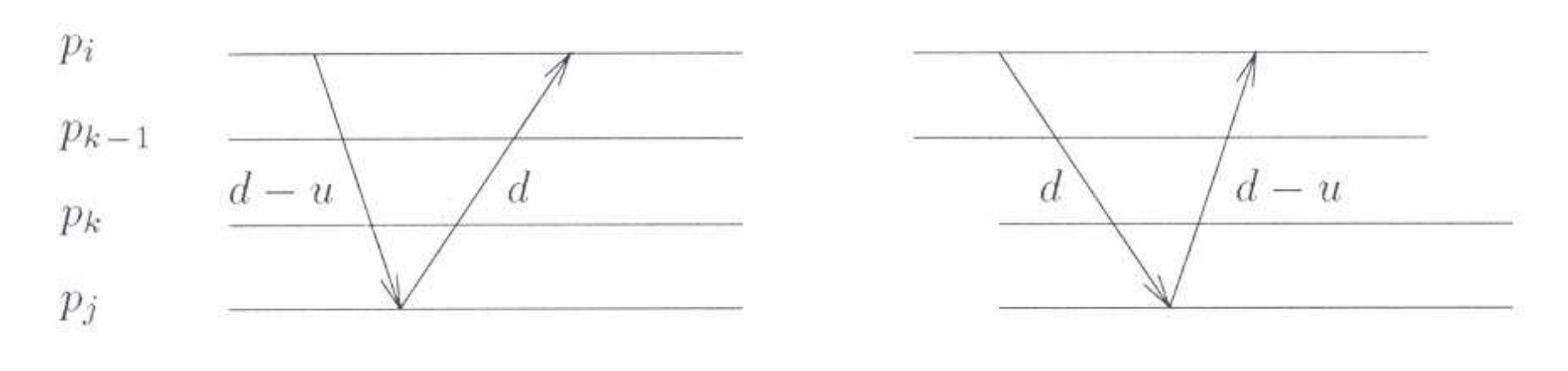
Έστω ότι p_i και p_j είναι δύο διεργασίες με $i < j$.

Αν $j \leq k-1$ ή $k \leq i$, τότε τα μηνύματα από την p_i στην p_j φθάνουν με καθυστέρηση d-u ενώ εκείνα από την p_j στην p_i φθάνουν με καθυστέρηση d.

Διαφορετικά, η καθυστέρηση μηνυμάτων από την p_i στην p_j = d και από την p_j στην p_i = d-u.

Ένα Κάτω Φράγμα

Απόδειξη Λήμματος (συνέχεια):



Επομένως, η α' είναι έγκυρη και έτσι ο Α πρέπει να επιτυγχάνει ε-συγχρονισμό των ρολογιών.

Αφού οι εκτελέσεις των διεργασιών μετακινούνται νωρίτερα σε πραγματικό χρόνο, η t είναι χρονική στιγμή που έπεται του τερματισμού στην α' $\Rightarrow AC_{k-1}'(t) \leq AC_k'(t) + \varepsilon$.

Έχουμε $AC_{k-1}'(t) = AC_{k-1}(t) + u$ και $AC_k'(t) = AC_k(t)$.

Από τα παραπάνω προκύπτει ότι:

$$AC_{k-1}(t) \leq AC_k(t) - u + \varepsilon.$$

Ένα Κάτω Φράγμα

Απόδειξη Θεωρήματος (συνέχεια):

Αφού ο Α υποτίθεται πως επιτυγχάνει ε-συγχρονισμό, ισχύει ότι $AC_{n-1}(t) \leq AC_0(t)$.

Εφαρμόζουμε επαναληπτικά το Λήμμα:

$$\begin{aligned} AC_{n-1}(t) &\leq AC_0(t) + \varepsilon \\ &\leq AC_1(t) - u + \varepsilon + \varepsilon = AC_1(t) - u + 2\varepsilon \\ &\leq AC_2(t) - u + \varepsilon - u + 2\varepsilon = AC_2(t) - 2u + 3\varepsilon \\ &\leq \ldots \\ &\ldots \\ &\leq AC_{n-1}(t) - (n-1)u + n\varepsilon \end{aligned}$$

$\Rightarrow (n-1)u \leq n\varepsilon$

$\Rightarrow \varepsilon \geq (1 - 1/n)u$, όπως απαιτείται!