Ατομικά Αντικείμενα

ΑΤΟΜΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ

Ένα ατομικό αντικείμενο κάποιου τύπου μοιάζει με μια διαμοιραζόμενη μεταβλητή αυτού του τύπου.

Ένα ατομικό αντικείμενο βρίσκεται σε μια κατάσταση και υποστηρίζει ένα σύνολο από λειτουργίες μέσω των οποίων μπορεί να αλλάξει η κατάστασή του.

Τα ατομικά αντικείμενα έχουν προταθεί σαν δομικά μπλοκ για την κατασκευή πολυ-επεξεργαστικών συστημάτων:

- Εκκίνηση με απλά βασικά ατομικά αντικείμενα που παρέχονται από το υλικό.
- Διαδοχική κατασκευή όλο και πιο πολύπλοκων ατομικών αντικειμένων από πιο απλά αντικείμενα.
- Το σύστημα που προκύπτει είναι απλό, καλά δομημένο, και είναι εύκολο να αποδειχθεί ότι είναι ορθό.

ΒΑΣΙΚΑ ΑΤΟΜΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ

Καταχωρητής Read/Write

Αποθηκεύει μια τιμή από κάποιο σύνολο και υποστηρίζει δύο λειτουργίες:

- read(R): επιστρέφει την τιμή που είναι αποθηκευμένη στον R χωρίς να τον αλλάξει
- write(R,v): γράφει την τιμή v στον R και επιστρέφει ack

Καταχωρητής Test-And-Set

Αποθηκεύει μια τιμή από το σύνολο {0,1} και υποστηρίζει δύο λειτουργίες:

- Test-And-Set(R): αποθηκεύει την τιμή 1 στον R και επιστρέφει την παλιά τιμή του
- reset(R): αποθηκεύει την τιμή 0 στον R

ΒΑΣΙΚΑ ΑΤΟΜΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ

Καταχωρητής Load-Link, Store-Conditional (LL/SC)

Αποθηκεύει μια τιμή από κάποιο σύνολο και υποστηρίζει δύο λειτουργίες LL και SC, όπου κάθε SC συσχετίζεται με την τελευταία LL που έχει εκτελεστεί από την ίδια διεργασία που εκτέλεσε την SC (δηλαδή, πριν μια διεργασία εκτελέσει την SC θα πρέπει να έχει εκτελέσει την LL που συσχετίζεται με την SC).

- ◊ LL(R): επιστρέφει την τιμή του R,
- ◊ SC(R,v): αποθηκεύει v στον R αν δεν έχει γίνει καμία αλλαγή στον R από την χρονική στιγμή που εκτελέστηκε η LL η οποία έχει συσχετισθεί με την SC.

Καταχωρητής Compare-And-Swap

Αποθηκεύει μια τιμή από κάποιο σύνολο και υποστηρίζει δύο λειτουργίες:

- ◊ read(R): επιστρέφει την τιμή του R,
- ◊ CAS(R,u,v): Αν η κατάσταση του R είναι u αλλάζει σε v και επιστρέφεται TRUE. Διαφορετικά δεν αλλάζει ο R και επιστρέφεται FALSE.

ΒΑΣΙΚΑ ΑΤΟΜΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ

Καταχωρητής Πολλαπλής-Εγγραφής (MW)

Όλες οι διεργασίες επιτρέπεται να εκτελούν λειτουργίες ενημέρωσης στον καταχωρητή

Καταχωρητής Απλής-Εγγραφής (SW)

Μόνο μια από τις διεργασίες επιτρέπεται να εκτελεί λειτουργίες ενημέρωσης στον καταχωρητή.

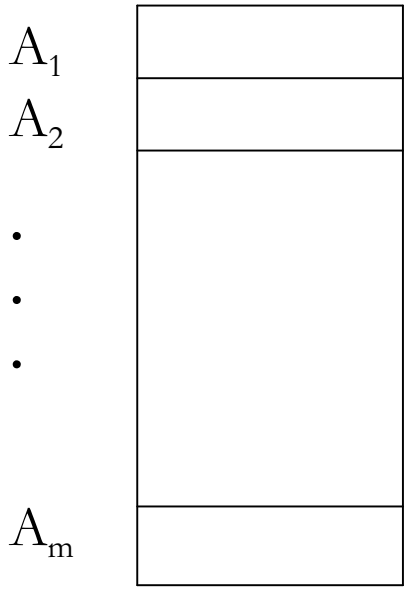
Χωρητικότητα καταχωρητή

- Ένας καταχωρητής είναι πεπερασμένης χωρητικότητας όταν το σύνολο τιμών που μπορεί να αποθηκεύει είναι πεπερασμένο. Στην αντίθετη περίπτωση είναι μη-πεπερασμένης χωρητικότητας.

$\Rightarrow$ Το υλικό πολλών συστημάτων υποστηρίζει κάποια από τα παραπάνω αντικείμενα. Το υλικό τότε εγγυάται ότι κάθε μια λειτουργία εκτελείται ατομικά.

ΑΤΟΜΙΚΑ ΣΤΙΓΜΙΟΤΥΠΑ ΜΝΗΜΗΣ

- Διαμοιραζόμενο αντικείμενο που αποτελείται από έναν πίνακα με m τμήματα/συνιστώσες που το καθένα αποθηκεύει μια τιμή.
- Υποστηρίζει δύο είδη ατομικών λειτουργιών:
  - ✓ UPDATE(i,v): εγγράφει στο τμήμα A_i του στιγμιοτύπου την τιμή υ.
  - ✓ SCAN: επιστρέφει ένα διάνυσμα τιμών, μία για κάθε τμήμα και είναι εγγυημένο πως οι τιμές αυτές είναι «συνεπείς».



Ατομικό Στιγμιότυπο A

Τα ατομικά στιγμιότυπα αποσκοπούν στην παροχή «συνεπών» όψεων των διαμοιραζόμενων μεταβλητών.

ΑΤΟΜΙΚΑ ΣΤΙΓΜΙΟΤΥΠΑ

Ατομικό Στιγμιότυπο Πολλαπλής-Εγγραφής (MW Atomic Snapshot)

Κάθε διεργασία μπορεί να εκτελεί UPDATES σε κάθε τμήμα.

Στιγμιότυπο Απλής-Εγγραφής (SW Atomic Snapshot)

Μόνο η διεργασία i μπορεί να εκτελεί UPDATES στο τμήμα A_i.

Τα ατομικά στιγμιότυπα είναι πιο ισχυρά αντικείμενα από τους καταχωρητές και διευκολύνουν τη σχεδίαση κατανεμημένων αλγορίθμων!

ΕΠΑΛΗΘΕΥΣΗ ΚΑΘΟΛΙΚΩΝ ΚΑΤΗΓΟΡΗΜΑΤΩΝ (GLOBAL PREDICATE EVALUATION)

Σε πολλά προβλήματα των κατανεμημένων συστημάτων μια ενέργεια πρέπει να συμβεί αν ένα κατηγόρημα (predicate) που αφορά την καθολική κατάσταση του συστήματος είναι αληθές.

Παραδείγματα

- Ανίχνευση αδιεξόδων (deadlock detection).
- Ανίχνευση τερματισμού (termination detection).
- Ανίχνευση απώλειας διακριτικού (token loss detection).
- Διαχείριση μη-προσβάσιμης ή αχρησιμοποίητης μνήμης (garbage collection).
- Εισαγωγή σημείων ελέγχου και επανεκκίνηση (checkpointing & restarting).
- Παρακολούθηση και αποσφαλμάτωση (monitoring & debugging).

ΟΡΘΟΤΗΤΑ & ΑΠΟΣΦΑΛΜΑΤΩΣΗ

- Η μεγαλύτερη δυσκολία στο να αποδειχθεί η ορθότητα λειτουργίας ενός κατανεμημένου αλγορίθμου είναι η αναγκαιότητα στήριξης της επιχειρηματολογίας σε μη συνεπείς όψεις των διαμοιραζόμενων μεταβλητών.

- Ο υπολογισμός συνεπών όψεων διευκολύνει το έργο επαλήθευσης της ορθότητας των κατανεμημένων αλγορίθμων αλλά δεν είναι εύκολο πρόβλημα.

ΕΦΑΡΜΟΓΕΣ

- Αμοιβαίος αποκλεισμός [Katseff - STOC'78, Lamport – JACM'86, Dolev & Gafni & Shavit – STOC'88].
- Επίτευξη ομοφωνίας [Aspnes - J. of Alg.'93, Aspnes & Herlihy – J. of Alg '90] & Approximate Agreement [Attiya, Lynch & Shavit – JACM'94]
- Κατασκευή συστημάτων παράλληλης ανάθεσης χρονοσφραγίδων [Dolev & Shavit - STOC'88].
- Αξιόπιστη και αποδοτική υλοποίηση κατανεμημένων δομών δεδομένων [Aspnes & Herlihy - SPAA'90, Herlihy – PODC'91].
- Υλοποίηση χρήσιμων κατανεμημένων αντικειμένων [Vitanyi & Awerbuch - FOCS'86, Bloom – PODC'87, Peterson & Burns – FOCS'87].

ΥΛΟΠΟΙΩΝΤΑΣ ΑΤΟΜΙΚΑ ΣΤΙΓΜΙΟΤΥΠΑ ΜΕ ΧΡΗΣΗ ΚΑΤΑΧΩΡΗΤΩΝ

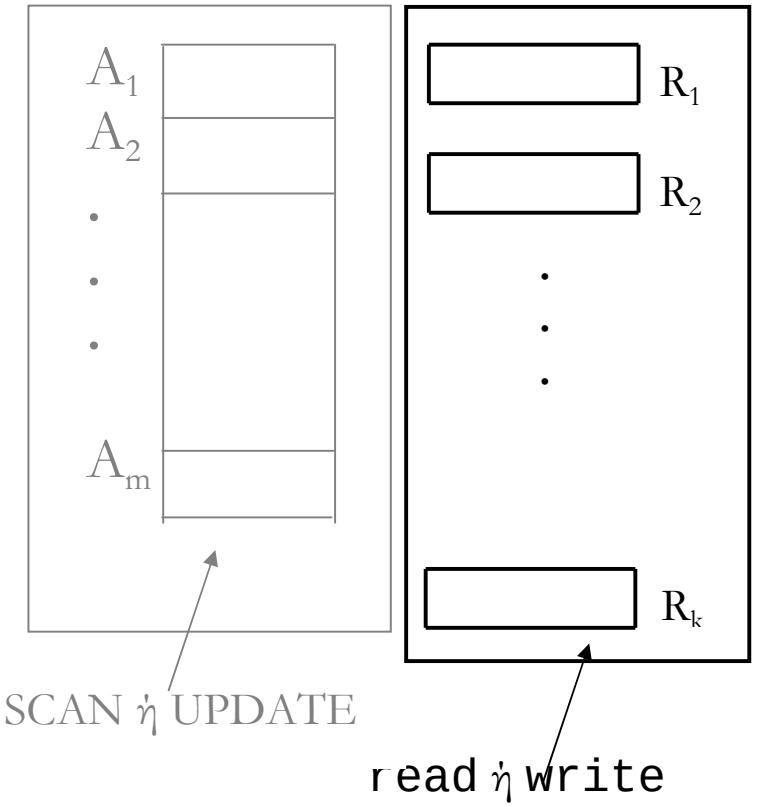
☹ Τα ατομικά στιγμιότυπα δεν παρέχονται από το υλικό.

? Παροχή αλγορίθμων για τις λειτουργίες SCAN και UPDATE.

? Αποτελεσματικότητα Υλοποίησης

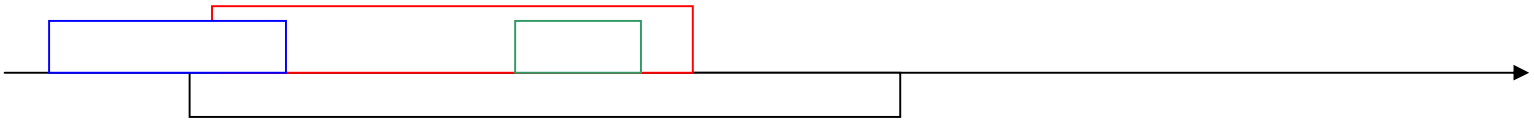
Χρονική πολυπλοκότητα μιας SCAN ή μιας UPDATE λειτουργίας: ο μέγιστος αριθμός βημάτων που εκτελούνται από οποιαδήποτε διεργασία σε οποιαδήποτε εκτέλεση προκειμένου να υλοποιηθεί η λειτουργία

Χωρική Πολυπλοκότητα υλοποίησης: # καταχωρητών που χρησιμοποιούνται



Χρήσιμοι Ορισμοί

◊ Διαφορετικά στιγμιότυπα των λειτουργιών του αντικειμένου μπορούν να εκτελούνται από διαφορετικές διεργασίες ταυτόχρονα.



◊ Σε κάθε εκτέλεση, οι κλήσεις και οι αποκρίσεις των λειτουργιών του ατομικού αντικειμένου μπορεί να πραγματοποιούνται με οποιαδήποτε σειρά. Πολλές κλήσεις μπορούν π.χ. να συμβούν πριν να συμβεί οποιαδήποτε απόκριση.

◊ Το διάστημα εκτέλεσης μιας λειτουργίας είναι το διάστημα από την κλήση της μέχρι την απόκρισή της. Τα διαστήματα εκτέλεσης των λειτουργιών ενός ατομικού στιγμιοτύπου μπορεί να επικαλύπτονται (μερικώς ή ολικώς).

◊ Μια ακολουθία από κλήσεις και αποκρίσεις είναι έγκυρη, αν κάθε κλήση μπορεί να συσχετισθεί με μια απόκριση.

ΤΕΡΜΑΤΙΣΜΟΣ & ΣΕΙΡΙΟΠΟΙΗΣΙΜΟΤΗΤΑ

Ιδιότητα Ελευθερίας Αναμονής (Wait-Freedom)

Κάθε μη-εσφαλμένη διεργασία εκτελεί έναν πεπερασμένο αριθμό βημάτων προκειμένου να διεκπεραιώσει τη λειτουργία της.

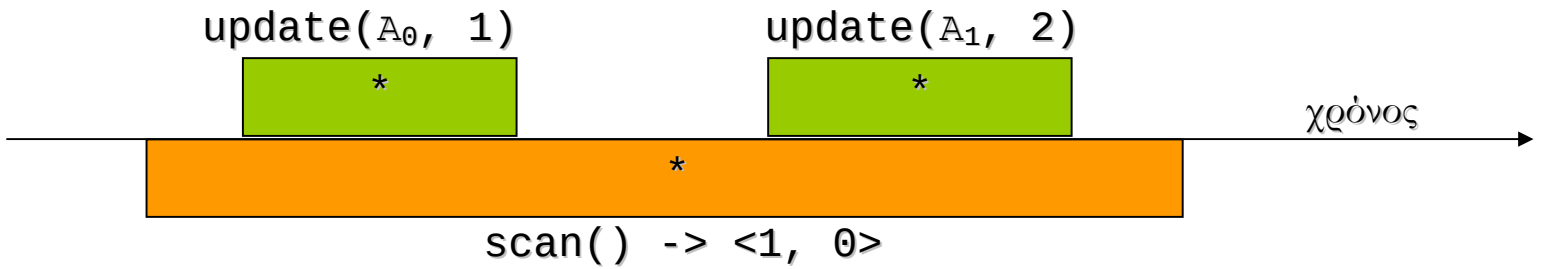
Σειριοποιησιμότητα (Linearizability)

Ακόμα και αν εκτελούνται παράλληλα λειτουργίες σε ένα διαμοιραζόμενο αντικείμενο, υπάρχει μία σειριακή εκτέλεση των λειτουργιών αυτών στην οποία όλες οι λειτουργίες έχουν τις ίδιες αποκρίσεις.

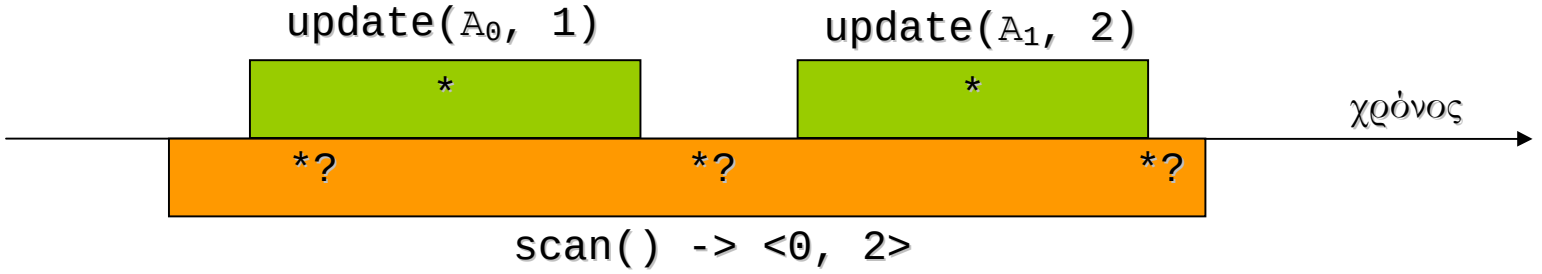
- Σε κάθε εκτέλεση, κάθε λειτουργία που εμπεριέχεται στην εκτέλεση έχει την ίδια απόκριση με μια σειριακή εκτέλεση της λειτουργίας σε κάποιο σημείο στο χρονικό διάστημα της εκτέλεσης της. Το σημείο αυτό λέγεται σημείο σειριοποίησης της λειτουργίας (linearization point).

ΣΕΙΡΙΟΠΟΙΗΣΙΜΟΤΗΤΑ

Παράδειγμα σειριοποιήσιμης εκτέλεσης



Παράδειγμα μη-σειριοποιήσιμης εκτέλεσης



Σειριοποιησιμότητα - Φορμαλιστικά

Έστω μια εκτέλεση β. Λέμε ότι η β είναι σειριοποιήσιμη αν τα ακόλουθα είναι δυνατά:

1. Για κάθε προσομοιούμενη λειτουργία π που έχει περατωθεί στη β, μπορούμε να επιλέξουμε ένα σημείο σειριοποίησης *π κάπου μεταξύ της κλήσης της π και της απόκρισής της.
2. Μπορούμε να επιλέξουμε ένα υποσύνολο Φ των λειτουργιών των οποίων η εκτέλεση δεν έχει περατωθεί στη β τ.ω. για κάθε λειτουργία π στο Φ:
   - μπορούμε να εισάγουμε μια απόκριση, και
   - μπορούμε να επιλέξουμε ένα σημείο σειριοποίησης *π, το οποίο να βρίσκεται σε κάποιο σημείο της εκτέλεσης που έπεται της κλήσης της π.

Διαισθητικά:

«Κάθε λειτουργία μοιάζει σαν να εκτελέστηκε στιγμιαία κάποια χρονική στιγμή μέσα στο διάστημα εκτέλεσής της.»

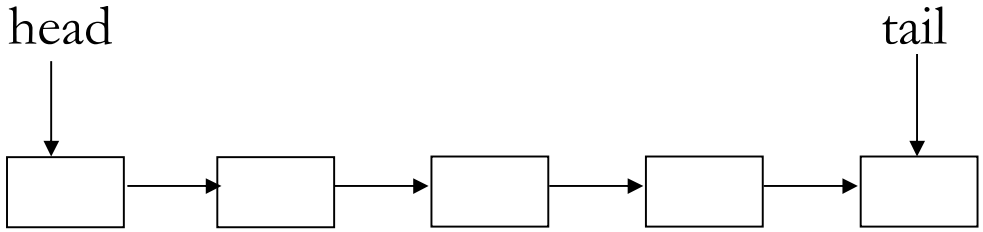
«Αν και η εκτέλεση κάποιων λειτουργιών μπορεί να συμβαίνει ταυτόχρονα, η υλοποίηση πρέπει να εγγυάται ότι οι λειτουργίες λαμβάνουν χώρα σειριακά, μια κάθε φορά, και η σειριακή αυτή διάταξη θα πρέπει να σέβεται τις κλήσεις και τις αποκρίσεις των λειτουργιών».

Σειριοποιησιμότητα – Περισσότερα Παραδείγματα

Μια ουρά FIFO Q υποστηρίζει δύο λειτουργίες:

- ◊ enq(Q,v): προσθέτει το στοιχείο v στο τέλος της ουράς Q
- ◊ deq(Q): αφαιρεί και επιστρέφει το πρώτο στοιχείο της ουράς Q

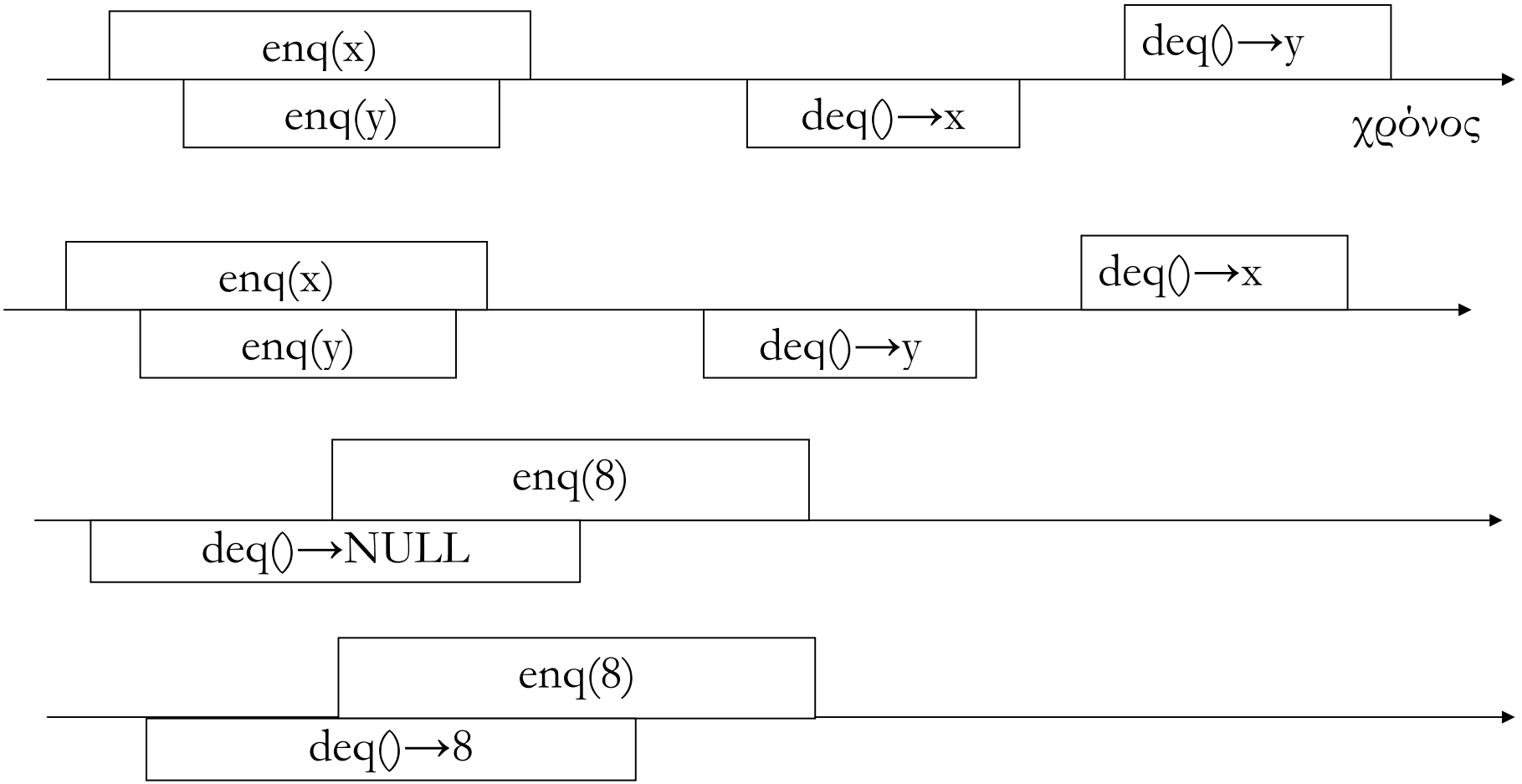
Σε μια παράλληλη ουρά πολλές διεργασίες μπορεί να προσπαθούν να εισάγουν (enq) και να εξάγουν(deq) στοιχεία ταυτόχρονα.



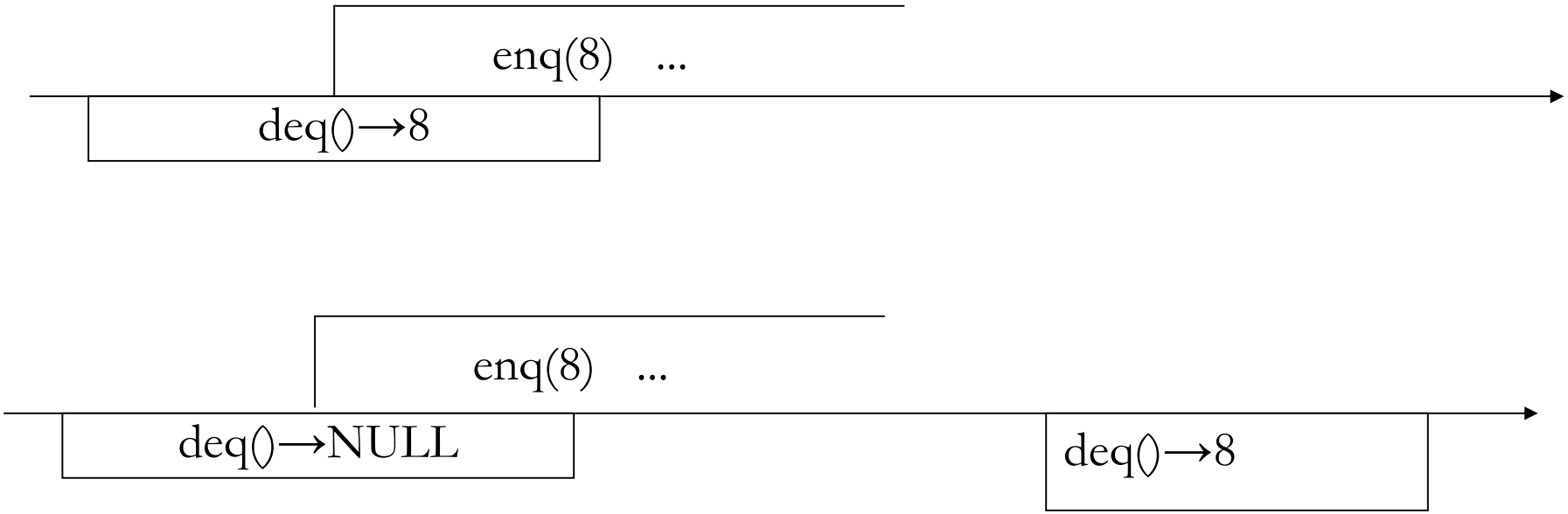
Ουρές μπορούν να υλοποιηθούν από απλούστερα αντικείμενα (π.χ., Test-And-Set και LL/SC registers).

Το ίδιο ισχύει και για άλλα αντικείμενα, π.χ., στοίβες, λίστες, λίστες παράλειψης, γράφους, κλπ.

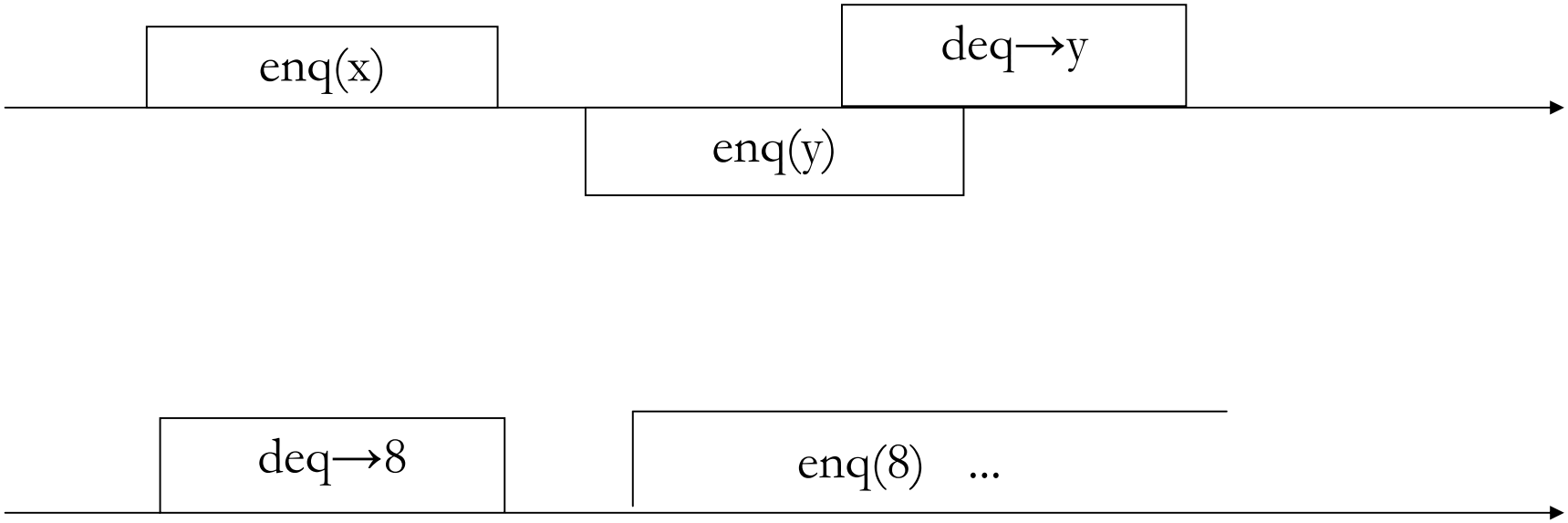
Σειριοποιήσιμες Εκτελέσεις



Σειριοποιήσιμες Εκτελέσεις



Μη-Σειριοποιήσιμες Εκτελέσεις



Σειριοποίησιμες Υλοποιήσεις

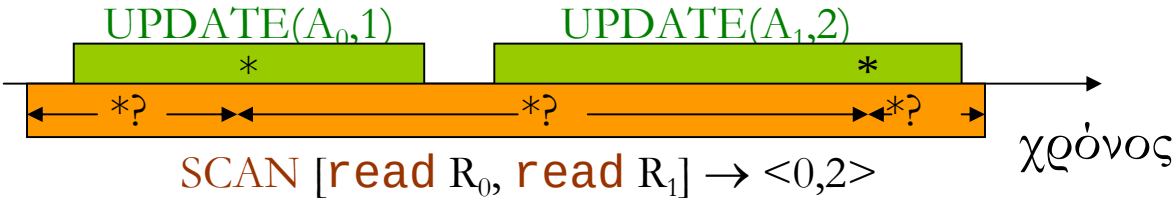
Μια υλοποίηση λέγεται σειριοποιήσιμη αν όλες οι εκτελέσεις που παράγει είναι σειριοποιήσιμες.

Στη συνέχεια του μαθήματος, μια υλοποίηση δεν θα θεωρείται ορθή αν δεν είναι σειριοποιήσιμη.

Ατομικά Στιγμιότυπα – Προφανής Λύση

Είναι δύσκολο για μια διεργασία να καταφέρει να υπολογίσει μια συνεπή όψη όλων των καταχωρητών.

Λειτουργίες Διεργασιών		Τιμές Καταχ/τών	
P	**Q**	**R_0**	**R_1**
		0	0
read $R_0 \rightarrow 0$			
	write(R_0,1)		
		1	0
	write(R_1,2)		
		1	2
read $R_1 \rightarrow 2$			



☹ Διαισθητικά: «Σε καμία χρονική στιγμή δεν συνυπάρχουν οι τιμές 0 και 2 στους καταχωρητές.»

☞ Τα ατομικά στιγμιότυπα έχουν σειριοποιήσιμες, ελεύθερες-αναμονής υλοποιήσεις χρησιμοποιώντας καταχωρητές. *[Afek et. al–JACM'93, Anderson–DistComp'93, Aspnes&Herlihy–SPAA'90]*

ΥΛΟΠΟΙΗΣΕΙΣ ΑΤΟΜΙΚΩΝ ΣΤΙΓΜΙΟΤΥΠΩΝ

- Anderson – PODC'90, PODC'93, DISC'91
- Afek, Attiya, Dolev, Gafni, Merritt and Shavit – PODC'90 + JACM'93
- Attiya, Herlihy & Rachman - DISC'92
- Attiya & Rachman – PODC'93 + SICOMP'98
- Israeli & Shaham – PODC'92
- Israeli, Shaham & Shirazi – DISC'93
- Inoue, Chen, Masuzawa & Tokura – DISC'94
- Afek, Stupp & Touitou – FOCS'99
- Afek, Attiya, Fouren, Stupp & Touitou – PODC'99
- Fatourou, Fich & Ruppert – PODC'02, STOC'03, STOC'06
- Fatourou & Kallimanis – PODC'06

ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Υπάρχουν n τμήματα, A_1, ..., A_n, ένα για κάθε διεργασία. Στο τμήμα A_i μπορεί να εκτελεί UPDATES μόνο η p_i.

Η υλοποίηση χρησιμοποιεί n καταχωρητές R_1, .., R_n, ένα για κάθε τμήμα. Ο καταχωρητής R_i έχει συσχετισθεί με το τμήμα A_i. UPDATES στο A_i γράφουν μόνο στον R_i. Κάθε καταχωρητής R_i μπορεί να εγγραφεί μόνο από την p_i αλλά να διαβαστεί από όλες τις διεργασίες ⇒ Οι καταχωρητές είναι single-writer.

Θεωρούμε πως κάθε καταχωρητής R_i είναι αρκετά μεγάλος ώστε να αποθηκεύει τα εξής πεδία:

◊ val_i: τιμή του τμήματος με το οποίο έχει συσχετισθεί

◊ tag_i: χρονοσφραγίδα την οποία χρησιμοποιεί η p_i για να ξεχωρίζει τις UPDATES της

◊ $view_i$: διάνυσμα n τιμών, μια για κάθε τμήμα.

Η υπόθεση αυτή δεν είναι ρεαλιστική (αφού τόσο μεγάλοι καταχωρητές δεν παρέχονται στην πράξη), αλλά προς το παρόν μας ενδιαφέρει απλά να σχεδιάσουμε μια απλή υλοποίηση ενός ατομικού στιγμιοτύπου μνήμης.

ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Ο αλγόριθμος UnboundedSnapshot

UPDATE στο τμήμα A_i με τιμή v:

view := SCAN;

increment p's tag;

`write`(R_i,<v, tag,view>);

SCAN από τη διεργασία p:

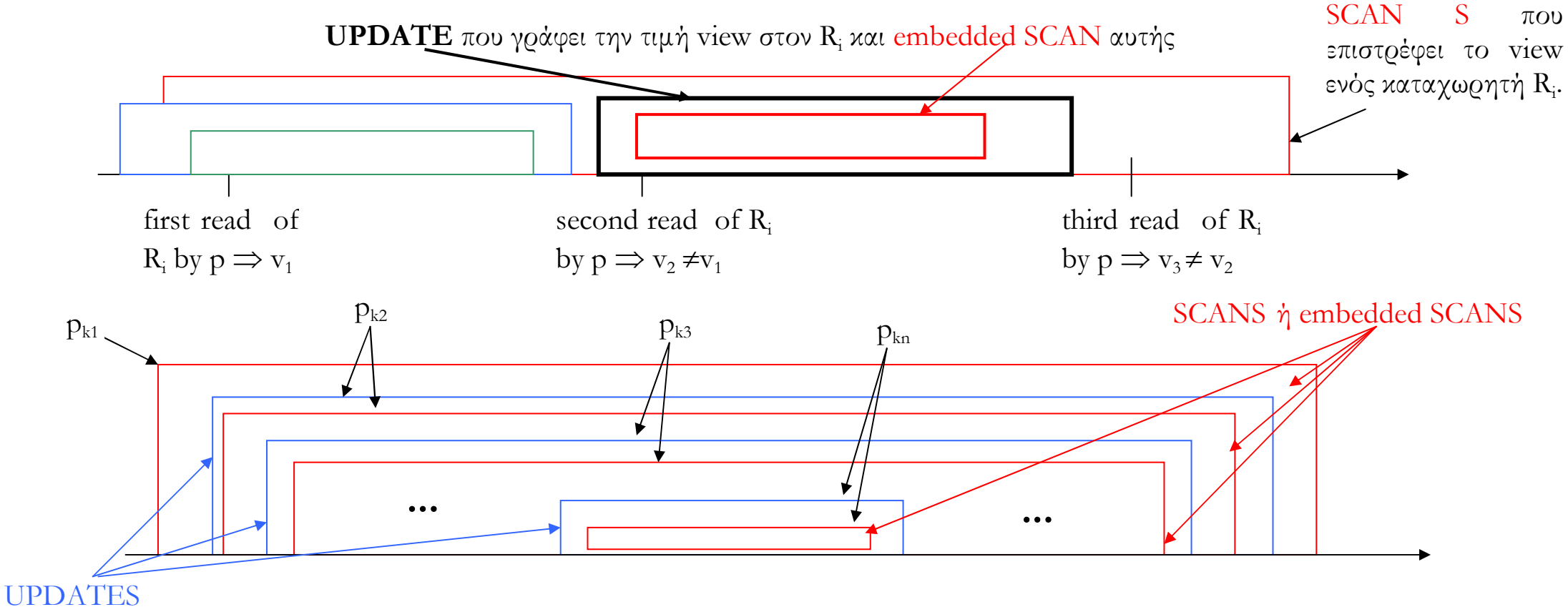
repeatedly `read` $R_1, \ldots, R_m$ until see

1. either the same vector of values twice (then, return the vector), or
2. three different values in some register R_i (return the view field of R_i read the third time);

☞ Χρονική Πολυπλοκότητα = $O(n^2)$ για τη SCAN και την UPDATE.

☞ Απαιτούνται *n* μη-πεπερασμένης χωρητικότητας MW καταχωρητές.

ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ



Όλες οι p_{k1}, …, p_{kn} είναι ενεργές ταυτόχρονα. Έχουμε n διεργασίες στο σύστημα. Άρα, η embedded SCAN που εκτελείται από την pkn πρέπει να τερματίζει βλέποντας το ίδιο διάνυσμα τιμών σε δύο συνεχόμενες ομάδες αναγνώσεων.

ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Θεώρημα: Ο παραπάνω αλγόριθμος είναι μια ορθή υλοποίηση ενός ατομικού στιγμιοτύπου μνήμης και επιτυγχάνει τερματισμό ελεύθερο-αναμονής.

Ορισμός: Μια SCAN επιστρέφει ένα **συνεπές διάνυσμα** τιμών αν, για κάθε τμήμα του στιγμιοτύπου, η τιμή που επιστρέφει η SCAN για το τμήμα είναι η τιμή που χρησιμοποίησε (ως παράμετρο) η τελευταία UPDATE στο τμήμα που σειριοποιείται πριν τη SCAN.

Για να αποδείξουμε το θεώρημα θα πρέπει:

- ◊ Να αποδώσουμε σημεία σειριοποίησης.
- ◊ Να δείξουμε ότι για κάθε λειτουργία (SCAN ή UPDATE), το σημείο σειριοποίησης της λειτουργίας βρίσκεται εντός του διαστήματος εκτέλεσής της.
- ◊ Να δείξουμε ότι οι SCANS επιστρέφουν συνεπή διανύσματα.

ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Απόδοση Σημείων Σειριοποίησης

⇒ Κάθε UPDATE σειριοποιείται στο σημείο που εκτελεί τη write εντολή της.

Για ευκολία αποδίδουμε σημεία σειριοποίησης όχι μόνο στις SCANS αλλά και στις embedded SCANS (δηλαδή σε αυτές που καλούνται από τις UPDATES). Στη συνέχεια θα τις ονομάζουμε όλες SCAN.

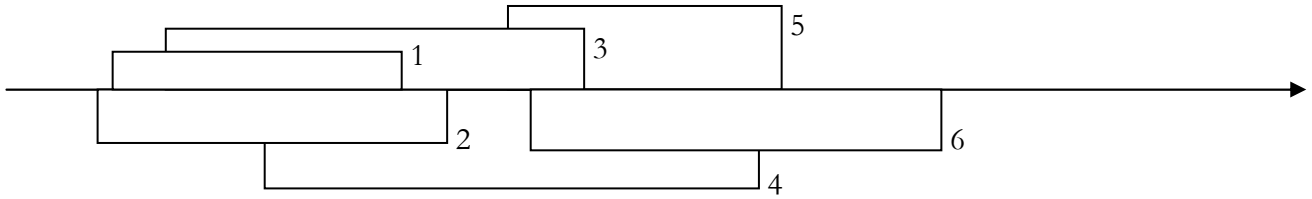
Χωρίζουμε τις SCAN σε δύο κατηγορίες:

Άμεσες: Αυτές που τερματίζουν λόγω της συνθήκης 1 στον ψευδο-κώδικα.

Έμμεσες: Αυτές που τερματίζουν λόγω της συνθήκης 2.

⇒ Σειριοποιούμε κάθε άμεση SCAN ανάμεσα στις δύο ομάδες αναγνώσεων (collect) που επιστρέφουν το ίδιο διάνυσμα.

⇒ Σειριοποιούμε τις έμμεσες SCAN με επαγωγή ως προς τα σημεία απόκρισης τους.



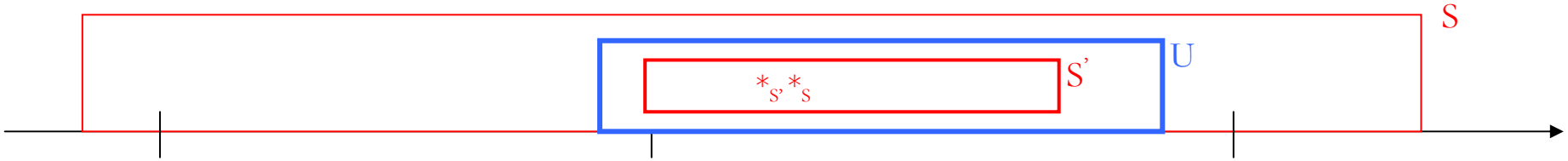
ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Βάση Επαγωγής: Απόδοση σημείου σειριοποίησης στην έμμεση SCAN (έστω S) της οποίας το σημείο απόκρισης είναι 1ο.

Η S επιστρέφει ένα διάνυσμα τιμών το οποίο έχει εγγραφεί από μια UPDATE U (δηλαδή το διάνυσμα αυτό υπολογίστηκε από την embedded SCAN S' της UPDATE). Η U και άρα και η S' (που εκτελείται από την U) εμπεριέχονται στην S.

Η S' είναι άμεση SCAN (αφού αν δεν ήταν τότε η έμεση SCAN της οποίας το σημείο απόκρισης είναι το 1ο θα ήταν η S' και όχι η S). Επομένως, έχει ήδη αποδοθεί σημείο σειριοποίησης στην S'.

Το σημείο σειριοποίησης της S τοποθετείται στο ίδιο σημείο με εκείνο της S'.



ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Επαγωγική Υπόθεση: Έστω S η έμμεση SCAN της οποίας το σημείο απόκρισης είναι το k-οστό. Ας υποθέσουμε ότι έχουμε αποδώσει σημεία σειριοποίησης σε όλες τις έμμεσες SCAN των οποίων τα σημεία απόκρισης προηγούνται εκείνου της S.

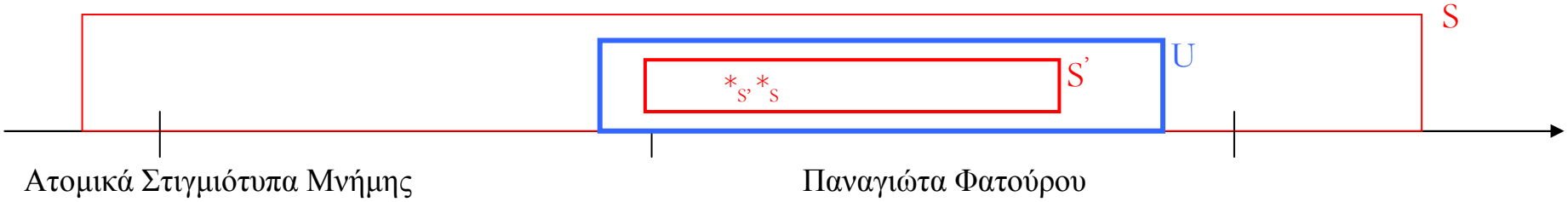
Επαγωγικό Βήμα: Αποδίδουμε σημείο σειριοποίησης στην S.

Η S επιστρέφει ένα διάνυσμα τιμών το οποίο έχει εγγραφεί από μια UPDATE U (δηλαδή το διάνυσμα αυτό υπολογίστηκε από την embedded SCAN S' της UPDATE). Η U (και άρα και η S') εμπεριέχονται στην S.

Αν η S' είναι άμεση SCAN, της έχει ήδη αποδοθεί σημείο σειριοποίησης.

Αν η S' είναι έμμεση SCAN, έχει τερματίσει πριν την S $\Rightarrow$ της έχει αποδοθεί σημείο σειριοποίησης (από επαγωγική υπόθεση).

Το σημείο σειριοποίησης της S τοποθετείται στο ίδιο σημείο με εκείνο της S'.



ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

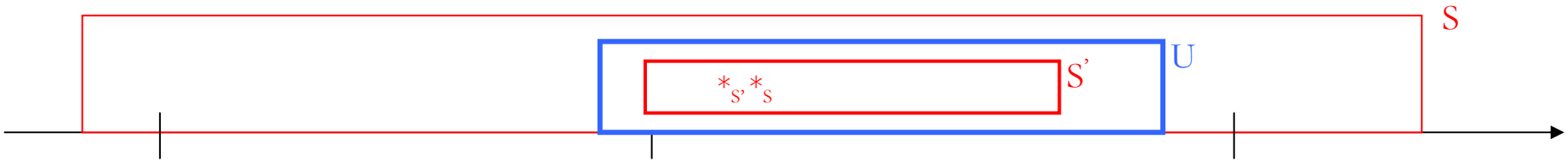
Λήμμα: Το σημείο σειριοποίησης κάθε SCAN ή UPDATE βρίσκεται μέσα στο διάστημα εκτέλεσης της.

Απόδειξη: Για τις UPDATES και τις άμεσες SCAN αυτό είναι προφανές.

Για τις έμμεσες SCANS αποδεικνύεται επαγωγικά ως προς τα σημεία απόκρισής τους.

Στη βάση της επαγωγής (θυμηθείτε ότι) η SCAN S της οποίας το σημείο απόκρισης είναι 1^{ο} σειριοποιείται στο ίδιο σημείο με την άμεση embedded SCAN S' της οποίας το διάνυσμα επιστρέφει. Επίσης, η S' εμπεριέχεται στην S.

Το σημείο σειριοποίησης της S' βρίσκεται στο διάστημα εκτέλεσής της $\Rightarrow$ το σημείο σειριοποίησης της S εμπεριέχεται στο διάστημα εκτέλεσής της.



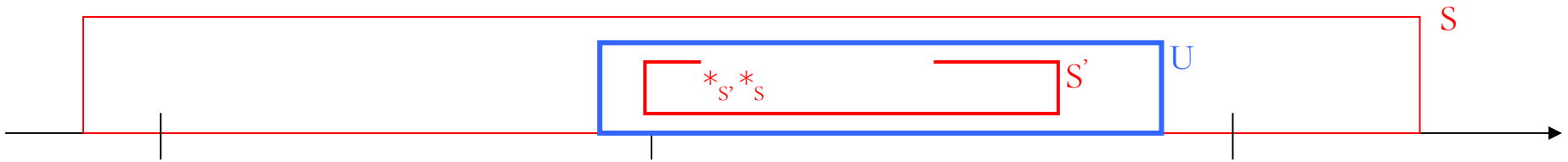
ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Επαγωγικό Βήμα: Αποδεικνύουμε τον ισχυρισμό για την έμμεση SCAN S της οποίας το σημείο σειριοποίησης είναι το k-οστό.

Η S επιστρέφει ένα διάνυσμα τιμών το οποίο έχει εγγραφεί από μια UPDATE U (δηλαδή το διάνυσμα αυτό υπολογίστηκε από την embedded SCAN S' της UPDATE). Η U (και άρα και η S') εμπεριέχονται στην S.

Αν η S' είναι άμεση, το σημείο σειριοποίησης είναι μέσα στο διάστημα εκτέλεσής της. Το ίδιο ισχύει αν η S' είναι έμμεση (από επαγωγική υπόθεση αφού η S' εμπεριέχεται στην S και άρα το σημείο απόκρισής της προηγείται εκείνου της S).

Το σημείο σειριοποίησης της S τοποθετείται στο ίδιο σημείο με εκείνο της S' $\Rightarrow$ το σημείο σειριοποίησης της S βρίσκεται μέσα στο διάστημα εκτέλεσής της.



ΑΠΛΗ ΥΛ/ΣΗ SW Snapshot ΜΕ ΧΡΗΣΗ n SW ΚΑΤΑΧΩΡΗΤΩΝ

Λήμμα: Σε κάθε εκτέλεση του αλγορίθμου, οι SCAN λειτουργίες επιστρέφουν συνεπή διανύσματα τιμών.

Απόδειξη: Ο ισχυρισμός ισχύει προφανώς για τις άμεσες SCAN.

Για έμμεσες SCAN θα αποδειχθεί με επαγωγή ως προς τα σημεία απόκρισης.

Στη βάση της επαγωγής (θυμηθείτε ότι) η SCAN S, της οποίας το σημείο απόκρισης είναι 1^o, σειριοποιείται στο ίδιο σημείο με την άμεση embedded SCAN S' της οποίας το διάνυσμα επιστρέφει. Αφού η S' επιστρέφει συνεπές διάνυσμα, το ίδιο και η S.

Επαγωγικό Βήμα: Αποδεικνύουμε τον ισχυρισμό για τη έμμεση SCAN S της οποίας το σημείο σειριοποίησης είναι το k-οστό. Η S επιστρέφει το ίδιο διάνυσμα τιμών που επιστρέφει μια embedded SCAN S' και σειριοποιείται στο ίδιο σημείο.

Αν η S' είναι άμεση, επιστρέφει προφανώς συνεπές διάνυσμα. Αν η S' είναι έμμεση επιστρέφει συνεπές διάνυσμα από επαγωγική υπόθεση (αφού σε αυτή την περίπτωση η S' τερματίζει πριν την S). Άρα και η S επιστρέφει συνεπές διάνυσμα.

A1	A2	A3	Vals	Scans
U(1,1)			[1,0,0]	
	U(2,1)		[1,1,0]	
				S1
U(1,3)			[3,1,0]	
	U(2,4)		[3,4,0]	
		U(3,5)	[3,4,5]	
		U(3,6)	[3,4,6]	
				S2
				S3
				S4

$*_{U(1,1)}^{[1,0,0]} *_{U(2,1)}^{[1,1,0]} *_{S1} *_{U(1,3)}^{[3,1,0]} *_{U(2,4)}^{[3,4,0]} *_{U(3,5)}^{[3,4,5]} *_{U(3,6)}^{[3,4,6]} *_{U(1,11)}^{[11,4,6]} *_{S2} *_{S3} *_{S4}$

time

Ατομικά Στιγμιότυπα Μνήμης

Παναγιώτα Φατούρου