Ομοφωνία σε σύστημα με αποτυχίες κατάρρευσης διεργασιών

Το Πρόβλημα Ομοφωνίας – Σύγχρονα Συστήματα Μεταβίβασης Μηνύματος

Μοντέλο Κατάρρευσης (crash model)

Οι διεργασίες μπορούν να σταματούν να εκτελούνται σε αυθαίρετα σημεία της εκτέλεσης. Μια διεργασία που έχει καταρρεύσει λέγεται εσφαλμένη (fault).

Επιτρεπτές Εκτελέσεις

- Όλες οι μη-εσφαλμένες διεργασίες εκτελούν απεριόριστο αριθμό βημάτων.
- Αν μια διεργασία αποτύχει σε κάποιο βήμα, δεν εκτελεί ξανά βήματα.
- Στο τελευταίο βήμα μιας εσφαλμένης διεργασίας, αποστέλλεται μόνο ένα αυθαίρετο υποσύνολο των εξερχόμενων μηνυμάτων της διεργασίας.

Υποθέσεις

Ο αριθμός των αποτυχιών είναι το πολύ f, όπου f είναι κάποιος θετικός ακέραιος. Το f ονομάζεται βαθμός ανθεκτικότητας (resiliency) του συστήματος.

Ο γράφος είναι κλίκα με n κόμβους.

Οι σύνδεσμοι (κανάλια επικοινωνίας) είναι αξιόπιστοι.

Το Πρόβλημα Ομοφωνίας – Σύγχρονα Συστήματα Μεταβίβασης Μηνύματος

Περιγραφή Προβλήματος

Κάθε διεργασία p_i έχει μια μεταβλητή εισόδου x_i και μια μεταβλητή εξόδου y_i.

Αρχικά, η x_i έχει μια τιμή από ένα σύνολο πιθανών εισόδων, ενώ η y_i δεν έχει αρχικοποιηθεί.

Η y_i θα πρέπει να εγγραφεί μία μόνο φορά (κάθε εγγραφή στην y_i είναι επομένως μη-αντιστρέψιμη).

Ένας αλγόριθμος που επιλύει το πρόβλημα ομοφωνίας πρέπει να εγγυάται τα εξής:

Ομοφωνία (agreement): Όλες οι μη-εσφαλμένες διεργασίες έχουν ως έξοδο την ίδια τιμή.

Εγκυρότητα (validity): Αν όλες οι διεργασίες έχουν την ίδια τιμή εισόδου, η τιμή εξόδου πρέπει να είναι η τιμή αυτή.

Τερματισμός (termination): Όλες οι μη-εσφαλμένες διεργασίες πρέπει να τερματίσουν μετά από έναν πεπερασμένο αριθμό βημάτων.

Το Πρόβλημα Ομοφωνίας – Σύγχρονα Συστήματα Μεταβίβασης Μηνύματος

Εφαρμογές

- Συστήματα ελέγχου πτήσεων.
- Οι διεργασίες ενός κατανεμημένου συστήματος χρειάζεται συχνά να συμφωνούν στο αν κάποιο μήνυμα έχει παραληφθεί.
- Διάγνωση αποτυχίας διεργασιών.

Πρόβλημα Ομοφωνίας – Ένας απλός αλγόριθμος

code for processor p_i, $0 \leq i \leq n-1$.

Initially $V = \{x\}$ // V contains p_i's input

round k, $1 \leq k \leq f+1$:

1: send $\{v \in V : p_i$ has not already sent $v\}$ to all processors
2: receive S_j from p_j, $0 \leq j \leq n-1$, $j \neq i$
3: $V := V \cup \bigcup_{j=0}^{n-1} S_j$
4: if $k = f+1$ then $y := \min(V)$ // decide

Κάθε διεργασία p διατηρεί ένα σύνολο με τιμές που έχει δει μέχρι την τρέχουσα χρονική στιγμή, το οποίο αρχικά περιέχει μόνο την τιμή εισόδου της διεργασίας.

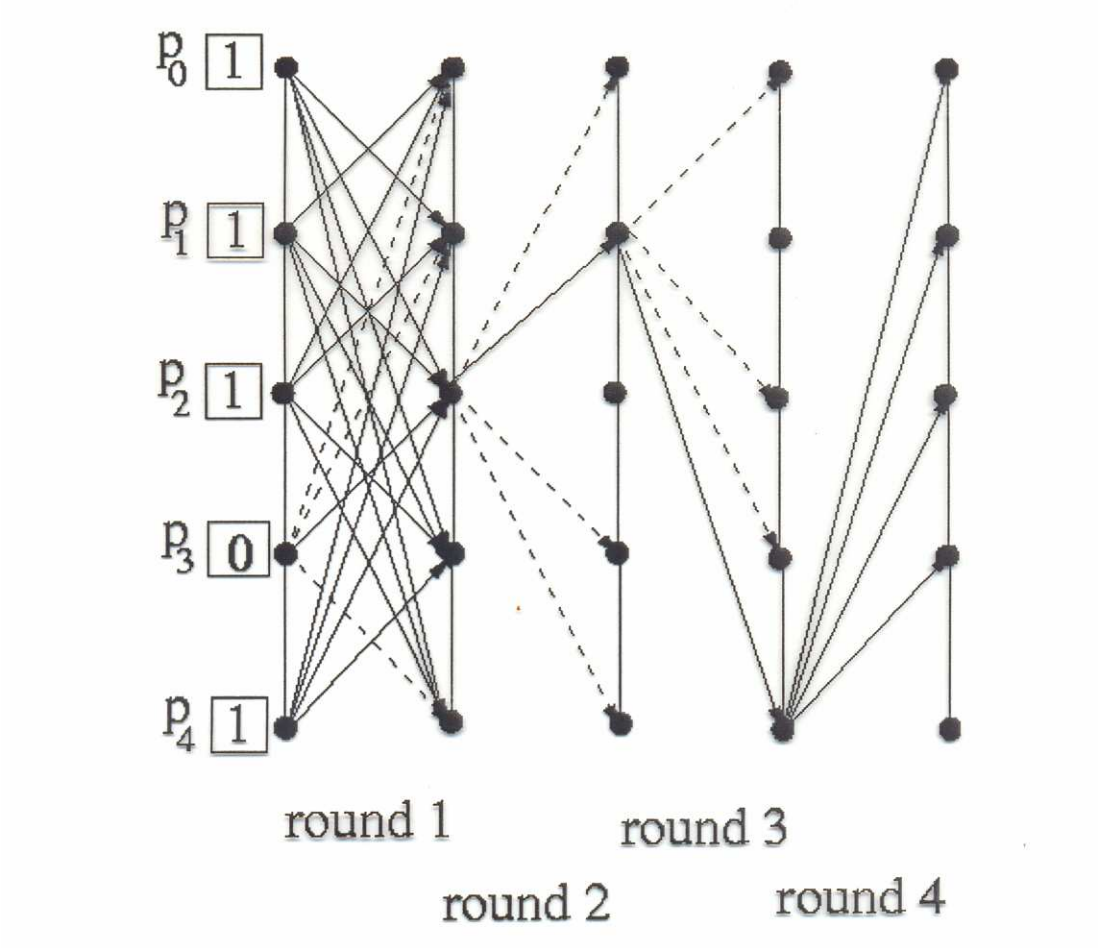
Για f+1 γύρους κάθε διεργασία:

- ενημερώνει το σύνολο της εκτελώντας την πράξη της ένωσης με τα σύνολα που λαμβάνει από άλλους καταχωρητές.
- εκπέμπει (broadcasts) οποιεσδήποτε προσθήκες έγιναν στο σύνολο σε όλες τις υπόλοιπες διεργασίες.

Μετά τους f+1 γύρους, η διεργασία αποφασίζει ως έξοδό της τη μικρότερη τιμή που περιέχεται στο σύνολό της.

Πρόβλημα Ομοφωνίας – Ένας απλός αλγόριθμος

Γιατί (f+1) γύροι;



f = 3, n = 4

Πρόβλημα Ομοφωνίας – Ένας απλός αλγόριθμος

Τερματισμός;

Εγκυρότητα;

Ομοφωνία:

- Ας υποθέσουμε ότι μια διεργασία p_i αποφασίζει μια μικρότερη τιμή, x, από κάποια άλλη p_j.
- Τότε, η x έχει παραμείνει «κρυφή» στην p_j για (f+1) γύρους.
- Έχουμε το πολύ f μη-εσφαλμένες διεργασίες. Άτοπο!!!

Αριθμός διεργασιών; $n > f$

Χρονική Πολυπλοκότητα; (f+1) γύρους

Πολυπλοκότητα Επικοινωνίας;

$n^2 * |V|$ μηνύματα, όπου V είναι το σύνολο τιμών εισόδου.

Αδυναμία Επίλυσης του Προβλήματος της Ομοφωνίας σε Ασύγχρονα Συστήματα Διαμοιραζομένης Μνήμης

- Δεν υπάρχει αλγόριθμος επίλυσης του προβλήματος της ομοφωνίας που να χρησιμοποιεί μόνο καταχωρητές ανάγνωσης-εγγραφής σε ασύγχρονο σύστημα στο οποίο έστω και μια διεργασία επιτρέπεται να αποτυγχάνει (με κατάρρευση).

Συνθήκες Τερματισμού

Τερματισμός Wait-free (ελευθερία-αναμονής): Ανεξάρτητα από τον αριθμό των αποτυχιών (που θα μπορούσαν να είναι ακόμη και n-1), οι μη-εσφαλμένες διεργασίες τερματίζουν μετά από έναν πεπερασμένο αριθμό βημάτων.

Τερματισμός f-failure: Αν συμβούν το πολύ f αποτυχίες, οι μη-εσφαλμένες διεργασίες τερματίζουν μετά από έναν πεπερασμένο αριθμό βημάτων.

- Θα αποδείξουμε ότι το αρνητικό αποτέλεσμα ισχύει για wait-free αλγορίθμους. (Αυτό είναι πιο ασθενές από τον αρχικό μας ισχυρισμό αλλά πιο εύκολο να αποδειχθεί!!!)

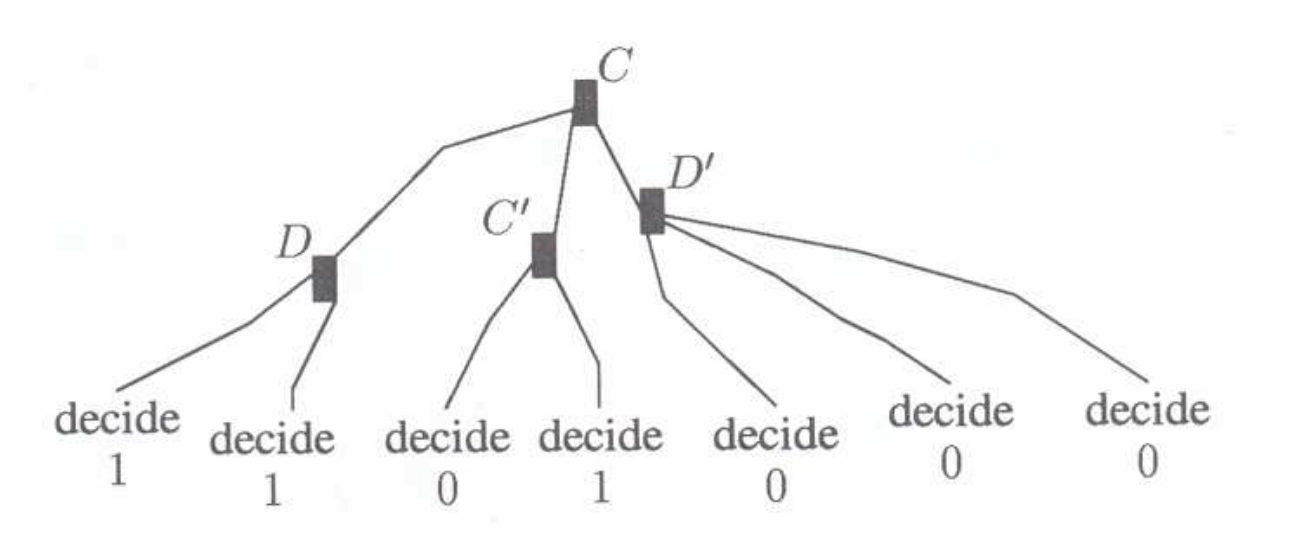
Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

Χρήσιμοι Ορισμοί

- Το *σθένος* (valency) μια καθολικής κατάστασης C είναι το σύνολο των τιμών που μπορούν να αποφασιστούν σε οποιαδήποτε εκτέλεση ξεκινά από τη C (ή από οποιαδήποτε άλλη προσβάσιμη από τη C καθολική κατάσταση). Το σθένος είναι *μονό* αν το σύνολο αυτό περιέχει μόνο το 0 ή μόνο το 1. Είναι *διπλό* αν το σύνολο περιέχει και το 0 και το 1.
- Μια καθολική κατάσταση C είναι *σθένους* 0 (ή 1) αν η μόνη τιμή που μπορεί να αποφασιστεί σε κάθε εκτέλεση που ξεκινά από την C (ή από οποιαδήποτε προσβάσιμη από τη C καθολική κατάσταση) είναι 0 (1, αντίστοιχα).
- Αν C είναι μια καθολική κατάσταση με διπλό σθένος, και η κατάσταση που προκύπτει επιτρέποντας σε μια διεργασία p να κάνει ένα βήμα από τη C είναι μονού σθένους, τότε λέμε ότι η p είναι *κρίσιμη* διεργασία στη C.

Θα αποδείξουμε το αρνητικό αποτέλεσμα με επαγωγή εις άτοπο. Έστω A ένας ασύγχρονος wait-free αλγόριθμος ομοφωνίας.

Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας



Λήμμα 1: Έστω ότι C_1 και C_2 είναι δύο καθολικές καταστάσεις μονού σθένους. Αν $C_1 \sim^p C_2$, για κάποια διεργασία p, τότε η C_1 είναι σθένους v αν και μόνο αν η C_2 είναι σθένους επίσης v, όπου $v \in \{0,1\}$.

Απόδειξη: Έστω ότι το σθένος της C_1 είναι v.

Έστω α μια άπειρη εκτέλεση ξεκινώντας από τη C_1 στην οποία μόνο η p κάνει βήματα.

Αφού ο A wait-free $\Rightarrow$ η p αποφασίζει στην α.

Αφού C_1 είναι σθένους v $\Rightarrow$ η p αποφασίζει v στην α.

Η α είναι έγκυρη και από την $C_2 \Rightarrow$ η C_2 είναι σθένους v.

Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

Λήμμα 2

Υπάρχει μια αρχική κατάσταση με διπλό σθένος.

Απόδειξη

Με εις άτοπο απαγωγή.

I_0: αρχική κατάσταση στην οποία όλες οι διεργασίες έχουν είσοδο $0 \Rightarrow I_0$ σθένους 0

I_1: αρχική κατάσταση στην οποία όλες οι διεργασίες έχουν είσοδο $1 \Rightarrow I_1$ σθένους 1

I_{01}: αρχική κατάσταση στην οποία η p_0 έχει είσοδο 0 και όλες οι άλλες διεργασίες έχουν είσοδο 1.

$I_{01} \sim^{p_0} I_0 \Rightarrow$ (από Λήμμα 1) I_{01} δεν μπορεί να είναι σθένους 1

$I_{01} \sim^{p_1} I_1 \Rightarrow$ (από Λήμμα 1) I_{01} δεν μπορεί να είναι σθένους 0

Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

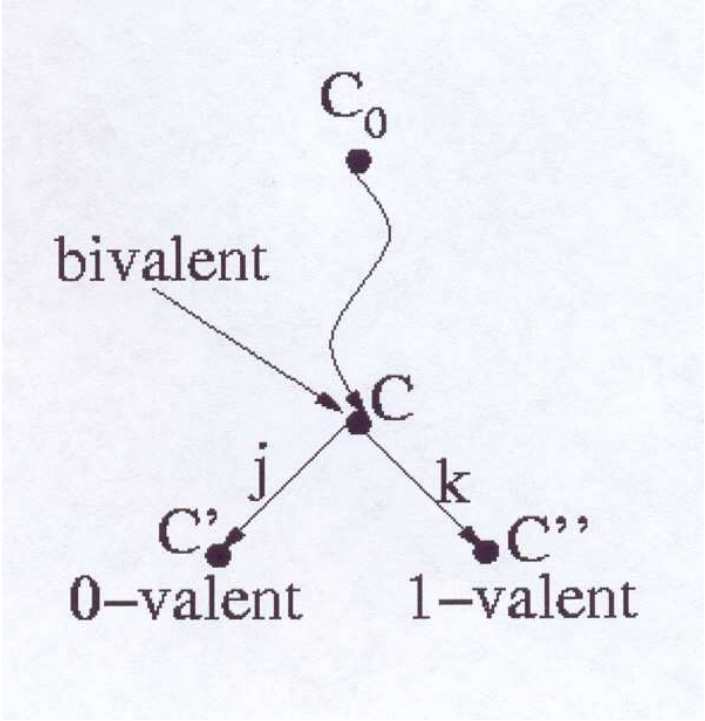
Λήμμα 3

Αν η C είναι μια καθολική κατάσταση διπλού σθένους, τότε τουλάχιστον μια διεργασία δεν είναι κρίσιμη στη C.

Απόδειξη

Με εις άτοπο απαγωγή. Έστω ότι όλες οι διεργασίες είναι κρίσιμες στη C.

Αφού C είναι διπλού σθένους και όλες οι διεργασίες κρίσιμες $\Rightarrow$ υπάρχουν p_j και p_k, τ.ω. αν η p_j κάνει ένα βήμα από τη C η προκύπτουσα κατάσταση έχει σθένος 0 και αν η p_k κάνει ένα βήμα από τη C η προκύπτουσα κατάσταση έχει σθένος 1.

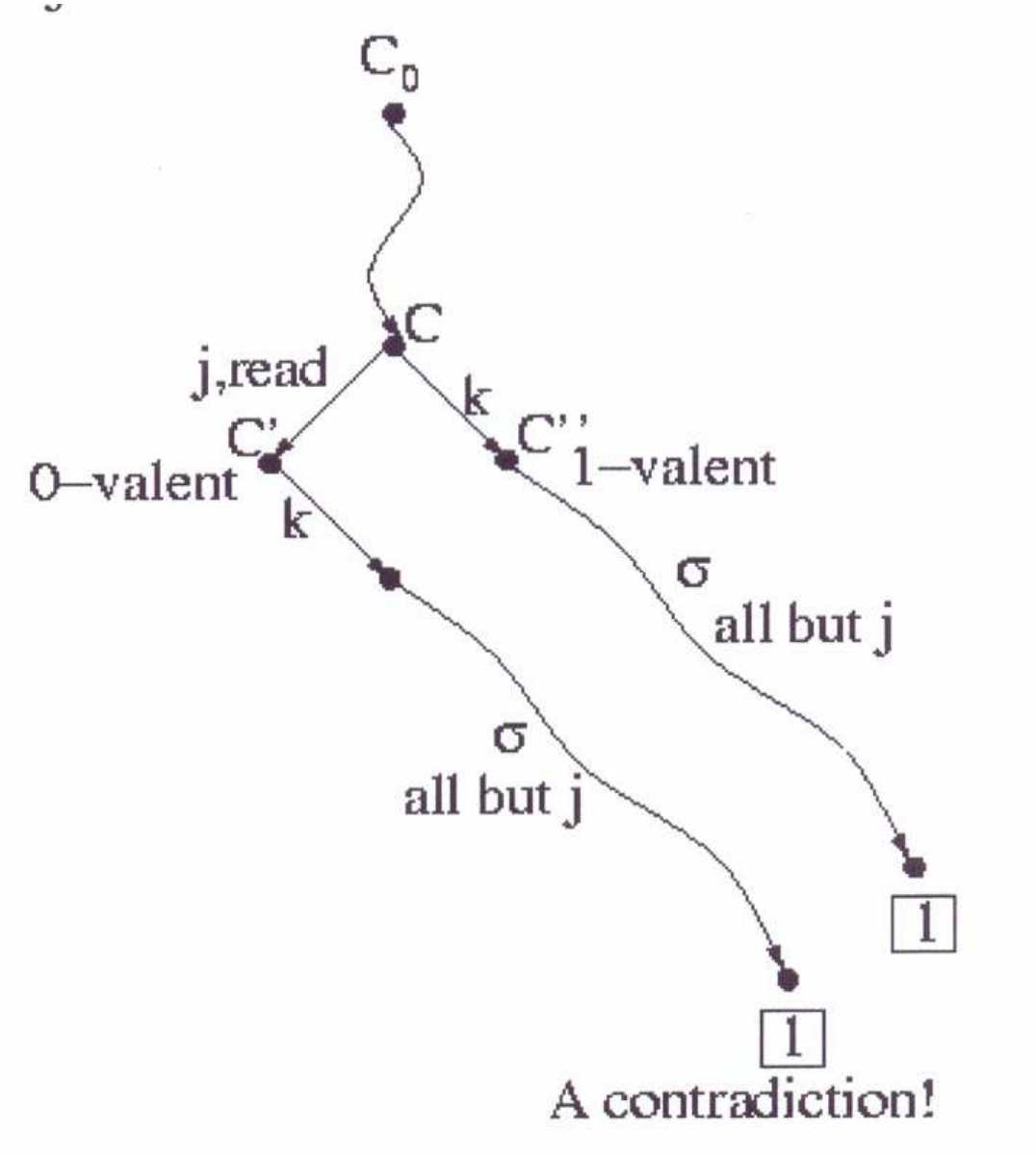


Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

Απόδειξη (συνέχεια)

Διακρίνω περιπτώσεις:

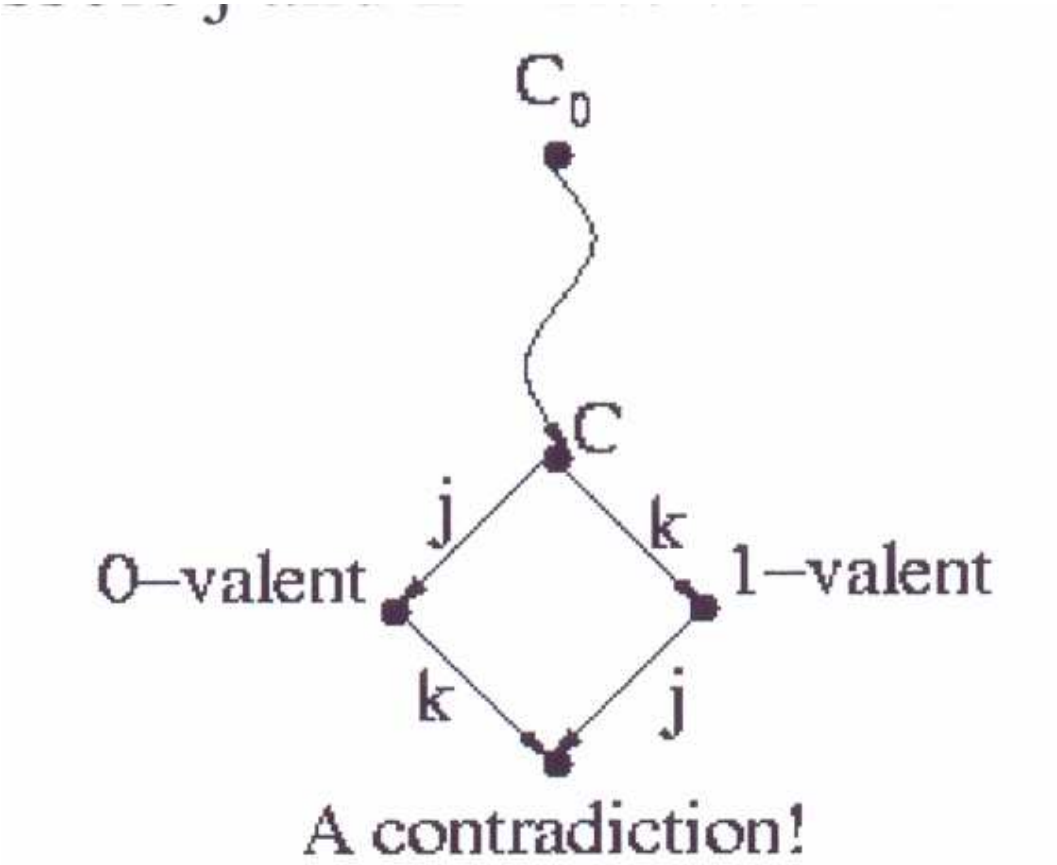
1. το πρώτο βήμα μιας από τις p_j, p_k είναι read (έστω π.χ., της p_j).



Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

Απόδειξη (συνέχεια)

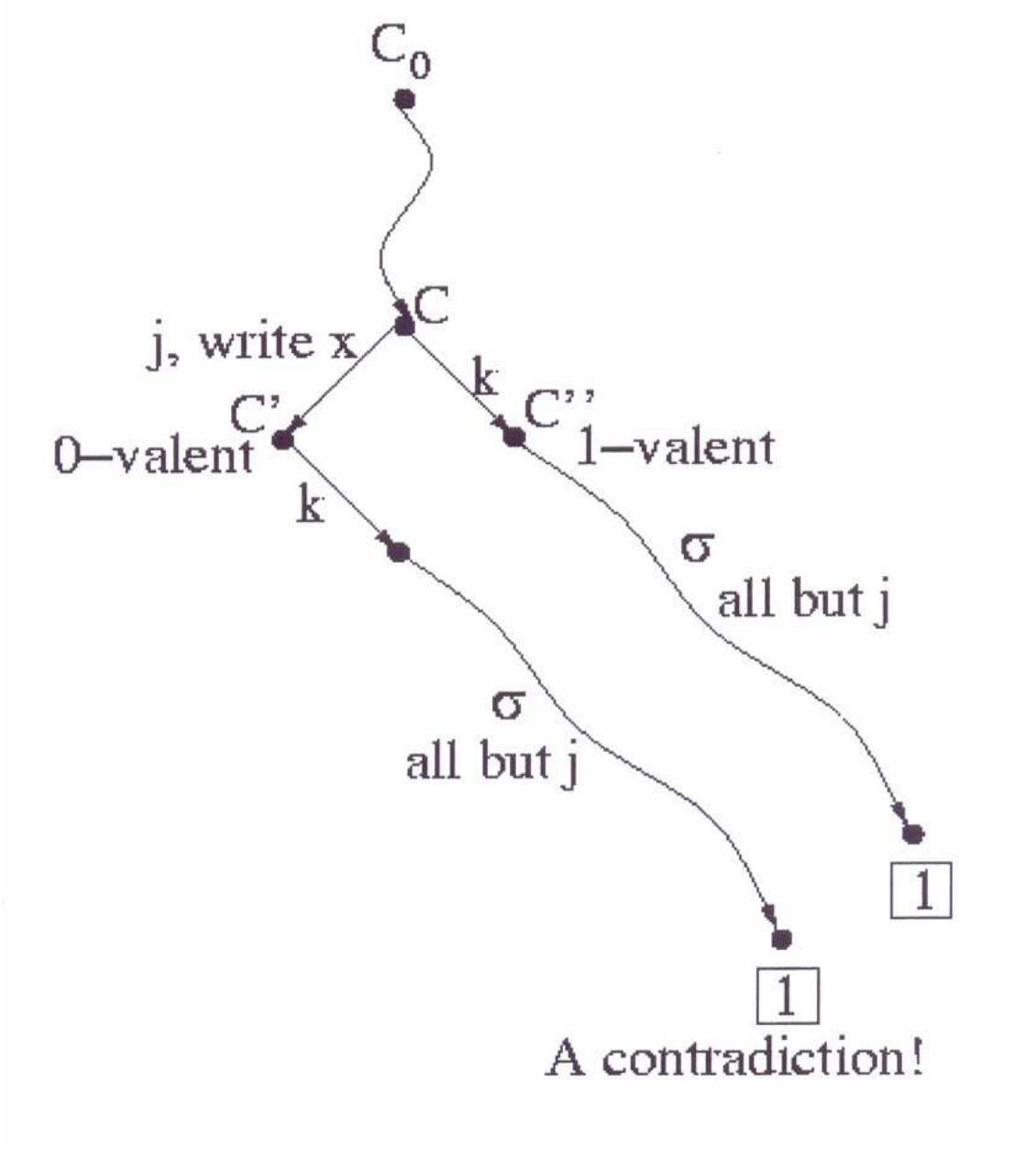
2. τα πρώτα βήματα των p_j, p_k είναι εγγραφές σε διαφορετικούς καταχωρητές



Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

Απόδειξη (συνέχεια)

3. τα πρώτα βήματα των p_j, p_k από τη C είναι εγγραφές στον ίδιο καταχωρητή.



Αδυναμία Σχεδιασμού Ασύγχρονου Αλγορίθμου Ομοφωνίας

Λήμμα 2 $\Rightarrow$ υπάρχει αρχική καθολική κατάσταση με διπλό σθένος.

Λήμμα 3 $\Rightarrow$ ξεκινώντας από μια κατάσταση με διπλό σθένος μπορώ να οδηγηθώ σε άλλη κατάσταση με διπλό σθένος αφήνοντας τη διεργασία που δεν είναι κρίσιμη να κάνει το επόμενο βήμα.

Αυτό επαναλαμβάνεται άπειρες φορές $\Rightarrow$ υπάρχει άπειρη ακολουθία στην οποία καμία διεργασία δεν τερματίζει (αφού μια καθολική κατάσταση στην οποία έστω μια διεργασία έχει τερματίσει δεν μπορεί να έχει διπλό σθένος, λόγω της ιδιότητας της ομοφωνίας).

Θεώρημα

Δεν υπάρχει wait-free αλγόριθμος που να επιλύει το πρόβλημα της ομοφωνίας σε ασύγχρονο σύστημα διαμοιραζόμενης μνήμης με n διεργασίες.