2o **PROJECT** ΑΠΟΚΕΝΤΡΩΜΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΣ & ΜΟΝΤΕΛΟΠΟΙΗΣΗ

ΛΕΚΚΑΣ ΓΕΩΡΓΙΟΣ ΑΜ:1067430 Έτος 5ο

Περιεχόμενα

Η συγγραφή της αναφοράς πραγματοποιήθηκε σε Latex με τη βοήθεια του TexStudio.

Α.Θεωρητική Άσκηση 1

Το πρώτο βήμα που πρέπει να γίνει για να αποδείχθεί αν ένας αλγόριθμος είναι αυτοσταθεροποιητικός ή όχι είναι να περιορίσουμε τον δαίμονα,δηλαδή το ¨περιβάλλον¨, αυτόν που καθορίζει τις εισόδους.Οπότε για τη συγκεκριμένη απόδειξη θα χρησιμοποιήσουμε έναν Locally Central Unfair Deamon.Locally Central διότι επιλέγουμε να μην ενεργοποιούνται δυο διαδοχικοί κόμβοι στο ίδιο βήμα και Unfair διότι δεν θα έχουμε κάποιο περιορισμό ως προς τη χρονική στιγμή ενεργοποίησης των κόμβων.

Έπειτα θα πρέπει να ορίσουμε την Variant Function του συστήματος μας. Γενικά η Variant Function απεικονίζει τις διαμορφώσεις του συστήματος και τη χρησιμοποιούμε για να δείξουμε πως ο αλγόριθμος μας συγκλίνει. Στη συγκεκριμένη άσκηση θα χρησιμοποιήσουμε ως Variant Function το εξής:

Variant Function: Το πλήθος των κόμβων που ενεργοποιούνται οι οποίοι δεν ανήκουν στο MIS και είναι γείτονες με κάποιον που είναι στο MIS.

Στη συνέχεια για λόγους ευκολίας και χρησιμότητας στην απόδειξη θα γράψουμε τον αλγόριθμο στη παρακάτω μορφή:

Algorithm:

Inputs: p.N: the set of p's neighbors, ID: every node's Id, We are assuming p belongs in the MIS.

Local Variable: I : contains the MIS, initially I$\neq$0.

Macros:
MIS(p) : p$\in$I
NOTMIS(p) : p$\notin$I
Leave(p) : $\exists$q$\in$p.N : q.ID<p.ID
Join(p) : $\exists$q$\in$p.N and $\exists$p.N$\in$I

Guard:
Problem(p) : $\exists$q$\notin$I with p.N$\in$I .

Action:
Problem(p) : (disable q and ignore q from getting in I) .

Terminal Configuration(Predicate):
MIS : I$\neq$0 , $\nexists$q$\in$I with p.N$\in$I .

Analysis:

Partial Correctness:

1. The predicate MIS holds in every terminal configuration

   Termination(let an execution e = d_0,d_1,...,d_i,...):

1. Let p a node.In every configuration , for any node p , p$\in$ I
2. Variant Function(def): Ability(d_i) = |p$\in$V: Problem(p) $\in d_i$|
3. If Problem(p) holds in d_i then there exists q$\in$p.N such that Problem(q) holds in d_i as well
4. Ability(d_i)=0 only at the terminal configuration
5. For every node q and every step $d_i \rightarrow d_{i+1}$, if $\neg$ Problem(q) holds in d_i, then $\neg$ Problem(q) holds in d_{i+1}

6. For every 2 configurations d_i and d_j such that $i \leq j$ it holds that Ability(d_i) $\geq$ Ability(d_j) because in every one neighboring node q that enables and has Problem(q), it gets disabled and ignored. So the count of anabled and active nodes becomes smaller

7. For every step $d_i \rightarrow d_{i+1}$ we have that Ability(d_{i+1}) $\leq$ Ability(d_i)

8. In the final step Ability(d_i)=0 and we conclude in a MIS

9. The execution terminates after n steps

10. For every execution e,e is finite(Termination)

Ο αλγόριθμος είναι silent διότι καθώς περνάμε από διαμόρφωση σε διαμόρφωση θα καταλήξουμε σε τερματική κατάσταση όπου το πρόβλημα έχει λυθεί.
Ο αλγόριθμος είναι self-stabilizing διότι καταλήγει σε ήδη γνωστό σχηματισμό, στη συγκεκριμένη περίπτωση σε ένα MIS.

Α.Θεωρητική Άσκηση 2

Για το πρωτόκολλο πληθυσμού που θα περιγραφτεί θα χρησιμοποιήσουμε έναν adversarial scheduler ο οποίος θα είναι strongly fair. Με αυτόν τον scheduler πρακτικά λαμβάνουμε υπόψη το χειρότερο σενάριο ενεργοποίησης των κόμβων(worst case) παρόλο που πάντοτε θα καταλήγει σε κάποια τερματική κατάσταση από την οποία μπορεί να εξαχθεί κάποιο συμπέρασμα ανάλογα με το πρωτόκολλο που έχει επιλεχθεί.
Δηλαδή το predicate θα είναι σταθερώς υπολογίσιμο αν το πρωτόκολλο πληθυσμού που θα περιγράψουμε, θα βρεθεί κάποια στιγμή σε μία τερματική διαμόρφωση. Στη δική μας περίπτωση άμα στο τέλος σύμφωνα με τους κανόνες που έχουμε θέσει μείνει μόνο ένας κόμβος με τιμή 1 τότε καταλήγουμε στο συμπέρασμα πως ξεκινήσαμε με άρτιο πλήθος από 1 στους αρχικούς κόμβους.