Ατομική Διπλωματική Εργασία

ACCED: ΜΙΑ ΑΛΓΕΒΡΑ ΔΙΕΡΓΑΣΙΩΝ ΓΙΑ ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΡΩΝ ΜΕ ΔΥΝΑΤΟΤΗΤΕΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΚΑΙ ΕΠΙΒΟΛΗΣ ΚΟΣΤΟΥΣ

Αλεξάνδρα Ιλεάνα Παπαηλιού

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ACCED: Μια άλγεβρα διεργασιών για συστήματα διαχείρισης πόρων με δυνατότητες επικοινωνίας και επιβολής κόστους

Αλεξάνδρα Ιλεάνα Παπαηλιού

Επιβλέπουσα Καθηγήτρια
Άννα Φιλίππου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2014

Ευχαριστίες

Ευχαριστώ ολόψυχα την επιβλέπουσα καθηγήτρια και σύμβουλό μου κα. Άννα Φιλίππου, πρωτίστως για την εμπιστοσύνη που μου έδειξε για την διεκπεραίωση αυτής της διπλωματικής εργασίας και που μου έδωσε την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον και χρήσιμο θέμα. Επίσης την ευχαριστώ για τη στήριξη, την καθοδήγηση, την υπομονή και τις συμβουλές που μου έδινε όχι μόνο σχετικά με την εκπόνηση της διπλωματικής μου εργασίας αλλά και κατά την τετραετή διάρκεια των σπουδών μου στο Πανεπιστήμιο Κύπρου.

Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου και ειδικότερα τη μητέρα μου, η οποία ήτανε πάντα δίπλα μου και με στήριζε σε όλες μου τις αποφάσεις, σε όλες τις φάσεις της ζωής μου.

Περίληψη

Το θέμα με το οποίο ασχολείται αυτή η διπλωματική εργασία είναι η ανάπτυξη ενός προτύπου άλγεβρας διεργασιών το οποίο μοντελοποιεί ένα σύστημα παροχής ενέργειας. Η άλγεβρα διεργασιών που υλοποιήθηκε για αυτό το σκοπό είναι η ACCED (Algebra for Communication and Costing Energy Demand), η οποία είναι μια χρονική άλγεβρα διεργασιών. Περιλαμβάνει: (α) εργασίες που ζητάνε κάποιες ποσότητες πόρων και έχουν προτεραιότητες, και (β) προμήθειες που χορηγούν προς τις εργασίες, κάποιους πόρους, έναντι κάποιου χρηματικού ποσού. Επίσης υπάρχουν δυνατότητες επικοινωνίας.

Στόχος μας ήταν να ελέγχουμε κατά πόσο μπορεί να χρονοπρογραμματιστεί ένα σύνολο από διεργασίες με βάση την έννοια του κόστους. Επιπρόσθετα, επειδή τα βασικά χαρακτηριστικά της άλγεβρας διεργασιών ACCED αποτελούν κάποια από τα κύρια χαρακτηριστικά του πεδίου του υπολογιστικού νέφους (Cloud Computing), έχουμε μοντελοποιήσει και κάποιους αλγόριθμους πολιτικών χρονοδρομολόγησης που έχουν εφαρμογή στο υπολογιστικό νέφος. Το πρότυπο αυτό αποτελεί μια προσπάθεια μοντελοποίησης του υπολογιστικού νέφους όπου εκμεταλλευόμενοι τις δυνατότητες επικοινωνίας που μπορούμε να έχουμε μοντελοποιήσαμε αλγόριθμους πολιτικών χρονοδρομολόγησης όπως είναι η πολιτική EDF, η μη-διακοπτόμενη και η διακοπτόμενη πολιτική, η πολιτική FIFO καθώς και άλλες πολιτικές που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους.

Πιο αναλυτικά, στην παρούσα διπλωματική εργασία υλοποιήθηκαν οι στόχοι που τέθηκαν, οδηγώντας έτσι στην νέα άλγεβρα διεργασιών ACCED. Συγκεκριμένα ορίστηκε η σύνταξη και η σημασιολογία της ACCED αξιοποιώντας στοιχεία από το βασικό πρότυπο της AESC σε συνδυασμό με στοιχεία από το βασικό πρότυπο της EDAUS ώστε να υποστηρίζεται η έννοια του κόστους αλλά και να παρέχονται δυνατότητες επικοινωνίας. Επίσης μοντελοποιήθηκαν διάφοροι αλγόριθμοι πολιτικών χρονοδρομολόγησης του υπολογιστικού νέφους βάσει της σύνταξης και της σημασιολογίας της ACCED. Ορίστηκε η έννοια του χρονοπρογραμματισμού καθώς και χρονοπρογραμματίσιμη ανάπτυξη μεθοδολογίας που στόχο είχε να βοηθήσει στην επίτευξη της απόδειξης του θεωρήματος της ταυτόχρονης εξυπηρέτησης δύο εργασιών από μια προμήθεια.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Συστήματα παροχής πόρων υπάρχουν παντού. Υπάρχουν υπολογιστικά συστήματα, συστήματα διαχείρισης ενέργειας, συστήματα του πεδίου του υπολογιστικού πλέγματος (Grid Computing) [1,11] και πιο πρόσφατα είναι τα συστήματα του πεδίου του υπολογιστικού νέφους (Cloud Computing) [1]. Όλα αυτά τα συστήματα παροχής πόρων έχουν κοινά χαρακτηριστικά και χρησιμοποιούνται για να επιλύουν παρόμοιας φύσης προβλήματα. Συγκεκριμένα διαχειρίζονται και παρέχουν πόρους στους χρήστες του συστήματος ανάλογα με τις απαιτήσεις των χρηστών και βάσει των διαθέσιμων ποσοτήτων πόρων του συστήματος. Στόχος μας ήτανε να δημιουργήσουμε ένα πρότυπο για την μοντελοποίηση και την ανάπτυξη τέτοιων συστημάτων, ιδιαιτέρα συστημάτων που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους [12]. Με τον όρο υπολογιστικό νέφος αναφερόμαστε σε ένα σύνολο από υπολογιστές, οι οποίοι συνδέονται μέσω ενός δικτύου επικοινωνίας πραγματικού χρόνου, όπως είναι το διαδίκτυο. Είναι μια συνώνυμη έννοια με την έννοια ενός δικτύου που περιέχει κατανεμημένα συστήματα υπολογισμού, τα οποία έχουν τη δυνατότητα να τρέχουν προγράμματα ή εφαρμογές σε πολλούς υπολογιστές που συνδέονται ταυτόχρονα.

Αυτή είναι και η κυριότερη υπηρεσία του υπολογιστικού νέφους. Δηλαδή οι χρήστες να έχουν την δυνατότητα να χρησιμοποιούν αυτή την υπηρεσία έναντι κάποιου χρηματικού ποσού, και επίσης μπορούν να καθορίσουν την ποσότητα που επιθυμούν καθώς και την επιθυμητή προθεσμία. Ένα από τα βασικά μας κίνητρα λοιπόν, ήτανε να αναπτύξουμε μια άλγεβρα διεργασιών η οποία να περιέχει τα βασικά χαρακτηριστικά του υπολογιστικού νέφους. Για την επίτευξη αυτού του στόχου αξιοποιήσαμε χαρακτηριστικά από τις άλγεβρες διεργασιών AESC [16] και EDAUS [17], και με επιπρόσθετα χαρακτηριστικά που μας επιτρέπουν να έχουμε την σχετική μοντελοποίηση συστημάτων του πεδίου του υπολογιστικού νέφους. Επιπρόσθετα, θέλαμε η άλγεβρα διεργασιών που θα αναπτύσσαμε να έχει τη δυνατότητα επικοινωνίας συντονισμού με το εξωτερικό περιβάλλον, ώστε να της προσδώσουμε πιο ρεαλιστικά χαρακτηριστικά. Αυτό το επιτυγχάνουμε με τη χρήση καναλιών. Επομένως θέλαμε η άλγεβρα διεργασιών μας να συνδυάζει τα βασικά χαρακτηριστικά του υπολογιστικού νέφους, ώστε να μοντελοποιεί αλγόριθμους που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους και ταυτόχρονα να μας παρέχει τη δυνατότητα επικοινωνίας συντονισμού.

1.2 Βασική Ιδέα της Διπλωματικής Εργασίας

Η κύρια ιδέα για την εκπόνηση αυτής της διπλωματικής εργασίας συνίσταται στην πιο πάνω επιθυμία, δηλαδή να δημιουργήσουμε ένα πρότυπο για την μοντελοποίηση και την ανάπτυξη τέτοιων συστημάτων, ιδιαιτέρα συστημάτων που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους, και ταυτόχρονα να μας παρέχει δυνατότητες επικοινωνίας συντονισμού με επιβολή κόστους.

Για τον λόγο αυτό αποφασίσαμε να δημιουργήσουμε ένα καινούργιο πρότυπο πάνω σε άλγεβρα διεργασιών, τη λεγόμενη ACCED (Algebra for Communication and Costing Energy Demand) η οποία βασίζεται στις άλγεβρες διεργασιών PADS, AESC και EDAUS. Η πρώτη άλγεβρα διαχειρίζεται τη ζήτηση και τη χορηγία των πόρων από τις διεργασίες που υφίστανται σε ένα σύστημα. Η δεύτερη επιπρόσθετα ορίζει και την ποσότητα του πόρου που μπορεί να προσφερθεί καθώς και τον ορισμό της ποσότητας που ζητά κάποια εργασία και η τρίτη επιπρόσθετα ορίζει την έννοια της αποθήκευσης και πως μπορούν οι αιτήσεις να αξιοποιούν τα ποσά ενέργειας τα οποία έχουν αποθηκευτεί.

Τα μειονεκτήματα των αλγεβρών AESC και EDAUS τα οποία θέλαμε να αντιμετωπίσουμε είναι πως δεν γινότανε καμία αναφορά στο κόστος και στην επικοινωνία συντονισμού, παράμετροι οι οποίοι είναι πολύ σημαντικοί. Σε αντίθεση με αυτές τις άλγεβρες η ACCED έρχεται να ενσωματώσει και την έννοια του κόστους ώστε οι αγοραστές να μπορούν να δουν αν όντως οι επιλογές που κάνουν είναι οι πιο συμφέρουσες για αυτούς και ίσως να ανακαλύψουν και νέες προοπτικές τις οποίες δεν είχαν αντιληφθεί, και αν προέβαιναν σε αυτές θα κάλυπταν πάλι τις ανάγκες τους αλλά σε πιο συμφέρουσες προσφορές. Επιπρόσθετα, έρχεται να ενσωματώσει και την έννοια της επικοινωνίας συντονισμού, προσδίδοντάς της έτσι πιο ρεαλιστικά χαρακτηριστικά.

Σημαντικά οφέλη που προκύπτουν από την επικοινωνία και την κοστολόγηση ενέργειας είναι τα ακόλουθα:

- Διασφάλιση των προσωπικών συμφερόντων του αγοραστή
- Μοντελοποίηση της έννοιας της κοστολόγησης και της δυνατότητας επιλογής προμηθευτή εκ μέρους του αγοραστή με βάσει το κόστος
- Εξασφάλιση επιθυμητού κέρδους για τον προμηθευτή λόγω του κόστους
- Προσθήκη πιο ρεαλιστικών χαρακτηριστικών
- Δυνατότητες επικοινωνίας και κατ’ επέκταση δυνατότητα μοντελοποίησης αλγορίθμων πολιτικών χρονοδρομολόγησης που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους

1.3 Χρησιμότητα της ACCED

Η άλγεβρα διεργασιών ACCED, έχει ως στόχο τη μοντελοποίηση συστημάτων παραγωγής και διαχείρισης ενέργειας βασιζόμενη στο κόστος αλλά και στην επικοινωνία συντονισμού. Όπως έχουμε ήδη αναφέρει, η άλγεβρα αυτή περιέχει τα βασικά χαρακτηριστικά της έννοιας του υπολογιστικού νέφους. Θα μπορούσε λοιπόν να εφαρμοστεί σε συστήματα που διαχειρίζονται πόρους και συνδέονται μέσω ενός δικτύου επικοινωνίας με σκοπό να εξυπηρετούν τις ανάγκες των χρηστών, διότι η ACCED μπορεί να αξιοποιηθεί ως εργαλείο για την μελέτη καινούργιων τεχνικών και αλγορίθμων και έτσι θα μπορούσε να προσφέρει βελτιωμένη απόδοση σε αυτούς τους αλγόριθμους και κατ’ επέκταση καλύτερη δυνατή εξυπηρέτηση των αναγκών των χρηστών. Για αυτό λοιπόν η άλγεβρα διεργασιών ACCED έχει πολλές δυνατότητες αφού μπορεί να αξιοποιηθεί ως εργαλείο μελέτης αλγορίθμων και

καινούργιων τεχνικών, που θα προσφέρουν βελτιωμένη απόδοση, και μπορεί να μοντελοποιήσει και αλγόριθμους οι οποίοι είναι βασισμένοι στο υπολογιστικό νέφος, βάσει της σύνταξης και της σημασιολογίας της. Επίσης η άλγεβρα διεργασιών ACCED παρέχει τη δυνατότητα επικοινωνίας συντονισμού με την χρήση καναλιών, γεγονός που της προσδίδει πιο ρεαλιστικά χαρακτηριστικά. Περισσότερες λεπτομέρειες θα δοθούν σε μετέπειτα κεφάλαια, τόσο για τη μοντελοποίηση διαφόρων αλγορίθμων πολιτικών χρονοδρομολόγησης βάσει του υπολογιστικού νέφους, όσο και για την επικοινωνία συντονισμού με τη χρήση καναλιών.

1.4 Δομή Διπλωματικής Εργασίας

Η δομή της εργασίας έχει ως εξής: Ακολουθεί στο Κεφάλαιο 2 η περιγραφή των αλγεβρών πάνω στις οποίες βασίζεται η ACCED. Ακολούθως στο Κεφάλαιο 3 δίνεται η σύνταξη και η σημασιολογία της ACCED, καθώς και η μοντελοποίηση κάποιων αλγορίθμων πολιτικών χρονοδρομολόγησης του υπολογιστικού νέφους και στο Κεφάλαιο 4 δίνεται ο ορισμός της έννοιας του χρονοπρογραμματισμού, όπου ενσωματώνουμε τη δυνατότητα επικοινωνίας συντονισμού με τη χρήση καναλιών και τέλος στο Κεφάλαιο 5 δίνεται η παρουσίαση των συμπερασμάτων και ενδεχόμενων μελλοντικών προεκτάσεων της διπλωματικής εργασίας.

Κεφάλαιο 2

Υπόβαθρο και Προηγούμενες Εργασίες

2.1 Άλγεβρες Διεργασιών

Όταν αναφερόμαστε σε άλγεβρες διεργασιών εννοούμε τη διατύπωση της συμπεριφοράς ενός συστήματος εντός ενός μαθηματικού πλαισίου με τη μορφή ενός συνόλου διεργασιών οι οποίες εκτελούνται παράλληλα και ταυτοχρόνως διαχειρίζονται τα δεδομένα του μοντελοποιημένου συστήματος.

Άλγεβρα Διεργασιών CCS

Μια από τις πρώτες άλγεβρες διεργασιών δημιουργήθηκε από τον Robin Milner(1973-1980). Ονομάζεται CCS (Calculus for Communication Systems) [10]. Περιγράφει την επικοινωνία που υπάρχει μεταξύ δύο διεργασιών οι οποίες τρέχουν παράλληλα.

Συγκεκριμένα η CCS έχει τους ακόλουθους τελεστές [18]:

Ακολουθιακός Τελεστής

Συμβολισμός: P=a.Q

Σημασιολογία: Η διεργασία P μπορεί να εκτελέσει την ενέργεια a και να εξελιχθεί στη νέα διεργασία Q.

Συμβολισμός διεργασίας τερματισμού: 0|FIN|NIL

Τελεστής Επιλογής

Συμβολισμός: $P=a.Q_1+b.Q_2$

Σημασιολογία: Η διεργασία P μπορεί να εκτελέσει την ενέργεια a και να εξελιχθεί στη νέα διεργασία Q_1 ή η διεργασία P μπορεί να εκτελέσει την ενέργεια b και να εξελιχθεί στην νέα διεργασία Q_2. Δηλαδή υπάρχει η δυνατότητα επιλογής μεταξύ των διαθέσιμων ενεργειών.

Τελεστής Παράλληλης Σύνθεσης

Συμβολισμός: P||Q όπου P=a.P' και Q=b.Q'

Σημασιολογία: Ο τελεστής είναι δυαδικός και εκφράζει τη δυνατότητα που έχουν δύο διεργασίες P και Q να εκτελέσουν παράλληλα τις ενέργειες τους και να δημιουργείται συντονισμός όταν η παράλληλη εκτέλεση των ενεργειών τους οδηγεί σε συμπληρωματικές ενέργειες.

2.2 Άλγεβρα Διεργασιών PADS

Η άλγεβρα διεργασιών PADS (Process Algebra for Demand and Supply) [13] είναι μια χρονική άλγεβρα διεργασιών όπου μοντελοποιείται η ζήτηση και η προμήθεια πόρων σε συστήματα πραγματικού χρόνου, π.χ. υπολογιστικά. Η άλγεβρα αυτή έχει επίσης επεκταθεί με τη χρήση προτεραιοτήτων.

2.2.1 Δομικά Στοιχεία και Έννοιες

Υπάρχουν δύο κατηγορίες στις οποίες διακρίνονται οι διεργασίες της PADS:

1. **Εργασίες**

Οι εργασίες είναι οι μονάδες οι οποίες αιτούνται να πάρουν κάποια ποσότητα πόρων από το σύστημα. Συμβολίζονται με T, π.χ. T_1, T_2, ….

2. Προμήθειες

Οι προμήθειες είναι οι μονάδες οι οποίες προσφέρουν κάποια ποσότητα πόρων στο σύστημα. Συμβολίζονται με S, π.χ. S_1, S_2, ….

Ένας πόρος r ∈ R, όπου R είναι ένα σύνολο από επαναχρησιμοποιήσιμους πόρους, μπορεί είτε να ζητηθεί (r), είτε να χορηγηθεί ($\overleftarrow{r}$), είτε να καταναλωθεί ($\vec{r}$) εάν ζητείται και χορηγείται την ίδια χρονική στιγμή.

Στην άλγεβρα διεργασιών PADS αξίζει να αναφέρουμε ότι ο χρόνος είναι διακριτός και για την ολοκλήρωση τους οι ενέργειες απαιτούν μία μονάδα χρόνου. Η αδράνεια στο σύστημα μίας μονάδας χρόνου συμβολίζεται με ∅.

2.2.2 Σύνταξη

Η άλγεβρα διεργασιών PADS περιλαμβάνει το σύνολο των εργασιών T, το σύνολο των προμηθειών S και το σύνολο των συστημάτων P:

T ::= FIN | $\sum_{i \in I} \rho_i : T_i$ | C
S ::= FIN | $\sum_{i \in I} \gamma_i : S_i$ | D
P ::= FIN | T | S | P‖P

Πίνακας 2.1 Σύνταξη Άλγεβρας Διεργασιών PADS

Με τον συμβολισμό ρ παρουσιάζονται τα στοιχεία του Act_r. Ορίζουμε ως Act_r το σύνολο που περιέχει μόνο αιτήσεις πόρων και με τον συμβολισμό γ παρουσιάζονται τα στοιχεία του Act_g. Ορίζουμε ως Act_g το σύνολο που περιέχει μόνο χορηγίες πόρων.

- Μια εργασία T μπορεί είτε να είναι αδρανής (FIN), είτε να αποτελείται από ένα σύνολο εργασιών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά εργασίας C.
- Μια προμήθεια S μπορεί είτε να είναι αδρανής (FIN), είτε να αποτελείται από ένα σύνολο προμηθειών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά προμήθειας D.

- Ένα σύστημα P μπορεί να είναι αδρανές (FIN), να αποτελείται είτε από μια εργασία T, είτε από μια προμήθεια S, είτε να αποτελεί την παράλληλη εκτέλεση δύο συστημάτων.

2.2.3 Σημασιολογία

Στην άλγεβρα διεργασιών PADS η σημασιολογία ορίζεται μέσω της σχέσης μεταβάσεων $\twoheadrightarrow$. Ακολούθως ορίζεται η σχέση προτιμήσεων με αποτέλεσμα να μετατρέπεται η σχέση μεταβάσεων σε σχέση μεταβάσεων με προτίμηση $\rightarrow$, με σκοπό την επίλυση του μη ντετερμινισμού που υπάρχει, στο μέγιστο δυνατό βαθμό που μπορεί.

Στον ακόλουθο πίνακα φαίνονται οι πέντε κανόνες μετάβασης της άλγεβρας διεργασιών PADS.

Idle: Μια διεργασία FIN επιτρέπει στον χρόνο να κυλά χωρίς την εκτέλεση κάποιας δράσης

SumT: Εκτέλεση των δράσεων και μετάβαση στην επόμενη κατάσταση που προβλέπεται

SumS: Εκτέλεση των δράσεων και μετάβαση στην επόμενη κατάσταση που προβλέπεται

Const: Μια σταθερά συμπεριφέρεται όπως την διεργασία στην οποία είναι ορισμένη

Par: Παράλληλη εκτέλεση των δράσεων δύο συστημάτων

(Idle)	$\mathrm{FIN} \overset{\emptyset}{\twoheadrightarrow} \mathrm{FIN}$	
(SumT)	$\Sigma_{i\in I}\rho_i : T_i \overset{\rho_i}{\twoheadrightarrow} T_i$	$I \neq \emptyset$
(SumS)	$\Sigma_{i\in I}\gamma_i : S_i \overset{\gamma_i}{\twoheadrightarrow} S_i$	$I \neq \emptyset$
(Const)	$\dfrac{P \overset{\alpha}{\twoheadrightarrow} P'}{C \overset{\alpha}{\twoheadrightarrow} P'}$	$C \overset{\mathrm{def}}{=} P$
(Par)	$\dfrac{P_1 \overset{\alpha_1}{\twoheadrightarrow} P_1' \quad P_2 \overset{\alpha_2}{\twoheadrightarrow} P_2'}{P_1 \| P_2 \overset{\alpha_1 \oplus \alpha_2}{\twoheadrightarrow} P_1' \| P_2'}$	$\mathsf{compatible}(\alpha_1, \alpha_2)$

Πίνακας 2.2 Κανόνες μετάβασης PADS

Για την εκτέλεση του κανόνα Par απαραίτητα πρέπει να ισχύει η συνθήκη compatible που ακολουθεί

$$\mathsf{compatible}(\alpha_1, \alpha_2) \quad = \bigwedge_{r \in res(\alpha_1) \cap res(\alpha_2)} (r \in \alpha_1^{\flat} \wedge \overline{r} \in \alpha_2^{\flat}) \vee (r \in \alpha_2^{\flat} \wedge \overline{r} \in \alpha_1^{\flat})$$

Αυτή η συνθήκη λέει ότι τα α_1 και α_2 είναι συμβατά αν η ενέργεια α_1 περιέχει τη ζήτηση ενός πόρου r και η ενέργεια α_2 περιέχει τη χορηγία αυτού του πόρου r ή αν η ενέργεια α_2 περιέχει τη ζήτηση ενός πόρου r και η ενέργεια α_1 περιέχει τη χορηγία αυτού του πόρου r.

Το $\alpha_1 \oplus \alpha_2$ είναι η ενέργεια η οποία μπορεί να προκύψει κατά την εκτέλεση του κανόνα μετάβασης Par και ορίζεται ως ακολούθως :

$$\begin{aligned}\alpha_1 \oplus \alpha_2 \quad = \quad & \{(r,p) \in \alpha_1 \cup \alpha_2 | \overline{r} \notin \alpha_1 \cup \alpha_2\} \cup \{\overline{r} \in \alpha_1 \cup \alpha_2 | (r,p) \notin \alpha_1 \cup \alpha_2\} \\ & \cup \{(\overleftrightarrow{r},p) | (r,p) \in \alpha_i, \overline{r} \in \alpha_{3-i}, i \in \{1,2\} \text{ or } (\overleftrightarrow{r},p) \in \alpha_1 \cup \alpha_2\}\end{aligned}$$

2.2.4 Αποτελέσματα

Για την άλγεβρα διεργασιών PADS έχει ορισθεί, μια θεωρία για τον χρονοπρογραμματισμό της. Συγκεκριμένα έχει δοθεί ο ορισμός για το πότε ένα σύνολο από εργασίες θεωρείται χρονοπρογραμματίσιμο από μια προμήθεια. Επίσης έχει δοθεί ένας εναλλακτικός ορισμός για την έννοια του χρονοπρογραμματισμού και αποδεικνύεται ότι αυτοί οι δύο ορισμοί είναι ισοδύναμοι. Τέλος παρουσιάζεται και αποδεικνύεται το θεώρημα της ταυτόχρονης εξυπηρέτησης δύο εργασιών, ένα θεώρημα που λέει ότι οι εργασίες T_1 και T_2 είναι χρονοπρογραμματίσιμες αν και μόνο αν οι αιτήσεις της εργασίας T_1 σε συνδυασμό με τις αιτήσεις της εργασίας T_2 δεν οδηγούν σε αδιέξοδο του συστήματος.

2.3 Άλγεβρα Διεργασιών AESC

Η άλγεβρα διεργασιών AESC (Algebra for Energy Supply and Consumption) [16] αποτελεί μια χρονική άλγεβρα διεργασιών που μοντελοποιεί συστήματα διανομής και παραγωγής ενέργειας. Επεκτείνει την PADS με τη δυνατότητα καθορισμού της ποσότητας του πόρου που προσφέρεται και του ορισμού της ποσότητας που ζητά κάποια εργασία.

Όπως και στην PADS, υπάρχουν δύο κατηγορίες διεργασιών στην AESC:

1) **Εργασίες**

   Οι εργασίες είναι οι μονάδες οι οποίες αιτούνται να πάρουν κάποια ποσότητα πόρων από το σύστημα. Οι αιτήσεις συμβολίζονται με (r,e,pr) όπου r είναι ο πόρος, e είναι η ποσότητα που ζητείται και pr είναι η προτεραιότητα

2) Προμήθειες

Οι προμήθειες είναι οι μονάδες οι οποίες προσφέρουν κάποια ποσότητα πόρων στο σύστημα. Οι χορηγίες ενέργειας συμβολίζονται με $(\bar{r}, c)$ όπου r είναι ο πόρος και c είναι η ποσότητα που χορηγείται.

Η σύνταξη της AESC είναι όμοια με αυτή της PADS με τη διαφορά ότι οι αιτήσεις και οι χορηγίες των πόρων, βάσει των οποίων δομούνται οι διεργασίες, περιέχουν και την έννοια της ποσότητας. Συγκεκριμένα κάθε πόρος σχετίζεται με την ζητούμενη ποσότητα, έτσι αντί ενός συνόλου από πόρους έχουμε σύνολα ζευγών (πόρος, ποσότητα).

Η άλγεβρα διεργασιών AESC περιλαμβάνει το σύνολο των εργασιών T, το σύνολο των προμηθειών S και το σύνολο των συστημάτων P:

$$T ::= NIL \mid \sum_{i \in I} \alpha_i : T_i \mid C$$

$$S ::= NIL \mid \sum_{i \in I} \beta_i : S_i \mid D$$

$$P ::= T \mid S \mid P||P$$

Πίνακας 2.3 Σύνταξη Άλγεβρας Διεργασιών AESC

Με τον συμβολισμό α παρουσιάζονται τα στοιχεία του Act_r. Ορίζουμε ως Act_r το σύνολο που περιέχει μόνο αιτήσεις πόρων και με τον συμβολισμό β παρουσιάζονται τα στοιχεία του Act_g. Ορίζουμε ως Act_g το σύνολο που περιέχει μόνο χορηγίες πόρων.

- Μια εργασία T μπορεί είτε να είναι αδρανής (NIL), είτε να αποτελείται από ένα σύνολο εργασιών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά εργασίας C.
- Μια προμήθεια S μπορεί είτε να είναι αδρανής (NIL), είτε να αποτελείται από ένα σύνολο προμηθειών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά προμήθειας D.
- Ένα σύστημα P μπορεί να αποτελείται από μια εργασία T, είτε να αποτελείται από μια προμήθεια S, είτε να αποτελείται από μια παράλληλη εκτέλεση δύο συστημάτων.

Περισσότερες λεπτομέρειες σχετικά με την σημασιολογία της AESC, καθώς και για παραδείγματα, δίνονται στην αναφορά [16] της βιβλιογραφίας.

Χρονοπρογραμματισμός Για τον χρονοπρογραμματισμό της AESC έχει ορισθεί, μια θεωρία για την έννοια του χρονοπρογραμματισμού ενός συστήματος. Επίσης έχουν δοθεί δύο ισοδύναμοι ορισμοί για το πότε ένα σύνολο από εργασίες θεωρείται χρονοπρογραμματίσιμο από μια προμήθεια. Ακόμη έχει ορισθεί η έννοια του υπολοίπου μιας προμήθειας η οποία εκφράζει την υπολειπόμενη ποσότητα πόρων που είναι διαθέσιμη για τις εργασίες ανά πάσα χρονική στιγμή. Τέλος παρουσιάζεται και αποδεικνύεται το θεώρημα της ταυτόχρονης εξυπηρέτησης δύο εργασιών από μια προμήθεια, ένα θεώρημα που λέει ότι εάν μια προμήθεια μπορεί να χρονοπρογραμματίσει μια εργασία και το υπόλοιπο της προμήθειας αυτής μπορεί να χρονοπρογραμματίσει μια άλλη εργασία, τότε η προμήθεια μπορεί να χρονοπρογραμματίσει και τις δύο εργασίες μαζί.

2.4 Άλγεβρα Διεργασιών EDAUS

Η άλγεβρα διεργασιών EDAUS (Energy Demand Analysis Using Storage) [17] αποτελεί μια χρονική άλγεβρα διεργασιών η οποία επεκτείνει την PADS με τη δυνατότητα αποθήκευσης ενέργειας σε περίπτωση που δεν καταναλωθεί άμεσα. Όπως και στην AESC, υπάρχει η δυνατότητα καθορισμού της ποσότητας του πόρου που προσφέρεται και του ορισμού της ποσότητας που ζητά κάποια εργασία.

Όπως και στην PADS, υπάρχουν οι διεργασίες των εργασιών και των προμηθειών. Επιπρόσθετα ορίζεται η διεργασία της αποθήκης. Επομένως υπάρχουν τρεις κατηγορίες διεργασιών στην EDAUS:

1) **Εργασίες**

   Οι εργασίες είναι οι μονάδες οι οποίες αιτούνται να πάρουν κάποια ποσότητα πόρων από το σύστημα. Ένα αίτημα συμβολίζεται με (r,e) οπου r είναι ο πόρος, και e είναι η ποσότητα που ζητείται.

2) **Προμήθειες**

   Οι προμήθειες είναι οι μονάδες οι οποίες προσφέρουν κάποια ποσότητα πόρων στο σύστημα. Μια χορηγία συμβολίζεται με $(\bar{r}, c)$ όπου r είναι ο πόρος και c είναι η ποσότητα που χορηγείται.

3) Αποθήκες

Οι αποθήκες είναι οι μονάδες οι οποίες αποθηκεύουν κάποια ποσότητα πόρων για μελλοντική χρήση στο σύστημα. Θεωρούμε ότι υπάρχει μόνο μια αποθήκη για κάθε πόρο στο σύστημα. Συμβολίζονται με E(r,v,max) όπου r είναι ο πόρος, v είναι η ποσότητα από τον πόρο r που περιέχει μέχρι στιγμής η αποθήκη και max η μέγιστη ποσότητα που μπορεί να αποθηκεύσει η αποθήκη.

Η σύνταξη της EDAUS είναι όμοια με αυτή της AESC με τη διαφορά ότι υπάρχει επιπρόσθετα και το σύνολο των αποθηκών E.

Η άλγεβρα διεργασιών EDAUS περιλαμβάνει το σύνολο των εργασιών T, το σύνολο των προμηθειών S, το σύνολο αποθηκών E και το σύνολο των συστημάτων P:

T ::= NIL | $\sum_{i\in I} \rho_i : T_i$ | A
S ::= NIL | $\sum_{i\in I} \gamma_i : S_i$ | B
E ::= E(r,u,max)
P ::= T | S | E | (P||P)

Πίνακας 2.4 Σύνταξη Άλγεβρας Διεργασιών EDAUS

Με τον συμβολισμό ρ παρουσιάζονται τα στοιχεία του Act_r. Ορίζουμε ως Act_r το σύνολο που περιέχει μόνο αιτήσεις πόρων και με τον συμβολισμό γ παρουσιάζονται τα στοιχεία του Act_g. Ορίζουμε ως Act_g το σύνολο που περιέχει μόνο χορηγίες πόρων.

- Μια εργασία T μπορεί είτε να είναι αδρανής (NIL), είτε να αποτελείται από ένα σύνολο εργασιών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά εργασίας A.
- Μια προμήθεια S μπορεί είτε να είναι αδρανής (NIL), είτε να αποτελείται από ένα σύνολο προμηθειών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά προμήθειας B.
- Μια αποθήκη E μπορεί να αποτελείται από μια αποθήκη που περιέχει κάποιο πόρο r, διαθέτει u μονάδες ενέργειας από τον πόρο r και έχει μέγιστη χωρητικότητα max μονάδες ενέργειας.

- Ένα σύστημα P μπορεί να αποτελείται είτε από μια εργασία T, είτε από μια προμήθεια S, είτε από μια αποθήκη E, είτε να αποτελεί την παράλληλη εκτέλεση δύο συστημάτων.

Περισσότερες λεπτομέρειες σχετικά με την σημασιολογία της EDAUS, καθώς και για παραδείγματα, δίνονται στην αναφορά [17] της βιβλιογραφίας.

Χρονοπρογραμματισμός Για τον χρονοπρογραμματισμό της EDAUS έχει δοθεί ο ορισμός του χρονοπρογραμματισμού ενός συστήματος. Δηλαδή αυτός ο ορισμός μας λέει πότε ένα σύνολο από αιτήματα ενέργειας είναι χρονοπρογραμματίσιμο από ένα σύνολο με προμήθειες και αποθήκες. Επίσης έχει ορισθεί η έννοια της υπολειπόμενης ενέργειας και τέλος έχει γίνει η απόδειξη του θεωρήματος της ταυτόχρονης εξυπηρέτησης δύο εργασιών. Το θεώρημα αυτό λέει ότι εάν ένα σύνολο από προμήθειες και αποθήκες μπορεί να χρονοπρογραμματίσει μια εργασία και η υπολειπόμενη ενέργεια μπορεί να χρονοπρογραμματίσει μια άλλη εργασία, τότε το σύνολο αυτό μπορεί να χρονοπρογραμματίσει και τις δύο εργασίες μαζί.

2.5 Τυπικές μέθοδοι και υπολογιστικό νέφος

Παρατηρούμε ότι οι υπηρεσίες οι οποίες στηρίζονται στο υπολογιστικό νέφος είναι ιδιαίτερα δημοφιλής [3]. Αξιοποιούνται τόσο από ιδιώτες, δηλαδή απλούς χρήστες όσο και από μεγάλους οργανισμούς επιχειρήσεων. Γι' αυτό το λόγο έχει δημιουργηθεί η ανάγκη χρήσης τυπικών μεθόδων, ώστε να μπορούμε με την δημιουργία και την χρήση των κατάλληλων εργαλείων να επαληθεύουμε την αξιοπιστία της παροχής αυτού του είδους υπηρεσιών. Θέλουμε δηλαδή τέτοιου είδους συστήματα να μπορούν να παρέχουν εγγυήσεις στους χρήστες και ότι δεν συμπεριφέρονται με απρόβλεπτο τρόπο. Επιπρόσθετα αυτού του είδους τα εργαλεία θα πρέπει να έχουν λειτουργική ορθότητα, διαθεσιμότητα, αξιοπιστία, απόδοση και εγγυήσεις ασφάλειας. Ήδη έχουν γίνει προσπάθειες για δημιουργία τέτοιου είδους εργαλείων, όμως ακόμη δεν έχουν τελειοποιηθεί. Για να τελειοποιηθούν θα μπορούσαν να γίνουν μελέτες καινούργιων τεχνικών και αλγορίθμων.

Η άλγεβρα διεργασιών ACCED έχει την δυνατότητα να αξιοποιηθεί ως εργαλείο για την μελέτη καινούργιων τεχνικών και αλγορίθμων, οι οποίοι θα προσφέρουν βελτιωμένη απόδοση. Επομένως η ACCED μπορεί και αυτή με την σειρά της να συμβάλλει στην επίτευξη της δημιουργίας κατάλληλων εργαλείων για την επαλήθευση και την αξιοπιστία της παροχής των υπηρεσιών του υπολογιστικού νέφους.

2.6 Συμπεράσματα

Συμπεραίνουμε λοιπόν ότι η άλγεβρα διεργασιών PADS μοντελοποιεί τη ζήτηση και την προμήθεια πόρων που υπάρχει σε ένα υπολογιστικό σύστημα, ενώ η AESC επεκτείνει την PADS και μπορεί επιπρόσθετα να καθορίσει και την ποσότητα ενός πόρου που ζητείται ή που χορηγείται. Η EDAUS επεκτείνει την PADS και επιπρόσθετα ορίζει την έννοια της αποθήκης που μας επιτρέπει να αποθηκεύουμε ενέργεια με σκοπό να χρησιμοποιηθεί στο μέλλον. Η άλγεβρα διεργασιών ACCED βασίζεται στις άλγεβρες διεργασιών PADS, AESC και EDAUS. Αυτό που έρχεται να προσθέσει η ACCED είναι η έννοια του κόστους και τις δυνατότητες επικοινωνίας με τη χρήση καναλιών, οι οποίες μας επιτρέπουν να μοντελοποιούμε αλγόριθμους που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους.

Κεφάλαιο 3

Άλγεβρα Διεργασιών ACCED

3.1 Περιγραφή

Σε αυτό το κεφάλαιο δίνεται η περιγραφή της άλγεβρας διεργασιών ACCED (Algebra for Communication and Costing Energy Demand) η οποία είναι μια γλώσσα που σκοπό έχει τη μοντελοποίηση των συστημάτων παροχής ενέργειας λαμβάνοντας υπόψη το κόστος και την επικοινωνία. Τέτοιου είδους συστήματα περιέχουν αγοραστές οι οποίοι ζητούν κάποιες ποσότητες ενέργειας και καλούνται να ικανοποιήσουν τις ανάγκες τους. Επίσης υπάρχουν οι

προμηθευτές οι οποίοι προμηθεύουν κάποιες ποσότητες ενέργειας έναντι κάποιου χρηματικού ποσού και αυτό το χρηματικό ποσό είναι κυμαινόμενο. Η άλγεβρα διεργασιών ACCED βασίζεται στις άλγεβρες διεργασιών PADS, AESC και EDAUS και επιπρόσθετα ορίζει την έννοια του κόστους και τις δυνατότητες επικοινωνίας με τη χρήση καναλιών, οι οποίες μας επιτρέπουν να μοντελοποιούμε αλγόριθμους που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους.

Στόχος μας είναι να δούμε αν είναι εφικτό να χρονοπρογραμματιστούν και να ικανοποιηθούν τα διάφορα αιτήματα των αγοραστών βάσει των ποσοτήτων που προσφέρουν οι προμηθευτές σε ένα σύστημα το οποίο έχει μοντελοποιηθεί βάσει της ACCED. Επίσης γίνεται μια αξιολόγηση της άλγεβρας διεργασιών ACCED ως προς την καταλληλότητα της για προβλήματα που προκύπτουν στο υπολογιστικό νέφος [1,4,11,15].
Την έννοια του χρονοπρογραμματισμού θα την μελετήσουμε στο επόμενο κεφάλαιο [7,8,9].

3.2 Δομικά Στοιχεία και Έννοιες της ACCED

Όπως και στην PADS, υπάρχουν τρεις δυνατοί τρόποι αξιοποίησης των πόρων μέσα σε αυτό το σύστημα. Αυτοί είναι η ζήτηση, η χορηγία και η κατανάλωση. Συμβολίζουμε με $r_i \in R$ τους πόρους οι οποίοι μπορούν να χρησιμοποιηθούν ως ακολούθως:
r_i : ζήτηση ενέργειας του πόρου r_i
$\bar{r}_i$: χορηγία ενέργειας του πόρου r_i
$\vec{r}_i$: κατανάλωση ενέργειας του πόρου r_i

Επίσης στην ACCED υπάρχουν και τρία είδη ενεργειών που μπορούμε να εκτελέσουμε και μια μονάδα ενέργειας αναφέρεται σε μια μονάδα του πόρου (slot).

Αίτηση
Παρουσιάζεται στο σύστημά μας όταν ένας αγοραστής ζητάει κάποιο ποσό ενέργειας από ένα συγκεκριμένο πόρο. Μια αίτηση ενέργειας είναι της μορφής (r,e,p) όπου r είναι το όνομα του πόρου, e είναι η ποσότητα του πόρου που ζητείται (σε αριθμό μονάδων του πόρου) και p είναι η προτεραιότητα του αιτήματος.

Χορηγία
Παρουσιάζεται στο σύστημά μας όταν ένας προμηθευτής προσφέρει κάποιο ποσό ενέργειας από ένα συγκεκριμένο πόρο. Δηλαδή εκτελεί ένα συγκεκριμένο αριθμό μονάδων του πόρου

ανά μονάδα χρόνου. Μια χορηγία ενέργειας είναι της μορφής ($\vec{r}$,s,c) όπου r είναι το όνομα του πόρου, s είναι ο αριθμός των μονάδων του πόρου που εκτελούνται/προσφέρονται ανά μονάδα χρόνου, και c είναι το κόστος του πόρου (κόστος ανά μονάδα του πόρου).

Κατανάλωση

Παρουσιάζεται στο σύστημά μας όταν υπάρχουν ταυτόχρονα στο σύστημά μας μια αίτηση και μια χορηγία. Μια κατανάλωση ενέργειας είναι της μορφής ($\vec{r}$,m,z,k) όπου r είναι το όνομα του πόρου, m είναι η ποσότητα του πόρου (σε αριθμό μονάδων του πόρου) που καταναλώνεται, z είναι η προτεραιότητα του αγοραστή που έκανε την κατανάλωση και k είναι το κόστος στο οποίο προμηθεύτηκε το ποσό του πόρου r (σε αριθμό μονάδων του πόρου) που καταναλώθηκε.

3.2.1 Δράσεις

Μια δράση α είναι ένα σύνολο από διάφορες ενέργειες (αιτήσεις, χορηγίες και καταναλώσεις), με τον περιορισμό εάν κάποιος πόρος r εμφανίζεται περισσότερες από μία φορές τότε οι εμφανίσεις αυτές πρέπει να είναι μια κατανάλωση και είτε μια χορηγία είτε μια αίτηση [14].

Συμβολίζουμε το σύνολο όλων των δράσεων με Act. Ακολουθούν κάποια παραδείγματα με πιθανές δράσεις που μπορούν να εμφανιστούν.

<u>Παραδείγματα δράσεων :</u>

- $\{(\vec{r}, 2,5,1)\ ,(\bar{r}, 5,2)\}$

  Στη δράση υπάρχει μια κατανάλωση 2 μονάδων ενέργειας από τον πόρο r με προτεραιότητα 5 η οποία καταναλώθηκε έναντι του χρηματικού ποσού του €1 ανά μονάδα ενέργειας και μια χορηγία του πόρου r όπου εκτελεί 5 μονάδες του πόρου ανά μονάδα χρόνου έναντι του ποσού των €2 ανά μονάδα ενέργειας

- $\{(r_1,5,3),(r_2,20,4)\}$

  Στη δράση υπάρχουν 2 αιτήσεις για ενέργεια, μία αίτηση 5 μονάδων ενέργειας από τον πόρο r_1 με προτεραιότητα 3, και μια αίτηση 20 μονάδων ενέργειας από τον πόρο r_2 με προτεραιότητα 4

- $\{(\vec{r}, 4,1,2)\ , (r, 6,2)\}$

  Στη δράση υπάρχει μια κατανάλωση 4 μονάδων ενέργειας από τον πόρο r με κόστος €2 ανά μονάδα ενέργειας και προτεραιότητα 1 και μια αίτηση 6 μονάδων ενέργειας από τον πόρο r με προτεραιότητα 2

3.2.2 Διεργασίες

Αρχικά ορίζουμε ως Act_r το σύνολο που περιέχει μόνο αιτήσεις πόρων και Act_g το σύνολο που περιέχει μόνο χορηγίες πόρων

Ένα σύστημα διατυπωμένο στην ACCED απαρτίζεται από τις ακόλουθες δύο κατηγορίες διεργασιών:

1. **Εργασίες**

   Οι εργασίες είναι οι μονάδες οι οποίες αιτούνται να πάρουν κάποια ποσότητα πόρων από το σύστημα. Συμβολίζονται με T_i , $i \in I$ και έχουν την δυνατότητα να εκτελούν δράσεις από το σύνολο Act_r.

2. **Προμήθειες**

   Οι προμήθειες είναι οι μονάδες οι οποίες προσφέρουν κάποιους πόρους στο σύστημα. Συμβολίζονται με S_i , $i \in I$ και έχουν τη δυνατότητα να εκτελούν δράσεις από το σύνολο Act_g

3.2.3 Χρόνος

Στο σύστημά μας ο χρόνος θα είναι διακριτός. Δηλαδή θα μετράμε σε γύρους όπου σε κάθε γύρο όλοι θα πρέπει να εκτελούν και από μια ενέργεια δηλαδή δεν γίνεται κάποιοι να είναι πιο μπροστά και κάποιοι πιο πίσω, αλλά όλοι πρέπει να προχωράνε μαζί [18]. Επίσης ένα άλλο πολύ βοηθητικό σημείο είναι ότι τα γεγονότα μπορούν να πραγματοποιηθούν μόνο σε ακέραιες χρονικές στιγμές. Επιπρόσθετα, κάθε δράση χρειάζεται μια μονάδα χρόνου και για να ορίσουμε δράσεις με μεγαλύτερη χρονική διάρκεια απλά επαναλαμβάνουμε τη δράση όσες φορές χρειαστεί [16]. Δηλαδή η άλγεβρα διεργασιών μας αναφέρεται σε ένα σύγχρονο μοντέλο.

3.3 Σύνταξη

Η άλγεβρα διεργασιών ACCED περιλαμβάνει το σύνολο των εργασιών T, το σύνολο των προμηθειών S, το σύνολο των γεγονότων e και το σύνολο των συστημάτων P. Με τον συμβολισμό ρ παρουσιάζονται τα στοιχεία του Act_r, όπως αυτό έχει ορισθεί προηγουμένως και με τον συμβολισμό γ παρουσιάζονται τα στοιχεία του Act_g , όπως αυτό έχει ορισθεί προηγουμένως.

$T ::= NIL \mid \sum_{i \in I} \rho_i : T_i \mid A \mid e.T$

$S ::= NIL \mid \sum_{i \in I} \gamma_i : S_i \mid B$

$e ::= (l?,ve) \mid (l!,ve) \mid (l?x,ve) \mid (l!x,ve)$

$P ::= T \mid S \mid (P \parallel P) \mid P\backslash F$

Πίνακας 3.1 Σύνταξη Άλγεβρας Διεργασιών ACCED

Τα l προέρχονται από ένα σύνολο καναλιών και μπορούν να χρησιμοποιηθούν για είσοδο και για έξοδο. Ακολουθούν οι επεξηγήσεις για την σύνταξη της ACCED καθώς και περισσότερες λεπτομέρειες για τις τέσσερις πιθανές μορφές του γεγονότος e.

- Μια εργασία T μπορεί είτε να είναι αδρανής (NIL), είτε να αποτελείται από ένα σύνολο εργασιών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά εργασίας A, είτε να σχετίζεται με ένα γεγονός το οποίο συμβαίνει εντός ενός καναλιού επικοινωνίας.
- Μια προμήθεια S μπορεί είτε να είναι αδρανής (NIL), είτε να αποτελείται από ένα σύνολο προμηθειών οι οποίες επιλέγονται κατά ένα μη ντετερμινιστικό τρόπο, είτε να είναι μια σταθερά προμήθειας B.
- Ένα γεγονός e μπορεί είτε να αξιοποιηθεί για είσοδο με το να παραλάβει ένα μήνυμα ve από το κανάλι επικοινωνίας l, είτε να αξιοποιηθεί για έξοδο με το να στείλει ένα μήνυμα ve από το κανάλι επικοινωνίας l, είτε να αξιοποιηθεί για είσοδο με το να παραλάβει ένα μήνυμα ve με τιμή x από το κανάλι επικοινωνίας l, είτε να αξιοποιηθεί για έξοδο με το να στείλει ένα μήνυμα ve με τιμή x από το κανάλι επικοινωνίας l. Το κάθε μήνυμα έχει τη δική του προτεραιότητα [2]. Επίσης με [ve] συμβολίζουμε την πραγματική τιμή που παίρνει η αριθμητική έκφραση ve.

- Ένα σύστημα P μπορεί είτε να αποτελείται από μια εργασία T, είτε να αποτελείται από μια προμήθεια S, είτε να αποτελείται από μια παράλληλη εκτέλεση δύο συστημάτων, είτε να αποτελείται από ένα δεσμευμένο κανάλι επικοινωνίας F.

Γενικά, η σύνταξη της άλγεβρας διεργασιών ACCED, είναι αρκετά ευέλικτη. Αυτό μας επιτρέπει να υποστηρίζουμε διάφορες πολιτικές χρονοδρομολόγησης.

3.4 Χρήση της ACCED για μοντελοποίηση της χρονοδρομολόγησης

Όπως έχουμε ήδη αναφέρει, η ACCED μπορεί να χρησιμοποιηθεί ως εργαλείο για την μοντελοποίηση διάφορων πολιτικών χρονοδρομολόγησης που έχουν εφαρμογή στο υπολογιστικό νέφος. Συγκεκριμένα το πρόβλημα το οποίο επιθυμούμε να λύσουμε είναι το γεγονός ότι θέλουμε να δούμε πως μπορούν να χρονοδρομολογηθούν οι διάφορες εργασίες ανάλογα με το είδος της πολιτικής χρονοδρομολόγησης που εφαρμόζουμε κάθε φορά. Χάρη στην ACCED μπορέσαμε να μοντελοποιήσουμε διάφορες πολιτικές χρονοδρομολόγησης. Ακολούθως επεξηγούνται μερικές από τις κυριότερες πολιτικές χρονοδρομολόγησης, και στη συνέχεια παρουσιάζονται παραδείγματα για κάποιες από αυτές τις πολιτικές μαζί με τα αντίστοιχα συστήματα μεταβάσεων:

- **EDF (Earliest Deadline First):** Το αίτημα με τη μικρότερη προθεσμία προηγείται, δηλαδή του δίνουμε μεγαλύτερη προτεραιότητα. Αυτό το επιτυγχάνουμε θέτοντας προτεραιότητα = $d_{max} - (d - t)$ όπου d_{max} είναι η μεγαλύτερη δυνατή προθεσμία που μπορεί να υπάρξει στο σύστημα μας, d είναι η προθεσμία, και t ο χρόνος που έχει περάσει.

- **Μη-διακοπτόμενη:** Ένα αίτημα όταν αρχίσει να εκτελείται δεν μπορεί να διακοπεί αν δεν ολοκληρωθεί.

- **Διακοπτόμενη:** Ένα αίτημα όταν αρχίζει να εκτελείται, μπορεί να διακοπεί και να συνεχίσει την εκτέλεσή του πολλές φορές μέχρι να ολοκληρωθεί.

- **Min-Min:** Το αίτημα που ζητά τη μικρότερη ποσότητα προηγείται.

- **Max-Min:** Το αίτημα που ζητά τη μεγαλύτερη ποσότητα προηγείται.

Ακολουθούν κάποια παραδείγματα στα οποία φαίνονται οι δυνατότητες που έχει η σύνταξη της άλγεβρας διεργασιών ACCED, και πως αλλάζουν οι διάφοροι παράμετροι ανάλογα με την πολιτική χρονοδρομολόγησης που θα ακολουθήσουμε.

3.4.1 Μοντελοποίηση χρονοδρομολόγησης με την πολιτική EDF

Στο παράδειγμα αυτό υποθέτουμε την ύπαρξη των εργασιών $T_1,..,T_n$. Για κάθε i, η εργασία T_i έχει τα πιο κάτω χαρακτηριστικά: t_i είναι ο χρόνος που έχει περάσει, d_i είναι η προθεσμία, x_i είναι η ποσότητα που ζητείται, u_i είναι η ποσότητα που χορηγείται, μπορεί να είναι μεταξύ του 0 και του x_i και $d_{max}=\max(d_i)$ $1\leq i \leq n$. Επιπλέον υποθέτουμε ότι οι εργασίες θα χρονοπρογραμματιστούν βάσει της πολιτικής EDF. Συμβολίζουμε την εργασία T_i, στην αρχική της κατάσταση ως $T_{EDF}(x,0,d)$ και την ορίζουμε βάσει των πιο κάτω εξισώσεων:

$$T_{EDF}(x,t,d) \stackrel{\text{def}}{=} \begin{cases} NIL & \alpha\nu\ (x=0) \\ \sum_{0\leq u\leq x} \{(r,u,d_{max}-(d-t))\}: T_{EDF}(x-u,t+1,d) & \alpha\nu\ (d-t>1) \\ \{(r,x,d_{max}-1)\}: NIL & \alpha\nu\ (d-t=1) \end{cases}$$

Δεδομένης της εργασίας T η οποία έχει προθεσμία d, ζητά x μονάδες ενέργειας και έχει περάσει χρόνος t, εάν έχει καλύψει τις ανάγκες της, τότε τερματίζει. Εάν έχει ακόμη περισσότερες από μια χρονικές στιγμές μέχρι την προθεσμία της τότε λαμβάνει u μονάδες ενέργειες, ποσότητα που κυμαίνεται μεταξύ 0 μέχρι και x και ακολούθως εξελίσσεται σε εργασία T πάλι. Εάν απέχει μόνο μια χρονική στιγμή από την προθεσμία της τότε θα ζητήσει όλη την ποσότητα που χρειάζεται τώρα διότι αν δεν το κάνει τώρα θα χάσει την προθεσμία της και ακολούθως τερματίζει.

3.4.2 Χρονοδρομολόγηση με προτεραιότητες και μη διακοπτόμενη πολιτική

Στο παράδειγμα αυτό υποθέτουμε την ύπαρξη των εργασιών $T_1,..,T_n$. Για κάθε i,η εργασία T_i έχει τα πιο κάτω χαρακτηριστικά: t_i είναι ο χρόνος που έχει περάσει, d_i είναι η προθεσμία, x_i είναι η ποσότητα που ζητείται, u_i είναι η ποσότητα που χορηγείται, μπορεί να είναι μεταξύ του 0 και του x_i και με p_i συμβολίζουμε την προτεραιότητα. Αφού ακολουθούμε τη μη-διακοπτόμενη πολιτική θα πρέπει να μεταβαίνουμε σε μια νέα κατάσταση, όπου η προτεραιότητα ισούται με +∞,διότι ο μεγαλύτερος αριθμός υποδηλώνει και μεγαλύτερη

προτεραιότητα. Αυτή η νέα κατάσταση λέγεται Started και διαφέρει με την κατάσταση T μόνο στην προτεραιότητα. Η διεργασία Started υποδεικνύει πότε μια εργασία έχει επιλεγεί και κατ' επέκταση δεν μπορεί να διακοπεί εάν δεν ολοκληρωθεί. Επιπλέον υποθέτουμε ότι οι εργασίες θα χρονοπρογραμματιστούν βάσει της μη-διακοπτόμενης πολιτικής (NPR-Non Pre-emptible). Συμβολίζουμε την εργασία $T_{i,}$ στην αρχική της κατάσταση ως $T_{NPR}(x,0,d,p)$ και συμβολίζουμε την αρχική κατάσταση της διεργασίας Started ως Started(x,0,d) και τις ορίζουμε βάσει των πιο κάτω εξισώσεων:

$T_{NPR}(x,t,d,p) \stackrel{def}{\equiv}$

$$\begin{cases} NIL & αν\ (x = 0) \\ \sum_{0<u\leq x} \{(r,u,p)\}: Started(x-u, t+1, d) + \emptyset: T_{NPR}\,(x, t+1, d, p) & αν\ (d - t > 1) \\ \{(r,x,p)\}: NIL & αν\ (d - t = 1) \end{cases}$$

$$\text{Started}(x,t,d) \stackrel{def}{\equiv} \begin{cases} NIL & αν\ (x = 0) \\ \sum_{0\leq u\leq x} \{(r,u,+\infty)\}: Started(x-u, t+1, d) & αν\ (d - t > 1) \\ \{(r,x,+\infty)\}: NIL & αν\ (d - t = 1) \end{cases}$$

Δεδομένης της εργασίας T η οποία έχει προθεσμία d, προτεραιότητα p, ζητά x μονάδες ενέργειας και έχει περάσει χρόνος t. Εάν έχει καλύψει τις ανάγκες της, τότε τερματίζει. Εάν έχει ακόμη περισσότερες από μια χρονικές στιγμές μέχρι την προθεσμία της τότε είτε λαμβάνει u μονάδες ενέργειες, ποσότητα που κυμαίνεται μεταξύ 0 μέχρι και x και ακολούθως εξελίσσεται σε διεργασία Started είτε αδρανεί. Εάν απέχει μόνο μια χρονική στιγμή από την προθεσμία της τότε θα ζητήσει όλη την ποσότητα που χρειάζεται τώρα διότι αν δεν το κάνει τώρα θα χάσει την προθεσμία της και ακολούθως τερματίζει. Επίσης δεδομένης της διεργασίας Started η οποία έχει προθεσμία d, ζητά x μονάδες ενέργειας και έχει περάσει χρόνος t, εάν έχει καλύψει τις ανάγκες της, τότε τερματίζει. Εάν έχει ακόμη περισσότερες από μια χρονικές στιγμές μέχρι την προθεσμία της τότε λαμβάνει u μονάδες ενέργειες, ποσότητα που κυμαίνεται μεταξύ 0 μέχρι και x και ακολούθως εξελίσσεται σε πάλι σε διεργασία Started. Εάν απέχει μόνο μια χρονική στιγμή από την προθεσμία της τότε θα ζητήσει όλη την ποσότητα που χρειάζεται τώρα διότι αν δεν το κάνει τώρα θα χάσει την προθεσμία της και ακολούθως τερματίζει.

3.4.3 Χρονοδρομολόγηση με προτεραιότητες και διακοπτόμενη πολιτική

Στο παράδειγμα αυτό υποθέτουμε την ύπαρξη των εργασιών $T_1,..,T_n$. Για κάθε i,η εργασία T_i έχει τα πιο κάτω χαρακτηριστικά: t_i είναι ο χρόνος που έχει περάσει, d_i είναι η προθεσμία, x_i είναι η ποσότητα που ζητείται, u_i είναι η ποσότητα που χορηγείται, μπορεί να είναι μεταξύ του 0 και του x_i και με p_i συμβολίζουμε την προτεραιότητα. Επιπλέον υποθέτουμε ότι οι εργασίες θα χρονοπρογραμματιστούν βάσει της διακοπτόμενης πολιτικής (PR-Pre-emptible). Συμβολίζουμε την εργασία $T_{i,}$ στην αρχική της κατάσταση ως $T_{PR}(x,0,d,p)$ και την ορίζουμε βάσει των πιο κάτω εξισώσεων:

$$T_{PR}(x,t,d,p) \stackrel{\text{def}}{=} \begin{cases} NIL & αν\ (x = 0) \\ \sum_{0 \le u \le x} \{(r,u,p)\} : T_{PR}(x-u, t+1, d, p) & αν\ (d - t > 1) \\ \{(r,x,p)\} : NIL & αν\ (d - t = 1) \end{cases}$$

Δεδομένης της εργασίας T η οποία έχει προθεσμία d, προτεραιότητα p, ζητά x μονάδες ενέργειας και έχει περάσει χρόνος t, εάν έχει καλύψει τις ανάγκες της, τότε τερματίζει. Εάν έχει ακόμη περισσότερες από μια χρονικές στιγμές μέχρι την προθεσμία της τότε λαμβάνει u μονάδες ενέργειες, ποσότητα που κυμαίνεται μεταξύ 0 μέχρι και x και ακολούθως εξελίσσεται σε εργασία T πάλι. Εάν απέχει μόνο μια χρονική στιγμή από την προθεσμία της τότε θα ζητήσει όλη την ποσότητα που χρειάζεται τώρα διότι αν δεν το κάνει τώρα θα χάσει την προθεσμία της και ακολούθως τερματίζει.

Ορισμός 3.1

$(S_1||S_2)||S_3 \equiv S_1||(S_2||S_3)$ και $(S_1||S_2) \equiv (S_2||S_1)$

Με τον συμβολισμό ≡ ορίζουμε την συντακτική ισοδυναμία. Με τον όρο συντακτική ισοδυναμία εννοούμε ότι όποια και να είναι η παρενθεσιοποίηση που έχουμε, θα οδηγούμαστε στις ίδιες πιθανές μεταβάσεις. Δηλαδή ισχύει η προσεταιριστική ιδιότητα. Επίσης οδηγούμαστε στις ίδιες πιθανές μεταβάσεις ανεξάρτητα από την σειρά της εμφάνισης των διεργασιών (εργασία, προμήθεια). Δηλαδή ισχύει και η αντιμεταθετική ιδιότητα.

3.5 Σημασιολογία

Η σημασιολογία της ACCED δίνεται σε δύο φάσεις. Στην πρώτη φάση ορίζουμε τους βασικούς κανόνες που σχετίζονται με τις δράσεις των διεργασιών. Στην δεύτερη φάση ορίζουμε τους βασικούς κανόνες που σχετίζονται με τις ενέργειες που μπορούν να εκτελέσουν οι διεργασίες. Ξεκινώντας από την πρώτη φάση στην οποία δίνεται η λειτουργία μέσω ενός συνόλου από κανόνες οι οποίοι είναι οι πρώτοι έξι κανόνες που δίνονται, καθώς και ο τελευταίος κανόνας. Ακολούθως οι υπόλοιποι κανόνες με εξαίρεση τον τελευταίο, αφορούν την δεύτερη φάση στην οποία δίνεται η λειτουργία μέσω ενός συνόλου που αφορά ενέργειες οι οποίες εκτυλίσσονται εντός ενός καναλιού επικοινωνίας.

Στον Πίνακα 3.2, παρουσιάζονται οι δεκαπέντε κανόνες μετάβασης της άλγεβρας διεργασιών ACCED, οι οποίοι εξηγούνται πιο κάτω:

Idle: Μια διεργασία NIL επιτρέπει στον χρόνο να κυλά χωρίς την εκτέλεση κάποιας δράσης
SumT: Εκτέλεση μιας εκ των δράσεων και μετάβαση στην επόμενη κατάσταση που προβλέπεται
SumS: Εκτέλεση μιας εκ των δράσεων και μετάβαση στην επόμενη κατάσταση που προβλέπεται
ConT: Μια σταθερά εργασίας συμπεριφέρεται όπως την διεργασία στην οποία είναι ορισμένη
ConS: Μια σταθερά προμήθειας συμπεριφέρεται όπως την διεργασία στην οποία είναι ορισμένη
Par: Παράλληλη εκτέλεση των δράσεων δύο συστημάτων
ActR: Παραλαβή μηνύματος στο κανάλι επικοινωνίας. Σχετίζεται με την πραγματική του τιμή που παίρνει η αριθμητική έκφραση ve η οποία δηλώνει ότι το κανάλι επικοινωνίας χρησιμοποιήθηκε για είσοδο
ActS: Αποστολή μηνύματος στο κανάλι επικοινωνίας. Σχετίζεται με την πραγματική του τιμή που παίρνει η αριθμητική έκφραση ve η οποία δηλώνει ότι το κανάλι επικοινωνίας χρησιμοποιήθηκε για έξοδο
ActRV: Παραλαβή μηνύματος με τιμή στο κανάλι επικοινωνίας. Σχετίζεται με την πραγματική του τιμή που παίρνει η αριθμητική έκφραση ve η οποία δηλώνει ότι το κανάλι επικοινωνίας χρησιμοποιήθηκε για είσοδο

ActSV: Αποστολή μηνύματος με τιμή στο κανάλι επικοινωνίας. Σχετίζεται με την πραγματική του τιμή που παίρνει η αριθμητική έκφραση ve η οποία δηλώνει ότι το κανάλι επικοινωνίας χρησιμοποιήθηκε για έξοδο

ParL: Εκτέλεση γεγονότος από την αριστερή συνιστώσα μιας παράλληλης σύνθεσης

ParR: Εκτέλεση γεγονότος από τη δεξιά συνιστώσα μιας παράλληλης σύνθεσης

ParC: Παράλληλη εκτέλεση των δράσεων δύο συστημάτων μέσω καναλιών επικοινωνίας όπου το ένα εκτελεί αποστολή και το άλλο εκτελεί παραλαβή μηνύματος με τιμή και αυτό εμφανίζεται ως εσωτερική ενέργεια στο σύστημα

Res: Εκτέλεση των δράσεων εντός του δεσμευμένου καναλιού επικοινωνίας, όπου ελέγχουμε αν το κανάλι που επιθυμούμε να δεσμεύσουμε δεν είναι ήδη δεσμευμένο

Equiv: Συντακτική ισοδυναμία μεταξύ των πιθανών μεταβάσεων των διεργασιών ενός συστήματος

Επίσης με γ(e) συμβολίζουμε το όνομα του καναλιού του γεγονότος e [2]. Δηλαδή γ(l?,ve)=l και ούτω καθεξής.

Πίνακας 3.2 Κανόνες μετάβασης Άλγεβρας Διεργασιών ACCED

Ονομασία	**Κανόνας**
Idle:	$\text{NIL} \xrightarrow{\emptyset} \text{NIL}$
SumT:	$\sum_{i \in I} \rho_i : \text{T}_i \xrightarrow{\rho_i} \text{T}_i \quad i \in I$
SumS:	$\sum_{i \in I} \gamma_i : \text{S}_i \xrightarrow{\gamma_i} \text{S}_i \quad i \in I$
ConT:	$\frac{T \xrightarrow{\rho} T'}{A \xrightarrow{\rho} T'} \qquad A \stackrel{\text{def}}{=} T$
ConS:	$\frac{S \xrightarrow{\gamma} S'}{B \xrightarrow{\gamma} S'} \qquad B \stackrel{\text{def}}{=} S$
Par:	$\frac{P_1 \xrightarrow{\alpha_1} P_1' \,,\, P_2 \xrightarrow{\alpha_2} P_2'}{P_1 \| P_2 \xrightarrow{a} P_1' \| P_2'} \qquad \alpha \in \alpha_1 \oplus \alpha_2$
ActR:	$(\text{l?,ve}).\text{P} \xrightarrow{(l?,[ve])} \text{P}$
ActS:	$(\text{l!,ve}).\text{P} \xrightarrow{(l!,[ve])} \text{P}$
ActRV:	$(\text{l?x,ve}).\text{P} \xrightarrow{(l?n,[ve])} \text{P[n/x]} \quad n \in \mathbb{Z}$
ActSV:	$(\text{l!ve}_2,\text{ve}_1).\text{P} \xrightarrow{(l![ve_2],[ve_1])} \text{P}$
ParL:	$\frac{P \xrightarrow{e} P'}{P \| Q \xrightarrow{e} P' \| Q}$
ParR:	$\frac{Q \xrightarrow{e} Q'}{P \| Q \xrightarrow{e} P \| Q'}$
ParC:	$\frac{P \xrightarrow{(l!k,m)} P' \,,\, Q \xrightarrow{(l?k,n)} Q'}{P \| Q \xrightarrow{(\tau,m+n)} P' \| Q'}$
Res:	$\frac{P \xrightarrow{e} P'}{P \backslash F \xrightarrow{e} P' \backslash F} \qquad (\gamma(e) \notin F)$
Equiv:	$\frac{S_1 \equiv S_2 , S_1 \xrightarrow{a} S_1'}{S_2 \xrightarrow{a} S_2'}$

Επεξήγηση κανόνα μετάβασης Par: Ο κανόνας μετάβασης Par επιτρέπει την παράλληλη εκτέλεση των δράσεων δύο συστημάτων. Βασίζεται στον τελεστή $\oplus$ ο οποίος συνδυάζει τις δράσεις των δύο επιμέρους συστημάτων. Κατ' ακρίβεια αν έχουμε δύο ενέργειες α_1 και α_2 υπάρχουν οι εξής πιθανές περιπτώσεις για το αποτέλεσμα της σύνθεσης $\alpha_1 \oplus \alpha_2$. Πρώτη πιθανή περίπτωση, είναι να μην υπάρχουν χορηγίες αλλά να υπάρχουν αιτήσεις για κάποιο συγκεκριμένο πόρο. Δεύτερη πιθανή περίπτωση, είναι να μην υπάρχουν αιτήματα αλλά να υπάρχουν χορηγίες για κάποιο συγκεκριμένο πόρο και τρίτη πιθανή περίπτωση, είναι να προϋπήρχαν κάποια αιτήματα και κάποιες χορηγίες για ένα συγκεκριμένο πόρο τα οποία θα εμφανιστούν ως καταναλώσεις αυτού του συγκεκριμένου πόρου. Συγκεκριμένα, αν παράλληλα ζητούνται a μονάδες ενέργειας από τον πόρο r και χορηγούνται b μονάδες ενέργειας από τον πόρο r, τότε αν a ισούται με b θα προκύψει μια κατανάλωση a μονάδων ενέργειας από τον πόρο r, αν a μεγαλύτερο από b τότε θα προκύψει μια κατανάλωση b μονάδων ενέργειας από τον πόρο r και μια αίτηση a-b μονάδων ενέργειας από τον πόρο r, ενώ αν b μεγαλύτερο από το a τότε θα προκύψει μια κατανάλωση a μονάδων ενέργειας από τον πόρο r και μια χορηγία b-a μονάδων ενέργειας από τον πόρο r. Επίσης ορίζουμε με $Act_r(\alpha)$ το σύνολο όλων των ενεργειών που αφορούν αιτήσεις του πόρου r στη δράση α.

Ορισμός 3.2

Έστω οι δράσεις α_1 και α_2. Η δράση $\alpha_1 \oplus \alpha_2$ που αποτελεί τη σύνθεση των δύο επιμέρους δράσεων περιέχει τις πιο κάτω ενέργειες:

0) Το σύνολο των καταναλώσεων που προϋπήρχαν

1) Το σύνολο όλων των αιτήσεων για πόρους τους οποίους δεν μπορεί να καλύψει η προμήθεια

2) Το σύνολο όλων των χορηγήσεων για πόρους για τους οποίους όμως δεν υπάρχει ζήτηση από καμία εργασία

3) Το σύνολο όλων των δυνατών καταναλώσεων πόρων που μπορούν να δημιουργηθούν από το συνδυασμό μεταξύ των αντίστοιχων αιτήσεων και χορηγιών

4) Το σύνολο όλων των δυνατών καταναλώσεων πόρων που μπορούν να δημιουργηθούν και της υπολειπόμενης χορηγίας που θα απομείνει μετά από αυτή την κατανάλωση

5) Το σύνολο όλων των δυνατών καταναλώσεων πόρων που μπορούν να δημιουργηθούν και μπορούν να καλυφθούν από μια προμήθεια, καθώς και το σύνολο όλων των αιτήσεων που στο τέλος δεν θα μπορούν να καλυφθούν από την προμήθεια και θα εμφανιστούν στο σύστημα ως αιτήματα. Σε περίπτωση που υπάρχουν δύο αιτήματα

με την ίδια προτεραιότητα τότε δίνεται προτεραιότητα στο αίτημα που ζητά την μεγαλύτερη ποσότητα. Συγκεκριμένα ικανοποιούνται οι καταναλώσεις σε φθίνουσα σειρά προτεραιότητας και τα υπόλοιπα αιτήματα που δεν μπόρεσαν να ικανοποιηθούν από τη χορηγία παραμένουν ως ανικανοποίητα αιτήματα.

Σημειώνουμε επίσης ότι αυτό είναι μια σχεδιαστική επιλογή. Το κίνητρο που μας ώθησε σε αυτή την επιλογή είναι το γεγονός ότι στο σύστημα μας δεν υπάρχουν αποθηκευτικές μονάδες ενέργειας. Αυτό έχει ως επακόλουθο εάν η ενέργεια δεν αξιοποιηθεί, δηλαδή καταναλωθεί άμεσα, τότε χάνεται, γεγονός που δεν είναι επιθυμητό. Επίσης οι προμηθευτές προτιμούν να πουλήσουν όσο το δυνατόν περισσότερες ποσότητες ενέργειας μπορούν και κατ' επέκταση αυτό θα εμφανιστεί στο σύστημα μας ως καταναλώσεις. Αυτό είναι επιθυμητό γιατί θα αποφέρει στους προμηθευτές περισσότερα χρήματα άρα και κέρδος. Για αυτό λοιπόν προτιμήσαμε αυτή τη σχεδιαστική επιλογή.

Συμβολικά

$\boldsymbol{\alpha_1} \oplus \boldsymbol{\alpha_2} =$

0) $\{\{(\vec{r}, e, \text{pr}, \text{c})\} \mid \text{ανήκει } (\vec{r}, e, \text{pr}, \text{c}) \in \alpha_i \cup \alpha_{3\text{-i}}\}$

$\cup$

1) $\{\{(\text{r},\text{e},\text{pr})\} \in \alpha_i \mid \nexists(\bar{r}, s, c) \in \alpha_{3\text{-i}}\}$

$\cup$

2) $\{\{(\bar{r}, s, c)\} \in \alpha_i \mid \nexists (\text{r},\text{e},\text{pr}) \in \alpha_{3\text{-i}}\}$

$\cup$

3) $\{\{(\vec{r}, e_1, \text{pr}_1, \text{c}), \ldots, (\vec{r}, e_n, \text{pr}_n, \text{c})\}\} \mid \{(\text{r},\text{e}_1,\text{pr}_1), \ldots (\text{r},\text{e}_n,\text{pr}_n)\} = \text{Act}_r(\alpha), (\bar{r}, s, c) \in \alpha_{3\text{-i}}, s = e_1 + \ldots + e_n\}$

$\cup$

4) $\{\{(\vec{r}, e_1, \text{pr}_1, \text{c}), \ldots, (\vec{r}, e_n, \text{pr}_n, \text{c}), (\bar{r}, \text{c}, \text{z})\}\} \mid \{(\text{r},\text{e}_1,\text{pr}_1), \ldots, (\text{r},\text{e}_n,\text{pr}_n)\} = \text{Act}_r(\alpha), (\bar{r}, s, c) \in \alpha_{3\text{-i}}, s = e_1 + \ldots + e_n + z\}$

$\cup$

5) $\{\{(\vec{r}, e_1, \text{pr}_1, \text{c}), \ldots, (\vec{r}, e_n', \text{pr}_n, \text{c}), (\text{r},\text{e}_n'', \text{pr}_n), (\text{r},\text{e}_{n+1}, \text{pr}_{n+1}), \ldots (\text{r},\text{e}_m, \text{pr}_m)\}\} \mid \{(\text{r},\text{e}_1,\text{pr}_1), \ldots, \ldots (\text{r},\text{e}_m,\text{pr}_m)\} = \text{Act}_r(\alpha), (\bar{r}, s, c) \in \alpha_{3\text{-i}}, s = e_1 + \ldots + e_n', e_n = e_n' + e_n'',$ και όπου $\text{pr}_1 \geq \ldots \geq \text{pr}_m$, και $\forall i$ αν $\text{pr}_i = \text{pr}_{i+1}$ τότε $e_i \geq e_{i+1}\}$

Παραδείγματα

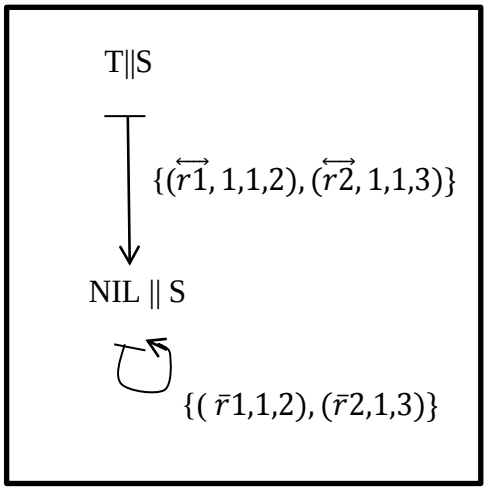
Στις ακόλουθες σελίδες παρουσιάζονται κάποια παραδείγματα, από πιθανές εκτελέσεις διεργασιών σε ένα σύστημα διατυπωμένο στην ACCED. Συγκεκριμένα παρουσιάζονται πέντε παραδείγματα που δείχνουν τις πιθανές αυτές εκτελέσεις των διεργασιών, προτού οριστεί η έννοια της σχέσης προτιμήσεως. Με τον όρο σχέση προτιμήσεως αναφερόμαστε στο γεγονός ότι βάσει κάποιων σημείων τα οποία θα επεξηγηθούν εκτενώς στη συνέχεια, μπορούμε να δούμε πότε μια δράση υπερνικά μια άλλη δράση. Αυτό έχει ως αποτέλεσμα αν δύο δράσεις, έστω α και β είναι οι δράσεις που προκύπτουν από τη παράλληλη εκτέλεση του ιδίου συστήματος τότε αν ισχύει η σχέση προτιμήσεως, είτε η δράση α θα αποτρέψει την εμφάνιση της δράσης β είτε η δράση β θα αποτρέψει την εμφάνιση της δράσης α. Αυτό θα εξαρτηθεί από τα σημεία της σχέσης προτιμήσεως, τα οποία όπως έχουμε προαναφέρει θα εξηγηθούν στη συνέχεια. Επίσης παρουσιάζονται κάποιοι γνωστοί αλγόριθμοι πολιτικών χρονοδρομολόγησης και πώς μπορούν να μοντελοποιηθούν βάσει της άλγεβρας διεργασιών ACCED.

Παράδειγμα 3.4

Στο σύστημα αυτό έχουμε την παράλληλη σύνθεση μιας εργασίας Τ και μιας προμήθειας S (Βλέπε Σχήμα 3.1). Η εργασία Τ ζητά 1 μονάδα ενέργειας από τον πόρο r_1, έχει προθεσμία 1 μονάδα χρόνου και προτεραιότητα 1 και ζητά 1 μονάδα ενέργειας από τον πόρο r_2, και έχει προθεσμία 1 μονάδα χρόνου και προτεραιότητα 1 και μετά τερματίζει. Η προμήθεια S προσφέρει 1 μονάδα ενέργειας από τον πόρο r_1,ανά μονάδα χρόνου έναντι του ποσού των €2 ανά μονάδα ενέργειας και προσφέρει 1 μονάδα ενέργειας από τον πόρο r_2, ανά μονάδα χρόνου έναντι του ποσού των €3 ανά μονάδα ενέργειας. Ακολούθως, η προμήθεια S εξελίσσεται πάλι στην προμήθεια S. Επομένως η προμήθεια S μπορεί να καλύψει τις ανάγκες της εργασίας Τ, δίχως να χάσει η εργασία Τ την προθεσμία της. Η προμήθεια S και η εργασία Τ συμβολίζονται ως εξής:

$S \stackrel{def}{=} \{(\bar{r}1,1,2),(\bar{r}2,1,3)\} : S$ $\quad$ $T \stackrel{def}{=} \{(r1,1,1),(r2,1,1)\} : NIL$

Παρατηρούμε ότι το σύστημα μεταβάσεων που ακολουθεί έχει μόνο μια πιθανή μετάβαση. Ακολούθως η εργασία Τ τερματίζει και η προμήθεια S μπορεί να χορηγεί τις διαθέσιμές της ποσότητες όπως έχουν ήδη αναφερθεί, συνεχώς.



Σχήμα 3.1 Σύστημα μεταβάσεων παραδείγματος 3.4

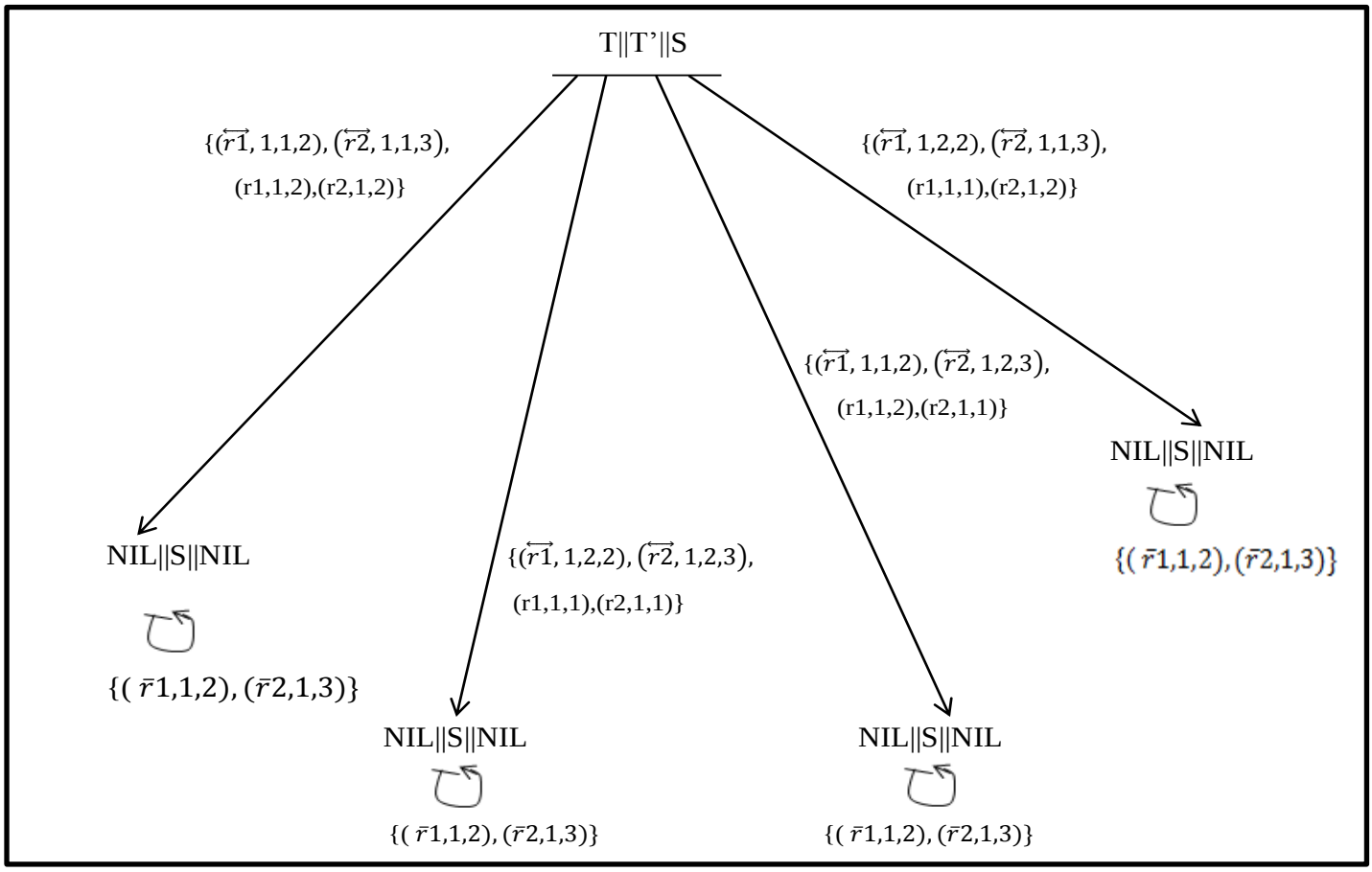
Παράδειγμα 3.5

Στο σύστημα αυτό έχουμε την παράλληλη σύνθεση δύο εργασιών, της Τ και της Τ' και μιας προμήθειας S (Βλέπε Σχήμα 3.2). Η εργασία Τ ζητά 1 μονάδα ενέργειας από τον πόρο r_1, έχει προθεσμία 1 μονάδα χρόνου και προτεραιότητα 1 και ζητά 1 μονάδα ενέργειας από τον πόρο r_2, και έχει προθεσμία 1 μονάδα χρόνου και προτεραιότητα 1 και μετά τερματίζει. Η εργασία Τ' ζητά 1 μονάδα ενέργειας από τον πόρο r_1, έχει προθεσμία 1 μονάδα χρόνου και προτεραιότητα 2 και ζητά 1 μονάδα ενέργειας από τον πόρο r_2,και έχει προθεσμία 1 μονάδα χρόνου και προτεραιότητα 2 και μετά τερματίζει. Η προμήθεια S προσφέρει 1 μονάδα ενέργειας από τον πόρο r_1,ανά μονάδα χρόνου έναντι του ποσού των €2 ανά μονάδα ενέργειας και προσφέρει 1 μονάδα ενέργειας από τον πόρο r_2,ανά μονάδα χρόνου έναντι του ποσού των €3 ανά μονάδα ενέργειας. Ακολούθως η προμήθεια S εξελίσσεται πάλι στην προμήθεια S. Επομένως η προμήθεια S δεν μπορεί να καλύψει τις ανάγκες της εργασίας Τ και τις ανάγκες της εργασίας Τ' ταυτόχρονα. Επομένως το σύστημα μας είναι μη-χρονοπρογραμματίσιμο. Η προμήθεια S και οι εργασίες Τ και Τ' συμβολίζονται ως εξής:

$S \stackrel{def}{=} \{(\bar{r}1,1,2), (\bar{r}2,1,3)\} : S$

$T \stackrel{def}{=} \{(r1,1,1), (r2,1,1)\} : NIL \qquad T' \stackrel{def}{=} \{(r1,1,2), (r2,1,2)\} : NIL$

Παρατηρούμε ότι το σύστημα μεταβάσεων που ακολουθεί αποτελείται από τέσσερις πιθανές μεταβάσεις. Όλες οι πιθανές μεταβάσεις καταλήγουν στον τερματισμό των εργασιών Τ και Τ' και η προμήθεια S μπορεί να χορηγεί τις διαθέσιμές της ποσότητες όπως έχουν ήδη αναφερθεί, συνεχώς.



Σχήμα 3.2 Σύστημα μεταβάσεων παραδείγματος 3.5

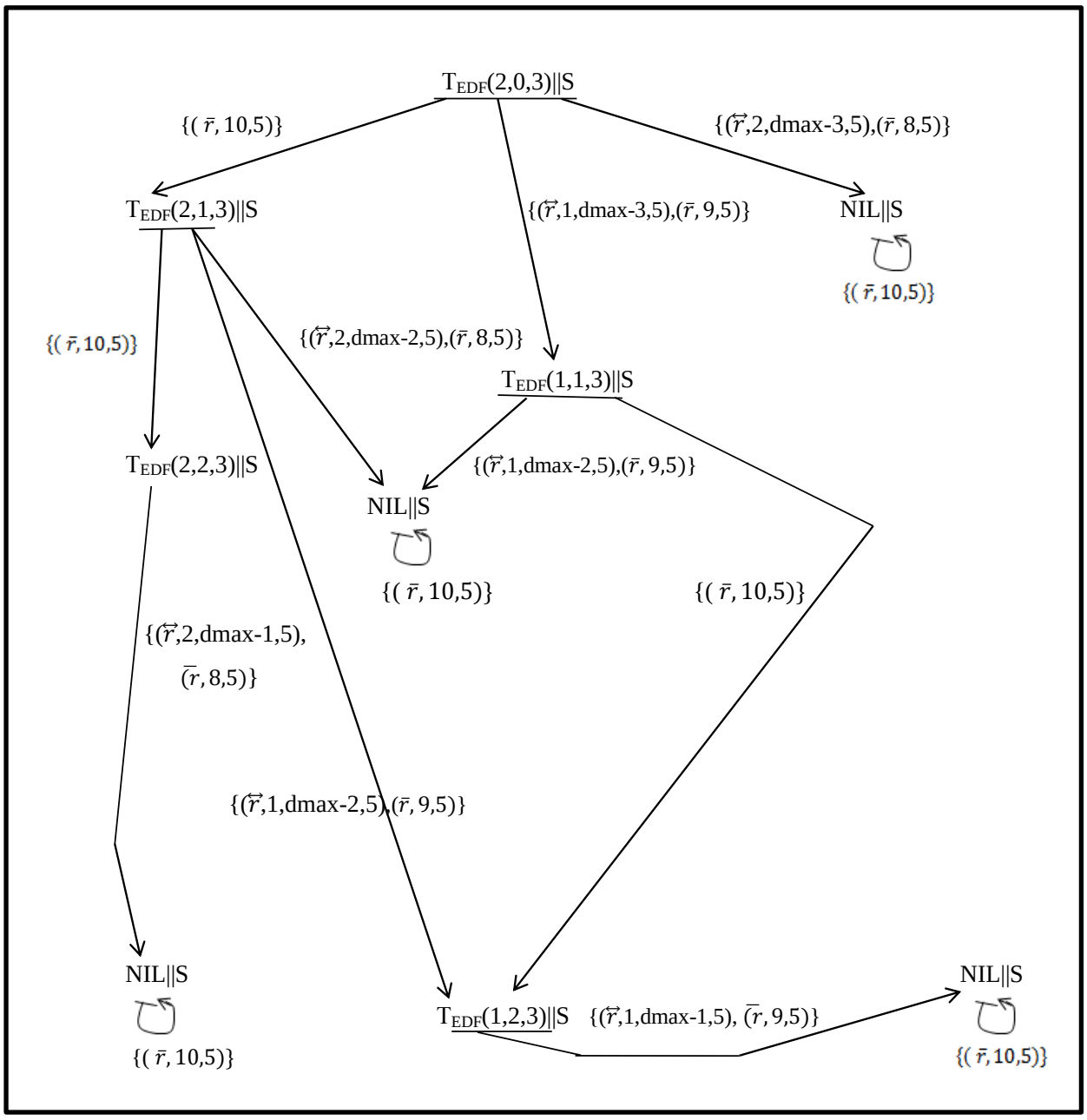
Παράδειγμα 3.6

Στο σύστημα αυτό έχουμε την παράλληλη σύνθεση μιας εργασίας T και μιας προμήθειας S (Βλέπε Σχήμα 3.3). Η εργασία T ζητά 2 μονάδες ενέργειας από τον πόρο r, έχει προθεσμία 3 μονάδες χρόνου και προτεραιότητα 5 και μετά τερματίζει και ορίζεται σύμφωνα με την πολιτική EDF όπως στο υποκεφάλαιο 3.4.1, δηλαδή θεωρούμε ότι $T=T_{EDF}(2,0,3)$. Η προμήθεια S προσφέρει 10 μονάδες ενέργειας από τον πόρο r, ανά μονάδα χρόνου έναντι του ποσού των €5 ανά μονάδα ενέργειας. Ακολούθως η προμήθεια S εξελίσσεται πάλι στην προμήθεια S. Επομένως η προμήθεια S μπορεί να καλύψει τις ανάγκες της εργασίας T, δίχως να χάσει η εργασία T την προθεσμία της. Ακολουθούμε την πολιτική EDF. Η προμήθεια S και η εργασία T συμβολίζονται ως εξής:

$S \stackrel{def}{\equiv} \{(\bar{r}, 10,5)\} : S \qquad T_{EDF}(2,0,3)$

Παρατηρούμε ότι το σύστημα μεταβάσεων που ακολουθεί αποτελείται αρχικά από τρεις πιθανές μεταβάσεις. Ακολούθως η πρώτη πιθανή μετάβαση εξελίσσεται σε τρεις νέες πιθανές μεταβάσεις, η δεύτερη σε δύο και γενικά δημιουργούνται οι μεταβάσεις όπως φαίνονται στο ακόλουθο σχήμα. Η τρίτη πιθανή μετάβαση καθώς και όλες οι άλλες τερματικές μεταβάσεις

οδηγούν σε τερματισμό της εργασίας $T_{EDF}(2,0,3)$ και ταυτόχρονα η προμήθεια S μπορεί να χορηγεί τις διαθέσιμές της ποσότητες όπως έχουν ήδη αναφερθεί, συνεχώς.



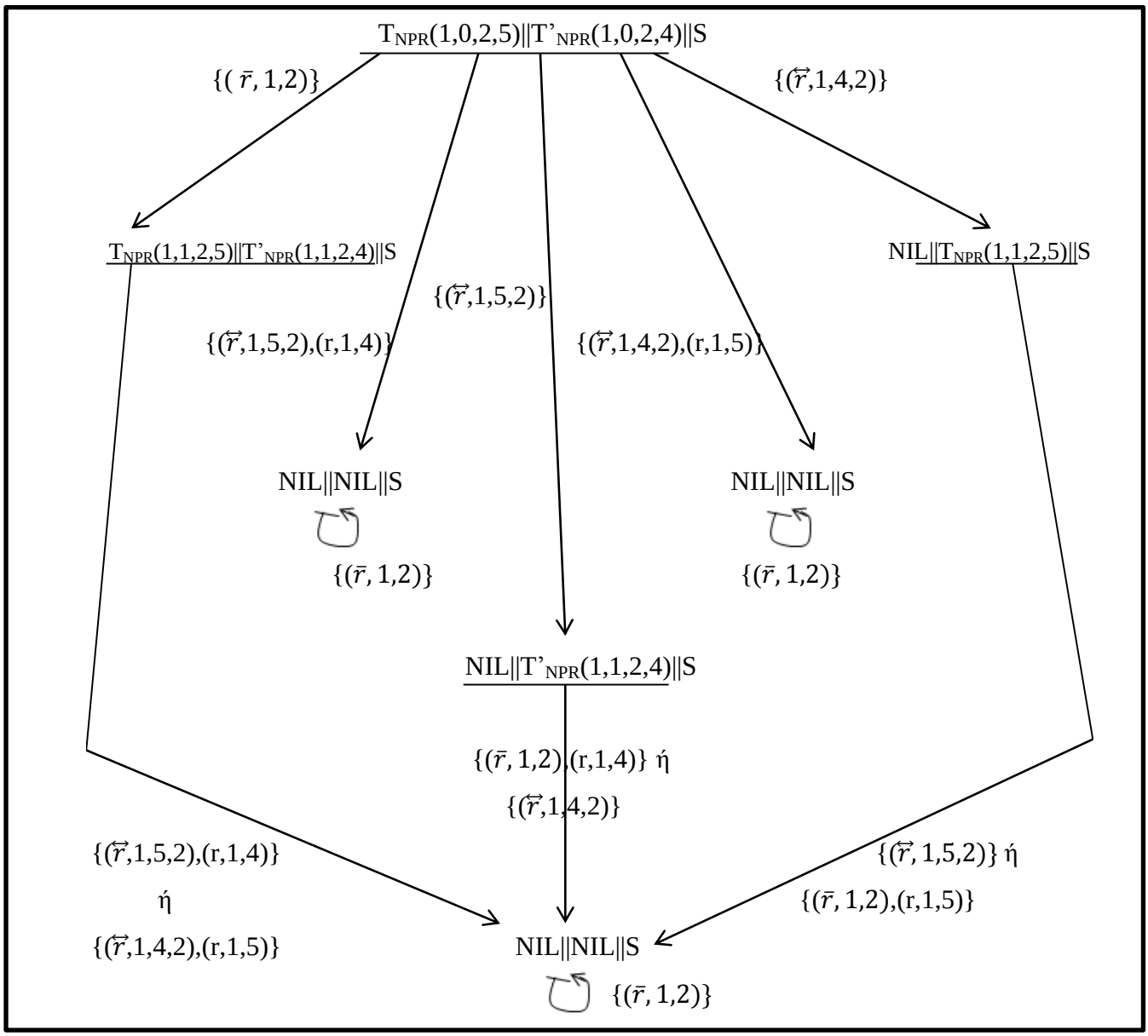
Σχήμα 3.3 Σύστημα μεταβάσεων παραδείγματος 3.6

Παράδειγμα 3.7

Στο σύστημα αυτό έχουμε την παράλληλη σύνθεση δύο εργασιών, της Τ και της Τ' και μιας προμήθειας S (Βλέπε Σχήμα 3.4). Η εργασία Τ ζητά 1 μονάδα ενέργειας από τον πόρο r, έχει προθεσμία 2 μονάδες χρόνου και προτεραιότητα 5 και μετά τερματίζει και ορίζεται σύμφωνα με τη μη διακοπτόμενη πολιτική όπως στο υποκεφάλαιο 3.4.2, δηλαδή θεωρούμε ότι $T=T_{NPR}(1,0,2,5)$. Η εργασία Τ' ζητά 1 μονάδα ενέργειας από τον πόρο r, έχει προθεσμία 2 μονάδες χρόνου και προτεραιότητα 4 και μετά τερματίζει και ορίζεται σύμφωνα με τη μη διακοπτόμενη πολιτική όπως στο υποκεφάλαιο 3.4.2, δηλαδή θεωρούμε ότι $T'=T'_{NPR}(1,0,2,4)$. Η προμήθεια S προσφέρει 1 μονάδα ενέργειας από τον πόρο r, ανά μονάδα χρόνου έναντι του ποσού των €2 ανά μονάδα ενέργειας. Ακολούθως η προμήθεια S εξελίσσεται πάλι στην προμήθεια S. Επομένως η προμήθεια S μπορεί να καλύψει τις ανάγκες της εργασίας Τ και τις ανάγκες της εργασίας Τ'. Ακολουθούμε τη μη-διακοπτόμενη πολιτική. Η προμήθεια S και οι εργασίες Τ και Τ' συμβολίζονται ως εξής:

$S \stackrel{\text{def}}{=} \{(\bar{r}, 1,2)\} : S \quad T_{NPR}(1,0,2,5) \quad T'_{NPR}(1,0,2,4)$

Παρατηρούμε ότι το σύστημα μεταβάσεων που ακολουθεί αποτελείται αρχικά από πέντε πιθανές μεταβάσεις. Ακολούθως κάποιες μεταβάσεις εξελίσσονται σε νέες μεταβάσεις όπως φαίνονται στο ακόλουθο σχήμα. Τελικά όλες οι πιθανές μεταβάσεις θα καταλήξουν σε τερματική μετάβαση όπου οι εργασίες $T_{NPR}(1,0,2,5)$ και $T'_{NPR}(1,0,2,4)$ τερματίζουν και ταυτόχρονα η προμήθεια S μπορεί να χορηγεί τις διαθέσιμές της ποσότητες όπως έχουν ήδη αναφερθεί, συνεχώς.



Σχήμα 3.4 Σύστημα μεταβάσεων παραδείγματος 3.7

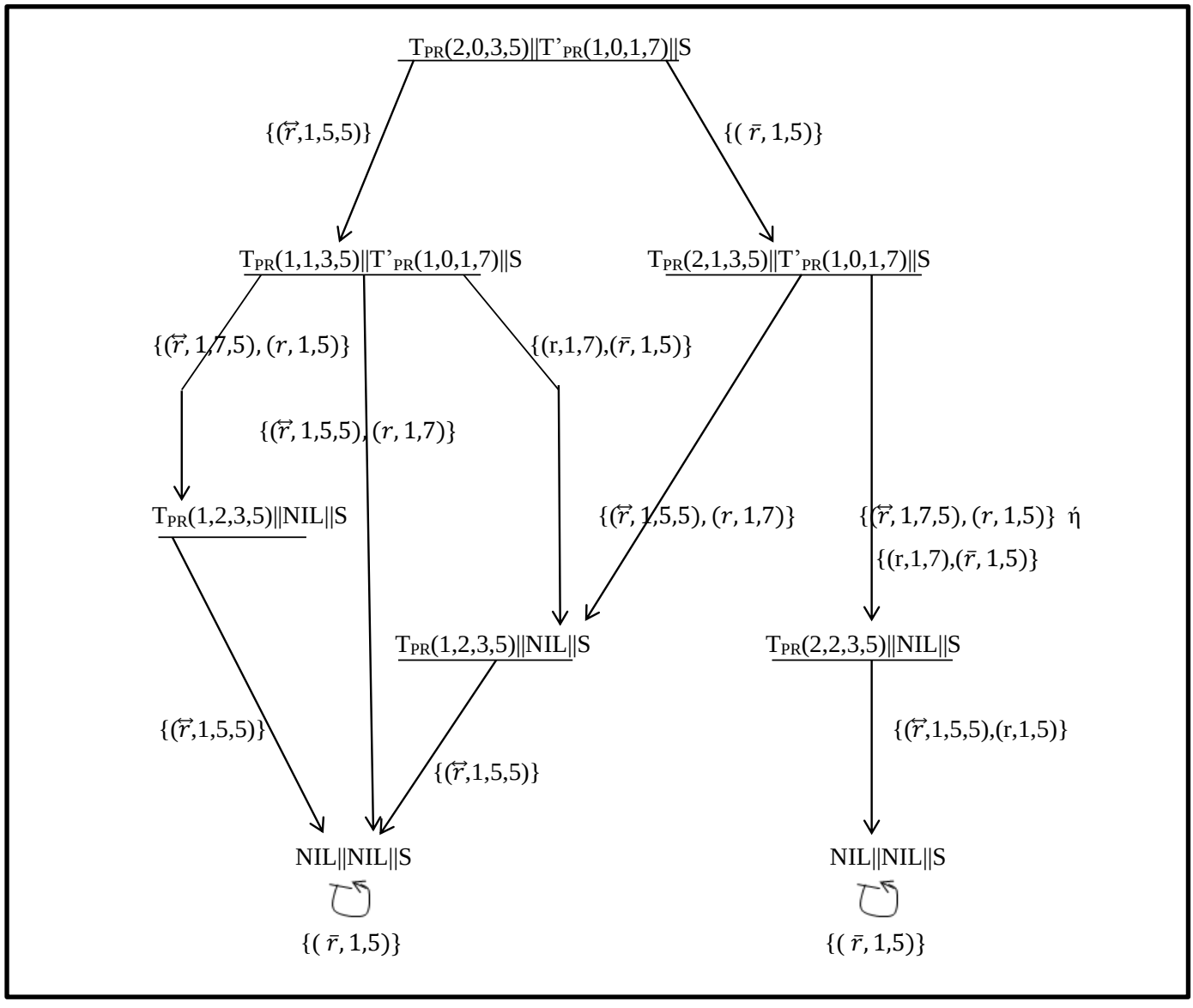
Παράδειγμα 3.8

Στο σύστημα αυτό έχουμε την παράλληλη σύνθεση δύο εργασιών, της T και της T' και μιας προμήθειας S (Βλέπε Σχήμα 3.5). Η εργασία T ζητά 2 μονάδες ενέργειας από τον πόρο r, έχει προθεσμία 3 μονάδες χρόνου και προτεραιότητα 5 και μετά τερματίζει και ορίζεται σύμφωνα με τη διακοπτόμενη πολιτική όπως στο υποκεφάλαιο 3.4.3, δηλαδή θεωρούμε ότι $T_{PR}(2,0,3,5)$. Η εργασία T' ζητά 1 μονάδα ενέργειας από τον πόρο r, έχει προθεσμία 1 μονάδα χρόνου και εμφανίζεται στο σύστημα μας μετά από 1 μονάδα χρόνου. Έχει προτεραιότητα 7 και μετά τερματίζει και ορίζεται σύμφωνα με τη διακοπτόμενη πολιτική όπως στο υποκεφάλαιο 3.4.3, δηλαδή θεωρούμε ότι $\emptyset^{1}$:$T'_{PR}(1,0,1,7)$. Η προμήθεια S προσφέρει 1 μονάδα ενέργειας από τον πόρο r, ανά μονάδα χρόνου έναντι του ποσού των €5 ανά μονάδα ενέργειας. Ακολούθως η προμήθεια S εξελίσσεται πάλι στην προμήθεια S.

Επομένως η προμήθεια S μπορεί να καλύψει τις ανάγκες της εργασίας T και τις ανάγκες της εργασίας T'. Ακολουθούμε την διακοπτόμενη πολιτική. Η προμήθεια S και οι εργασίες T και T' συμβολίζονται ως εξής:

$S \stackrel{def}{\equiv} \{(\bar{r}, 1,5)\} : S \quad T_{PR}(2,0,3,5) \quad \emptyset^1 : T'_{PR}(1,0,1,7)$

Παρατηρούμε ότι το σύστημα μεταβάσεων που ακολουθεί αποτελείται αρχικά από δύο πιθανές μεταβάσεις. Ακολούθως κάποιες μεταβάσεις εξελίσσονται σε νέες μεταβάσεις όπως φαίνονται στο ακόλουθο σχήμα. Τελικά όλες οι πιθανές μεταβάσεις θα καταλήξουν σε τερματική μετάβαση όπου οι εργασίες $T_{PR}(2,0,3,5)$ και $T'_{PR}(1,0,1,7)$ τερματίζουν και ταυτόχρονα η προμήθεια S μπορεί να χορηγεί τις διαθέσιμές της ποσότητες όπως έχουν ήδη αναφερθεί, συνεχώς.



Σχήμα 3.5 Σύστημα μεταβάσεων παραδείγματος 3.8

3.6 Σχέση προτιμήσεων

Αν παρατηρήσουμε το Παράδειγμα 3.6 βλέπουμε ότι επιθυμούμε το σύστημα μεταβάσεων το οποίο ικανοποιεί τις απαιτήσεις της εργασίας και ταυτόχρονα εντός της επιθυμητής προθεσμίας.

Αυτό που θέλουμε να πετύχουμε είναι βάσει των κανόνων των σχέσεων προτιμήσεων να αποκλείονται οι δράσεις οι οποίες δεν μας οδηγούν στο επιθυμητό σύστημα μεταβάσεων. Για αυτό λοιπόν τον λόγο ορίζουμε ακολούθως τη σχέση προτιμήσεων ώστε να αποφεύγουμε τις ανεπιθύμητες μεταβάσεις.

Πριν δώσουμε τον ορισμό μας, ας υποθέσουμε αρχικά ότι τα α και β είναι οι δράσεις που δημιουργήθηκαν από την παράλληλη εκτέλεση του ιδίου συστήματος. Θεωρούμε ότι αν ισχύει η σχέση προτιμήσεων $\alpha \prec \beta$ τότε η δράση β θα πρέπει να αποτρέψει την εμφάνιση της δράσης α.

Αρχικά με τον συμβολισμό ρ παρουσιάζονται τα στοιχεία του Act_r, όπου Act_r είναι το σύνολο που περιέχει μόνο αιτήσεις πόρων. Επίσης ορίζουμε ως α^{μ} το σύνολο των πόρων με την μορφή που εμφανίζονται στη δράση α. Παραδείγματος χάρη $\{(\overleftarrow{r1}, 2,3,5), (\overline{r2}, 1,3), (r3,2,4)\}^{\mu} = \{\overleftarrow{r1}, \overline{r2}, r3\}$

Ορισμός 3.3

Έστω α και β δράσεις. Ορίζουμε τη σχέση προτιμήσεων $\prec$ ως τη σχέση όπου $\alpha \prec \beta$ αν ικανοποιείται ένα από τα ακόλουθα σημεία:

1) Αν σε δύο δράσεις α και β, η α και η β έχουν τις ίδιες χορηγίες και καταναλώσεις και η α έχει ακόμη ανικανοποίητες αιτήσεις ενώ η β όχι, δηλαδή $\alpha = \cup_{i \in I}\{(r_i, e_i, p_i)\} \cup \beta$ τότε $\alpha \prec \beta$

2) Αν σε δύο δράσεις α και β όπου η α και η β είναι ακριβώς ίδιες όσον αφορά την προτεραιότητα και τους πόρους που εμπλέκουν για αίτηση, χορηγία και κατανάλωση, τότε αν η β καταναλώνει λιγότερους πόρους από την α, τότε α≺β

$$a = \bigcup_{i \in I} \{(r_i, e_i, p_i)\} \bigcup_{j \in J} \{(\overline{r_j}, c_j)\} \bigcup_{k \in K} \{(\overleftrightarrow{r_k}, e_k, p_k)\},$$

$$\beta = \bigcup_{i \in I} \{(r_i, e_i, q_i)\} \bigcup_{j \in J} \{(\overline{r_j}, c_j)\} \bigcup_{k \in K} \{(\overleftrightarrow{r_k}, e_k, q_k)\},$$ και $\forall\, k \in K,\ q_k \le p_k$

3) Αν σε δύο δράσεις α και β, η α και η β είναι ακριβώς ίδιες όσον αφορά στους πόρους που εμπλέκουν για αίτηση, χορηγία και κατανάλωση, και η δράση β δίνει μεγαλύτερη ή ίση προτεραιότητα σε όλες τις καταναλώσεις πόρων και υπάρχει τουλάχιστον μια προτεραιότητα στις καταναλώσεις της δράσης β που να είναι αυστηρά μεγαλύτερη από της δράσης α τότε α≺β

$$a = \bigcup_{i \in I} \{(r_i, e_i, p_i)\} \bigcup_{j \in J} \{(\overline{r_j}, c_j)\} \bigcup_{k \in K} \{(\overleftrightarrow{r_k}, e_k, p_k)\},$$

$$\beta = \bigcup_{i \in I} \{(r_i, e_i, q_i)\} \bigcup_{j \in J} \{(\overline{r_j}, c_j)\} \bigcup_{k \in K} \{(\overleftrightarrow{r_k}, e_k, q_k)\} \quad \forall i \in I\ p_i \le q_i, \forall k \in K\ p_k \le q_k$$

και $\exists l \in I \cup K$ τέτοιο ώστε $p_l < q_l$

Ακολουθούν οι επεξηγήσεις των σημείων.

Σημείο (1): Με αυτό το σημείο θέλουμε η κάθε εργασία να επιλέγει τις δράσεις που ικανοποιούν άμεσα τις αιτήσεις της, εάν υπάρχουν τέτοιες επιλογές.

Σημείο (2): Αυτό το σημείο μας εξυπηρετεί στο γεγονός ότι αν επιλέγουμε τις λιγότερες αιτήσεις ή όταν έχουμε τον ίδιο αριθμό αιτήσεων και επιλέγουμε εκείνη με τη μικρότερη κατανάλωση, τότε θα μπορούμε να ικανοποιήσουμε όσο το δυνατό περισσότερους πελάτες και έτσι με αυτό τον τρόπο όσο περισσότερους ικανοποιημένους πελάτες έχουμε, τόσο μεγαλύτερη είναι η πιθανότητα να αποκτήσουμε πιστούς πελάτες και κατ' επέκταση να έχουμε μεγαλύτερο κέρδος.

Σημείο (3): Τέλος αυτό το σημείο χειρίζεται τις περιπτώσεις προτεραιότητας του συστήματος. Δηλαδή αν δύο δράσεις περιέχουν τους ίδιους πόρους και βρίσκονται στην ίδια

μορφή (αίτηση, χορηγία, κατανάλωση) τότε η δράση β υπερνικά την δράση α αν για κάθε κατανάλωση πόρου έχει μεγαλύτερη ορισμένη προτεραιότητα.

Ακολούθως θα ορίσουμε την σχέση μετάβασης με προτεραιότητα.

Ορισμός 3.4

Η σχέση μετάβασης $\longrightarrow$, καθορίζεται ως ακολούθως: $P \xrightarrow{\alpha} P'$ αν και μόνο αν

1) $P \overset{\alpha}{\mapsto} P'$ είναι μια σχέση μετάβασης χωρίς προτεραιότητα και
2) Δεν υπάρχει σχέση μετάβασης χωρίς προτεραιότητα $P \overset{\beta}{\mapsto} P''$ τέτοια ώστε $\alpha \prec \beta$

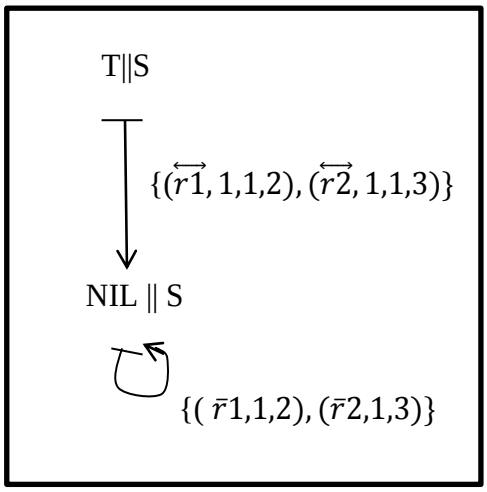
Ακολούθως θα δώσουμε ξανά τα παραδείγματα 3.4-3.8 απλά τώρα θα εφαρμόσουμε στα συστήματα μεταβάσεων την σχέση προτιμήσεων, ώστε να δούμε ποιες μεταβάσεις παραμένουν βάσει του ορισμού της σχέσης προτίμησης.

Παράδειγμα 3.9

Το σύστημα μεταβάσεων του παραδείγματος 3.4 μετασχηματίστηκε ως εξής: Αφού αυτή είναι η μοναδική μετάβαση, η σχέση προτιμήσεως ισχύει τετριμμένα. Επομένως αυτό είναι το επιθυμητό σύστημα μεταβάσεων (Βλέπε Σχήμα 3.6). Παρατηρούμε δηλαδή ότι η προμήθεια S μπορεί να ικανοποιήσει τις ανάγκες της εργασίας T.

$S \overset{\text{def}}{\equiv} \{(\bar{r}1,1,2), (\bar{r}2,1,3)\} : S$

$T \overset{\text{def}}{\equiv} \{(r1,1,1), (r2,1,1)\} : NIL$



Σχήμα 3.6 Σύστημα μεταβάσεων παραδείγματος 3.9 με χρήση προτεραιότητας

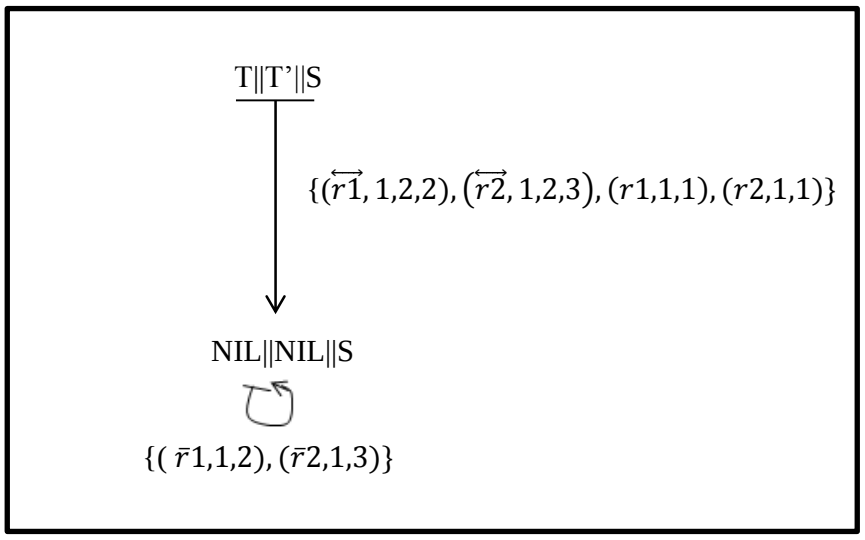
Παράδειγμα 3.10

Το σύστημα μεταβάσεων του παραδείγματος 3.5 μετασχηματίστηκε ως εξής: Αυτή είναι η επιθυμητή μετάβαση, βάσει του 3ου σημείου του ορισμού της σχέσης προτιμήσεων. Δηλαδή αν σε δύο δράσεις α και β, η α και η β έχουν τις ίδιες αιτήσεις, χορηγίες και καταναλώσεις, και η δράση β δίνει μεγαλύτερη ή ίση προτεραιότητα σε όλες τις καταναλώσεις πόρων και υπάρχει τουλάχιστον μια προτεραιότητα στις καταναλώσεις της δράσης β που να είναι αυστηρά μεγαλύτερη από της δράσης α τότε $\alpha \prec \beta$. Για αυτό το λόγο αυτό είναι το επιθυμητό σύστημα μεταβάσεων (Βλέπε Σχήμα 3.7). Παρατηρούμε δηλαδή ότι η προμήθεια S δεν μπορεί να ικανοποιήσει τις ανάγκες των εργασιών T και T'. Ουσιαστικά αυτό το σύστημα μεταβάσεων μας δείχνει ότι το σύστημα μας, δεν είναι χρονοπρογραμματίσιμο, όπως θα εξηγήσουμε εκτενώς στο κεφάλαιο 4.

S $\stackrel{\text{def}}{=}$ $\{(\bar{r}1,1,2),(\bar{r}2,1,3)\}$: S

T $\stackrel{\text{def}}{=}$ $\{(r1,1,1),(r2,1,1)\}$: NIL

T' $\stackrel{\text{def}}{=}$ $\{(r1,1,2),(r2,1,2)\}$: NIL



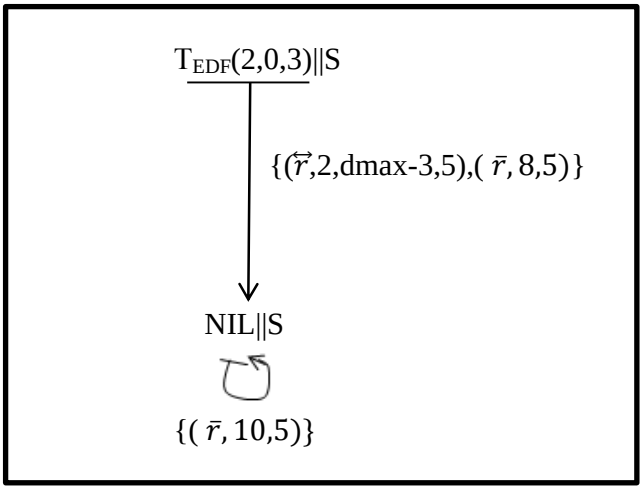
Σχήμα 3.7 Σύστημα μεταβάσεων παραδείγματος 3.10 με χρήση προτεραιότητας

Παράδειγμα 3.11

Το σύστημα μεταβάσεων του παραδείγματος 3.6 μετασχηματίστηκε ως εξής: Αυτή είναι η επιθυμητή μετάβαση, βάσει του $1^{ου}$ σημείου του ορισμού της σχέσης προτιμήσεων. Δηλαδή σε δύο δράσεις α και β, η α και η β έχουν τις ίδιες χορηγίες και καταναλώσεις και η α έχει ακόμη ανικανοποίητες αιτήσεις ενώ η β όχι τότε $\alpha \prec \beta$. Για αυτό το λόγο, αυτό είναι το επιθυμητό σύστημα μεταβάσεων (Βλέπε Σχήμα 3.8). Παρατηρούμε δηλαδή ότι η προμήθεια S μπορεί να ικανοποιήσει τις ανάγκες της εργασίας $T_{EDF}(2,0,3)$.

$S \stackrel{def}{=} \{(\bar{r}, 10,5)\} : S$

$T_{EDF}(2,0,3)$



Σχήμα 3.8 Σύστημα μεταβάσεων παραδείγματος 3.11 με χρήση προτεραιότητας

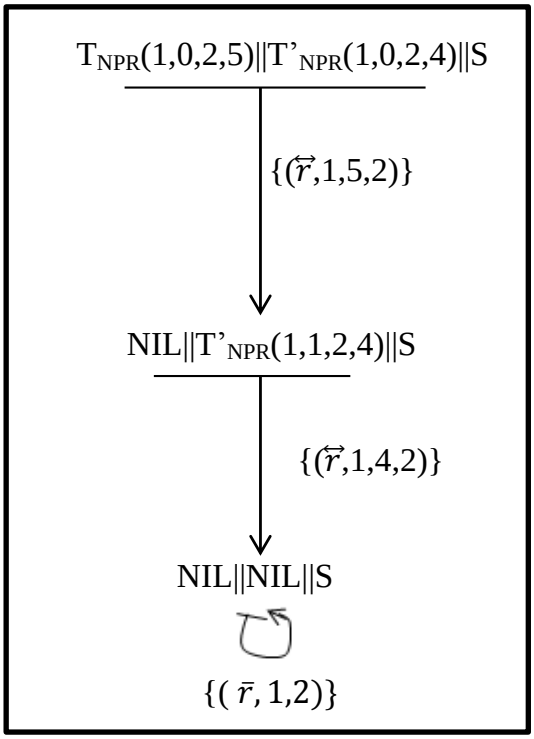
Παράδειγμα 3.12

Το σύστημα μεταβάσεων του παραδείγματος 3.7 μετασχηματίστηκε ως εξής: Αυτή είναι η επιθυμητή μετάβαση, βάσει του 3^{ου} και του 1^{ου} σημείου του ορισμού της σχέσης προτιμήσεων. Δηλαδή βάσει του 3^{ου}, αν σε δύο δράσεις α και β, η α και η β έχουν τις ίδιες αιτήσεις, χορηγίες και καταναλώσεις, και η δράση β δίνει μεγαλύτερη ή ίση προτεραιότητα σε όλες τις καταναλώσεις πόρων και υπάρχει τουλάχιστον μια προτεραιότητα στις καταναλώσεις της δράσης β που να είναι αυστηρά μεγαλύτερη από της δράσης α τότε α≺β, για αυτό οδηγηθήκαμε στη μετάβαση $\{(\vec{r},1,5,2)\}$ και βάσει του 1^{ου} , αν σε δύο δράσεις α και β, η α και η β έχουν τις ίδιες χορηγίες και καταναλώσεις και η α έχει ακόμη ανικανοποίητες αιτήσεις ενώ η β όχι, τότε α≺β για αυτό το λόγο οδηγηθήκαμε στη μετάβαση $\{(\vec{r},1,4,2)\}$. Για αυτούς τους λόγους αυτό είναι το επιθυμητό σύστημα μεταβάσεων (Βλέπε Σχήμα 3.9). Παρατηρούμε δηλαδή ότι η προμήθεια S μπορεί να ικανοποιήσει τις ανάγκες των εργασιών $T_{NPR}(1,0,2,5)$ και $T'_{NPR}(1,0,2,4)$.

$S \stackrel{def}{=} \{(\bar{r},1,2)\} : S$

$T_{NPR}(1,0,2,5)$

$T'_{NPR}(1,0,2,4)$

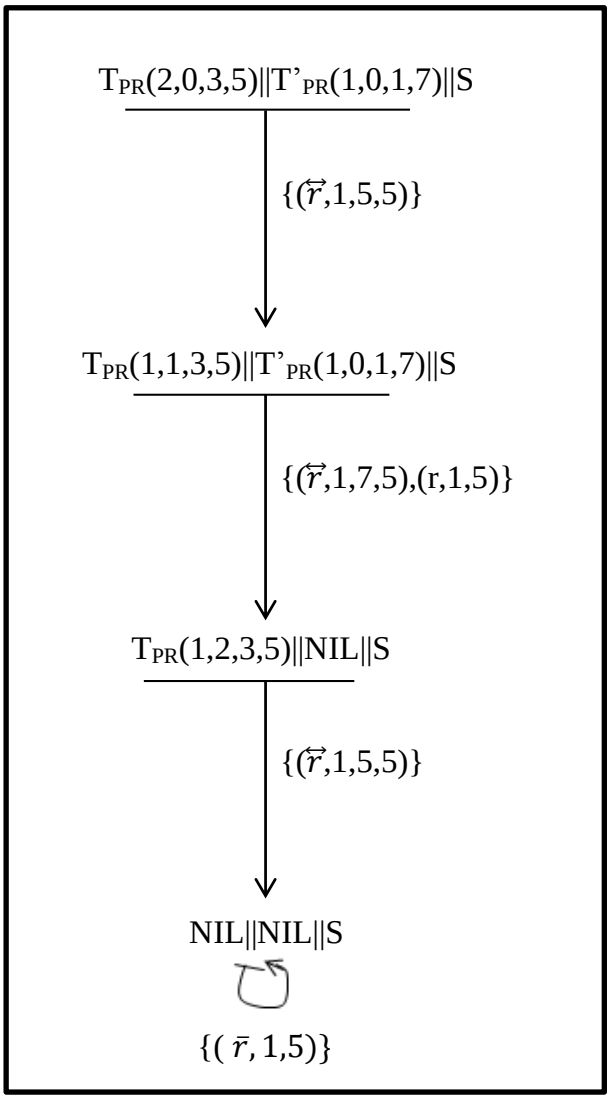


Σχήμα 3.9 Σύστημα μεταβάσεων παραδείγματος 3.12 με χρήση προτεραιότητας

Παράδειγμα 3.13

Το σύστημα μεταβάσεων του παραδείγματος 3.8 μετασχηματίστηκε ως εξής: Αυτή είναι η επιθυμητή μετάβαση, βάσει του 3ου σημείου του ορισμού της σχέσης προτιμήσεων. Δηλαδή αν σε δύο δράσεις α και β, η α και η β έχουν τις ίδιες αιτήσεις, χορηγίες και καταναλώσεις, και η δράση β δίνει μεγαλύτερη ή ίση προτεραιότητα σε όλες τις καταναλώσεις πόρων και υπάρχει τουλάχιστον μια προτεραιότητα στις καταναλώσεις της δράσης β που να είναι αυστηρά μεγαλύτερη από της δράσης α τότε α≺β. Για αυτό το λόγο οδηγηθήκαμε αρχικά στη μετάβαση $\{(\vec{r},1,5,5)\}$, έπειτα στη μετάβαση $\{(\vec{r},1,7,5),(r,1,5)\}$ και τέλος στην μετάβαση $\{(\vec{r},1,5,5)\}$. Για αυτούς τους λόγους αυτό είναι το επιθυμητό σύστημα μεταβάσεων (Βλέπε Σχήμα 3.10). Παρατηρούμε δηλαδή ότι η προμήθεια S μπορεί να ικανοποιήσει τις ανάγκες των εργασιών $T_{PR}(2,0,3,5)$ και $T'_{PR}(1,0,1,7)$.

$S \stackrel{def}{=} \{(\bar{r},1,5)\} : S$ $\quad T_{PR}(2,0,3,5)$ $\quad \emptyset^1 : T'_{PR}(1,0,1,7)$



Σχήμα 3.10 Σύστημα μεταβάσεων παραδείγματος 3.13 με χρήση προτεραιότητας

3.7 Αξιολόγηση βάσει του υπολογιστικού νέφους

3.7.1 Αλγόριθμος του υπολογιστικού νέφους

Ακολούθως θα δώσουμε ένα αλγόριθμο που έχει εφαρμογές στο υπολογιστικό νέφος. Ο συγκεκριμένος αλγόριθμος όπως θα τον παρουσιάσουμε αποτελεί μια απλοποιημένη μορφή του αλγορίθμου που παρουσιάζεται στο άρθρο [12].

Αφού δοθεί ο ορισμός του προβλήματος, στη συνέχεια δίνονται η περιγραφή του αλγορίθμου και η μοντελοποίηση βάσει διάφορων πολιτικών χρονοδρομολόγησης.

Ορισμός του προβλήματος Όπως έχουμε ήδη αναφέρει ο αλγόριθμός μας αποτελεί μια απλοποιημένη μορφή του αλγορίθμου που παρουσιάζεται στο άρθρο [12]. Ο συγκεκριμένος αλγόριθμος του άρθρου αυτού προέρχεται από την έννοια του MapReduce [5]. Με τον όρο MapReduce αναφερόμαστε σε μια μέθοδο αλγοριθμικής προσέγγισης η οποία έχει υποβοηθητικό ρόλο π.χ. μια βιβλιοθήκη υλοποιημένη σε διάφορες γλώσσες προγραμματισμού. Επίσης η μέθοδος του MapReduce μας προσφέρει παράλληλη επεξεργασία δεδομένων πάνω σε μεγάλου όγκου δεδομένα, με σκοπό να γίνει η επεξεργασία αυτών των δεδομένων όσο το δυνατό γρηγορότερα. Επίσης τις πλείστες φορές τα δεδομένα προς επεξεργασία είναι ανεξάρτητα μεταξύ τους αλλά και στη περίπτωση που υπάρχει εξάρτηση μεταξύ των δεδομένων, παρέχονται δυνατότητες συντονισμού από το ίδιο το MapReduce, και έτσι δεν δημιουργείται πρόβλημα. Επομένως ο αλγόριθμος του άρθρου που μελετήσαμε, στηρίζεται σε σύνδεση πραγματικού χρόνου χρονοδρομολόγησης του MapReduce των εργασιών οι οποίες εκτελούνται από το Hadoop [6]. Με την έννοια του Hadoop αναφερόμαστε σε ένα πλαίσιο λογισμικού ανοικτού κώδικα, δηλαδή μια πλατφόρμα ενδιάμεσου λογισμικού, για αποθήκευση και μεγάλης κλίμακας επεξεργασίας συνόλων δεδομένων πάνω σε συστάδες (clusters) εμπορεύσιμου υλικού. Επίσης αποτελεί εργαλείο υλοποίησης του MapReduce, δηλαδή το MapReduce είναι το πρότυπο και το Hadoop μια υλοποίησή του. Το Hadoop ασχολείται με τη διαχείριση των δεδομένων (data), δηλαδή σε ποιον σκλάβο (Slave) θα ανατεθεί το κάθε δεδομένο. Για αυτό το λόγο λοιπόν ο αλγόριθμος του άρθρου [12] θεωρεί ότι αυτό το ζευγάρωμα έχει ήδη γίνει. Έτσι και εμείς στο δικό μας απλοποιημένο αλγόριθμο θεωρούμε ότι το ζευγάρωμα μεταξύ Jobs και Slaves ή LocalJobs και Slaves έχει ήδη γίνει. Ακολουθεί η αντιστοιχεία του αρχικού αλγορίθμου με τον δικό μας αλγόριθμο.

Ο συγκεκριμένος αλγόριθμος που έχουμε μελετήσει όπως έχουμε ήδη αναφέρει αποτελεί μια απλοποιημένη μορφή του πιο πάνω αλγορίθμου. Συγκεκριμένα στο σύστημα υπάρχουν οι δουλειές (Jobs) οι οποίες ζητούν κάποιες ποσότητες από πόρους και έχουν προτεραιότητες και προθεσμίες. Επίσης στο σύστημα υπάρχουν οι προμήθειες (Slaves) οι οποίες χορηγούν κάποιες ποσότητες πόρων, συγκεκριμένα προσφέρουν κάποιο δεδομένο αριθμό από πόρους ανά μονάδα χρόνου, έναντι κάποιου χρηματικού ποσού (σε €). Ακόμη οι προμήθειες πρώτα πρέπει να ικανοποιήσουν τις τοπικές δουλειές (LocalJobs) που έχουν πρωτού να ικανοποιήσουν τις υπόλοιπες δουλειές. Οι τοπικές δουλειές διαφέρουν από τις υπόλοιπες δουλειές στην προτεραιότητα, δηλαδή έχουν μεγαλύτερη προτεραιότητα ώστε να ικανοποιηθούν πρώτα αυτές και μετά οι υπόλοιπες. Σημειώνουμε ότι μεγαλύτερος αριθμός υποδηλώνει και μεγαλύτερη προτεραιότητα. Επιπρόσθετα κάθε δουλειά μπορεί να αποτελείται από μια εως πολλές εργασίες (Tasks) . Η κάθε εργασία ανάλογα με το αν είναι τοπική η όχι στέλνεται περιοδικά στο σύστημα από την αντίστοιχη οντότητα (Activator ή LocalActivator).

3.7.2 Περιγραφή αλγορίθμου

Ο αλγόριθμός μας περιγράφει λοιπόν το σύστημα το οποίο αποτελείται όπως έχουμε ήδη αναφέρει από δουλειές, τοπικές ή/και μη και προμήθειες. Οι δουλειές, τοπικές ή/και μη, χρονοδρομολογούνται βάσει του χρονοδρομολογητή με σκοπό να ικανοποιηθούν τα αιτήματα των τοπικών δουλειών ή/και μη και οι προμήθειες να έχουν τη μέγιστη δυνατή αξιοποίηση. Δηλαδή δεν θέλουμε να έχουμε προμήθειες που μένουν ανεκμετάλλευτες με το να αδρανούν.

Τέλος ανάλογα με την πολιτική χρονοδρομολόγησης που εφαρμόζουμε υφίστανται κάποιες τροποποιήσεις. Οι τροποποιήσεις αυτές εμφανίζονται σε κάθε αλγόριθμο ανάλογα με την πολιτική χρονοδρομολόγησης, στις συναρτήσεις BreakTiesLJ (για τις τοπικές δουλειές) και BreakTiesJ (για τις δουλειές). Αυτές οι τροποποιήσεις θα επεξηγηθούν αργότερα στην μοντελοποίηση με περισσότερη λεπτομέρεια.

Ακολουθούν τα βήματα του αλγορίθμου:

1. Αρχικά έρχονται Jobs και LocalJobs στο σύστημα
2. Κάθε φορά που έρχεται ένα Job ή LocalJob στο σύστημα, άφου πρώτα γίνει το αντίστοιχο ζευγάρωμα μεταξύ Slave και Job ή LocalJob, κάνουμε κάποιους ελέγχους για τους Slaves
3. Ελέγχουμε για κάθε Slave αν έχει LocalJobs και αν έχει τότε επιλέγει από όλα τα LocalJobs που του αντιστοιχούν εκείνο που έχει τη μεγαλύτερη προτεραιότητα. Η προτεραιότητα εξαρτάται κάθε φορά από την πολιτική χρονοδρομολόγησης
4. Εάν δεν έχει LocalJobs τότε επιλέγει από όλα τα Jobs που του αντιστοιχούν εκείνο που έχει τη μεγαλύτερη προτεραιότητα. Η προτεραιότητα εξαρτάται κάθε φορά από την πολιτική χρονοδρομολόγησης
5. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να μην υπάρχουν άλλα Jobs και LocalJobs στο σύστημα

Ακολουθεί στην επόμενη σελίδα ο ψευδοκώδικας του αλγορίθμου. Ο αλγόριθμος αυτός εφαρμόζεται σε κάθε πολιτική χρονοδρομολόγησης που μοντελοποιήσαμε πιο κάτω. Απλά έχει μια μικρή τροποποίηση κάθε φορά στο πώς γίνεται ο υπολογισμός της προτεραιότητας ανάλογα με την πολιτική χρονοδρομολόγησης που υλοποιούμε. Αυτός ο αλγόριθμος εκτελείται από τον χρονοδρομολογητή, ο οποίος τον στέλνει σε κάθε $Slave_i$,$1 \leq i \leq N$ για να τον εκτελέσει. Επίσης θεωρούμε ότι το J είναι μια λίστα που περιέχει τα Jobs και τα LocalJobs των οποίων τα αιτήματα ακόμη δεν έχουν ικανοποιηθεί.

∀ J ∈ Jobs ∪ LocalJobs do
Κάνε το ζευγάρωμα του J με τον αντίστοιχο Slave

∀ $Slave_i$ όπου $1 \leq i \leq N$
Εάν LocalJobs του $Slave_i \neq \emptyset$ τότε εκτέλεσε
την συνάρτηση BreakTiesLJ (η οποία υπολογίζει την προτεραιότητα του κάθε $LocalJobs_i$) και επέλεξε για εκτέλεση το $LocalJobs_i$ που έχει την μεγαλύτερη προτεραιότητα. (Ο ορισμός της προτεραιότητας εξαρτάται κάθε φορά από την πολιτική χρονοδρομολόγησης που υλοποιείται)

Εάν LocalJobs του $Slave_i = \emptyset$ τότε εκτέλεσε
την συνάρτηση BreakTiesJ (η οποία υπολογίζει την προτεραιότητα του κάθε $Jobs_i$) και επέλεξε για εκτέλεση το $Jobs_i$ που έχει την μεγαλύτερη προτεραιότητα. (Ο ορισμός της προτεραιότητας εξαρτάται κάθε φορά από την πολιτική χρονοδρομολόγησης που υλοποιείται)

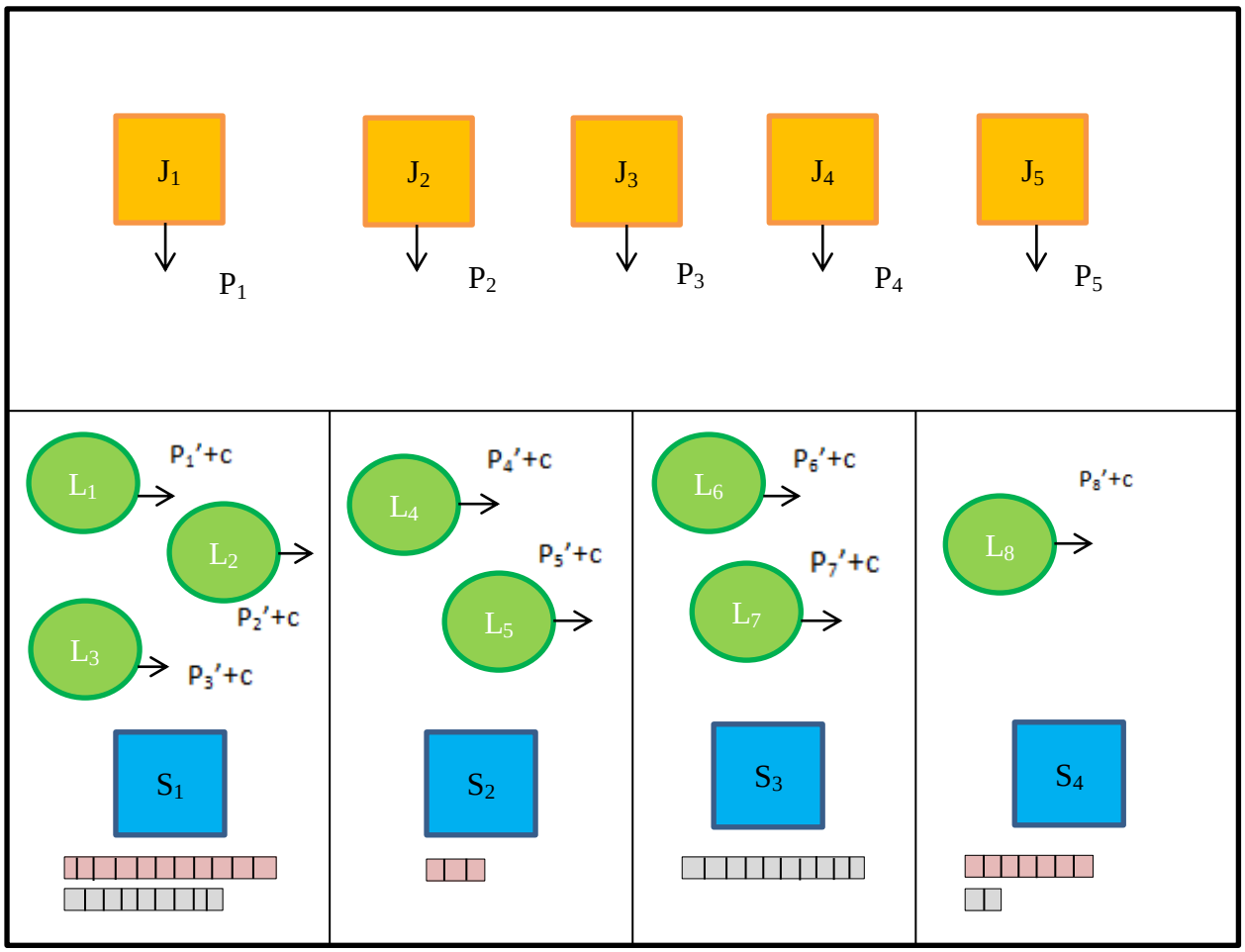
end do

Επεξηγηματικός πίνακας

Ακολούθως δίνουμε για την κάθε οντότητα που υπάρχει στο σύστημά μας μια μικρή περιγραφή, τις παραμέτρους από τις οποίες χαρακτηρίζεται καθώς και τη σχηματική της αναπαράσταση με την οποία θα συμβολίζεται (Βλέπε Σχήμα 3.11).

ΑΑ	**Όνομα Οντότητας**	**Σχηματικά**	**Παράμετροι οντότητας**	**Μικρή περιγραφή**
1	Slaves		$\{(\bar{r}, s, c)\}$ r: όνομα πόρου s: ποσότητα c: κόστος	Οι Slaves αντιστοιχούν στις προμήθειες που υπάρχουν στο σύστημα. Χορηγούν ποσότητες από πόρους έναντι κάποιου χρηματικού ποσού
2	Jobs		{(r,e,p)} r: όνομα πόρου e: ποσότητα p: προτεραιότητα	Τα Jobs αντιστοιχούν στα αιτήματα που υπάρχουν στο σύστημα. Ζητούν κάποιες ποσότητες από τους πόρους και έχουν προτεραιότητες
3	LocalJobs		{(r,e,p)} r: όνομα πόρου e: ποσότητα p: προτεραιότητα	Τα LocalJobs αντιστοιχούν στα τοπικά αιτήματα του κάθε Slave τα οποία ζητούν και αυτά κάποιες ποσότητες πόρων. Οι ανάγκες τους αυτές πρέπει να καλυφθούν από τον αντίστοιχο Slave πρωτού ικανοποιήσει κάποιο άλλο Job

Πίνακας 3.3 Πληροφορίες για τις οντότητες Slaves, Jobs και LocalJobs



Σχήμα 3.11 Σχηματική αναπαράσταση των Slaves, των Jobs και των LocalJobs

Στο Σχήμα 3.11 παρατηρούμε ότι στο σύστημά μας υπάρχουν 5 δουλειές, 8 τοπικές δουλειές και 4 προμήθειες. Συγκεκριμένα η $1^{η}$ προμήθεια έχει τοπικά τις δουλειές L1,L2 και L3, η $2^{η}$ προμήθεια έχει τοπικά τις δουλειές L4 και L5, η $3^{η}$ προμήθεια έχει τοπικά τις δουλειές L6 και L7, και η $4^{η}$ προμήθεια έχει τοπικά τη δουλειά L8. Αυτές οι τοπικές δουλειές πρέπει να εκτελεστούν από τις προμήθειες πρωτού εξυπηρετήσουν κάποια από τις υπόλοιπες δουλειές. Η κάθε προμήθεια και η κάθε εργασία, τοπική ή/και μη έχει έναν υποδείκτη που υποδηλώνει σε ποια προμήθεια ανήκει ο κάθε πόρος και από ποια προμήθεια ζητείται ο κάθε πόρος αντίστοιχα. Για παράδειγμα ο συμβολισμός $\bar{r}_1^4$ αναφέρεται στην χορηγία του πόρου r_1 από την προμήθεια S_4. Ενώ ο συμβολισμός r_2^3 αναφέρεται στο αίτημα για τον πόρο r_2 από την προμήθεια S_3. Ο λόγος που χρησιμοποιούμε αυτούς τους υποδείκτες είναι επειδή θεωρούμε ότι το ζευγάρωμα μεταξύ προμηθειών και εργασιών τοπικών ή/και μη έχει ήδη γίνει. Επίσης κάθε δουλειά τοπική ή/και μη, έχει μια προτεραιότητα και $c = +\infty$. Ακολουθούν τα ακριβή χαρακτηριστικά της κάθε οντότητας:

$S_1 \overset{def}{=} \{(\overline{r_1}^1, 11,2), (\overline{r_2}^1, 9,3)\} : S_1$

$S_2 \overset{def}{=} \{(\overline{r_1}^2, 3,4)\} : S_2$

$S_3 \overset{def}{=} \{(\overline{r_2}^3, 10,3)\} : S_3$

$S_4 \overset{def}{=} \{(\overline{r_1}^4, 7,1), (\overline{r_2}^4, 2,1)\} : S_4$

$J_1 \overset{def}{=} \{(r_1^1, 5,1), (r_2^1, 5,1)\} : NIL$

$J_2 \overset{def}{=} \{(r_2^3, 10,9)\} : NIL$

$J_3 \overset{def}{=} \{(r_1^4, 2,1), (r_2^4, 1,4)\} : NIL$

$J_4 \overset{def}{=} \{(r_2^1, 5,8)\} : NIL$

$J_5 \overset{def}{=} \{(r_1^1, 8,5), (r_2^1, 1,7)\} : NIL$

$L_1 \overset{def}{=} \{(r_1^1, 4,1 + c)\} : NIL$

$L_2 \overset{def}{=} \{(r_1^1, 1,2 + c), (r_2^1, 8,3 + c)\} : NIL$

$L_3 \overset{def}{=} \{(r_2^1, 9,1 + c)\} : NIL$

$L_4 \overset{def}{=} \{(r_1^2, 1,5 + c)\} : NIL$

$L_5 \overset{def}{=} \{(r_1^2, 2,6 + c)\} : NIL$

$L_6 \overset{def}{=} \{(r_2^3, 7,2 + c)\} : NIL$

$L_7 \overset{def}{=} \{(r_2^3, 6,3 + c)\} : NIL$

$L_8 \overset{def}{=} \{(r_1^4, 1,4 + c), (r_2^4, 1,5 + c)\} : NIL$

3.7.3 Μοντελοποίηση

Ακολουθεί η μοντελοποίηση αλγορίθμων διάφορων πολιτικών χρονοδρομολόγησης. Για όλες τις πολιτικές χρονοδρομολόγησης θεωρούμε ότι το ζευγάρωμα μεταξύ εργασιών (τοπικών ή/και μη) και προμηθειών έχει ήδη γίνει. Αυτό που αλλάζει σε κάθε αλγόριθμο είναι οι συναρτήσεις BreakTiesJ και BreakTiesLJ οι οποίες υπολογίζουν την προτεραιότητα και επιστρέφουν το Job ή το LocalJob αντίστοιχα, που έχει τη μεγαλύτερη προτεραιότητα. Ο υπολογισμός της προτεραιότητας διαφοροποιείται ανάλογα με την πολιτική χρονοδρομολόγησης που εφαρμόζεται κάθε φορά. Για όλες τις πολιτικές χρονοδρομολόγησης που ακολουθούν, το c αποτελεί μια μεγάλη τιμή. Το άθροισμα της προτεραιότητας με το c για τις τοπικές δουλειές, έχει ως αποτέλεσμα να αποκτούν οι τοπικές δουλειές μεγαλύτερη προτεραιότητα σε σχέση με τις δουλειές και έτσι να εκτελούνται οι τοπικές δουλειές πριν από τις δουλείες λόγω μεγαλύτερης προτεραιότητας.

Πολιτική χρονοδρομολόγησης EDF (Earliest Deadline First): Σε αυτή την πολιτική, μεγαλύτερη προτεραιότητα έχουν οι δουλειές οι οποίες έχουν τη μικρότερη ευχέρεια χρόνου (laxity).

BreakTiesJ:

```
    ∀ Job_i ∈ Jobs
Υπολόγισε το laxity_i του Job_i
laxity_i = χρόνος μέχρι την προθεσμία – χρόνος επεξεργασίας
του Job_i
Επέστρεψε το min(laxity_i)  1 ≤ i ≤ N
```

BreakTiesLJ:

∀ $\text{LocalJob}_i \in \text{LocalJobs}$

Υπολόγισε το laxity_i του LocalJob_i

laxity_i = χρόνος μέχρι την προθεσμία − χρόνος επεξεργασίας του LocalJob_i + c

Επέστρεψε το $\min(\text{laxity}_i)$ $1 \leq i \leq N$

Για την πολιτική χρονοδρομολόγησης EDF η ευχέρεια χρόνου ορίζεται ως ο χρόνος μέχρι την προθεσμία μείον το χρόνο επεξεργασίας (processing time) της αντίστοιχης εργασίας για τα Jobs και ορίζεται ως ο χρόνος μέχρι την προθεσμία μείον το χρόνο επεξεργασίας της αντίστοιχης τοπικής εργασίας συν μια σταθερά c (c = + ∞) για τα LocalJobs

$$\text{Slave}_i(x,c) \stackrel{\text{def}}{=} \begin{cases} \emptyset : Slave_i(x,c) & \alpha\nu\ (x = 0) \\ \{(\bar{r}_i, x, c)\} : Slave_i(x,c) & \alpha\nu\ (x \neq 0) \end{cases}$$

Οι Slaves χορηγούν κάποια ποσότητα x έναντι κάποιου χρηματικού ποσού c ανά μονάδα ενέργειας

$$\text{JobsEDF}_{i,j}(x,t,d) \stackrel{\text{def}}{=}$$

$$\begin{cases} Job_i(x,t,d) & \alpha\nu\ (x = 0) \\ \sum_{0 \leq u \leq x} \left\{ \left(r_j, u, (d-t) - \left\lceil \frac{t}{x} \right\rceil \right) \right\} : JobsEDF_{i,j}(x-u, t+1, d) & \alpha\nu\ (d - t > 1) \\ \{(r_j, x, dmax - 1)\} : Job_i(x,t,d) & \alpha\nu\ (d - t = 1) \end{cases}$$

Τα Jobs ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα (d-t) - $\left\lceil \frac{t}{x} \right\rceil$, τον χρόνο που πέρασε t και dmax = $\max(d_i)$ $1 \leq i \leq N$ όπου d_i είναι η προθεσμία. Η προτεραιότητα ορίζεται ως ο χρόνος που έχει περάσει μέχρι την προθεσμία μείον το χρόνο επεξεργασίας. Ο συμβολισμός $\text{JobsEDF}_{i,j}$ σημαίνει ότι η εργασία JobsEDF_i έχει ζευγαρωθεί με την προμήθεια Slave_j

$LocalJobsEDF_{i,j}(x,t,d) \stackrel{def}{=}$

$$\begin{cases} LocalJob_i(x,t,d) & \alpha\nu\ (x=0) \\ \sum_{0\le u\le x}\left\{\left(r_j,u,(d-t)-\left\lceil\frac{t}{x}\right\rceil+c\right)\right\}:LocalJobsEDF_{i,j}(x-u,t+1,d) & \alpha\nu\ (d-t>1) \\ \left\{\left(r_j,x,(d-t)-\left\lceil\frac{t}{x}\right\rceil+c\right)\right\}:LocalJob_i(x,t,d) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα LocalJobs ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα $(d-t)-\left\lceil\frac{t}{x}\right\rceil+c$ όπου c = +∞, και τον χρόνο που πέρασε t. Ο συμβολισμός $LocalJobsEDF_{i,j}$ σημαίνει ότι η τοπική εργασία $LocalJobsEDF_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

$Job_i(x,t,d) \stackrel{def}{=} \emptyset:Job_i(x,t,d) + (start_i?,1).JobsEDF_{i,j}(x,t,d)$

Το Job είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

$Activator_i \stackrel{def}{=} (start_i!,1).\emptyset^{p_i}: Activator_i$

Ο Activator στέλνει περιοδικά ένα μήνυμα start που υποδηλώνει το χρόνο έναρξης και έχει προτεραιότητα 1 για μια δουλειά που περιμένει μέχρι να φτάσει η προθεσμία της και ακολούθως εξελίσσεται πάλι σε Activator. Επίσης p_i είναι η περίοδος που έχει το κάθε $Task_i$

$Task_j(x,t,d) \stackrel{def}{=} (Job_i(x,t,d) \parallel Activator_i) \backslash \{start_i\}$

Κάθε Job μπορεί να αποτελείται από 1 έως πολλά Tasks. Κάθε Task δεσμεύει τη χρήση του καναλιού $start_i$

$LocalActivator_i \stackrel{def}{=} (start_i!,1).\emptyset^{p_i}: LocalActivator_i$

Ο LocalActivator στέλνει περιοδικά ένα μήνυμα start που υποδηλώνει το χρόνο έναρξης και έχει προτεραιότητα 1 για μια τοπική δουλειά που περιμένει μέχρι να φτάσει η προθεσμία της και ακολούθως εξελίσσεται πάλι σε LocalActivator. Επίσης p_i είναι η περίοδος που έχει το κάθε $LocalTask_i$

$LocalTask_j(x,t,d) \stackrel{def}{=} (LocalJob_i(x,t,d) \parallel LocalActivator_i) \backslash \{start_i\}$

Κάθε LocalJob μπορεί να αποτελείται από 1 έως πολλά LocalTasks. Κάθε LocalTask δεσμεύει την χρήση του καναλιού $start_i$

$LocalJob_i(x,t,d) \stackrel{def}{=} \emptyset:LocalJob_i(x,t,d) + (start_i?,1).LocalJobsEDF_{i,j}(x,t,d)$

Το LocalJob είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

System $\stackrel{def}{=}$
$(Task_j(x,t,d)||JobsEDF_{i,j}(x,t,d)||LocalJobsEDF_{i,j}(x,t,d)||Slave_i(x,c)||LocalTask_j(x,t,d))$

Η διεργασία System αναπαριστά ολόκληρο το σύστημα

Μη-διακοπτόμενη πολιτική χρονοδρομολόγησης: Όταν αρχίσει να εκτελείται μια δουλειά δεν μπορεί να διακοπεί αν δεν ολοκληρωθεί.

Ακολούθως ορίζουμε μόνο τις διεργασίες που έχουν διαφοροποιηθεί σε αυτή την πολιτική χρονοδρομολόγησης. Οι διεργασίες $Slave_i(x,c)$, $Job_i(x,t,d)$, $Activator_i$, $Task_j(x,t,d)$, $LocalActivator_i$, $LocalTask_j(x,t,d)$ και System είναι ίδιες με αυτές που έχουμε ορίσει στην πολιτική χρονοδρομολόγησης EDF. Η μόνη διαφοροποίηση είναι ότι όσες διεργασίες καλούσανε τη διεργασία $JobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $JobsNPR_{i,j}(x,t,d,p)$, όσες διεργασίες καλούσανε τη διεργασία $LocalJobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJobsNPR_{i,j}(x,t,d,p)$, όσες διεργασίες καλούσανε τη διεργασία $Job_i(x,t,d)$ πλέον θα καλούν τη διεργασία $Job'_i(x,t,d,p)$ (εκτός από τη διεργασία $StartedNPR_{i,j}(x,t,d)$) και όσες διεργασίες καλούσανε τη διεργασία $LocalJob_i(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJob'_i(x,t,d,p)$

BreakTiesJ:

$\forall\ Job_i \in Jobs$
Υπολόγισε το $priority_i$ του Job_i
$priority_i = p_i$ του Job_i ή $+\infty$ του Job_i (αν βρίσκεται στην κατάσταση Started)
Επέστρεψε το $max(priority_i)$ $1 \leq i \leq N$

BreakTiesLJ:

```
∀ LocalJob_i ∈ LocalJobs
Υπολόγισε το priority_i του LocalJob_i
priority_i = p_i του LocalJob_i + c
Επέστρεψε το max(priority_i)  1 ≤ i ≤ N
```

Για τη μη-διακοπτόμενη πολιτική χρονοδρομολόγησης η προτεραιότητα ορίζεται ως το p_i του Job_i ή $+\infty$ του Job_i (αν βρίσκεται στην κατάσταση Started) για τα Jobs και ορίζεται ως το p_i του $LocalJob_i$ συν μια σταθερά c ($c = +\infty$) για τα LocalJobs. Το p_i είναι μια τυχαία τιμή με την οποία το Job_i ή το $LocalJob_i$ έρχεται στο σύστημα

$JobsNPR_{i,j}(x,t,d,p) \stackrel{def}{=}$

$$\begin{cases} Job'_i(x,t,d,p) & αν\ (x=0) \\ \sum_{0<u\le x}\{(r_j,u,p)\}:StartedNPR_{i,j}(x-u,t+1,d)+\emptyset:JobsNPR_{i,j}(x,t+1,d,p) & αν\ (d-t>1) \\ \{(r_j,x,p)\}:Job'_i(x,t,d,p) & αν\ (d-t=1) \end{cases}$$

$$StartedNPR_{i,j}(x,t,d) \stackrel{def}{=} \begin{cases} Job_i(x,t,d) & αν\ (x=0) \\ \sum_{0\le u\le x}\{(r_j,u,+\infty)\}:StartedNPR_{i,j}(x-u,t+1,d) & αν\ (d-t>1) \\ \{(r_j,x,+\infty)\}:Job_i(x,t,d) & αν\ (d-t=1) \end{cases}$$

Τα JobsNPR ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα p και τον χρόνο που πέρασε t. Η διεργασία Started ζητά κάποια ποσότητα x, έχει προθεσμία d και τον χρόνο που πέρασε t και υποδηλώνει ότι η δουλειά έχει επιλεγεί προς εκτέλεση και δεν θα διακοπεί εάν δεν ικανοποιηθεί. Ο συμβολισμός $JobsNPR_{i,j}$ σημαίνει ότι η εργασία $JobsNPR_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

$LocalJobsNPR_{i,j}(x,t,d,p) \stackrel{def}{=}$

$$\begin{cases} LocalJob'_i(x,t,d,p) & αν\ (x=0) \\ \sum_{0\le u\le x}\{(r_j,u,p+c)\}:LocalJobsNPR_{i,j}(x-u,t+1,d,p) & αν\ (d-t>1) \\ \{(r_j,x,p+c)\}:LocalJob'_i(x,t,d,p) & αν\ (d-t=1) \end{cases}$$

Τα LocalJobs ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα $p+c$ όπου $c = +\infty$, και τον χρόνο που πέρασε t. Ο συμβολισμός $LocalJobsNPR_{i,j}$ σημαίνει ότι η τοπική εργασία $LocalJobsNPR_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

$$Job'_i(x,t,d,p) \stackrel{def}{\equiv} \emptyset{:}Job'_i(x,t,d,p) + (start_i?,1).JobsNPR_{i,j}(x,t,d,p)$$

Το Job' είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

$$LocalJob'_i(x,t,d,p) \stackrel{def}{\equiv} \emptyset{:}LocalJob'_i(x,t,d,p) + (start_i?,1).LocalJobsNPR_{i,j}(x,t,d,p)$$

Το LocalJob' είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

Διακοπτόμενη πολιτική χρονοδρομολόγησης: Όταν αρχίζει να εκτελείται μια δουλειά, μπορεί να διακοπεί και να συνεχίσει την εκτέλεσή της πολλές φορές μέχρι να ολοκληρωθεί.

Ακολούθως ορίζουμε μόνο τις διεργασίες που έχουν διαφοροποιηθεί σε αυτή την πολιτική χρονοδρομολόγησης. Οι διεργασίες $Slave_i(x,c)$, $Activator_i$, $Task_j(x,t,d)$, $LocalActivator_i$, $LocalTask_j(x,t,d)$ και System είναι ίδιες με αυτές που έχουμε ορίσει στην πολιτική χρονοδρομολόγησης EDF. Η μόνη διαφοροποίηση είναι ότι όσες διεργασίες καλούσανε τη διεργασία $JobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $JobsPR_{i,j}(x,t,d,p)$ και όσες διεργασίες καλούσανε τη διεργασία $LocalJobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJobsPR_{i,j}(x,t,d,p)$, όσες διεργασίες καλούσανε τη διεργασία $Job_i(x,t,d)$ πλέον θα καλούν τη διεργασία $Job'_i(x,t,d,p)$ και όσες διεργασίες καλούσανε τη διεργασία $LocalJob_i(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJob'_i(x,t,d,p)$

BreakTiesJ:

$\forall\ Job_i \in Jobs$

Υπολόγισε το $priority_i$ του Job_i

$priority_i = p_i$ του Job_i

Επέστρεψε το $max(priority_i)$ $\quad 1 \leq i \leq N$

BreakTiesLJ:

∀ $LocalJob_i$ ∈ LocalJobs

Υπολόγισε το $priority_i$ του $LocalJob_i$

$priority_i$ = p_i του $LocalJob_i$ + c

Επέστρεψε το max($priority_i$) $1 \leq i \leq N$

Για τη διακοπτόμενη πολιτική χρονοδρομολόγησης η προτεραιότητα ορίζεται ως το p_i του Job_i για τα Jobs και ορίζεται ως το p_i του $LocalJob_i$ συν μια σταθερά c (c = +∞) για τα LocalJobs .Το p_i είναι μια τυχαία τιμή με την οποία το Job_i ή το $LocalJob_i$ έρχεται στο σύστημα

$$\text{JobsPR}_{i,j}(x,t,d,p) \stackrel{\text{def}}{=} \begin{cases} Job'_i(x,t,d,p) & αν\ (x=0) \\ \sum_{0\le u\le x} \{(r_j,u,p)\}: JobsPR_{i,j}(x-u,t+1,d,p) & αν\ (d-t>1) \\ \{(r_j,x,p)\}: Job'_i(x,t,d,p) & αν\ (d-t=1) \end{cases}$$

Τα JobsPR ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα p και τον χρόνο που πέρασε t. Ο συμβολισμός $\text{JobsPR}_{i,j}$ σημαίνει ότι η εργασία JobsPR_i έχει ζευγαρωθεί με την προμήθεια Slave_j

$$\text{LocalJobsPR}_{i,j}(x,t,d,p) \stackrel{\text{def}}{=}$$

$$\begin{cases} LocalJob'_i(x,t,d,p) & αν\ (x=0) \\ \sum_{0\le u\le x} \{(r_j,u,p+c)\}: LocalJobsPR_{i,j}(x-u,t+1,d,p) & αν\ (d-t>1) \\ \{(r_j,x,p+c)\}: LocalJob'_i(x,t,d,p) & αν\ (d-t=1) \end{cases}$$

Τα LocalJobs ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα $p+c$ όπου c = +∞ ,και τον χρόνο που πέρασε t. Ο συμβολισμός $\text{LocalJobsPR}_{i,j}$ σημαίνει ότι η τοπική εργασία LocalJobsPR_i έχει ζευγαρωθεί με την προμήθεια Slave_j

$$\text{Job'}_i(x,t,d,p) \stackrel{\text{def}}{=} \emptyset:\text{Job'}_i(x,t,d,p) + (\text{start}_i?,1).\text{JobsPR}_{i,j}(x,t,d,p)$$

Το Job' είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

$$\text{LocalJob'}_i(x,t,d,p) \stackrel{\text{def}}{=} \emptyset:\text{LocalJob'}_i(x,t,d,p) + (\text{start}_i?,1).\text{LocalJobsPR}_{i,j}(x,t,d,p)$$

Το LocalJob’ είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

Πολιτική χρονοδρομολόγησης Min-Min: Σε αυτή την πολιτική, μεγαλύτερη προτεραιότητα έχουν οι δουλειές οι οποίες ζητούν τη μικρότερη ποσότητα. [2]

Ακολούθως ορίζουμε μόνο τις διεργασίες που έχουν διαφοροποιηθεί σε αυτή την πολιτική χρονοδρομολόγησης. Οι διεργασίες $Slave_i(x,c)$, $Job_i(x,t,d)$, $Activator_i$, $Task_j(x,t,d)$, $LocalActivator_i$, $LocalTask_j(x,t,d)$, $LocalJob_i(x,t,d)$ και System είναι ίδιες με αυτές που έχουμε ορίσει στην πολιτική χρονοδρομολόγησης EDF. Η μόνη διαφοροποίηση είναι ότι όσες διεργασίες καλούσανε τη διεργασία $JobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $JobsMin_{i,j}(x,t,d)$ και όσες διεργασίες καλούσανε τη διεργασία $LocalJobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJobsMin_{i,j}(x,t,d)$

BreakTiesJ:

∀ Job_i ∈ Jobs
Υπολόγισε το $priority_i$ του Job_i
$priority_i$ = 1/ ποσότητα που ζητά το Job_i
Επέστρεψε το max($priority_i$) $1 \leq i \leq N$

BreakTiesLJ:

∀ $LocalJob_i$ ∈ LocalJobs
Υπολόγισε το $priority_i$ του $LocalJob_i$
$priority_i$ = 1/ ποσότητα που ζητά το $LocalJob_i$ + c
Επέστρεψε το max($priority_i$) $1 \leq i \leq N$

Για την πολιτική χρονοδρομολόγησης Min-Min η προτεραιότητα ορίζεται ως 1/ ποσότητα που ζητά το Job_i για τα Jobs και ορίζεται ως 1/ ποσότητα που ζητά το $LocalJob_i$ συν μια σταθερά c (c= +∞) για τα LocalJobs

$$\text{JobsMin}_{i,j}(x,t,d) \stackrel{\text{def}}{=} \begin{cases} Job_i(x,t,d) & \alpha\nu\ (x=0) \\ \sum_{0\le u\le x} \left\{\left(r_j, u, \frac{1}{x}\right)\right\} : JobsMin_{i,j}(x-u, t+1, d) & \alpha\nu\ (d-t>1) \\ \left\{\left(r_j, x, \frac{1}{x}\right)\right\} : Job_i(x,t,d) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα JobsMin ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα $\frac{1}{x}$ και τον χρόνο που πέρασε t. Ο συμβολισμός $\text{JobsMin}_{i,j}$ σημαίνει ότι η εργασία JobsMin_i έχει ζευγαρωθεί με την προμήθεια Slave_j

$$\text{LocalJobsMin}_{i,j}(x,t,d) \stackrel{\text{def}}{=}$$
$$\begin{cases} LocalJob_i(x,t,d) & \alpha\nu\ (x=0) \\ \sum_{0\le u\le x} \left\{\left(r_j, u, \frac{1}{x}+c\right)\right\} : LocalJobsMin_{i,j}(x-u, t+1, d) & \alpha\nu\ (d-t>1) \\ \left\{\left(r_j, x, \frac{1}{x}+c\right)\right\} : LocalJob_i(x,t,d) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα LocalJobs ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα $\frac{1}{x}+c$ όπου c = +∞ , και το χρόνο που πέρασε t. Ο συμβολισμός $\text{LocalJobsMin}_{i,j}$ σημαίνει ότι η τοπική εργασία LocalJobsMin_i έχει ζευγαρωθεί με την προμήθεια Slave_j

Πολιτική χρονοδρομολόγησης Max-Min: Σε αυτή τη πολιτική, μεγαλύτερη προτεραιότητα έχουν οι δουλειές οι οποίες ζητούν τη μεγαλύτερη ποσότητα. [2]

Ακολούθως ορίζουμε μόνο τις διεργασίες που έχουν διαφοροποιηθεί σε αυτή την πολιτική χρονοδρομολόγησης. Οι διεργασίες $\text{Slave}_i(x,c)$, $\text{Job}_i(x,t,d)$, Activator_i, $\text{Task}_j(x,t,d)$, LocalActivator_i, $\text{LocalTask}_j(x,t,d)$, $\text{LocalJob}_i(x,t,d)$ και System είναι ίδιες με αυτές που έχουμε ορίσει στην πολιτική χρονοδρομολόγησης EDF. Η μόνη διαφοροποίηση είναι ότι όσες διεργασίες καλούσανε τη διεργασία $\text{JobsEDF}_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $\text{JobsMax}_{i,j}(x,t,d)$ και όσες διεργασίες καλούσανε τη διεργασία $\text{LocalJobsEDF}_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $\text{LocalJobsMax}_{i,j}(x,t,d)$

BreakTiesJ:

∀ Job_i ∈ Jobs

Υπολόγισε το $priority_i$ του Job_i

$priority_i$ = ποσότητα που ζητά το Job_i

Επέστρεψε το max($priority_i$) $1 \leq i \leq N$

BreakTiesLJ:

∀ $LocalJob_i$ ∈ LocalJobs

Υπολόγισε το $priority_i$ του $LocalJob_i$

$priority_i$ = ποσότητα που ζητά το $LocalJob_i$ + c

Επέστρεψε το max($priority_i$) $1 \leq i \leq N$

Για την πολιτική χρονοδρομολόγησης Max-Min η προτεραιότητα ορίζεται ως η ποσότητα που ζητά το Job_i για τα Jobs και ορίζεται ως η ποσότητα που ζητά το $LocalJob_i$ συν μια σταθερά c (c= +∞) για τα LocalJobs

$$JobsMax_{i,j}(x,t,d) \stackrel{def}{=} \begin{cases} Job_i(x,t,d) & \alpha\nu\ (x=0) \\ \sum_{0\leq u\leq x} \{(r_j,u,x)\}: JobsMax_{i,j}(x-u,t+1,d) & \alpha\nu\ (d-t>1) \\ \{(r_j,x,x)\}: Job_i(x,t,d) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα JobsMax ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα x και τον χρόνο που πέρασε t. Ο συμβολισμός $JobsMax_{i,j}$ σημαίνει ότι η εργασία $JobsMax_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

$$LocalJobsMax_{i,j}(x,t,d) \stackrel{def}{=}$$

$$\begin{cases} LocalJob_i(x,t,d) & \alpha\nu\ (x=0) \\ \sum_{0\leq u\leq x} \{(r_j,u,x+c)\}: LocalJobsMax_{i,j}(x-u,t+1,d) & \alpha\nu\ (d-t>1) \\ \{(r_j,x,x+c)\}: LocalJob_i(x,t,d) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα LocalJobs ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα $x+c$ όπου c = +∞ , και το χρόνο που πέρασε t. Ο συμβολισμός $LocalJobsMax_{i,j}$ σημαίνει ότι η τοπική εργασία $LocalJobsMax_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

Πολιτική χρονοδρομολόγησης FIFO: Σε αυτή την πολιτική, μεγαλύτερη προτεραιότητα έχουν οι δουλειές οι οποίες αφίχθησαν πρώτες στο σύστημα. Συγκεκριμένα για αυτή την πολιτική κάνουμε χρήση καναλιών, για τον ορισμό του num. Το num καθορίζει την προτεραιότητα της κάθε τοπική ή/και μη τοπικής δουλειάς. Αρχικά ισούται με ένα και κάθε φορά που έρχεται μια δουλεία τοπική ή/και μη στο σύστημα, στέλνεται η τιμή του num στην αντίστοιχη τοπική ή/και μη δουλειά διαμέσου του καναλιού επικοινωνίας. Ακολούθως η τιμή του num αυξάνεται κατά ένα.

Ακολούθως ορίζουμε μόνο τις διεργασίες που έχουν διαφοροποιηθεί σε αυτή την πολιτική χρονοδρομολόγησης. Οι διεργασίες $Slave_i(x,c)$, $Activator_i$, $Task_j(x,t,d)$, $LocalActivator_i$, $LocalTask_j(x,t,d)$ και System (με κάποιες τροποποιήσεις) είναι ίδιες με αυτές που έχουμε ορίσει στην πολιτική χρονοδρομολόγησης EDF. Η μόνη διαφοροποίηση είναι ότι όσες διεργασίες καλούσανε τη διεργασία $JobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $JobsFIFO_{i,j}(x,t,d,num)$, όσες διεργασίες καλούσανε τη διεργασία $LocalJobsEDF_{i,j}(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJobsFIFO_{i,j}(x,t,d,num)$, όσες διεργασίες καλούσανε τη διεργασία $Job_i(x,t,d)$ πλέον θα καλούν τη διεργασία $Job'_i(x,t,d,num)$ και όσες διεργασίες καλούσανε τη διεργασία $LocalJob_i(x,t,d)$ πλέον θα καλούν τη διεργασία $LocalJob'_i(x,t,d,num)$

BreakTiesJ:

$\forall\ Job_i \in Jobs$

Υπολόγισε το $priority_i$ του Job_i

$priority_i = {}^{1}/_{num}$ που πήρε το Job_i

Επέστρεψε το $max(priority_i)$ $\quad 1 \leq i \leq N$

BreakTiesLJ:

$\forall\ LocalJob_i \in LocalJobs$

Υπολόγισε το $priority_i$ του $LocalJob_i$

$priority_i = {}^{1}/_{num}$ που πήρε το $LocalJob_i$ + c

Επέστρεψε το $max(priority_i)$ $\quad 1 \leq i \leq N$

Για την πολιτική χρονοδρομολόγησης FIFO η προτεραιότητα ορίζεται ως ${}^{1}/_{num}$ που πήρε το Job_i για τα Jobs και ορίζεται ως ${}^{1}/_{num}$ που πήρε το $LocalJob_i$ συν μια σταθερά c (c= + ∞) για τα LocalJobs

$JobsFIFO_{i,j}(x,t,d,1/num) \stackrel{def}{=}$

$$\begin{cases} Job'_i(x,t,d,1/num) & \alpha\nu\ (x=0) \\ \sum_{0\le u\le x}\{(r_j,u,1/num)\}:JobsFIFO_{i,j}(x-u,t+1,d,1/num) & \alpha\nu\ (d-t>1) \\ \{(r_j,x,1/num)\}:Job'_i(x,t,d,1/num) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα JobsFIFO ζητούν κάποια ποσότητα x, έχουν προθεσμία d και προτεραιότητα 1/num και τον χρόνο που πέρασε t. Ο συμβολισμός $JobsFIFO_{i,j}$ σημαίνει ότι η εργασία $JobsFIFO_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

$LocalJobsFIFO_{i,j}(x,t,d,1/num) \stackrel{def}{=}$

$$\begin{cases} LocalJob'_i(x,t,d,1/num) & \alpha\nu\ (x=0) \\ \sum_{0\le u\le x}\{(r_j,u,1/num+c)\}:LocalJobsFIFO_{i,j}(x-u,t+1,d,1/num) & \alpha\nu\ (d-t>1) \\ \{(r_j,x,1/num+c)\}:LocalJob'_i(x,t,d,1/num) & \alpha\nu\ (d-t=1) \end{cases}$$

Τα LocalJobs ζητούν κάποια ποσότητα x έχουν προθεσμία d και προτεραιότητα $1/num+c$ όπου c = +∞ ,και τον χρόνο που πέρασε t. Ο συμβολισμός $LocalJobsFIFO_{i,j}$ σημαίνει ότι η τοπική εργασία $LocalJobsFIFO_i$ έχει ζευγαρωθεί με την προμήθεια $Slave_j$

$Job'_i(x,t,d,num) \stackrel{def}{=} \emptyset:Job'_i(x,t,d,num) + (start_i?,1).JobsFIFO_{i,j}(x,t,d,1/num)$

Το Job' είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

$LocalJob'_i(x,t,d,num) \stackrel{def}{=} \emptyset:LocalJob'_i(x,t,d,num) + (start_i?,1).LocalJobsFIFO_{i,j}(x,t,d,\frac{1}{num})$

Το LocalJob' είναι υπεύθυνο να δέχεται μηνύματα και αναλόγως να καλεί την αντίστοιχη διεργασία

$J_i(x,t,d) \stackrel{def}{=} (st?num,p).JobsFIFO_{i,j}(x,t,d,1/num)$

Η διεργασία J παραλαμβάνει το μήνυμα st με προτεραιότητα p, το οποίο περιέχει την τιμή num που αντιστοιχεί στην προτεραιότητα της αντίστοιχης δουλειάς

$LJ_i(x,t,d) \stackrel{def}{=} (st?num,p).LocalJobsFIFO_{i,j}(x,t,d,1/num)$

Η διεργασία LJ παραλαμβάνει το μήνυμα st με προτεραιότητα p, το οποίο περιέχει την τιμή num που αντιστοιχεί στην προτεραιότητα της αντίστοιχης τοπικής δουλειάς

$Coordinator_i(num) \stackrel{def}{\equiv} (st!num,p):Coordinator_i(num+ +)$

Η διεργασία Coordinator σκοπό έχει να συντονίζει τις δουλειές (τοπικές ή/και μη) που έρχονται στο σύστημα. Δηλαδή τους στέλνει μέσω του καναλιού ένα μήνυμα st με προτεραιότητα p, το οποίο περιέχει την τιμή num που αντιστοιχεί στην προτεραιότητα της αντίστοιχης δουλειάς (τοπικής ή/και μη). Δηλαδή όσο πιο νωρίς έρθει μια δουλειά (τοπική ή/και μη) στο σύστημα τόσο μικρότερο num θα πάρει και κατά επέκταση τόσο μεγαλύτερη προτεραιότητα θα έχει αφού η προτεραιότητα = $1/_{num}$. Έτσι με αυτό τον τρόπο βάζει τις δουλειές (τοπικές ή/και μη) στη σωστή σειρά χρονικής άφιξης μιας και ο χρόνος είναι σχετικός. Αρχικά num = 1

$C_i(x,t,d) \stackrel{def}{\equiv} (J_i(x,t,d) \,||\, LJ_i(x,t,d)||\, Coordinator_i(num)) \backslash \{st\}$

Η διεργασία C συντονίζει τις διεργασίες J ,LJ και Coordinator. Επίσης δεσμεύει την χρήση του καναλιού st

$System \stackrel{def}{\equiv} (Task_j(x,t,d)||JobsFIFO_{i,j}(x,t,d,1/num)||$
$LocalJobsFIFO_{i,j}(x,t,d,1/num)||Slave_i(x,c)||LocalTask_j(x,t,d)||C_i(x,t,d))$

Η διεργασία System αναπαριστά ολόκληρο το σύστημα

3.8 Γενικά Συμπεράσματα

Κλείνοντας αυτό το κεφάλαιο, παρατηρούμε ότι η άλγεβρα διεργασιών ACCED είναι μια εκφραστική γλώσσα που έχει τη δυνατότητα να μοντελοποιήσει προβλήματα παροχής και ζήτησης πόρων και πολιτικών χρονοδρομολόγησης σε αυτό, όπως για παράδειγμα, θα μπορούσε να μοντελοποιήσει νέους αλγόριθμους από το πεδίο του υπολογιστικού νέφους άλλα ίσως και κάποιου άλλου πεδίου που έχει να κάνει με παροχή και διαχείριση πόρων και ακολούθως να ελέγξουμε και να δοκιμάσουμε αυτούς τους αλγόριθμους ή μπορούμε να ελέγξουμε υπάρχοντες αλγόριθμους αν όντως λειτουργούν ορθά ή αν μπορούμε να τους βελτιώσουμε. Οι προτεραιότητες έπαιξαν καθοριστικό ρόλο και συνέβαλλαν ώστε να μπορούμε να επιλέξουμε ποια θα είναι η επόμενη εργασία προς εκτέλεση και κατ'επέκταση για να μπορέσουμε να χρονοδρομολογήσουμε όλες τις εργασίες του συστήματος. Επίσης ο ορισμός της ποσότητας μας βοήθησε να αποφασίσουμε για το ποιος σκλάβος έχει την απαιτούμενη χορηγούμενη ποσότητα για να ικανοποιήσει τα αιτήματα της κάθε εργασίας. Δηλαδή μας βοήθησαν στον εντοπισμό του καταλληλότερου σκλάβου και έτσι οδηγηθήκαμε στη μέγιστη δυνατή αξιοποίηση των προμηθειών του συστήματος μας. Επιπρόσθετα έχουμε

ορίσει και κάποιες καινούργιες έννοιες στη γλώσσα όπως είναι η έννοια των συντονισμών και η έννοια του κόστους. Οι συντονισμοί, μας επιτρέπουν να υπάρχει επικοινωνία μεταξύ των διεργασιών διαμέσου των καναλιών επικοινωνίας όταν χρειάζεται. Δηλαδή συντονίζουν τις δράσεις τους ώστε να μπορούμε να υλοποιούμε τους αλγόριθμους πολιτικών χρονοδρομολόγησης που επιθυμούμε. Η ενσωμάτωση του κόστους μας επιτρέπει να μοντελοποιούμε την κοστολόγηση και κατ' επέκταση να εξασφαλίζουμε το μεγαλύτερο δυνατό κέρδος. Όπως έχουμε ήδη αναφέρει, δυνατότητες συντονισμού παρέχονται και στην αλγοριθμική προσέγγιση του MapReduce, η οποία έχει εφαρμογή στο πεδίο του υπολογιστικού νέφους. Συγκεκριμένα παρατηρούμε ότι η άλγεβρα διεργασιών ACCED έχει κοινά χαρακτηριστικά με το πεδίο του υπολογιστικού νέφους. Δηλαδή στο πεδίο του υπολογιστικού νέφους θέλουμε να έχουμε τη μέγιστη δυνατή κατανάλωση όπως και στην ACCED, δηλαδή αυτό σημαίνει να έχουμε πολλούς χρήστες και να μπορούμε να τους ικανοποιούμε, ώστε να έχουμε και μεγαλύτερο κέρδος. Επίσης τα αιτήματα των χρηστών στο πεδίο του υπολογιστικού νέφους όπως και στην ACCED, έχουν κάποια προτεραιότητα και αν το αίτημα α έχει μεγαλύτερη προτεραιότητα από το αίτημα β τότε προηγείται το αίτημα α του αιτήματος β. Ακόμη οι εικονικές μηχανές (virtual machines) στο πεδίο του υπολογιστικού νέφους ικανοποιούν πρώτα τα αιτήματα που είναι χρονοπρογραμματίσιμα, δηλαδή που δεν θα περιέχουν ανικανοποίητα αιτήματα, όπως συμβαίνει και στην ACCED. Επομένως υπάρχει ένας παραλληλισμός μεταξύ της άλγεβρας διεργασιών ACCED και των χαρακτηριστικών του πεδίου του υπολογιστικού νέφους. Επίσης η άλγεβρα διεργασιών ACCED επεκτείνει την άλγεβρα διεργασιών AESC, διότι μπορέσαμε να απαλλαγούμε από τα σύνολα λιστών τα οποία ήτανε ορισμένα στην AESC, χάρη στον κανόνα Equiv, δηλαδή της συντακτικής ισοδυναμίας μεταξύ όλων των πιθανών μεταβάσεων των διεργασιών ενός συστήματος. Επιπρόσθετα για την έννοια του χρονοπρογραμματισμού την οποία θα μελετήσουμε στο τέταρτο κεφάλαιο, μπορέσαμε να διορθώσουμε το πρόβλημα που υπήρχε στην απόδειξη του θεωρήματος της ταυτόχρονης εξυπηρέτησης δύο εργασιών από μια προμήθεια της AESC, χάρη στον ορισμό της σχέσης προτιμήσεως που δώσαμε και επίσης μπορέσαμε να αποδείξουμε το θεώρημα της ταυτόχρονης εξυπηρέτησης δύο εργασιών από μια προμήθεια εντός ενός πιο γενικευμένου μοντελού, συγκριτικά με τα μοντέλα των άλγεβρων διεργασιών AESC και EDAUS. Άρα συμπεραίνουμε ότι η άλγεβρα διεργασιών ACCED μας παρέχει πολλές δυνατότητες.

Κεφάλαιο 4

Χρονοπρογραμματισμός

4.1 Περιγραφή

Σε αυτό το κεφάλαιο θα ορίσουμε την έννοια του χρονοπρογραμματισμού ενός συστήματος. Συγκεκριμένα θα δούμε πότε ένα σύνολο από αιτήματα είναι χρονοπρογραμματίσιμο από ένα σύνολο από προμήθειες και από τις δυνατότητες συντονισμού. Ακολούθως θα ορίσουμε την έννοια του υπολοίπου προμήθειας και τη σημασία που έχει. Τέλος θα δώσουμε τον ορισμό της ταυτόχρονης εξυπηρέτησης δύο εργασιών από μια προμήθεια, δηλαδή όταν μια προμήθεια S, με τις δυνατότητες συντονισμού P μπορεί να χρονοπρογραμματίσει μια εργασία T και η υπολειπόμενη χορηγούμενη ποσότητα της προμήθειας με τις διαθέσιμες δυνατότητες συντονισμού μπορεί να χρονοπρογραμματίσει μια εργασία T’, τότε η προμήθεια S, με τις δυνατότητες συντονισμού P μπορεί να χρονοπρογραμματίσει ταυτόχρονα τις εργασίες T και T’, δεδομένου ότι αν οι εργασίες T και T’ ζητούν το ίδιο μήνυμα τότε πρέπει να έχουν ντετερμινιστική δομή.

4.2 Ορισμός χρονοπρογραμματισμού

Για να γίνει πιο κατανοητός ο ορισμός του χρονοπρογραμματισμού αρχικά θα ορίσουμε κάποιες βασικές βοηθητικές έννοιες.

Αρχικά θα ορίσουμε τη γραμματική των εργασιών T, των προμηθειών S και των δυνατοτήτων συντονισμού P

T::= NIL | e:T | α:T |T||T |$\sum_{i\in I} e_i: T_i$ | $\sum_{i\in I} a_i: T_i$
S::= NIL | β:S | S||S | $\sum_{i\in I} \beta_i: S_i$
P::= NIL | Ø:P | e:P | P||P

Από τη γραμματική παρατηρούμε ότι οι δυνατότητες συντονισμού P, δεν έχουν την δυνατότητα να αποτελούνται από άθροισμα γεγονότων.

Επίσης ορίζουμε ως T^* το σύνολο όλων των παράλληλων συνθέσεων που έχουν την μορφή $T_1||T_2||\ldots||T_n$ όπου $n \geq 1$
Ακόμη ορίζουμε ως S^* το σύνολο όλων των παράλληλων συνθέσεων που έχουν την μορφή $S_1||S_2||\ldots||S_n$ όπου $n \geq 1$
Τέλος ορίζουμε ως P^* το σύνολο όλων των παράλληλων συνθέσεων που έχουν την μορφή $P_1||P_2||\ldots||P_n$ όπου $n \geq 1$

Επιπρόσθετα για τον ορισμό του α^{μ} ισχύει ακριβώς ότι ορίσαμε στο κεφάλαιο 3, στο υποκεφάλαιο 3.6.

Ακολούθως θα ορίσουμε την έννοια του χρονοπρογραμματισμού, δίνοντας το διαισθητικό ορισμό της έννοιας του χρονοπρογραμματισμού.

Ορισμός 4.1

Μια εργασία $T\in T^*$ είναι χρονοπρογραμματίσιμη από μια προμήθεια $S\in S^*$ σε συνδυασμό με τις δυνατότητες συντονισμού $P\in P^*$ αν ισχύει πως για κάθε μετάβαση $T||S||P \Rightarrow X$ τότε αν $X \xrightarrow{\alpha}$ έχουμε $\alpha^{\mu} \cap R = \emptyset$. Με το συμβολισμό $\Rightarrow$ αναφερόμαστε σε μεταβάσεις της μορφής $X \xrightarrow{*} X'$.

Δηλαδή μια εργασία T είναι χρονοπρογραμματίσιμη από μια προμήθεια S σε συνδυασμό με τις δυνατότητες συντονισμού P εάν σε κάθε δράση που εμφανίζεται στο σύστημα μας δεν εμπεριέχεται κανένα ανικανοποίητο αίτημα.

Αν μια εργασία T είναι χρονοπρογραμματίσιμη από μια προμήθεια S σε συνδυασμό με τις δυνατότητες συντονισμού P τότε το συμβολίζουμε ως εξής : $S||P \models T$. Επίσης λέμε ότι η εργασία T είναι χρονοπρογραμματίσιμη από μια προμήθεια S αν $S||NIL \models T$.

Παράδειγμα 4.1

Έστω οι ακόλουθες εργασίες και προμήθειες:

$T_1 \stackrel{def}{=} \{(r_1,10,2)\}:NIL$
$T_2 \stackrel{def}{=} \{(r_1,5,3),(r_2,10,2)\}:NIL + \{(r_2,20,4)\}:NIL$
$S_1 \stackrel{def}{=} \{(\overline{r1},20,3)\}:NIL$
$S_2 \stackrel{def}{=} \{(\overline{r1},5,2),(\overline{r2},10,3)\}:S_2$
$P \stackrel{def}{=} NIL$

Η εργασία T_1 ζητά 10 μονάδες ενέργειας από τον πόρο r_1. Έχει προτεραιότητα 2. Η εργασία T_2 ζητά 5 μονάδες ενέργειας από τον πόρο r_1 και έχει προτεραιότητα 3 και 10 μονάδες ενέργειας από τον πόρο r_2 και έχει προτεραιότητα 2 ή ζητά 20 μονάδες ενέργειας από τον πόρο r_2.Έχει προτεραιότητα 4. Η προμήθεια S_1 χορηγεί 20 μονάδες ενέργειας ανά μονάδα χρόνου από τον πόρο r_1 έναντι του ποσού των €3. Η προμήθεια S_2 χορηγεί 5 μονάδες ενέργειας ανά μονάδα χρόνου από τον πόρο r_1 έναντι του ποσού των €2 και χορηγεί 10 μονάδες ενέργειας ανά μονάδα χρόνου από τον πόρο r_2 έναντι του ποσού των €3. Τέλος δεν υπάρχουν δυνατότητες συντονισμού P.

Από την παράλληλη σύνθεση της εργασίας T_2 και της προμήθειας S_2 προκύπτουν δύο πιθανές μεταβάσεις. Η πρώτη πιθανή μετάβαση είναι η $[\ T_2||S_2 \xrightarrow{\{(\overleftarrow{r1},5,3,2),(\overleftarrow{r2},10,2,3)\}} NIL||S_2\ ]$, δηλαδή προκύπτει μια κατανάλωση 5 μονάδων ενέργειας από τον πόρο r_1 και μια κατανάλωση 10 μονάδων ενέργειας από τον πόρο r_2. Η δεύτερη πιθανή μετάβαση είναι η $[T_2||S_2 \xrightarrow{\{(\overleftarrow{r1},5,2),(\overleftarrow{r2},10,4,3),(r2,10,4)\}} NIL||S_2]$, δηλαδή προκύπτει μια χορηγία 5 μονάδων ενέργειας από τον πόρο r_1 έναντι του ποσού των €2, μια κατανάλωση 10 μονάδων ενέργειας από τον πόρο r_2 και μια αίτηση 10 μονάδων ενέργειας από τον πόρο r_2. Αλλά από το 2ο σημείο του ορισμού της σχέσης προτιμήσεως γνωρίζουμε ότι ισχύει $\{(\overleftarrow{r1},5,2),(\overleftarrow{r2},10,4,3),(r2,10,4)\} \prec \{(\overleftarrow{r1},5,3,2),(\overleftarrow{r2},10,2,3)\}$.

Επομένως η πρώτη δυνατή μετάβαση αποκλείει την εμφάνιση της δεύτερης δυνατής μετάβασης και έτσι προκύπτει ότι το σύστημα μας είναι χρονοπρογραμματίσιμο.

Από την παράλληλη σύνθεση της εργασίας T_1 και της προμήθειας S_1 προκύπτει μόνο μία πιθανή μετάβαση η εξής: $[T_1||S_1 \xrightarrow{\{(\overrightarrow{r1},10,3),(\overrightarrow{r1},10,2,3)\}} NIL||NIL]$ και έτσι προκύπτει ότι το σύστημα μας είναι χρονοπρογραμματίσιμο.

Σε αντίθεση με την παράλληλη σύνθεση της εργασίας T_1 και της προμήθειας S_2 προκύπτει μόνο μία πιθανή μετάβαση η εξής ακόλουθη: $[T_1||S_2 \xrightarrow{\{(\overrightarrow{r2},10,3),(\overrightarrow{r1},5,2,2),(r1,5,2)\}} NIL||S_2]$ και έτσι προκύπτει ότι το σύστημα μας είναι μη χρονοπρογραμματίσιμο αφού η μετάβαση περιέχει ανικανοποίητα αιτήματα.

Επιπρόσθετα από την παράλληλη σύνθεση της εργασίας T_2 και της προμήθειας S_1 προκύπτουν δύο πιθανές μεταβάσεις. Η πρώτη πιθανή μετάβαση είναι η $[T_2||S_1 \xrightarrow{\{(\overrightarrow{r1},5,3,3),(\overrightarrow{r1},15,3),(r2,10,2)\}} NIL||NIL]$, και η δεύτερη πιθανή μετάβαση είναι η $[T_2||S_1 \xrightarrow{\{(\overrightarrow{r1},20,3),(r2,20,4)\}} NIL||NIL]$. Επομένως παρατηρούμε ότι τόσο η πρώτη όσο και η δεύτερη πιθανή μετάβαση περιέχουν ανικανοποίητα αιτήματα επομένως προκύπτει ότι το σύστημα μας είναι μη χρονοπρογραμματίσιμο.

Παράδειγμα 4.2

Έστω η προμήθεια $S \stackrel{def}{=} \{(\bar{r}, 4,5)\}$:NIL η οποία προσφέρει 4 μονάδες ενέργειας από τον πόρο r έναντι του ποσού των € 5 ανά μονάδα ενέργειας και μετά τερματίζει. Επίσης έστω η εργασία $T \stackrel{def}{=} a?:\{(r,2,3)\}$:NIL η οποία ζητά να λάβει ένα μήνυμα a και έπειτα ζητά 2 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 3 και ακολούθως τερματίζει. Και τέλος οι δυνατότητες συντονισμού $P_1 \stackrel{def}{=} a!:P_1'$, οι οποίες στέλνουν το μήνυμα a και ακολούθως εξελίσσονται σε P_1' και οι δυνατότητες συντονισμού $P_2 \stackrel{def}{=} NIL$, οι οποίες δεν προσφέρουν κάποιο μήνυμα.

Επομένως από την παράλληλη σύνθεση της εργασίας T, της προμήθειας S και των δυνατοτήτων συντονισμού P_1, προκύπτει μια πιθανή μετάβαση η εξής: $[T||S||P_1] \xrightarrow{\{(\bar{r},2,3,5)\}} [NIL||NIL||P_1']$, και έτσι προκύπτει ότι το σύστημα μας είναι χρονοπρογραμματίσιμο.

Ενώ από την παράλληλη σύνθεση της εργασίας T, της προμήθειας S και των δυνατοτήτων συντονισμού P_2, προκύπτει επίσης μια πιθανή μετάβαση η εξής: $[T||S||P_2] \xrightarrow{\{(\bar{r},4,5),(r,2,3))\}}$ [NIL||NIL||NIL], και έτσι προκύπτει ότι το σύστημα μας είναι μη-χρονοπρογραμματίσιμο αφού η μετάβαση περιέχει ανικανοποίητα αιτήματα.

4.3 Υπόλοιπο προμήθειας

Έστω προμήθεια S, οι δυνατότητες συντονισμού P και εργασία T όπου $S||P \models T$. Ορίζουμε ως υπόλοιπο προμήθειας τους πόρους που απομένουν στην προμήθεια S μετά την ικανοποίηση των αιτήσεων της εργασίας T.

Μελετώντας λοιπόν τις εργασίες T, τις χορηγίες S και τις δυνατότητες συντονισμού P καταλήξαμε στο συμπέρασμα πως πρέπει να διαχειριζόμαστε μαζί τις εργασίες T με τις προμήθειες S και τις δυνατότητες συντονισμού P. Ακολουθεί η περιγραφή της συνάρτησης για το rem:

rem(S||P,T): Η συνάρτηση αυτή υπολογίζει τη διαθέσιμη προμήθεια και τους εναπομείναντες συντονισμούς έπειτα από την ικανοποίηση όλων των σχετικών απαιτήσεων της εργασίας T από την προμήθεια S και τους διαθέσιμους συντονισμούς P.

Πριν δώσουμε τον ορισμό του υπολοίπου προμήθειας, θα δώσουμε κάποιους βοηθητικούς ορισμούς

Ορισμός 4.2

‘Εστω οι δυνατότητες συντονισμού P. Ορίζουμε ως before(P) τα γεγονότα που είναι διαθέσιμα μέχρι την πρώτη μονάδα χρόνου.

$$\text{before(P)} \stackrel{\text{def}}{=} \begin{cases} NIL & αν \quad P = NIL \\ \prod_{i \in I} before(P_i) & αν \quad P = \prod_{i \in I} P_i \\ e: before(P') & αν \quad P = e: P' \end{cases}$$

Ο υπολογισμός των διαθέσιμων γεγονότων μέχρι την πρώτη μονάδα χρόνου γίνεται με τη χρήση της συνάρτησης before(P). Υπάρχουν οι εξής περιπτώσεις:

1) Αν P=NIL τότε δεν υπάρχουν διαθέσιμα γεγονότα μέχρι την πρώτη μονάδα χρόνου.

2) Αν $P = \prod_{i \in I} P_i$ τότε τα διαθέσιμα γεγονότα μέχρι την πρώτη μονάδα χρόνου είναι ίσα με την παράλληλη σύνθεση των μη-ντετερμινιστικών επιλογών του αναδρομικού υπολογισμού του before(P_i).
3) Αν $P = e: P'$ τότε τα διαθέσιμα γεγονότα μέχρι την πρώτη μονάδα χρόνου είναι ίσα με τον υπολογισμό του e:before(P').

Ορισμός 4.3

'Εστω οι δυνατότητες συντονισμού P. Ορίζουμε ως after(P) τα γεγονότα που είναι διαθέσιμα μετά την πρώτη μονάδα χρόνου.

$$\text{after(P)} \stackrel{\text{def}}{=} \begin{cases} NIL & \alpha\nu \quad P = NIL \\ \prod_{i \in I} after(P_i) & \alpha\nu \quad P = \prod_{i \in I} P_i \\ after(P') & \alpha\nu \quad P = e: P' \end{cases}$$

Ο υπολογισμός των διαθέσιμων γεγονότων μετά την πρώτη μονάδα χρόνου γίνεται με τη χρήση της συνάρτησης after(P). Υπάρχουν οι εξής περιπτώσεις:

1) Αν P=NIL τότε δεν υπάρχουν διαθέσιμα γεγονότα μετά την πρώτη μονάδα χρόνου.
2) Αν $P = \prod_{i \in I} P_i$ τότε τα διαθέσιμα γεγονότα μετά την πρώτη μονάδα χρόνου είναι ίσα με την παράλληλη σύνθεση των μη-ντετερμινιστικών επιλογών του αναδρομικού υπολογισμού του after(P_i).
3) Αν $P = e: P'$ τότε τα διαθέσιμα γεγονότα μετά την πρώτη μονάδα χρόνου είναι ίσα με τον αναδρομικό υπολογισμό του after(P').

Ορισμός 4.4

'Εστω η εργασία T. Ορίζουμε ως split(T) τα νέα ζεύγη που δημιουργούνται και που συνδέουν κάθε "before" με το σχετικό "after".

$$\text{split (T)} \stackrel{\text{def}}{=} \begin{cases} \{(NIL, NIL)\} & \alpha\nu \quad T = NIL \\ \{(NIL, a: T')\} & \alpha\nu \quad T = a: T' \\ \{(e: B, A) || (B, A) \in split(T')\} & \alpha\nu \quad T = e: T' \\ \bigcup_{i \in I} split(T') & \alpha\nu \quad T = \sum_{i \in I} T_i \\ split(T') & \alpha\nu \quad T = C, C \stackrel{\text{def}}{=} T' \end{cases}$$

Ο υπολογισμός των νέων ζευγών που δημιουργούνται και που συνδέουν κάθε "before" με το σχετικό "after" γίνεται με τη χρήση της συνάρτησης split(T). Υπάρχουν οι εξής περιπτώσεις:

1) Αν $T = NIL$ τότε τα ζεύγη είναι τα $\{(NIL, NIL)\}$, δηλαδή δεν υπάρχουν διαθέσιμα ζεύγη.
2) Αν $T = a{:}\,T'$ τότε τα ζεύγη είναι τα $\{(NIL, a{:}\,T')\}$, δηλαδή υπάρχουν μόνο διαθέσιμα "after".
3) Αν $T = e{:}\,T'$ τότε τα ζεύγη είναι τα $\{(e{:}\,B, A)||(B, A) \in split(T')\}$, δλαδή υπάρχουν διαθέσιμα και "before" και "after".
4) Αν $T = \sum_{i\in I} T_i$ τότε τα ζεύγη είναι τα $\bigcup_{i\in I} split(T')$, δηλαδή τα ζεύγη ισούνται με τις ενώσεις όλων των συναρτήσεων split(T').
5) Αν $T = C, C \stackrel{\text{def}}{=} T'$ τότε τα ζεύγη είναι ίσα με τον αναδρομικό υπολογισμό της συνάρτησης split(T').

Ο Ορισμός 4.4 είναι σημαντικός λόγω του γεγονότος ότι διαφορετικές ακολουθίες γεγονότων οδηγούν σε διαφορετικές καταστάσεις.

Για παράδειγμα, αν $T \stackrel{\text{def}}{=} e_1{:}(e_2{:}T_1 + e_3{:}T_2)$ θα έχουμε split(T) = $\{(e_1{:}e_2{:}NIL, T_1), (e_1{:}e_3{:}NIL, T_2)\}$

Ορισμός 4.5

Έστω μια προμήθεια S, οι δυνατότητες συντονισμού P και μια εργασία T, όπου ισχύει S||P ⊨ T

rem(S||P,T)$\stackrel{\text{def}}{=}$

$$\begin{cases} S||P & \alpha\nu \quad T = NIL \\ rem(S||P, T') & \alpha\nu \quad T = C\ , C \stackrel{\text{def}}{=} T' \\ rem(S'||P, T) & \alpha\nu \quad S = D\ , D \stackrel{\text{def}}{=} S' \\ rem(S||P', T) & \alpha\nu \quad P = E\ , E \stackrel{\text{def}}{=} P' \\ before(P)||\sum_{j\in J}(\beta - \alpha_j).rem(S'||after(P), T_j) & \alpha\nu \quad S = \beta : S' \ \kappa\alpha\iota\ T = \sum_{i\in I}\alpha_i : T_i \\ & \quad \text{όπου}\ \ J = \{j \in I, \forall\, i \in I, \neg(\beta \oplus \alpha_j \prec \beta \oplus \alpha_i)\} \\ \sum_{i\in I} rem(\beta_i : S_i||P, T) & \alpha\nu \quad S = \sum_{i\in I}\beta_i : S_i \\ \sum_{(P_i,T_i)\in split(T)} rem(S||P||P_i, T') & \alpha\nu \quad P = \emptyset : P' \ \kappa\alpha\iota\ T = e : T' \quad \text{ή} \\ & \quad P = \prod_{i\in I} e_i : P_i \ \ \kappa\alpha\iota\ K = \{i \in I \mid e_i = e!\} = \emptyset \\ \sum_{k\in K} rem(S||\prod_{j\in I-k} e_i : P_i||P_k, T') & \alpha\nu \quad T = e : T' \ \kappa\alpha\iota\ \ P = \prod_{i\in I} e_i : P_i \\ & \quad \text{όπου} \quad K = \{i \in I \mid e_i = e!\} \neq \emptyset \\ \sum_{i\in I} rem(S||P, e_i : T_i) & \alpha\nu \quad T = \sum_{i\in I} e_i : T_i \end{cases}$$

όπου β-α={($\bar{r}$,s,c) ∈ β⊕α} και e ∈ {a?,a!} και αν e ∈ P τότε e=a!

Ο υπολογισμός του υπολοίπου μιας προμήθειας γίνεται με τη χρήση της συνάρτησης rem(S||P,T). Υπάρχουν οι εξής περιπτώσεις:

1) Αν T=NIL τότε το υπόλοιπο ορίζεται ως S||P.
2) Αν $T = C\ , C \stackrel{\text{def}}{=} T'$ τότε το υπόλοιπο ισούται με τον υπολογισμό του rem(S||P,T').
3) Αν $S = D\ , D \stackrel{\text{def}}{=} S'$ τότε το υπόλοιπο ισούται με τον υπολογισμό του rem(S'||P,T).
4) Αν $P = E\ , E \stackrel{\text{def}}{=} P'$ τότε το υπόλοιπο ισούται με τον υπολογισμό του rem(S||P',T).
5) Αν $S = \beta : S'\ \kappa\alpha\iota\ T = \sum_{i\in I}\alpha_i : T_i$ τότε το υπόλοιπο ισούται με τον υπολογισμό του before(P) παράλληλα με το άθροισμα όλων των δυνατών υπολοίπων που προκύπτουν από τις μη-ντετερμινιστικές δράσεις της εργασίας T (εάν i=1 τότε η εργασία T έχει

μόνο μια δράση). Ο υπολογισμός κάθε υπολοίπου γίνεται ξεχωριστά από τη διαφορά της δράσης της προμήθειας με τη δράση της ζήτησης (β-α_j) και τον αναδρομικό υπολογισμό του rem(S'||after(P),T_j). Επίσης βάσει του περιορισμού των τιμών του δείκτη j υπολογίζονται μόνο όλες οι πιθανές υπολειπόμενες ενέργειες που μπορούν να υπάρξουν βάσει των προτεραιοτήτων της σχέσης προτιμήσεως που έχουμε ορίσει.

6) Αν $S = \sum_{i\in I} \beta_i : S_i$ τότε το υπόλοιπο ισούται με το άθροισμα όλων των διαφορετικών υπολοίπων που προκύπτουν από τις μη-ντετερμινιστικές επιλογές της προμήθειας S (εάν i=1 τότε η προμήθεια S έχει μόνο μια δράση). Ο υπολογισμός κάθε υπολοίπου γίνεται ξεχωριστά βάσει του αναδρομικού υπολογισμού του rem(β_i:S_i||P,T).
7) Αν $P = \emptyset : P' \; και \; T = e : T'$ ή $P = \prod_{i\in I} e_i : P_i \; και \; K = \{i \in I \mid e_i = e!\} = \emptyset$ τότε το υπόλοιπο ισούται με το άθροισμα όλων των αναδρομικών υπολογισμών των rem(S||P||P_i,T') για τα οποία (P_i,T_i)∈split(T).
8) Αν $T = e : T' \; και \; P = \prod_{i\in I} e_i : P_i$ τότε το υπόλοιπο ισούται με με το άθροισμα όλων των αναδρομικών υπολογισμών των $rem(S || \prod_{j\in I-k} e_i : P_i || P_k, T')$ για τα οποία ισχύει ότι k∈ K και $K = \{i \in I \mid e_i = e!\} \neq \emptyset$.
9) Αν $T = \sum_{i\in I} e_i : T_i$ τότε το υπόλοιπο ισούται με το άθροισμα όλων των διαφορετικών υπολοίπων που προκύπτουν από τις μη-ντετερμινιστικές επιλογές της εργασίας T (εάν i=1 τότε η εργασία T έχει μόνο μια δράση). Ο υπολογισμός κάθε υπολοίπου γίνεται ξεχωριστά βάσει του αναδρομικού υπολογισμού του rem(S||P,e_i:T_i).

Πιο κάτω παρουσιάζουμε μερικά παραδείγματα στα οποία εφαρμόζουμε τον υπολογισμό του υπολοίπου προμήθειας όπως προκύπτει από τον ορισμό.

Παράδειγμα 4.3

Έστω η προμήθεια S $\stackrel{\text{def}}{=}$ {($\bar{r}_1$,25,3)}:{($\bar{r}_2$,10,4)}:NIL η οποία προσφέρει 25 μονάδες ενέργειας από τον πόρο r_1 έναντι του ποσού των € 3 ανά μονάδα ενέργειας την $1^{η}$ χρονική στιγμή ενώ την $2^{η}$ χρονική στιγμή προσφέρει 10 μονάδες ενέργειας από τον πόρο r_2 έναντι του ποσού των € 4 ανά μονάδα ενέργειας και μετά τερματίζει. Και έστω επίσης η εργασία
T $\stackrel{\text{def}}{=}$ {(r_1,5,2)}:{(r_2,8,3)}:NIL η οποία ζητά 5 μονάδες ενέργειας από τον πόρο r_1 την πρώτη χρονική στιγμή και έχει προτεραιότητα 2 και την δεύτερη χρονική στιγμή ζητά 8 μονάδες

ενέργειας από τον πόρο r_2 ,έχει προτεραιότητα 3 και ακολούθως τερματίζει. Οι δυνατότητες συντονισμού P μπορεί να είναι οτιδήποτε. Επομένως αν εφαρμόσουμε τη συνάρτηση του υπολοίπου rem(S||P,T) έχουμε:

$rem(S||P,T) \stackrel{def}{=} [P \parallel \{(\bar{r}_1,20,3)\}:\{(\bar{r}_2,2,4)\}:NIL]$

Παράδειγμα 4.4

Έστω η προμήθεια $S \stackrel{def}{=} \{(\bar{r}_1,10,2),(\bar{r}_2,15,6)\}:NIL$ η οποία προσφέρει 10 μονάδες ενέργειας από τον πόρο r_1 έναντι του ποσού των € 2 ανά μονάδα ενέργειας και προσφέρει 15 μονάδες ενέργειας από τον πόρο r_2 έναντι του ποσού των € 6 ανά μονάδα ενέργειας και μετά τερματίζει. Και έστω επίσης η εργασία $T \stackrel{def}{=} \{(r_1,5,7)\}:NIL+ \{(r_2,5,8)\}:NIL$ η οποία ζητά είτε 5 μονάδες ενέργειας από τον πόρο r_1 με προτεραιότητα 7, είτε ζητά 5 μονάδες ενέργειας από τον πόρο r_2 με προτεραιότητα 8 και ακολούθως τερματίζει. Οι δυνατότητες συντονισμού P μπορεί να είναι οτιδήποτε. Επομένως αν εφαρμόσουμε τη συνάρτηση του υπολοίπου rem(S||P,T) έχουμε:

$rem(S||P,T) \stackrel{def}{=} [P \parallel \{(\bar{r}_1,10,2),(\bar{r}_2,10,6)\}:NIL] + [P\|\{(\bar{r}_1,5,2),(\bar{r}_2,15,6)\}:NIL]$

Παράδειγμα 4.5

Έστω η προμήθεια $S \stackrel{def}{=} \{(\bar{r},5,3)\}:S$ η οποία προσφέρει 5 μονάδες ενέργειας από τον πόρο r έναντι του ποσού των € 3 ανά μονάδα ενέργειας και μετά εξελίσσεται πάλι σε S. Και έστω επίσης η εργασία $T \stackrel{def}{=} e_1!:\{(r,3,2)\}:NIL + e_2!:\{(r,2,4)\}:NIL$ η οποία είτε στέλνει ένα μήνυμα e_1 και μετά ζητά 3 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 2 και ακολούθως τερματίζει, είτε στέλνει ένα μήνυμα e_2 και μετά ζητά 2 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 4 και ακολούθως τερματίζει. Επίσης οι δυνατότητες συντονισμού P ισούνται με $P \stackrel{def}{=} NIL$. Επομένως αν εφαρμόσουμε τη συνάρτηση του υπολοίπου rem(S||P,T) έχουμε:

$rem(S||P,T) \stackrel{def}{=} [P\|e_1!\|\{(\bar{r},2,3)\}:S] + [P \parallel e_2! \parallel \{(\bar{r},3,3)\}:S]$

4.4 Ταυτόχρονη εξυπηρέτηση δύο εργασιών από μια προμήθεια

Θεώρημα 4.1-Αρχική διατύπωση

Εάν (S || P) $\models$ T_1 και rem(S||P,T_1) $\models$$T_2$ τότε (S||P) $\models$ T_1||T_2

Δηλαδή εάν η παράλληλη εκτέλεση της προμήθειας S με τις δυνατότητες συντονισμού P μπορεί να χρονοπρογραμματίσει μια εργασία T_1 και αν η υπολειπόμενη ενέργεια της προμήθειας S παράλληλα με τις διαθέσιμες δυνατότητες συντονισμού P που υπάρχουν για το εξωτερικό περιβάλλον έπειτα από την ικανοποίηση των αναγκών της εργασίας T_1, μπορεί να χρονοπρογραμματίσει μια εργασία T_2, τότε αυτό σημαίνει ότι η προμήθεια S παράλληλα με τις δυνατότητες συντονισμού P μπορεί να χρονοπρογραμματίσει τις εργασίες T_1 και T_2 ταυτόχρονα.

Παράδειγμα 4.6

Έστω η προμήθεια S$\stackrel{def}{=}\{(\bar{r},25,2)\}$:S η οποία προσφέρει 25 μονάδες ενέργειας από τον πόρο r έναντι του ποσού των € 2 ανά μονάδα ενέργειας και μετά εξελίσσεται πάλι σε S. Και έστω επίσης οι εργασίες T_1 και T_2. Η εργασία $T_1\stackrel{def}{=}$a?:{(r,5,4)}:T_1' ζητά να λάβει το μήνυμα a και μετά ζητά 5 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 4 και ακολούθως εξελίσσεται σε T_1'. Η εργασία $T_2\stackrel{def}{=}$a?:{(r,6,7)}:T_2' + b?:{(r,4,2)}:T_2'' η οποία είτε ζητά να λάβει ένα μήνυμα a και μετά ζητά 6 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 7 και ακολούθως εξελίσσεται σε T_2', είτε ζητά να λάβει ένα μήνυμα b και μετά ζητά 4 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 2 και ακολούθως εξελίσσεται σε T_2''. Επίσης οι δυνατότητες συντονισμού P ισούνται με P$\stackrel{def}{=}$ a!:b!:NIL, δηλαδή μπορούν να προσφέρουν τα μηνύματα a και b. Επομένως αν εφαρμόσουμε τη συνάρτηση του υπολοίπου rem(S||P,T_1) και rem(S||P,T_2) έχουμε:

rem(S||P,T_1) $\stackrel{def}{=}$ $\{(\bar{r},20,2)\}$:S rem(S||P,T_2) $\stackrel{def}{=}$ $\{(\bar{r},19,2)\}$:S + $\{(\bar{r},21,2)\}$:S

Όμως αν έχουμε την παράλληλη σύνθεση των T_1||T_2||S||P και γνωρίζοντας ότι ο τελεστής της παράλληλης σύνθεσης είναι δυαδικός, επομένως αν εκτελεστεί πρώτα η παράλληλη σύνθεση των T_2||P και η εργασία T_2 πάρει το μήνυμα a από τις δυνατότητες συντονισμού P, τότε μετά rem(S||P,T_2)$\not\models$ T_1. Ενώ αν η εργασία T_2 πάρει το μήνυμα b από τις δυνατότητες συντονισμού P, τότε μετά rem(S||P,T_2)$\models$ T_1.

Η άνωθεν παρατήρηση επομένως μας παρουσιάζει ένα πρόβλημα που έχει δημιουργηθεί. Για την επίλυση αυτού του προβλήματος θα δώσουμε ένα βοηθητικό ορισμό.

Ορισμός 4.6

Η διεργασία T είναι *a-ντετερμινιστική* αν για κάθε T⇒ T' και T'$\xrightarrow{\alpha}$ T'' τότε T' = α.T''. Αν a ∈ κανάλια(T_i)∩ κανάλια(T_j), τότε T_i,T_j πρέπει να είναι a-ντετερμινιστικές.

Ακολουθεί ένα παράδειγμα όπου εφαρμόζουμε τον Ορισμό 4.6.

Παράδειγμα 4.7

Έστω η προμήθεια S≝^{def}{$(\bar{r}, 10,2)$}:S η οποία προσφέρει 10 μονάδες ενέργειας από τον πόρο r έναντι του ποσού των € 2 ανά μονάδα ενέργειας και μετά εξελίσσεται πάλι σε S. Και έστω επίσης οι εργασίες T_1 και T_2. Η εργασία T_1≝^{def}a?:{(r,2,4)}:T_1' ζητά να λάβει το μήνυμα a και μετά ζητά 2 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 4 και ακολούθως εξελίσσεται σε T_1'. Η εργασία T_2≝^{def}a?:{(r,6,7)}:T_2' η οποία την πρώτη μονάδα χρόνου, ζητά να λάβει ένα μήνυμα a και μετά ζητά 6 μονάδες ενέργειας από τον πόρο r με προτεραιότητα 7 και ακολούθως εξελίσσεται σε T_2'. Επίσης οι δυνατότητες συντονισμού P ισούνται με P≝^{def} a!:a!:NIL, δηλαδή μπορούν να προσφέρουν το μήνυμα a δύο φορές. Βάσει του Ορισμού 4.6 αφού οι εργασίες T_1 και T_2 ζητούν το ίδιο μήνυμα a, επομένως οι εργασίες T_1 και T_2 πρέπει να είναι a-ντετερμινιστικές. Παρατηρούμε ότι οι εργασίες T_1 και T_2 είναι a-ντετερμινιστικές, επομένως αν εφαρμόσουμε τη συνάρτηση του υπολοίπου rem(S||P,T_1) και rem(S||P,T_2) έχουμε:

rem(S||P,T_1) ≝^{def} {$(\bar{r}, 8,2)$}:S rem(S||P,T_2) ≝^{def} {$(\bar{r}, 4,2)$}:S

Παρατηρούμε λοιπόν ότι χάρη στον Ορισμό 4.6 μπορούμε να επιλύουμε τα προβλήματα της μορφής που παρουσιάστηκαν στο Παράδειγμα 4.6. Συγκεκριμένα με τον Ορισμό 4.6, εξασφαλίζουμε ότι οι εργασίες θα έχουν ντετερμινιστική δομή σε περίπτωση που επιθυμούν να λάβουν το ίδιο μήνυμα. Κατ' επέκταση αν μια προμήθεια S με τις δυνατότητες συντονισμού P μπορεί να ικανοποιήσει τα αιτήματα μιας εργασίας και η υπολειπόμενη ενέργεια από την προμήθεια S μαζί με τους εναπομείναντες συντονισμούς του P μπορούν να

ικανοποιήσουν τα αιτήματα μιας άλλης εργασίας, τότε η προμήθεια S μαζί με τις δυνατότητες συντονισμού P μπορεί ταυτόχρονα να ικανοποιήσει τα αιτήματα των δύο αυτών εργασιών ταυτόχρονα δίχως να προκύψει κάποιο πρόβλημα, όπως προηγουμένως.

Σχολιασμός Ορισμού 4.6 Ο Ορισμός 4.6 μπορεί από την μια μεριά να μας βοηθά να εξασφαλίζουμε ότι δεν θα παρουσιαστούν προβλήματα κατά την παράλληλη σύνθεση, όμως από την άλλη μεριά μας δημιουργεί κάποιους περιορισμούς. Συγκεκριμένα δεν αφήνουμε στις εργασίες να έχουν τη δυνατότητα της επιλογής, όταν ζητούν να λάβουν το ίδιο μήνυμα, γεγονός που μας προσδίδει ένα μικρό μειονέκτημα. Όμως ο συγκεκριμένος περιορισμός δεν είναι τόσο σημαντικός διότι απλά κάνει τον ανταγωνισμό μεταξύ των εργασιών πιο δίκαιο, επειδή αν μια εργασία ζητούσε να λάβει είτε το μήνυμα a είτε το μήνυμα b και μια άλλη εργασία ζητούσε να λάβει μόνο το μήνυμα a, τότε η πρώτη εργασία θα μπορούσε να κλέψει το μήνυμα a από την δεύτερη εργασία και αυτό θα είχε ως αποτέλεσμα να έμενε η δεύτερη εργασία με ανικανοποίητα αιτήματα αφού δεν θα μπορούσε να λάβει τα μήνυμα a που ήθελε. Ενώ τώρα δεν υπάρχει περίπτωση μια εργασία να κλέψει το μήνυμα μιας άλλης εργασίας.

Ακολουθεί η επαναδιατύπωση του αρχικού Θεωρήματος 4.1, κάνοντας χρήση του ντετερμινισμού.

Θεώρημα 4.1-Επαναδιατύπωση

Εάν T_1 και T_2 εργασίες τέτοιες ώστε αν $a \in$ κανάλια(T_1) $\cap$ κανάλια(T_2) τότε οι εργασίες T_1,T_2 είναι a-ντετερμινιστικές, και επιπρόσθετα $(S \parallel P) \models T_1$ και $rem(S||P,T_1) \models T_2$ τότε $(S||P) \models T_1||T_2$.

Δηλαδή αν οι εργασίες T_1 και T_2 είναι a-ντετερμινιστικές σε κάθε κοινό τους κανάλι a, δηλαδή έχουν ντετερμινιστική δομή και επιπρόσθετα εάν η παράλληλη εκτέλεση της προμήθειας S με τις δυνατότητες συντονισμού P μπορεί να χρονοπρογραμματίσει μια εργασία T_1 και αν η υπολειπόμενη ενέργεια της προμήθειας S παράλληλα με τις διαθέσιμες δυνατότητες συντονισμού P που υπάρχουν για το εξωτερικό περιβάλλον έπειτα από την ικανοποίηση των αναγκών της εργασίας T_1, μπορεί να χρονοπρογραμματίσει μια εργασία T_2, τότε αυτό σημαίνει ότι η προμήθεια S παράλληλα με τις δυνατότητες συντονισμού P μπορεί να χρονοπρογραμματίσει τις εργασίες T_1 και T_2 ταυτόχρονα.

Στην συνέχεια θα αποδείξουμε δύο λήμματα, τα οποία θα μας βοηθήσουν στην απόδειξη του Θεωρήματος 4.1

Λήμμα 4.1

Έστω μια προμήθεια S, οι δυνατότητες συντονισμού P και μια εργασία T, για τις οποίες ισχύει η σχέση S||P⊨T. Τότε rem(S||P,T)$\xrightarrow{a'}$ R αν και μόνο αν S$\xrightarrow{\beta}$ S', T$\xrightarrow{a}$ T', P$\xrightarrow{\emptyset}$ P', T||S||P$\xrightarrow{a\oplus\beta}$ T'||S'||P' και α'=β-α, R=rem(S'||P',T').

Απόδειξη:

Η απόδειξη θα γίνει με επαγωγή στη δομή των διεργασιών S και T. Βασίζεται στον Ορισμό 4.5 του υπολοίπου προμήθειας rem(S||P,T)

1) Αν T=NIL

   Τότε rem(S||P,T) = S||P$\xrightarrow{a'}$ R, T$\xrightarrow{a}$ NIL όπου α=∅ και για κάθε δράση S$\xrightarrow{\beta}$ S', α'=β-∅ και R=rem(S'||P,NIL)

2) Αν T=C, C$\stackrel{\text{def}}{=}$T'

   Τότε rem(S||P,T) = rem(S||P,T') και το ζητούμενο ισχύει από την υπόθεση της επαγωγής

3) Αν S=D, D$\stackrel{\text{def}}{=}$S'

   Τότε rem(S||P,T) = rem(S'||P,T) και το ζητούμενο ισχύει από την υπόθεση της επαγωγής

4) Αν S=β:S'

   Υπάρχει μια υποπερίπτωση:

   - T=$\sum_{i\in I} a_i : T_i$

     Τότε rem(S||P,T)=before(P)||$\sum_{j\in J}(\beta - \alpha_j)$.rem(S'||after(P),$T_j$) όπου $J = \{j \in I, \forall\, i \in I, \neg(\beta \oplus \alpha_j \prec \beta \oplus \alpha_i)\}$. Έχουμε ότι rem(S||P,T)$\xrightarrow{a'}$ R αν και μόνο αν

α'=β-α_i και before(P)=NIL, R=rem(S'||after(P),T_i) για κάποιο i, όπου για κανένα j∈I, $\beta \oplus \alpha_i \prec \beta \oplus \alpha_j$. Προφανώς, ισχύει ότι S$\xrightarrow{\beta}$ S', T$\xrightarrow{\alpha_i}$ T_i', P$\xrightarrow{\emptyset}$ P'=after(P), και S||T||P$\xrightarrow{\beta \oplus \alpha_i}$ S'||T'||P' όπου α'=β-α_i, R=rem(S'||P',T_i) που είναι και το ζητούμενο

5) Αν S=$\sum_{i \in I} \beta_i$:S_i

   Τότε rem(S||P,T)=$\sum_{i \in I} rem(\beta_i : S_i || P, T)$. Επιπλέον rem(S||P,T)$\xrightarrow{a'}$ R αν και μόνο αν για κάποιο i∈I , rem(β_i:S_i||P,T)$\xrightarrow{a'}$R. Από την επαγωγική υπόθεση αυτό συμβαίνει αν και μόνο αν ισχύουν οι απαιτήσεις του Λήμματος.

Αυτό ολοκληρώνει την απόδειξη του Λήμματος 4.1 □

Λήμμα 4.2

Έστω μια προμήθεια S, οι δυνατότητες συντονισμού P και μια εργασία T, για τις οποίες ισχύει η σχέση S||P ⊨ T. Τότε rem(S||P,T) $\xrightarrow{\tau}$ R αν και μόνο αν P $\xrightarrow{e!}$ P', T $\xrightarrow{e?}$ T' και R=rem(S||P',T').

Απόδειξη:

Η απόδειξη θα γίνει με επαγωγή στη δομή των διεργασιών P και T. Βασίζεται στον Ορισμό 4.5 του υπολοίπου προμήθειας rem(S||P,T)

1) Αν T=C, C$\overset{\text{def}}{=}$T'

   Τότε rem(S||P,T) = rem(S||P,T') και το ζητούμενο ισχύει από την υπόθεση της επαγωγής

2) Αν P=E, E$\overset{\text{def}}{=}$P'

   Τότε rem(S||P,T) = rem(S||P',T) και το ζητούμενο ισχύει από την υπόθεση της επαγωγής

3) Αν P= $\prod_{i\in I} e_i : P_i'$

   Υπάρχει μια υποπερίπτωση:

   - T=e:T'

     Τότε rem(S||P,T) = $\sum k \in K\, rem(S||\prod_{j\in I-k} e_i : P_i||P_k, T')$ και επομένως rem(S||P,T)$\xrightarrow{\tau}$ R αν και μόνο αν P$\xrightarrow{e!}$P', T$\xrightarrow{e?}$ T', επομένως για κάποιο k όπου k $\in K$, και $K = \{i \in I \mid e_i = e!\} \neq \emptyset$, R= $rem(S||\prod_{j\in I-k} e_i : P_i||P_k, T')$ δηλαδή R=rem(S||P',T')

4) Αν $T = \sum_{i\in I} e_i : T_i$

   Τότε rem(S||P,T) = $\sum_{i\in I} rem(S||P, e_i : T_i)$ και επομένως rem(S||P,T)$\xrightarrow{\tau}$ R αν και μόνο αν για κάποιο i, $rem(S||P, T_i) \xrightarrow{\tau}$ R. Από την επαγωγική υπόθεση αυτό συμβαίνει αν και μόνο αν P$\xrightarrow{e!}$P', $T_i\xrightarrow{e?}$ T_i', R = $rem(S||P', T_i')$, και προφανώς ισχύει ότι P$\xrightarrow{e!}$P', $T_i\xrightarrow{e?}$ T_i' και R=rem(S||P',T')

Αυτό ολοκληρώνει την απόδειξη του Λήμματος 4.2 □

Απόδειξη Θεωρήματος 4.1

Υπάρχουν 2 περιπτώσεις:

Για να ισχύει η σχέση (S||P)$\models$ $T_1||T_2$ πρέπει να δείξουμε ότι 1) Αν S||P||$T_1||T_2\xrightarrow{a}$ X πρέπει να ισχύει $\alpha^{\mu} \cap R = \emptyset$ και το X να ικανοποιεί τις συνθήκες του Θεωρήματος 4.1. 2) Αν S||P||$T_1||T_2\xrightarrow{\tau}$ X πρέπει να ισχύει τότε ότι το X ικανοποιεί τις συνθήκες του Θεωρήματος 4.1.

1η Περίπτωση

Η απόδειξη θα γίνει με αντίφαση. Έστω ότι υπάρχει κάποια μετάβαση T||S||P$\Longrightarrow$ X όπου X$\xrightarrow{\alpha}$ και $\alpha^{\mu} \cap R \neq \emptyset$, δηλαδή υπάρχουν ανικανοποίητα αιτήματα. Γνωρίζουμε ότι η εργασία T_1 είναι χρονοπρογραμματίσιμη από την προμήθεια S και τις δυνατότητες συντονισμού P επομένως S||P $\models$ T_1, άρα μέχρι στιγμής δεν έχουμε κάποιο ανικανοποίητο αίτημα. Επιπρόσθετα από το Λήμμα 4.1 γνωρίζουμε ότι ισχύει το rem(S||P,T_1)$\models$ T_2, επομένως ούτε

μετά την ικανοποίηση των αιτημάτων της εργασίας T_2 υπάρχουν ανικανοποίητα αιτήματα. Επομένως βρήκαμε μια δράση α η οποία δεν περιέχει ανικανοποίητα αιτήματα. Επίσης από τον Ορισμό 3.6 της σχέσης προτιμήσεως γνωρίζουμε ότι η δράση α πρέπει να περιέχει τις ελάχιστες απαιτήσεις των εργασιών T_1 και T_2 σε πόρους, και έτσι με αυτό τον τρόπο θα υπερνικήσει όλες τις άλλες δράσεις που περιέχουν περισσότερες απαιτήσεις και επίσης πάλι βάσει του Ορισμού 3.6 της σχέσης προτίμησης, αφού η δράση α δεν περιέχει ανικανοποίητα αιτήματα θα υπερνικήσει και όλες τις άλλες δράσεις που περιέχουν ανικανοποίητα αιτήματα. Επομένως δεν μπορούμε ποτέ να οδηγηθούμε σε κάποια δράση α η οποία να περιέχει ανικανοποίητα αιτήματα. Άρα φτάσαμε σε αντίφαση και έτσι αποδείξαμε ότι αν $S||P||T_1||T_2 \xrightarrow{a} X$ πρέπει να ισχύει $\alpha^{\mu} \cap R=\emptyset$ και επίσης αποδείξαμε ότι το X να ικανοποιεί τις συνθήκες του Θεωρήματος 4.1.

2η Περίπτωση

Έστω ότι $S||P||T_1||T_2 \xrightarrow{\tau} X$ τότε το X πρέπει να ικανοποιεί τις συνθήκες του Θεωρήματος 4.1. Ακολουθεί η ανάλυση όλων των δυνατών υποπεριπτώσεων:

1η) Έστω ότι $P \xrightarrow{e!} P'$ και $T_1 \xrightarrow{e?} T_1'$, τότε θα έχουμε εσωτερική ενέργεια μεταξύ των δυνατοτήτων συντονισμού P και της εργασίας T_1. Από το Λήμμα 4.2 γνωρίζουμε ότι ισχύει $S||P' \models T_1'$ και επίσης η υπολειπόμενη ενέργεια μπορεί να ικανοποιήσει τα αιτήματα της εργασίας T_2. Επομένως ισχύει ότι $rem(S||P',T_1') \models T_2$ και άρα οι δυνατότητες συντονισμού P πρέπει να έχουν τουλάχιστον δύο μηνύματα e. Επομένως ικανοποιούνται οι συνθήκες του Θεωρήματος 4.1.

2η) Έστω ότι $P \xrightarrow{e!} P'$ και $T_2 \xrightarrow{e?} T_2'$, τότε θα έχουμε εσωτερική ενέργεια μεταξύ των δυνατοτήτων συντονισμού P και της εργασίας T_2. Από τον Ορισμό 4.6 γνωρίζουμε ότι για να ισχύει $S||P' \models T_1$ θα πρέπει αν οι εργασίες T_1 και T_2 ζητούν να λάβουν το ίδιο μήνυμα e, να είναι a-ντετερμινιστικές. Επίσης από το Λήμμα 4.2 γνωρίζουμε ότι ισχύει $S||P' \models T_1$ και επίσης η υπολειπόμενη ενέργεια μπορεί να ικανοποιήσει τα αιτήματα της εργασίας T_2. Επομένως ισχύει ότι $rem(S||P',T_1) \models T_2'$ και άρα οι δυνατότητες συντονισμού P πρέπει να έχουν τουλάχιστον δύο μηνύματα e. Επομένως ικανοποιούνται οι συνθήκες του Θεωρήματος 4.1.

3η) Έστω ότι $T_1 \xrightarrow{e!} T_1$' και $T_2 \xrightarrow{e?} T_2$', τότε η εργασία T_1 θα στείλει το μήνυμα e στις δυνατότητες συντονισμού P και στην συνέχεια οι δυνατότητες συντονισμού P θα στείλουν το μήνυμα e στην εργασία T_2. Ακολούθως συνεχίζουμε όπως συνεχίσαμε και στην δεύτερη υποπερίπτωση.

4η) Έστω ότι $T_2 \xrightarrow{e!} T_2$' και $T_1 \xrightarrow{e?} T_1$', τότε η εργασία T_2 θα στείλει το μήνυμα e στις δυνατότητες συντονισμού P και στην συνέχεια οι δυνατότητες συντονισμού P θα στείλουν το μήνυμα e στην εργασία T_1. Ακολούθως συνεχίζουμε όπως συνεχίσαμε και στην πρώτη υποπερίπτωση.

5η) Έστω ότι $T_1 \xrightarrow{e?} T_1$' και $T_2 \xrightarrow{e!} T_2$', τότε είτε η εργασία T_2 θα στείλει απευθείας το μήνυμα e στην εργασία T_1 και οι δυνατότητες συντονισμού P θα παραμείνουν με τουλάχιστον ένα μήνυμα e αφού από το Λήμμα 4.2 γνωρίζουμε ότι πρέπει να ισχύει $S||P \models T_1$', είτε η εργασία T_1 θα λάβει το μήνυμα e από τις δυνατότητες συντονισμού P αφού από το Λήμμα 4.2 γνωρίζουμε ότι πρέπει να ισχύει $S||P' \models T_1$' και ακολούθως η εργασία T_2 θα στείλει το μήνυμα e στις δυνατότητες συντονισμού P, επομένως και πάλι οι δυνατότητες συντονισμού P παραμένουν με τουλάχιστον ένα μήνυμα e, επομένως ικανοποιούνται οι συνθήκες του Θεωρήματος 4.1.

6η) Έστω ότι $T_2 \xrightarrow{e?} T_2$' και $T_1 \xrightarrow{e!} T_1$'. Συμμετρικά όπως την πέμπτη υποπερίπτωση.

Αυτό ολοκληρώνει την απόδειξη του Θεωρήματος 4.1 □

4.5 Συμπεράσματα

Σε ένα σύστημα, όπως το δικό μας, που διαχειρίζεται πόρους, ο υπολογισμός του υπολοίπου μιας προμήθειας είναι πολύ μεγάλης σημασίας. Με τον υπολογισμό του υπολοίπου, ένα σύστημα γνωρίζει πόσοι και ποιοι πόροι του έχουν απομείνει. Συνεπώς θα μπορεί να ελέγξει κατά πόσο είναι σε θέση να ικανοποιήσει ένα νέο αίτημα για ενέργεια σε πραγματικό χρόνο και ακόμη αν έχει τη δυνατότητα να εξυπηρετήσει παράλληλα πολλά αιτήματα, εάν τα αιτήματα αυτά έχουν έρθει ταυτόχρονα στο σύστημα, όπως αποδείξαμε στο θεώρημα ταυτόχρονης εξυπηρέτησης δύο εργασιών από μια προμήθεια.

Ένα από τα κυριότερα χαρακτηριστικά ενός συστήματος που διαχειρίζεται πόρους είναι να μπορεί να ικανοποιήσει παράλληλα όσο το δυνατό περισσότερα αιτήματα. Αυτό οφείλεται στο γεγονός ότι ένα τέτοιο σύστημα επιθυμεί να παρέχει ενέργεια σε όσο το δυνατό περισσότερους αγοραστές ταυτόχρονα. Αυτό θα εξασφαλίσει στο σύστημα το μέγιστο δυνατό κέρδος, το οποίο είναι και το ζητούμενο.

Έτσι με τον υπολογισμό του υπολοίπου μπορεί να δημιουργηθεί ένας δυναμικός αλγόριθμος χρονοπρογραμματισμού συστημάτων, ο οποίος να προσαρμόζει την υπολειπόμενη ενέργεια που διαθέτει το σύστημα, έπειτα από την εξυπηρέτηση της κάθε αίτησης, και με βάση την ποσότητα αυτή να αποφασίζει κατά πόσο θα αποδεχτεί καινούργιους καταναλωτές που θα προσεγγίσουν το σύστημα. Θα τους αποδεχτεί εφόσον είναι χρονοπρογραμματίσιμο από την διαθέσιμη/υπολειπόμενη ενέργεια.

Κεφάλαιο 5

Συμπεράσματα

5.1 Συμπεράσματα

Συστήματα παροχής πόρων υπάρχουν παντού. Υπάρχουν υπολογιστικά συστήματα, συστήματα διαχείρισης ενέργειας, συστήματα του πεδίου του υπολογιστικού πλέγματος (Grid Computing) και πιο πρόσφατα είναι τα συστήματα του πεδίου του υπολογιστικού νέφους (Cloud Computing). Όλα αυτά τα συστήματα παροχής πόρων έχουν κοινά χαρακτηριστικά και χρησιμοποιούνται για να επιλύουν παρόμοιας φύσης προβλήματα. Συγκεκριμένα διαχειρίζονται και παρέχουν πόρους στους χρήστες του συστήματος ανάλογα με τις απαιτήσεις των χρηστών και βάσει των διαθέσιμων ποσοτήτων πόρων του συστήματος.

Εμείς σε αυτή την διπλωματική εργασία θέλαμε να δημιουργήσουμε ένα πρότυπο για την μοντελοποίηση και την ανάπτυξη τέτοιων συστημάτων, ιδιαιτέρα συστημάτων που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους. Επίσης θέλαμε αυτή μας η μοντελοποίηση να γίνει όσο πιο ρεαλιστική γίνεται για αυτό το λόγο ενσωματώσαμε στο μοντέλο μας και δυνατότητες επικοινωνίας. Μοντελοποιήσαμε μια απλοποιημένη μορφή ενός αλγορίθμου που έχει εφαρμογές στο υπολογιστικό νέφος και ταυτόχρονα δώσαμε την μοντελοποίηση διαφόρων πολιτικών χρονοδρομολόγησης όπως την πολιτική EDF, τη μη-διακοπτόμενη πολιτική, τη διακοπτόμενη πολιτική, την πολιτική Min-Min, την πολιτική Max-Min και την πολιτική FIFO η οποία κάνει χρήση καναλιών. Επιπρόσθετα μπορέσαμε να ενσωματώσουμε την έννοια της δυνατότητας επικοινωνίας συντονισμού με την χρήση των καναλιών.

Η ενσωμάτωση στην μοντελοποίηση της επικοινωνίας συντονισμού με την χρήση των καναλιών, ήταν ιδιαίτερα δύσκολη, διότι έπρεπε να επιτύχουμε συντονισμό μεταξύ των εργασιών και των δυνατοτήτων συντονισμού. Πιο συγκεκριμένα το πρόβλημα το οποίο αντιμετωπίσαμε ήταν στην συνθετική ανάλυση της άλγεβρας διεργασιών, στο σημείο όπου υπήρχε ανταγωνισμός στο ποιος θα λάβει τους διαθέσιμους συντονισμούς. Για να λύσουμε το πρόβλημα εισαγάγαμε τον Ορισμό 4.6.

Επίσης δώσαμε τον ορισμό του χρονοπρογραμματισμού, ο οποίος δηλώνει πότε μια διεργασία μπορεί να χρονοπρογραμματιστεί από ένα σύνολο προμηθειών και δυνατοτήτων συντονισμού. Στη συνέχεια δώσαμε τον ορισμό του υπολοίπου προμήθειας. Η δυσκολία που αντιμετωπίσαμε στον ορισμό του υπολοίπου προμήθειας, ήταν το γεγονός ότι έπρεπε να διαχειριζόμαστε ταυτόχρονα τις διαθέσιμες δυνατότητες συντονισμού που υπήρχαν διαθέσιμες αυτή την χρονική στιγμή για το σύστημα με τις διαθέσιμες ποσότητες πόρων που παρείχε το σύστημα, αλλά και να γνωρίζουμε τις διαθέσιμες δυνατότητες που θα είναι διαθέσιμες για το εξωτερικό περιβάλλον μετά την πρώτη μονάδα χρόνου. Το πρόβλημα αυτό το λύσαμε με χρήση βοηθητικών συναρτήσεων.

Τέλος αποδείξαμε ότι αν ένα σύνολο από προμήθειες και δυνατότητες συντονισμού μπορεί να χρονοπρογραμματίσει μια εργασία και το υπόλοιπο της προμήθειας μια άλλη εργασία, τότε το σύνολο αυτό μπορεί να χρονοπρογραμματίσει και τις δύο εργασίες μαζί. Για την απόδειξη αυτού του θεωρήματος, αντιμετωπίσαμε πολλές δυσκολίες λόγω του γεγονότος ότι ήταν αρκετά σύνθετος ο ορισμός του υπολοίπου προμήθειας. Συγκεκριμένα χρειάστηκε να ορίσουμε δύο βοηθητικά λήμματα, όπου το πρώτο χειριζόταν τις περιπτώσεις που σχετίζονταν με την προμήθεια και το δεύτερο τις περιπτώσεις που σχετίζονταν με τις εσωτερικές ενέργειες. Για όλα τα λήμματα χρειάστηκε να δώσουμε μια ιδιαίτερα λεπτομερή ανάλυση όλων των δυνατών υποπεριπτώσεων που υπήρχαν.

Η απόδειξη αυτή, του άνωθεν θεωρήματος είναι μεγίστης σημασίας, για τους λόγους που έχουμε αναφέρει και εξηγήσει στο τέλος του τέταρτου κεφαλαίου.

Συμπεραίνουμε λοιπόν ότι η χρονική άλγεβρα, ACCED (Algebra for Communication and Costing Energy Demand) που αναπτύξαμε, μας επιτρέπει να μοντελοποιούμε την ζήτηση, την χορηγία και την κατανάλωση, καθώς μας δίνει την δυνατότητα να μοντελοποιούμε, να

ελέγχουμε και να επαληθεύουμε αλγόριθμους που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους άλλα και συναφών πεδίων που σχετίζονται με διαχείριση πόρων. Επίσης μας παρέχει τη δυνατότητα επικοινωνίας συντονισμού με την χρήση καναλιών. Η άλγεβρα αυτή λοιπόν μας δίνει όλα τα οφέλη που αναφέρθηκαν παραπάνω.

5.2 Μελλοντικές Εργασίες

Υπάρχουν αρκετές κατευθύνσεις για μελλοντικές μελέτες στις οποίες θα μπορούσε να επεκταθεί η συγκεκριμένη διπλωματική εργασία. Μερικές από αυτές είναι οι ακόλουθες. Καταρχάς θα μπορούσε να γίνει μοντελοποίηση και άλλων αλγορίθμων που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους άλλα και άλλων παρόμοιων πεδίων όπου έχουν να κάνουν με τη διαχείριση και την παροχή πόρων. Επίσης θα μπορούσε να γίνει και μοντελοποίηση και άλλων πολιτικών χρονοδρομολόγησης, που κάνουν χρήση καναλιών και χρειάζονται επικοινωνία συντονισμού μεταξύ των καναλιών, π.χ. η πολιτική χρονοδρομολόγησης LIFO. Ακόμη θα μπορούσαμε να δημιουργήσουμε προσομοιώσεις των αλγορίθμων που έχουμε μοντελοποιήσει ώστε να τους συγκρίνουμε μεταξύ τους και να βρίσκουμε ποιος είναι ο καλύτερος ανάλογα με την παράμετρο που θέλουμε να αξιολογήσουμε κάθε φορά [12]. Επιπρόσθετα θα μπορούσε να γίνει και υλοποίηση παράλληλων αλγορίθμων που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους ή άλλων συναφών πεδίων που έχουν να κάνουν με τη διαχείριση και την παροχή πόρων, καθώς και σχεδιασμός και ανάλυση νέων αλγορίθμων μέσω της άλγεβρας διεργασιών ACCED. Επίσης θα μπορούσαμε να κάνουμε ακόμη πιο ρεαλιστική την μοντελοποίηση, ενσωματώνοντας την έννοια του προϋπολογισμού, δηλαδή οι αγοραστές να διαθέτουν ένα ορισμένο χρηματικό ποσό βάσει του οποίου να ελέγχεται εάν μπορούν να ικανοποιήσουν τις ανάγκες τους. Τέλος θα μπορούσε να γίνει μελέτη σε περισσότερο βάθος των δυνατοτήτων συντονισμού που μας παρέχει η αλγοριθμική προσέγγιση του MapReduce, με σκοπό τη μοντελοποίηση και άλλων αλγόριθμων που έχουν εφαρμογή στο πεδίο του υπολογιστικού νέφους.

Βιβλιογραφία

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing", Communication of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[2] H. Ben-Abdallah, J. Y. Choi, D. Clarke, Y. S. Kim, I. Lee, and H. L. Xie, "A Process Algebraic Approach to the Schedulability Analysis of Real-Time Systems", Real-Time Systems, vol. 15, no. 3, pp. 189-219, 1998.

[3] S. Bouchenak, G. Chockler, H. Chockler, G. Gheorghe, N. Santos, and A. Shraer, "Verifying cloud services: present and future", Operating Systems Review, vol. 47, no. 2, pp. 6-19, 2013.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rosa, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software-Practice and Experience, vol. 41, no.1, pp. 23-50, 2011.

[5] J. Dean, and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters", OSDI 2004: pp. 137-150, 2004.

[6] Hadoop: http://hadoop.apache.org/.

[7] I. Lee, P. Brémond-Grégoire, and R. Gerber, "A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems", in Proceedings of the IEEE 1994: pp. 158-171, 1994.

[8] I. Lee, A. Philippou, and O. Sokolsky, "A General Resource Framework for Real-Time Systems", RISSEF 2002: pp. 234-248, 2002.

[9] I. Lee, A. Philippou, and O. Sokolsky, “Process Algebraic modeling and analysis of power-aware real-time systems”, Computing and Control Engineering Journal, vol. 13, no. 4, pp. 180-188, 2002.

[10] R. Milner, “Communication and Concurrency”, Prentice Hall, London, 1989.

[11] I. A. Moschakis, and H. D. Karantza, “Evaluation of gang scheduling performance and cost in a cloud computing system”, The Journal of Supercomputing, vol. 59, no. 2, pp. 975-992, 2012.

[12] L. T. X. Phan, Z. Zhang, Q. Zheng, B. T. Loo, and I. Lee, “An empirical analysis of scheduling techniques for real-time cloud-based data processing”, SOCA 2011: pp. 1-8, 2011.

[13] A. Philippou, I. Lee, and O. Sokolsky, “PADS: An approach to modeling resource demand and supply for the formal analysis of hierarchical scheduling”, Theoretical Computer Science, vol. 413, no. 1, pp. 2-20, 2012.

[14] A. Philippou, I. Lee, O. Sokolsky and J. Y. Choi, “A Process Algebraic Framework for Modeling Resource Demand and Supply”, FORMATS 2010: pp. 183-197, 2010.

[15] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges”, Journal Internet Services and Applications, vol. 1, no. 1, pp. 7-18, 2010.

[16] Φ. Ιωάννου, “AESC: Μια άλγεβρα διεργασιών για την μοντελοποίηση και ανάλυση συστημάτων διαχείρισης πόρων”, Διπλωματική Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, Μάιος 2012.

[17] Μ. Μιχαήλ, “Επέκταση προτύπου για την ανάλυση της ζήτησης σε συστήματα διαχείρισης πόρων με δυνατότητα αποθήκευσης”, Διπλωματική Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, Μάιος 2013.

[18] Α. Φιλίππου, Διάλεξη: “Άλγεβρες Διεργασιών”, Σημειώσεις μαθήματος ΕΠΛ664: “Ανάλυση και Επαλήθευση Συστημάτων”, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, Γενάρης 2012.