Επαλήθευση Έξυπνης Σύμβασης με χρήση του περιβάλλοντος **UPPAAL**

Πτυχιακή Εργασία

Βαϊδομαρκάκης Παναγιώτης

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών
Εθνικό Μετσόβιο Πολυτεχνείο
Τομέας Μαθηματικών

Επιβλέπων: Στεφανέας Πέτρος, Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα 2020

Επαλήθευση Έξυπνης Σύμβασης με χρήση του περιβάλλοντος **UPPAAL**

Πτυχιακή Εργασία

Βαϊδομαρκάκης Παναγιώτης

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών
Εθνικό Μετσόβιο Πολυτεχνείο
Τομέας Μαθηματικών

Επιτηρητής: Στεφανέας Πέτρος, Επίκουρος Καθηγητής Ε.Μ.Π.
Εγκρίθηκε από τριμελή επιτροπή στις 7 Οκτωβρίου 2020

Στεφανέας Πέτρος	Παγουρτζής Αριστείδης	Ζάχος Ευστάθιος
Επ. Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Ομ. Καθηγητής Ε.Μ.Π.

Αθήνα 2020

(Υπογραφή)

.....
Βαϊδομαρκάκης Παναγιώτης

Abstract

The purpose of this thesis is the development of a smart contract and its verification through the UPPAL system.

First, a short review and explanation of a smart contract's use and application is presented. Then, its basic characteristics along with its respective advantage against the pre-existing way of contract is analyzed. In the end of the second chapter, a cost analysis highlights the smart contract's superior use.

Second, a short overview of database definition and the way a database works is presented. Specifically, this thesis's focus is on the interaction among various databases, the way data is stored and the respective data security protocols such smart contracts offer.

Third, an analysis of the UPPAL system and a short overview of various system definitions is presented. A thorough overview of the timed automata connection along with the specifics of their programming language is then analyzed. Definition of how time passes by in UPPAAL and how the graphical environment works (with the use of the editor, the simulator and the verifier) is in the last sections of this chapter.

In the final chapter, two simultaneous transactions are modeled in UPPAAL using a two-phase locking system. The basic framework of two transactions is designed, followed by the necessary modifications that make the model functional using the two-phase locking system. In the end, after executing the simulation, the two-phase locking system model is verified and the two transactions are isolated.

Ευχαριστίες

Για την διεκπεραίωση της παρούσας πτυχιακής εργασίας, θα ήθελα να ευχαριστήσω θερμά τον επιβλέπον καθηγητή μου κύριο Στεφανέα Πέτρο για την πολύτιμη βοήθειά του καθώς και τη συμφοιτήτριά μου Αναστασιάδη Έλλη που με ενέπνευσε να ασχοληθώ με το θέμα αυτό.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την ακλόνητη συμπαράστασή τους.

Περιεχόμενα

1 Εισαγωγή 3

2 Έξυπνη Σύμβαση 4
2.1 Εισαγωγή 4
2.2 Η ιστορία των έξυπνων συμβάσεων 4
2.3 Ορισμός και Χαρακτηριστικά 5
2.4 Βασικά Στοιχεία έξυπνης σύμβασης 6
2.5 Ροές δεδομένων έξυπνων συμβάσεων 7
2.6 Ταξινόμηση των έξυπνων συμβάσεων 8
2.7 Τα πλεονεκτήματα των έξυπνων συμβάσεων 9
2.8 Προκλήσεις για βελτίωση 10
2.9 Ανάλυση κόστους-οφέλους της έξυπνης σύμβασης 11
2.9.1 Ανάλυση Κόστους 11
2.9.1.1 Κόστος της έξυπνης σύμβασης 11
2.9.1.2 Κόστος αποδοχής των μπλοκ αλυσίδων 12
2.9.2 Ανάλυση οφέλους 13
2.9.3 Ωφέλιμα συμπεράσματα 14

3 Βάσεις Δεδομένων 15
3.1 Ορολογία 15
3.2 Ιστορία 17
3.3 Αλληλεπίδραση βάσεων δεδομένων 18
3.3.1 Σύστημα διαχείρισης βάσης δεδομένων 18
3.3.2 Εφαρμογή σύνδεσης με σύστημα διαχείρισης βάσης δεδομένων 19
3.3.3 Διεπαφή Προγράμματος Εφαρμογής 20
3.3.4 Γλώσσες Βάσεων Δεδομένων 20
3.4 Αποθήκευση βάσεων δεδομένων 21
3.5 Ασφάλεια βάσεων δεδομένων 21
3.6 Συναλλαγές και ταυτοχρονισμός 22
3.6.1 Αισιόδοξος έλεγχος ταυτοχρονισμού 23
3.6.1.1 Φάσεις ελέγχου αισιοδοξίας 23
3.6.1.2 Χρήση στον ιστό 24
3.6.1.3 Σύστημα Vector 24

4 **UPPAAL** 26
4.1 Εισαγωγή 26
4.2 Χρονισμένα Αυτόματα 26
4.3 Η γλώσσα μοντελοποίησης 27
4.3.1 Δίκτυα χρονισμένων αυτομάτων 27
4.3.2 Χρονισμένα Αυτόματα Στην UPPAAL 29
4.3.3 Εκφράσεις στο UPPAAL 30
4.4 Η γλώσσα επερώτησης στο UPPAAL 31
4.5 Κατανοώντας το χρόνο 34
4.6 Επισκόπηση της Εργαλειοθήκης του UPPAAL 36
4.6.1 Γραφικό περιβάλλον της Java 36
4.6.1.1 Ο Συντάκτης 36
4.6.1.2 Ο Προσομοιωτής 37
4.6.1.3 Ο Επαληθευτής 39
4.6.2 Ο αυτόνομος Επαληθευτής 39

5 Επαλήθευση κλειδώματος δύο φάσεων στο **UPPAAL** 41
5.1 Η έννοια της συναλλαγής 41
5.2 Έλεγχος ταυτότητας 42
5.3 Μοντελοποίηση ελέγχου επικαιρότητας συναλλαγής 43
5.4 Βασικός σκελετός συναλλαγής 44
5.5 Σκελετοί και σχέδια ελέγχου ταυτότητας 45
5.6 Έλεγχος Μοντέλου επικαιρότητας υπό αυστηρό κλείδωμα δύο φάσεων 47
5.7 Επαλήθευση επικαιρότητας 50
5.8 Μοντέλο ελέγχου απομόνωσης συναλλαγής 51
5.8.1 Μία επισκόπηση της απομόνωσης 51
5.9 Ένα μοντέλο παρατηρητή για απομόνωση 53
5.9.1 Συμπεράσματα για την επαλήθευση (απομόνωση) του μοντέλου 55
5.10 Μελλοντικές επεκτάσεις 57

Κεφάλαιο 1

Εισαγωγή

Σκοπός αυτής της πτυχιακής εργασίας είναι η δημιουργία μίας έξυπνης σύμβασης καθώς επίσης και η επαλήθευση της ότι λειτουργεί μέσω του περιβάλλοντος UPPAAL.

Αρχικά, γίνεται μία σύντομη αναδρομή και επεξήγηση για τη χρησιμότητα και τις λειτουργίες των έξυπνων συμβάσεων. Αναλύονται τα βασικά τους χαρακτηριστικά, τι πλεονεκτήματα προσφέρουν έναντι των συμβάσεων που προϋπήρχαν και στο τέλος του δευτέρου κεφαλαίου γίνεται μία σύντομη ανάλυση κόστους αυτών με σκοπό την ανάδειξη της χρησιμότητά τους.

Στη συνέχεια, γίνεται μία σύντομη αναδρομή και επεξήγηση για το τι είναι και πως λειτουργεί μία βάση δεδομένων. Συγκεκριμένα, γίνεται εμβάθυνση στην αλληλεπίδραση μεταξύ των βάσεων δεδομένων, αναλύεται ο τρόπος αποθήκευσης πληροφοριών καθώς και το είδος ασφάλειας που προσφέρουν.

Ακολούθως, ξεκινάει η ανάλυση του περιβάλλοντος UPPAAL. Γίνεται μία σύντομη ανασκόπηση διαφόρων ορισμών όπως για παράδειγμα τα χρονισμένα αυτόματα. Αναλύεται πλήρως η σύνδεση χρονισμένων αυτομάτων, ο τρόπος γραφής στο UPPAAL, οι εκφράσεις που χρησιμοποιεί, η γλώσσα επερώτησης που υπάρχει, πως διαχειρίζεται το χρόνο και ολοκληρώνεται με τη χρήση του γραφικού περιβάλλοντος του UPPAAL όπου υπάρχει ο συντάκτης, ο προσομοιωτής και ο επαληθευτής.

Στο τελευταίο κεφάλαιο, μέσω του προγράμματος UPPAAL, γίνεται η μοντελοποίηση δύο ταυτόχρονων συναλλαγών που χρησιμοποιούν το μοντέλο κλείδωμα δύο φάσεων. Σχεδιάζεται ο βασικός σκελετός δύο συναλλαγών με την χρήση του UPPAAL. Στη συνέχεια γίνονται τροποποιήσεις για να λειτουργήσουν οι συναλλαγές με κλείδωμα δύο φάσεων και στο τέλος, εκτελώντας την προσομοίωση των συναλλαγών, επαληθεύεται το μοντέλο που δημιουργήθηκε όπως επίσης και η απομόνωση των συναλλαγών.

Κεφάλαιο 2

Έξυπνη Σύμβαση

2.1 Εισαγωγή

Μία έξυπνη σύμβαση (Smart Contract) είναι μία συμφωνία με χρήση ηλεκτρονικών υπολογιστών που έχει σαν στόχο να διευκολύνει μέσω των ψηφιακών συστημάτων, να κάνει επαλήθευση και να επιβληθεί, για να γίνει μία διαπραγμάτευση και εκτέλεση μίας σύμβασης. Οι έξυπνες συμβάσεις εκτελούν αξιόπιστες συναλλαγές χωρίς μεσάζοντες. Οι συναλλαγές αυτές είναι ανιχνεύσιμες αλλά δεν είναι αναστρέψιμες.

Άνθρωποι που υποστηρίζουν τις "έξυπνες συμβάσεις" θεωρούν ότι με την δημιουργία τους, μπορούνε να αυτοματοποιηθούν πολλά είδη συμβάσεων ή μέρος αυτών. Ο στόχος για τις έξυπνες συμβάσεις είναι η παροχή μεγαλύτερης ασφάλειας από το παραδοσιακό δίκαιο των συμβάσεων και η μείωση του κόστους συναλλαγών που είναι αλληλένδετα με την εκτέλεση συμβάσεων. Διάφορα κρυπτονομίσματα έχουν εφαρμόσει τύπους έξυπνων συμβάσεων.

2.2 Η ιστορία των έξυπνων συμβάσεων

Οι έξυπνες συμβάσεις προτάθηκαν αρχικά στις αρχές του 1990 από έναν υπολογιστικό επιστήμονα, δικηγόρο και κρυπτογράφο, τον Nick Szabo, ο οποίος εφάρμοσε τον όρο. Με τις παρούσες εφαρμογές, με βάση τις αλυσίδες μπλοκ(Blockchains), το "έξυπνο συμβόλαιο" χρησιμοποιείται ως επί το πλείστων πιο συγκεκριμένα υπό την έννοια του υπολογισμού γενικού σκοπού που λαμβάνει χώρα σε μία αλυσίδα μπλοκ ή σε ένα κατανεμημένο σύνολο αλυσίδων μπλοκ. Ένα έξυπνο συμβόλαιο λειτουργεί ως ένα πρόγραμμα που χρησιμοποιείται μέσω υπολογιστή και όχι με την κλασσική χειρόγραφη μορφή που υπάρχει μέχρι σήμερα, σύμφωνα με ιδρύματα όπως το Ethereum ή την IBM.

Μία έξυπνη σύμβαση αποτελεί μία απόλυτα ασφαλής διεργασία αφού τόσο η εκτέλεσή της όσο και τα αποτελέσματα τα οποία παράγει, γίνονται μέσα από αυστηρή διαδικασία και δεν μπορούν να εμπλακούν τρίτα άτομα. Για παράδειγμα, οι λεπτομέρειες μίας μεταβίβασης ανεξαρτήτου αξίας που μοιράζεται σε κάποια άτομα, αποθηκεύονται σε μία αλυσίδα μπλοκ ή σε ένα κατανεμημένο καθολικό. Αυτό

συμβαίνει διότι η πλατφόρμα εκτελεί πλήρως ελεγχόμενα τη σύμβαση και όχι τρίτα προγράμματα που έχουνε σύνδεση με το σύστημα.

Το 2018, μία έκθεση της Γερουσίας των ΗΠΑ δήλωσε: «Ενώ οι έξυπνες συμβάσεις μπορεί να ακούγονται καινούργιες, η έννοια της πηγάζει από το βασικό δίκαιο που ισχύει για τις κλασσικές συμβάσεις. Γενικά, το δικαστικό σύστημα εκδικάζει τις διαφορές των συμβάσεων, επιβάλλοντας συγκεκριμένους όρους, αλλά υπάρχει επίσης και άλλη μέθοδος διαιτησίας, για τις διεθνείς συναλλαγές. Με έξυπνες συμβάσεις, ένα πρόγραμμα επιβάλλει τη σύμβαση ενσωματωμένη στον κώδικα.

Η Λευκορωσία χαρακτηρίστηκε ως το πρώτο κράτος που έθεσε σε νομιμότητα τα έξυπνα συμβόλαια. Ο δικηγόρος Aleinikov Denis χαρακτηρίζεται ως ο πρώτος συντάκτης που χρησιμοποίησε τις νομικές γνώσεις του για να θεσπίσει τα έξυπνα συμβόλαια στη Λευκορωσία και να αναπτυχθεί η ψηφιακή οικονομία.

Όπως αναφέραμε και παραπάνω, ο δημιουργός των έξυπνων συμβάσεων θεωρείται ότι είναι ο κρυπτογράφος Szabo Nick όπου τις δημιούργησε στην Αμερική στα μέσα της δεκαετίας του ′90 (1994). Μάλιστα, αναφερότανε συχνά σε ένα παράδειγμα με ένα ενοικιαζόμενο αυτοκίνητο και ένα έξυπνο συμβόλαιο, έτσι ώστε ο έλεγχος του αυτοκινήτου να επιστρέφει στον ιδιοκτήτη του αυτοκινήτου όταν ένας ενοικιαστής πληρώνει τις οφειλές του. Τα έξυπνα συμβόλαια μπορούν να οριστούν ως αυτόνομα προγράμματα ηλεκτρονικών υπολογιστών (κώδικες που εκτελούνται μόνοι τους) όπου, μόλις ξεκινήσουν, εκτελούν αυτόματα και κατά τρόπο υποχρεωτικό τις βασικές προϋποθέσεις, όπως η διευκόλυνση, επαλήθευση ή επιβολή της διαπραγμάτευσης ή της εκτέλεσης α σύμβαση, εκτέλεση μίας συναλλαγής πληρωμής και ούτω καθεξής.

Το βασικό όφελος που υπάρχει μέσω της ανάπτυξης έξυπνων συμβάσεων είναι οι αλυσίδες μπλοκ όπου αποτελούν εγγύηση για την έξυπνη σύμβαση, καθώς δεν υπάρχει τρόπος να τροποποιηθούν. Οι αλυσίδες μπλοκ είναι αδύνατο να παραβιαστούν ή να παραβιάσουν τους όρους της σύμβασης. Έτσι, έξυπνα συμβόλαια που αναπτύχθηκαν με μία αλυσίδα μπλοκ αναμένεται να μειώσουν το κόστος της επαλήθευσης, της εκτέλεσης, της διαιτησίας καθώς και την πρόληψη της απάτης. Εφόσον έχουν εξαλείψει τον ηθικό κίνδυνο, τα έξυπνα συμβόλαια μπορούν να φανούν χρήσιμα.

2.3 Ορισμός και Χαρακτηριστικά

Ένα έξυπνο συμβόλαιο ορίζεται ως μία ψηφιακή αλλά ταυτόχρονα και αξιόπιστη σύμβαση ανάμεσα σε δύο ή περισσότερα μέλη. Μία τρίτη εικονική ˝οντότητα˝, ένας λογισμικός πράκτορας, αποφασίζει και πραγματοποιεί, κάποιες από τις προϋποθέσεις αυτής της σύμβασης. Στο πλαίσιο της αλυσίδας μπλοκ, μία έξυπνη σύμβαση είναι ένα πρόγραμμα που καθοδηγείται από γεγονότα, με καταστάσεις, που τρέχει σε ένα αναπαραγόμενο, κοινό μητρώο και το οποίο μπορεί να πάρει την επιμέλεια επί περιουσιακών στοιχείων σε αυτό.

Τα έξυπνα συμβόλαια που κάνουνε χρήση των αλυσίδων μπλοκ, δημιουργήθηκαν από ανθρώπους που ξέρουν να προγραμματίζουν με χρήση ηλεκτρονικού υπολογιστή και επομένως, χρησιμοποιήθηκαν διάφορες γλώσσες προγραμματισμού για να γραφτεί η ψηφιακή μορφή τους. Ανάλογα με την περίσταση, η έξυπνη σύμ-

βαση έχει διαφορετικές προϋποθέσεις, επομένως και συνέπειες. Όπως έχει και μία χειρόγραφη νομική σύμβαση, η οποία δηλώνει ακριβώς τι παρέχει και τι κυρώσεις αναμένουν αυτούς που δεν θα την τηρήσουν. Η μεγαλύτερη διαφορά βρίσκεται στον τρόπο με τον οποίο εκτελείται, αυτόματα και με την μέγιστη ασφάλεια για την ελαχιστοποίηση κινδύνου. Ο κώδικας της έξυπνης σύμβασης έχει ορισμένα μοναδικά χαρακτηριστικά:

- Προσδιοριστικός (**Deterministic**):Δεδομένου ότι ένας έξυπνος κώδικας συμβάσεων εκτελείται σε πολλαπλούς κατανεμημένους κόμβους ταυτόχρονα, πρέπει να είναι προσδιοριστικός, δηλαδή δεδομένης μίας εισόδου, όλοι οι κόμβοι θα πρέπει να παράγουν το ίδιο αποτέλεσμα. Αυτό συνεπάγεται ότι στον κώδικας της έξυπνης σύμβασης είναι απαραίτητο να μην υπάρχει καμία τυχαιότητα. Οφείλει να μην εξαρτάται από το χρόνο (πρέπει να εκτελείται ανάμεσα σε χρονικά όρια, για το λόγο ότι το πρόγραμμα μπορεί να εκτελείται με μία ελάχιστη διαφορά ώρας για άλλη σύμβαση) και τέλος, οφείλει να είναι επαναληπτικός, δηλαδή να εκτελείται σωστά όσες φορές και να τον τρέξουμε.

- Αμετάβλητος (**Immutable**): Ο κώδικας ενός έξυπνου συμβολαίου οφείλει να παραμένει αμετάβλητος. Επομένως, με το που δημιουργηθεί και αναπτυχθεί ο κώδικας, τότε δεν μπορούμε να μεταβάλλουμε κανένα μέρος του. Παρ΄ όλο που στην θεωρία ακούγεται συναρπαστικό, διότι έτσι είμαστε βέβαιοι ότι θα είναι ένας ασφαλής κώδικας, αυτόματα δημιουργούνται ορισμένα προβλήματα, όπως για παράδειγμα δεν μπορεί να γίνει διόρθωση ενός λάθους στον κώδικα. Άρα, χρειάζεται αυξημένη επιμέλεια για τη δημιουργία του.

- Επαληθεύσιμος (**Verifiable**): Μετά την ανάπτυξή του, ο κώδικας του έξυπνου συμβολαίου αποκτά μία διεύθυνση. Μία μικρή ομάδα ατόμων ή άτομα τα οποία ενδιαφέρονται για την ανάπτυξή του οφείλουν να τον δουν και να κάνουν επαλήθευση πριν γίνει δημόσια η χρήση του.

2.4 Βασικά Στοιχεία έξυπνης σύμβασης

Μία έξυπνη σύμβαση μπορεί να χωριστεί σε δύο βασικά στοιχεία:

- Κώδικας Έξυπνης Σύμβασης: Ο κώδικας που αποθηκεύεται ελεγμένος και εκτελούμενος σε μία αλυσίδα μπλοκ.

- Έξυπνες Νομικές Συμβάσεις: Η χρήση του κώδικα έξυπνης σύμβασης που μπορεί να χρησιμοποιηθεί ως συμπλήρωμα ή υποκατάστατο των νομικών συμβάσεων.

Παρακάτω είναι μία σύντομη περιγραφή για το πως λειτουργούν οι έξυπνες συμβάσεις διαιρούμενες σε τρία μέρη, την εισαγωγή, τη διαδικασία και το αποτέλεσμα.(Kehrli, J. [2016])

- Κωδικοποίηση (τι συμβαίνει σε μία έξυπνη σύμβαση): Επειδή μία έξυπνη σύμβαση λειτουργεί όπως ένα πρόγραμμα του υπολογιστή, είναι πολύ σημαντικό να κάνει ακριβώς αυτό που ορίζουν τα συμβαλλόμενα μέλη να κάνει.

Αυτό μπορούμε να το πετύχουμε με χρήση κατάλληλης λογικής όταν ξεκινάει η ανάπτυξη της έξυπνης σύμβασης. Ο κώδικας συμπεριφέρεται με προκαθορισμένους τρόπους και δεν έχει τις γλωσσικές αποχρώσεις των ανθρώπινων γλωσσών. Επίσης, στις μέρες μας έχει γίνει αυτόματο το: «εάν γίνει αυτό, έπειτα κάνε εκείνο» κομμάτι των κλασσικών συμβάσεων.

- Καθολικό Σύστημα (πως ένα έξυπνο συμβόλαιο διαμοιράζεται): Μετά την κρυπτογράφηση του κώδικα, γίνεται η διανομή του με χρήση ενός δικτύου κόμβων, το οποίο εκτελείται σε ένα καθολικό σύστημα.

- Εκτέλεση (πως επεξεργάζεται): Όταν οι συνδεδεμένοι υπολογιστές έχουν τον κώδικα, τότε πρέπει να γίνει ομόφωνη αποδοχή από όλα τα μέλη που εμπεριέχει η σύμβαση με μοναδικό κριτήριο τα αποτελέσματα που βγήκαν κατά την εκτέλεσή του. Έπειτα, το καθολικό σύστημα ενημερώνεται από το δίκτυο για την καταγραφή της εφαρμογής της σύμβασης και αμέσως μετά, κάνει τον έλεγχο των προϋποθέσεων της συμφωνίας. Με αυτόν τον τύπο συστήματος, αποτρέπεται ο χειρισμός από μεμονωμένα μέλη, επειδή ο έλεγχος της εκτέλεσης της έξυπνης σύμβασης δεν είναι πλέον δυνατή εφόσον η εκτέλεση αυτή δεν είναι στα χέρια ενός μεμονωμένου μέλους.

2.5 Ροές δεδομένων έξυπνων συμβάσεων

Τα μαντεία (Oracles) είναι αξιόπιστες οντότητες που υπογράφουν ισχυρισμούς για την κατάσταση του κόσμου. Δεδομένου ότι η επαλήθευση των υπογραφών μπορεί να είναι προσδιοριστική, επιτρέπει στις προσδιοριστικές έξυπνες συμβάσεις να αντιδρούν στον (απροσδιόριστο) έξω κόσμο. Τα μαντεία είναι απαραίτητα για να συνδεθούν οι έξυπνες συμβάσεις με ροές δεδομένων, οποιοδήποτε είδος διεπαφής προγραμματισμού εφαρμογών Ιστού (Web API) είτε οποιαδήποτε μέθοδο πληρωμής που είναι αποδεκτή. Τα έξυπνα συμβόλαια εξετάζουν όλες τις συνθήκες πριν ξεκινήσει η εκτέλεσή τους. Αυτές οι προϋποθέσεις καθορίζονται από την αρχή στον κώδικα. Το μοναδικό πρόβλημα που προκύπτει είναι η επικύρωση των όρων εκτέλεσης.
Τα δυνατά σενάρια είναι δύο:

- Οι προϋποθέσεις εφαρμογής του συμβολαίου συνδέονται με διαφορετικές καταχωρήσεις στην αλυσίδα μπλοκ ή είναι χρονικά όρια. Όταν βρισκόμαστε σε αυτήν την επιλογή, ο έλεγχος των όρων εκτέλεσης είναι αρκετά απλός: το συμβόλαιο είναι προγραμματισμένο να κάνει έλεγχο για την ύπαρξη των καταχωρήσεων είτε ότι το χρονικό όριο εκτέλεσης έχει περάσει, και εκτελείται όταν ο παραπάνω έλεγχος περάσει.

- Οι προϋποθέσεις εφαρμογής του συμβολαίου βρίσκονται εκτός της αλυσίδας μπλοκ. Σε αυτήν την περίπτωση, η εφαρμογή της σύμβασης απαιτεί τη χρήση ενός εμπιστευμένου τρίτου, ενός μαντείου.

Ένας χρησμός κατασκευάζεται για να εισάγει τις πληροφορίες των αλυσίδων μπλοκ αξιόπιστα, έτσι ώστε η σύμβαση να μπορέσει να τρέξει κατάλληλα και μπορεί να συγκροτηθεί με διάφορους τρόπους (Kehrli, J. [2016]):

- Προγενέστερος προσδιορισμός ενός εμπιστευτικού τρίτου μέλους που είναι γνωστό από όλα τα συμβαλλόμενα μέρη.

- Κάνει αναφορά σε βάση πληροφοριών που χαρακτηρίζονται αξιόπιστες. (δηλαδή στην περίπτωση των στοιχηματικών αθλημάτων, η δυνατότητα να καταγράφεται το αποτέλεσμα σε μία περιοχή κάποιας αθλητικής εφημερίδας).

- Χρησιμοποιώντας μία αποκεντρωμένη υπηρεσία μαντείου. Είναι μία υπάρχουσα υπηρεσία στην αλυσίδα μπλοκ με πολλούς συμμετέχοντες. Κάθε συμμετέχων ψηφίζει για το αποτέλεσμα που θεωρεί ότι είναι ακριβές και είναι η συναίνεση μεταξύ των συμμετεχόντων που καθορίζει τον τελικό αποτέλεσμα που στέλνεται στη σύμβαση. Έχουν ήδη υπάρξει έργα αποκεντρωμένου μαντείου, σημαντικά όπως το έργο **Oraclize**.

Το μαντείο υπάρχει ανάμεσα στις πληροφορίες εξωτερικού κόσμου και στη διεπαφή προγραμματισμού εφαρμογών καθώς και του έξυπνου συμβολαίου.

2.6 Ταξινόμηση των έξυπνων συμβάσεων

Θα ταξινομήσουμε τις έξυπνες συμβάσεις με βάση την περιοχή εφαρμογής όσο αναφορά τις **Bitcoin** και τις **Ethereum** συμβάσεις, για τις πρώτες βασισμένοι στην έρευνα ιστοσελίδας τους και τα σχετικά φόρουμ συζήτησης και στη χειρωνακτική επιθεώρηση του κώδικα πηγής στερεότητας για οι συμβάσεις **Ethereum** όπως περιγράφεται στο σχετικό έγγραφο (**Bartoletti, M. and Pompianu, L.** [2017]). Τα αποτελέσματα αυτής της έρευνας είναι η κατηγοριοποίηση των έξυπνων συμβάσεων σε πέντε τομείς που αναλύονται παρακάτω:

i Χρηματοοικονομική :
Το κύριο χαρακτηριστικό αυτού του είδους των συμβάσεων είναι η διαχείριση συγκεκριμένου ποσού ή η επαλήθευση της ιδιοκτησίας αγαθών. Άλλες συμβάσεις χρησιμοποιούνται για τη συλλογή χρηματικού ποσού που έχουν ως αποτέλεσμα τη παροχή χρημάτων για συγκεκριμένα έργα. Ένα χρηματοοικονομικό συμβόλαιο είναι για παράδειγμα ένα υψηλής απόδοσης επενδυτικό πρόγραμμα που έχει σαν στόχο να βρει νέους επενδυτές μέσα από υποσχέσεις υψηλών επιτοκίων. Κάποια συμβόλαια εγγυώνται ασφάλεια όταν δεν πετύχουν και οι αποδείξεις τους είναι ψηφιακές ενώ άλλα έχουν σχέση με διαφημίσεις.

ii Συμβολαιογραφική :
Σε αυτόν τον τομέα, τα συμβόλαια κάνουν χρήση της σταθερότητας της μπλοκ αλυσίδας για να αποθηκεύσουν πληροφορίες και σε κάποιες περιπτώσεις, να κάνουν πιστοποιήσεις ιδιοκτησίας. Κάποιες από τις συμβάσεις αφήνουν τους χρήστες να γράφουν κομμάτια εγγράφων στην μπλοκ αλυσίδα, με σκοπό αργότερα να έχουν αποδείξεις σχετικά με την ύπαρξη ή την ακεραιότητα ενός εγγράφου. Άλλες συμβάσεις επιτρέπουν τη δήλωση πνευματικών δικαιωμάτων σε ψηφιακά μέσα όπως φωτογραφίες και μουσική ενώ μερικές

άλλες συμβάσεις επιτρέπουν στους χρήστες να καταγράφουν στις μπλοκ αλυσίδες μηνύματα που ο καθένας μπορεί να διαβάσει. Αυτός ο τομέας έχει συμβόλαια τα οποία κάνουν συσχετίσεις χρηστών με διευθύνσεις, με σκοπό να γίνει πιστοποίηση ταυτότητας.

iii Παιχνίδια :
Οι συμβάσεις σε αυτή την κατηγορία περιλαμβάνουν τυχερά παιχνίδια και παιχνίδια δεξιοτήτων που οι χρήστες επιθυμούν να συμμετάσχουν.

iv Πορτοφόλι :
Αυτές οι συμβάσεις χειρίζονται κλειδιά, διαχειρίζονται χρήματα που προορίζονται για συναλλαγές και αναπτύσσουν συμβάσεις, καθιστώντας την αλληλεπίδραση με την μπλοκ αλυσίδα απλή. Τα πορτοφόλια έχουν τη δυνατότητα να διαχειρίζονται από πολλούς χρήστες ταυτόχρονα, γεγονός που γίνεται εφικτό μέσω πολλαπλών εγκρίσεων.

v Βιβλιοθήκη :
Τα συμβόλαια αυτά χρησιμοποιούνται για διάφορους τύπους δραστηριοτήτων με σκοπό να είναι χρηστικά για άλλες συμβάσεις.

2.7 Τα πλεονεκτήματα των έξυπνων συμβάσεων

Για μεγάλο εύρος εφαρμογών, οι έξυπνες συμβάσεις που κατασκευάστηκαν με βάση τις μπλοκ αλυσίδες έχουν πολλά πλεονεκτήματα τα οποία αναφέρονται παρακάτω:

i Ταχύτητα και ενημέρωση σε πραγματικό χρόνο : Για το λόγο ότι τα έξυπνα συμβόλαια προσπαθούν να αυτοματοποιήσουν χειροκίνητες ενέργειες μέσω του κώδικα που έχουν γραφτεί, πολλές επιχειρηματικές διαδικασίες επωφελούνται από αυτό.

ii Ακρίβεια : Με την αυτοματοποίηση των συναλλαγών, όχι μόνο υπάρχει αύξηση της ταχύτητας, αλλά επιτυγχάνονται και λιγότερα σε πλήθος χειροκίνητα σφάλματα.

iii Χαμηλότερος κίνδυνος εκτέλεσης : Η αποκεντρωμένη διαδικασία της εκτέλεσης εξαλείφει ουσιαστικά τον κίνδυνο χειραγώγησης, της μη αποτελεσματικότητας ή σφαλμάτων, δεδομένου ότι η εκτέλεση ρυθμίζεται αυτόματα από το δίκτυο και όχι από ένα μεμονωμένο μέλος.

iv Λιγότεροι μεσάζοντες : Τα έξυπνα συμβόλαια μειώνουν ή εξαλείφουν τους μεσάζοντες που προσφέρουν υπηρεσίες "έμπιστοσύνης", παραδείγματος χάριν οικονομικές εγγυήσεις ανάμεσα στα συμβαλλόμενα μέλη.

v Χαμηλότερο κόστος : Με την μείωση της ανθρώπινης παρέμβασης όπως επίσης και με την μείωση ή την εξάλειψη των μεσαζόντων που επιτυγχάνεται μέσω των έξυπνων συμβάσεων, λογικό είναι ότι επιτυγχάνεται και μείωση στο συνολικό κόστος της διαδικασίας.

vi Νέα επιχειρηματικά ή επιχειρησιακά μοντέλα : Επειδή οι έξυπνες συμβάσεις παρέχουν έναν χαμηλού κόστους τρόπο ώστε να διασφαλίζεται ότι οι συναλλαγές πραγματοποιούνται αξιόπιστα όπως έχει συμφωνηθεί, θα επιτρέψει τη δημιουργία νέων τύπων επιχειρήσεων, από τις εμπορικές συναλλαγές ισοδύναμων ανανεώσιμων πηγών ενέργειας σε αυτοματοποιημένη πρόσβαση σε οχήματα και μονάδες αποθήκευσης.

2.8 Προκλήσεις για βελτίωση

Οι έξυπνες συμβάσεις βρίσκονται ακόμα στα πρώτα τους βήματα, τόσο τεχνολογικά όσο και επιχειρησιακά. Από τεχνολογικής άποψης, συγκεκριμένα πλεονεκτήματα που υπάρχουν θα διευκολύνουν τη διεύρυνση των εφαρμογών και την υιοθέτηση έξυπνων συμβάσεων. Κάποιες προκλήσεις που πρέπει να βελτιώσουν οι μπλοκ αλυσίδες είναι οι παρακάτω (Kehrli, J. [2016]):

i Επεκτασιμότητα : Η επεκτασιμότητα εξακολουθεί να θεωρείται μη αποδεδειγμένη όσο αναφορά τις πλατφόρμες έξυπνων συμβολαίων. Οι προγραμματιστές έχουν γνώση αυτού του γεγονότος και προσπαθούν να το επιλύσουν με διάφορες μεθόδους. Παραμένει να δούμε αν μπορεί κάποια από τις προσεγγίσεις να διατηρήσει τα οφέλη μίας δημόσιας μπλοκ αλυσίδας.

ii Πρόσβαση σε επίκαιρες πληροφορίες : Όπως αναφέρθηκε παραπάνω, επειδή τα έξυπνα συμβόλαια μπορούν να αναφέρουν μόνο πληροφορίες στη μπλοκ αλυσίδα, χρειάζονται τα μαντεία για να μπορούν να ωθούν πληροφορίες στη μπλοκ αλυσίδα. Παρόλο που υπάρχουν κάποιες ενέργειες όπου είναι πολλά υποσχόμενες, είναι ακόμα αναγκαίο να δημιουργηθούν μαντεία.

iii Ιδιωτικότητα : Ο κώδικας με τον οποίο γράφεται μία έξυπνη σύμβαση παραμένει ορατός για όλα τα μέλη του συμβολαίου. Παρόλα αυτά, δεν πρέπει να γίνεται πάντα αυτό διότι, παραδείγματος χάριν έμποροι λιανικής πώλησης μπορεί να μην θέλουν να φαίνονται δημόσια οι διαπραγματεύσεις τους με ποικίλους προμηθευτές.

iv Καθυστέρηση και Απόδοση : Οι μπλοκ αλυσίδες υποφέρουν από μεγάλη καθυστέρηση, δεδομένου του χρόνου που περνά για κάθε επαληθευμένο μπλοκ συναλλαγών που θα προστεθεί στο γενικό σύστημα. Παραδείγματος χάριν στην μπλοκ αλυσίδα Ethereum, ο χρόνος ανανέωσης είναι 17 δευτερόλεπτα, χρόνος που είναι πολύ μεγαλύτερος από τα χιλιοστά του δευτερολέπτου που υπάρχει σε συστήματα βάσεις δεδομένων δίχως μπλοκ αλυσίδα.

v Αδειοδότηση: : Ενώ ο ενθουσιασμός για έξυπνες συμβάσεις αυξάνεται τόσο στον τομέα των μη αδειοδοτημένων όσο και των αδειοδοτημένων μπλοκ αλυσίδων, το τελευταίο είναι πιθανότερο να δει ταχύτερη υιοθέτηση στη βιομηχανία, δεδομένης της πολυπλοκότητας όσον αφορά την εμπιστοσύνη, την ιδιωτικότητα και την επεκτασιμότητα που μπορούν να επιλυθούν ευκολότερα, μέσα σε μία κοινοπραξία γνωστών μελών.

vi Όρια εφαρμογής : Υπάρχουν συχνά εύλογοι λόγοι για την παροχή επιλογών ενώ γράφονται συμβάσεις. Σε πολλές από αυτές, οι ρήτρες γράφονται επίτηδες σε πράγματα με σκοπό να δημιουργηθεί ένα κανάλι διαιτησίας. Επιπλέον, κάποιες επιχειρήσεις από την φύση τους δεν μπορούν να επωφεληθούν από τις έξυπνες συμβάσεις ή άλλες μπλοκ αλυσίδες.

vii Διακυβέρνηση : Εάν οι μπλοκ αλυσίδες θέλουμε να είναι βιώσιμες μακροπρόθεσμα, πρέπει να εξεταστούν σοβαρά οι κατάλληλοι μηχανισμοί διακυβέρνησης. Η κατανομή της μεταλλευτικής ισχύος και των κρυπτονομισμάτων μαζί με τα ψευδώνυμα των κατόχων λογαριασμών αποτελούν ένα ισχυρό κίνητρο για να εξαπατήσεις το σύστημα, το οποίο καθίσταται επιρρεπές σε εξαπάτησεις, μη λογοδοσίες και απάτες.

2.9 Ανάλυση κόστους-οφέλους της έξυπνης σύμβασης

2.9.1 Ανάλυση Κόστους

Η προσπάθεια ανάλυσης κόστους περιλαμβάνει το κόστος για την έξυπνη σύμβαση όπως και τα κόστη υιοθέτησης μπλοκ αλυσίδων. Επειδή ποικίλοι παράγοντες επηρεάζουν τις δαπάνες λόγω της μοναδικότητας κάθε σύμβασης, προσπαθούμε παρακάτω να κάνουμε μία ανάλυση κόστους και να συμπεριλάβουμε όσο το δυνατό περισσότερες περιπτώσεις παρέχοντας όπου είναι δυνατόν ποσοτικές εξηγήσεις.

2.9.1.1 Κόστος της έξυπνης σύμβασης

Από γενικής απόψεως, οι έξυπνες συμβάσεις αποσκοπούν στη μείωση του κόστους εμπιστοσύνης. Όμως, επειδή οποιοσδήποτε μπορεί να συντάξει ένα, δεν συνεπάγεται ότι οι έξυπνες συμβάσεις δεν έχουνε κόστος. Τα σημαντικότερα κόστη των έξυπνων συμβάσεων κατηγοριοποιούνται στα ακόλουθα:

- Το κόστος της σύνταξης μιας σύμβασης.

  Αυτό το κόστος διαμορφώνεται ανάλογα τον προγραμματιστή καθώς επίσης και κάθε άλλου ατόμου που εμπλέκεται στη δημιουργία ή τη συμπλήρωση μίας σύμβασης.

- Το κόστος της αγοράς της σύμβασης.

  Το κόστος αγοράς μίας σύμβασης εξαρτάται από την πολιτική του πωλητή αλλά το κόστος του να γραφτεί μία περιλαμβάνει μεταξύ άλλων:

  · Η πολυπλοκότητα του κώδικα της έξυπνης σύμβασης.
  · Ο έλεγχος που απαιτείται για την ασφάλεια μίας σύμβασης.
  · Το κόστος σύνδεσης με εξωτερικούς πόρους, όπως τα μαντεία.

2.9.1.2 Κόστος αποδοχής των μπλοκ αλυσίδων

Πέρα από τις χρεώσεις που είναι άμεσα συσχετισμένες με την ίδια τη σύμβαση και τη διαδικασία συναλλαγής, υπάρχουν επίσης δαπάνες σχετικά με την υιοθέτηση των μπλοκ αλυσίδων σε έναν οργανισμό. Αυτά περιλαμβάνουν:

- Το κόστος μηχανημάτων υπολογιστών :

  Αυτά τα έξοδα σχετίζονται με τα πρόσθετα μηχανήματα υπολογιστών που χρειάζεται ενδεχομένως πάνω από τα υφιστάμενα που υπάρχουν σε μία εταιρεία.

- Το κόστος λογισμικού :

  Αυτά τα έξοδα σχετίζονται με το πρόσθετο λογισμικό που χρειάζεται ενδεχομένως πάνω από το υφιστάμενο που υπάρχει στην εταιρεία και το κόστος της ίδιας της σύμβασης.

- Το κόστος εφαρμογής του συστήματος :

  Η κατηγορία αυτή περιλαμβάνει το κόστος τροποποίησης και το κόστος ανάπτυξης των μπλοκ αλυσίδων προκειμένου να χρησιμοποιηθούνε για την έξυπνη σύμβαση. Οι παράμετροι ασφαλείας ως καθώς και πιθανό λογισμικό και μηχανήματα υπολογιστών, πρέπει να προστεθούν εδώ.

- Επιχειρησιακό κόστος :

  Το επιχειρησιακό κόστος αφορά το κόστος των πόρων που διατίθενται για την διαδικασία σύναψης έξυπνων συμβάσεων.

- Κόστος συντήρησης :

  Το κόστος συντήρησης είναι η ελάχιστη δαπάνη που είναι αναγκαία για τη διατήρηση ολόκληρου του συστήματος μπλοκ αλυσίδας. Οι κύριοι παράγοντες που διαμορφώνουν αυτές τις δαπάνες είναι οι εξής:

  · Το μέγεθος της μπλοκ αλυσίδας.
  · Τη σημαντικότητα της έξυπνης σύμβασης που θα βοηθήσει τις αποστολές της οργάνωσης.

  Λόγω της πολυπλοκότητας του κόστους των έξυπνων συμβάσεων, πριν από την υιοθέτηση τους, μία διεξοδική ανάλυση του συνολικού κόστους ιδιοκτησίας μίας πλατφόρμας μπλοκ αλυσίδων για έξυπνες συμβάσεις είναι σημαντικό να γίνει προκειμένου να προσδιοριστούν οι σχετικές δαπάνες.

2.9.2 Ανάλυση οφέλους

Η ανάλυση οφελών για τις εφαρμογές διαφοροποιείται ανάμεσα σε επιχειρήσεις. Αυτό οφείλεται στο γεγονός ότι μέχρι τώρα, δεν υπάρχουν εκτεταμένες επιχειρηματικές περιπτώσεις για τη διεξαγωγή έρευνας με πραγματικά δεδομένα και ως εκ τούτου δεν μπορούμε να παράσχουμε λεπτομερή ανάλυση των οφελών. Η ανάλυση οφέλους μας επισημαίνει τα εξής:

- Ο διαμεσολαβητής :

  Δεδομένου ότι οι έξυπνες συμβάσεις εξαλείφουν τους διαμεσολαβητές, το όφελος θεωρείται ότι είναι ίσο με το κόστος διαμεσολάβησης ενός διαμεσολαβητή και οι χρεώσεις διαφέρουν ανάλογα τον τύπο της επιχείρησης. Όσον αφορά αυτή την προσέγγιση, πρέπει να επισημάνουμε ότι η ο διαμεσολαβητής εξακολουθεί να υπάρχει σε έξυπνες συμβάσεις με μικρότερη επιρροή και χρέωση και αυτό στοχεύει να αντισταθμίσουν την ίδια τη διαμεσολάβηση και όχι άλλες συναφείς υπηρεσίες που θα μπορούσαν να προσφέρονται στον πραγματικό κόσμο.

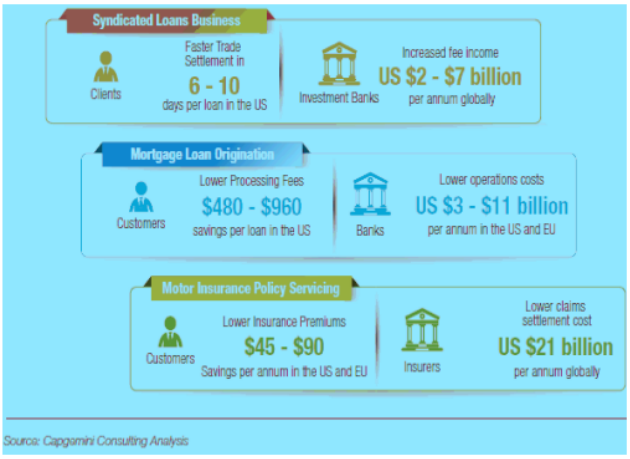
- Η ίδια η συναλλαγή :

  Θεμελιώδες όφελος αποτελεί η μείωση του χρόνου καθώς και η μείωση πιθανών σφαλμάτων. Παρακάτω, θα γίνει μια σύντομη παρουσίαση ανάλυσης των οικονομικών οφελών, όπως αναλύεται στην αναφορά (CapGemini [2017]), με χρήση έξυπνων συμβολαίων στον οικονομικό τομέα. Αυτά τα οφέλη, απεικονίζονται στις επόμενες περιπτώσεις χρήσης. Πρέπει να το επισημάνουμε ότι όλο και περισσότερες περιπτώσεις ανακύπτουν στον πραγματικό κόσμο και ότι μελλοντικά θα υπάρξει πιο ακριβής ανάλυση οφέλους.

  - Τραπεζική επένδυση : Κατά τη διαπραγμάτευση και τον διακανονισμό κοινοπρακτικών δανείων, οι εταιρικοί πελάτες θα μπορούσαν να επωφεληθούν από μικρότερους κύκλους διακανονισμού. Με τις έξυπνες συμβάσεις, μειώνεται ο χρόνος αναμονής σε 6 έως 10 ημέρες, έναντι των σημερινών καταστάσεων που χρειάζονται 20 ή και παραπάνω ημέρες. Τα πρόσθετα εισοδήματα μπορούν να φτάσουν από 2 έως 7 δισεκατομμύρια, εφόσον με λιγότερη αναμονή θα υπάρξει άνοδος της ζήτησης μεταξύ 5−6%. Τα λειτουργικά κόστη θα μειωθούν τόσο σε επενδυτικές τράπεζες της Ευρώπης όσο και τις ΗΠΑ.
  - Λιανική τραπεζική : Με την ανάπτυξη των έξυπνων συμβάσεων, τα στεγαστικά δάνεια θα είχανε σημαντικά οφέλη. Οι καταναλωτές θα μπορούσαν να αναμένουν εξοικονόμηση από 480 έως 960 δολάρια ανά δάνειο και οι τράπεζες θα μπορούσαν να μειώσουν το κόστος από 3 δισεκατομμύρια δολάρια έως 11 δισεκατομμύρια δολάρια ετησίως μειώνοντας το κόστος επεξεργασίας στις διαδικασίες δημιουργίας στις αμερικανικές και ευρωπαϊκές αγορές.
  - Ασφάλιση : Χρήση έξυπνων συμβολαίων στον ιδιωτικό κλάδο ασφάλισης αυτοκινήτων μόνο, θα μπορούσε να οδηγήσει σε ετήσια εξοικονόμηση κόστους ύψους 21 δισεκατομμυρίων δολαρίων παγκοσμίως μέσω αυ-

τοματοποίησης και μειωμένων γενικών εξόδων επεξεργασίας στις διαδικασίες διεκπεραίωσης αξιώσεων. Δίνοντας ένα μερίδιο οι ασφαλιστές από τις ετήσιες αποταμιεύσεις στους καταναλωτές, οι τελευταίοι θα είχανε πιθανόν μείωση ασφαλίστρων.

Αυτή η πιθανή εξοικονόμηση κόστους εμφανίζεται στο παρακάτω σχήμα που λαμβάνεται από την έκθεση "CapGemini" (CapGemini [2017]).



Σχήμα 2.1

2.9.3 Ωφέλιμα συμπεράσματα

Τα δεδομένα από το προηγούμενο σχήμα δείχνουν ότι:

- Θα πρέπει να ληφθεί προσεκτική οικονομική αξιολόγηση του προτεινόμενου συστήματος αλυσίδων συστοιχιών.
- Το κόστος των συναλλαγών μπορεί να μειωθεί υιοθετώντας έξυπνες συμβάσεις.
- Υπάρχουν κόστη που σχετίζονται με την υιοθέτηση έξυπνων συμβάσεων και αλυσίδων συστοιχιών που πρέπει να λαμβάνονται υπόψη από τους οργανισμούς.
- Δεν είναι κάθε επιχείρηση κατάλληλη για έξυπνες συμβάσεις για να έχει οφέλη από οικονομικής απόψεως.

Κεφάλαιο 3

Βάσεις Δεδομένων

3.1 Ορολογία

Μία βάση δεδομένων είναι μία μεθοδική συγκέντρωση πληροφοριών που αποθηκεύονται και προσπελάζονται, με τη χρήση ενός ηλεκτρονικού συστήματος υπολογιστή. Οι βάσεις δεδομένων είναι σύνθετες και συνήθως υλοποιούνται με τη χρήση επισήμων συγκεκριμένων τεχνικών, τόσο για τον σχεδιασμό τους όσο και για την μοντελοποίησή τους. Το σύστημα που διαχειρίζεται τις βάσεις δεδομένων (DBMS) είναι το διαμεσολαβητικό όργανο μεταξύ των τελικών χρηστών, των προγραμμάτων και της βάσης δεδομένων προκειμένου να συλλέξουμε δεδομένα. Το σύνολο προγραμμάτων συστημάτων διαχείρισης βάσεων δεδομένων εμπεριέχει επίσης τις κύριες λειτουργίες που χρειάζονται για να ελεγχθεί μία βάση δεδομένων.Όλη η βάση δεδομένων, το σύστημα διαχείρισης βάσεων δεδομένων και οι σχετικές εφαρμογές μπορούν να αναφέρονται ως "σύστημα βάσης δεδομένων". Συχνά, η φράση "βάση δεδομένων", αναφέρεται σε καθένα από τα συστήματα διαχείρισης βάσεων δεδομένων, το σύστημα βάσης δεδομένων ή ένα πρόγραμμα που συνδέεται άμεσα με βάσεις δεδομένων.

Τα συστήματα που διαχειρίζονται βάσεις δεδομένων γίνεται να ταξινομηθούν βάσει των προτύπων των προτύπων των βάσεων της πληροφορικής που υποστηρίζουν. Οι σχετικές βάσεις δεδομένων κατέστησαν κυρίαρχες το 1980. Αυτές οι πληροφορίες μοντελοποιούνται σαν γραμμές και στήλες, παράγοντας πίνακες και χρησιμοποιώντας SQL για τη γραφή και την αναζήτηση δεδομένων. Στο πέρασμα του χρόνου και πιο συγκεκριμένα στον 21^o αιώνα αρκετά διαδεδομένες έγιναν οι μη σχεσιακές βάσεις δεδομένων, γνωστές ως NoSQL, για τον λόγο ότι χρησιμοποιούν διαφορετικές γλώσσες για να δημιουργούν ερωτήματα.

Επίσημα, μία "βάση δεδομένων" αναφέρεται σε ένα σύνολο σχετικών δεδομένων και στον τρόπο με τον οποίο είναι οργανωμένα. Η είσοδος στις πληροφορίες αυτές προσφέρεται κατά βάση από το "σύστημα διαχείρισης βάσεων δεδομένων" (DBMS) το οποίο έχει δημιουργηθεί από μία ολοκληρωμένη συλλογή λογισμικών, που δίνει την άδεια στους χρήστες να αλληλεπιδρούν με τουλάχιστον μία βάση δεδομένων και επιτρέπει την είσοδο σε όλες τις πληροφορίες που εμπεριέχονται μέσα σε μία βάση

δεδομένων(παρόλο που μπορεί να θέτονται όρια που να περιορίζουν την πρόσβαση σε συγκεκριμένες πληροφορίες). Το σύστημα διαχείρισης βάσεων δεδομένων εμπεριέχει ποικίλες ενέργειες που διασφαλίζουν την εισαγωγή, την φύλαξη και την ανάγνωση πολλών δεδομένων, παρέχοντας μεθόδους που διαχειρίζονται τον τρόπο οργάνωσης αυτών των δεδομένων.

Ως εκ τούτου, χρησιμοποιούμε τον όρο "βάση δεδομένων" σχετικά με μία βάση δεδομένων αλλά και του συστήματος διαχείρισης βάσεων δεδομένων το οποίο χρησιμοποιεί για τον χειρισμό του.

Γενικότερα με τον όρο "βάση δεδομένων" εννοούμε οτιδήποτε έχει σχέση με τη συγκέντρωση σχετικών δεδομένων διότι οι προϋποθέσεις χρήσης αναζητούν ένα σύστημα διαχείρισης βάσεων δεδομένων. (Ullman, J. and Widom, J. [1997])

Τα υπάρχοντα συστήματα διαχείρισης βάσεων δεδομένων εμπεριέχουν ποικίλες ενέργειες που δίνουν τη δυνατότητα να υπάρξει διαχείριση μίας βάσης δεδομένων και των πληροφοριών της, τα οποία κατηγοριοποιούνται σε τέσσερα βασικά σύνολα:

- Ορισμός δεδομένων : Δημιουργούνται, τροποποιούνται και καταργούνται ορισμοί που συμβάλλουν στην διάταξη των πληροφοριών.
- Ενημέρωση : Εμπεριέχει ενέργειες για να εισαχθούν, να τροποποιηθούν και να διαγραφούν πραγματικά δεδομένα. (Webster, M. [ς])
- Ανάκτηση : Διάδοση δεδομένων για να χρησιμοποιηθεί άμεσα ή για περαιτέρω επεξεργασία από άλλες υλοποιήσεις. Οι ανακτηθέντες πληροφορίες διατίθενται σε παρόμοια μορφή με αυτή που αποθηκεύτηκαν στη βάση δεδομένων είτε υπό καινούρια μορφή που έλαβε με τη μεταβολή ή την ένωση πληροφοριών από τη βάση δεδομένων. (Webster, M. [β])
- Διοίκηση : Καταγράφει και παρακολουθεί χρήστες, επιβάλλοντας την ασφάλεια των πληροφοριών, παρακολουθώντας την απόδοση, διατηρώντας την ακεραιότητα των πληροφοριών, αντιμετωπίζοντας τον έλεγχο ταυτόχρονης λειτουργίας και ανάκτησης πληροφοριών που έχουν αλλοιωθεί από συμβάν, σαν μία απρόβλεπτη ατυχία ενός ή πολλών συστημάτων. (Webster, M. [α])

Μια βάση δεδομένων και το σύστημα διαχείρισης βάσεων δεδομένων της συντονίζονται στα πλαίσια ενός προτύπου βάσης δεδομένων (Tsitchizris, Dionysios C. and Lochovsky, Fred H. [1982]). Το "Σύστημα Βάσης Δεδομένων" παραπέμπει στο πρότυπο βάσης δεδομένων, στο σύστημα που τη διαχειρίζεται καθώς και στην ίδια τη βάση πληροφοριών (Beynon-Davies, P. [2003]).

Προφανώς οι διαμεσολαβητές βάσεων δεδομένων αποτελούνται από υπολογιστές που περιέχουν αποκλειστικά τις αληθινές βάσεις δεδομένων και χρησιμοποιούν το σύστημα διαχείρισης βάσεων δεδομένων και το συναφές λειτουργικό πρόγραμμα. Οι διαμεσολαβητές βάσεων δεδομένων είναι συχνά υπολογιστές πολλών πυρήνων επεξεργασίας με μεγάλες σε χωρητικότητα μνήμες και συνδεδεμένους πολλούς σκληρούς δίσκους RAID. Τα συστήματα διαχείρισης βάσεων δεδομένων είναι ο πυρήνας για τις περισσότερες εφαρμογές. Τα συστήματα διαχείρισης βάσεων δεδομένων αναπτύσσονται γύρω από προσαρμοσμένους πυρήνες που εκτελούν διαφορετικές εργασίες(multitasking) συνδεδεμένο σε δίκτυο. Τα τωρινά συστήματα

διαχείρισης βάσεων δεδομένων βασίζονται κατά κύριο λόγο σε λειτουργικά σύστημα για την εκτέλεση αυτών των εργασιών.

Ως εκ τούτου, τα συστήματα διαχείρισης βάσεων δεδομένων κατέχουν μεγάλο μερίδιο της αγοράς. Έτσι οι έμποροι ηλεκτρονικών ειδών συνήθως λαμβάνουν υπόψη τους τις προϋποθέσεις των συστημάτων διαχείρισης βάσεων δεδομένων για τον στρατηγικό σχεδιασμό της ανάπτυξής τους. (Nelson, A.F. and Nelson, W.H.M. [2001])

Οι κατηγορίες των βάσεων δεδομένων καθώς και των συστημάτων διαχείρισης βάσεων δεδομένων γίνονται με γνώμονα τα μοντέλα που χρησιμοποιούν (παραδείγματος χάριν σχεσιακό είτε με χρήση της γλώσσας σήμανσης XML (eXtensible Markup Language)), το είδος του υπολογιστή με τον οποίο γίνονται οι εκτελέσεις (παραδείγματος χάριν συμπλέγματα διακομιστών είτε με τη χρήση κινητού τηλεφώνου), τη γλώσσα που χρησιμοποιείται για να διατυπωθούν τα ερωτήματα εισόδου (παραδείγματος χάριν SQL είτε XQuery) και την διάταξή τους εσωτερικά, με βάση την οποία επηρεάζεται η απόδοση, η επεκτασιμότητα, η ανθεκτικότητα και η ασφάλεια.

3.2 Ιστορία

Ραγδαία αύξηση έχουν σημειωθεί στις διαστάσεις, στις ικανότητες και στις επιδόσεις τόσο των βάσεων δεδομένων όσο και στα αντίστοιχά τους συστήματα διαχείρισης βάσεων δεδομένων. Αυτές οι αυξήσεις των επιδόσεων ενεργοποιήθηκαν από την πάροδο της εξέλιξης των πυρήνων επεξεργασίας, των μικρών καταχωρητών, των μέσων αποθήκευσης και των συνδέσεων των ηλεκτρονικών υπολογιστών. Οι βάσεις δεδομένων αναπτύχθηκαν σε τρία στάδια βάση του μοντέλου ή της δομής δεδομένων : πρώτα πλοήγηση (Bachman, C.W. [1973]), μετά SQL / σχεσιακή και μετά μετα-σχεσιακή.

Στα δύο βασικά πρότυπα πρώιμης πλοήγησης έχουμε το ιεραρχικό μοντέλο και το μοντέλο CODASYL (μοντέλο δικτύου).

Το σχεσιακό πρότυπο, που εμφανίστηκε αρχικά τη δεκαετία του ′70 με δημιουργό του τον Edgar F. Codd, απομακρύνθηκε από αυτή την παράδοση επιμένοντας ότι οι υλοποιήσεις πρέπει να εξετάζουν τις πληροφορίες με βάση το περιεχόμενο και όχι με το να ακολουθούν συνδέσμους. Το σχεσιακό μοντέλο χρησιμοποιεί σύνολα πινάκων στυλ βιβλίου, το καθένα από τα οποία είναι χρήσιμο για άλλη κατηγορίας ύπαρξης. Μόνο ενδιάμεσα το 1980 το υλικό πληροφορικής έγινε αρκετά ισχυρό για να επιτρέψει την ευρεία ανάπτυξη σχεσιακών συστημάτων (συστήματα διαχείρισης βάσεων δεδομένων και εφαρμογές). Ωστόσο, στις αρχές της δεκαετίας του 1990, τα σχεσιακά συστήματα κυριάρχησαν ανάμεσα σε διάφορες εφαρμογές παγκόσμιας έκτασης ανάλυσης πληροφοριών και μέχρι το 2020 βρίσκονται πρώτες σε προτίμηση οι ακόλουθες: IBM DB2, Oracle, MySQL και Microsoft SQL Server είναι τα κυρίαρχα συστήματα διαχείρισης βάσεων δεδομένων (Carbonnelle, P. [2020]). Ο βασικός τρόπος επικοινωνίας με μία βάση δεδομένων είναι η συμβατική SQL για το σχεσιακό μοντέλο, παρόλο που έχει διαμορφώσει τρόπους επικοινωνίας με βάσεις δεδομένων για διαφορετικά μοντέλα πληροφοριών.

Οι βάσεις δεδομένων που ήταν αντικειμενοστραφείς δημιουργήθηκαν στα μέσα

του ′80 για να μειώσουν το χάσμα αναντιστοιχίας που υπάρχει μεταξύ αντικειμενοστραφής και σχεσιακής, που κατέληξε στην επινόηση του όρου "post relational" όπως και στο να αναπτυχθούν υβριδικές αντικείμενο-σχεσιακές βάσεις δεδομένων.

Η επόμενη έκδοση μετά-σχεσιακών βάσεων δεδομένων αναπτύχθηκε πριν το 2000 και ονομάστηκε NoSQL, αναπτύσσοντας γρήγορα βασικής αξίας αποθηκεύσεις και έγγραφο-προσανατολισμένες βάσεις δεδομένων. Μία εξεζητημένη νέα έκδοση (NewSQL) υλοποιήθηκε για την διατήρηση του σχεσιακού / SQL μοντέλου, ενώ στόχο είχε να εναρμονιστεί με την αποδοτικότητα του NoSQL συγκριτικά με τα είδη υπάρχοντα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων.

3.3 Αλληλεπίδραση βάσεων δεδομένων

3.3.1 Σύστημα διαχείρισης βάσης δεδομένων

Ο Connolly και η Begg ορίζουν το σύστημα διαχείρισης βάσεων δεδομένων (DBMS ή Σ.Δ.Β.Δ.) ως "συλλογή προγραμμάτων όπου δίνει άδεια στους χρήστες να προσδιορίζουν, να κατασκευάσουν, να συντηρούν και να αξιολογούν την είσοδο στη βάση δεδομένων" (Connolly, T.M. and Begg, C.E. [2014]). Η MySQL, PostgreSQL, MSSQL, Oracle Database και Microsoft Access αποτελούν παραδείγματα συστημάτων διαχείρισης βάσεων δεδομένων.

Αναλόγως ποιο μοντέλο υποδεικνύεται σε μία βάση δεδομένων, το ακρωνύμιο DBMS προσαρμόζεται διαφορετικά. Με RDBMS συμβολίζουμε το σχεσιακό, με O-ODBMS συμβολίζουμε το αντικειμενοστραφές και με ORDBMS συμβολίζουμε το αντικείμενο-σχεσιακό. Χρησιμοποιώντας διαφορετικές επεκτάσεις υποδηλώνουμε διαφορετικά χαρακτηριστικά. Παραδείγματος χάριν, το αρκτικόλεξο DDBMS χρησιμοποιείται σε συστήματα κατανεμημένης διαχείρισης.

Η λειτουργίες που παρέχει ένα σύστημα διαχείρισης βάσεων δεδομένων μπορεί να είναι πολύ διαφορετικές. Η βασική λειτουργικότητα είναι η εγγραφή, η ανάγνωση και η μεταβολή πληροφοριών. Ο Codd πρότεινε τις ακόλουθες λειτουργίες και υπηρεσίες: ένα ολοκληρωμένο σύστημα διαχείρισης βάσεων δεδομένων γενικού σκοπού πρέπει να παρέχει (Connolly, T.M. and Begg, C.E. [2014]):

- Αποθήκευση δεδομένων, ανάκτηση και ενημέρωση
- Κατάλογος χρήστη ή λεξικό δεδομένων προσβάσιμο από τον χρήστη που περιγράφει τα μεταδεδομένα
- Υποστήριξη συναλλαγών και ταυτοχρονισμού
- Οι εγκαταστάσεις για την ανάκτηση της βάσης δεδομένων θα πρέπει να καταστραφούν
- Υποστήριξη για εξουσιοδότηση πρόσβασης και ενημέρωση δεδομένων
- Υποστήριξη πρόσβασης από απομακρυσμένες τοποθεσίες
- Η επιβολή περιορισμών για τη διασφάλιση των δεδομένων στη βάση δεδομένων συμμορφώνεται με ορισμένους κανόνες

Είναι επίσης γενικά αναμενόμενο ότι το σύστημα διαχείρισης βάσεων δεδομένων θα παρέχει ένα σύνολο βοηθητικών προγραμμάτων για σκοπούς που μπορεί να είναι απαραίτητοι για την αποτελεσματική διαχείριση της βάσης δεδομένων, συμπεριλαμβανομένων των βοηθητικών προγραμμάτων εισαγωγής, εξαγωγής, παρακολούθησης, ανασυγκρότησης και ανάλυσης (Connolly, T.M. and Begg, C.E. [2014]). Το βασικό μέρος του συστήματος διαχείρισης βάσεων δεδομένων που αλληλεπιδρά μεταξύ της βάσης δεδομένων και της διεπαφής εφαρμογής αναφέρεται μερικές φορές ως μηχανή βάσης δεδομένων. Μηχανή βάσης δεδομένων συνήθως συνιστά το κύριο τμήμα του συστήματος διαχείρισης βάσεων δεδομένων που συνδέει τη βάση δεδομένων με την εφαρμογή.

Συχνά τα συστήματα διαχείρισης βάσεων δεδομένων θα έχουν παραμέτρους διαμόρφωσης που μπορούν να συντονιστούν στατικά και δυναμικά, για παράδειγμα η μέγιστη ποσότητα κύριας μνήμης σε διακομιστή που μπορεί να χρησιμοποιήσει η βάση δεδομένων. Σκοπός είναι να ελαχιστοποιήσουμε τη χειρωνακτική διαμόρφωση και σε περίπτωση ενσωματωμένων βάσεων δεδομένων, να μηδενιστεί η διαχείριση.

Οι μεγάλες κυρίαρχες επιχειρήσεις συστημάτων διαχείρισης βάσεων δεδομένων κάνουν προσπάθειες αύξησης του μεγέθους και της λειτουργικότητάς τους, έχοντας διανύσει μεγάλη πορεία εξέλιξης.

Η αρχική έκδοση συστήματος διαχείρισης βάσεων δεδομένων άφηνε την εφαρμογή να είναι συνδεδεμένη μόνο με τον ίδιο υπολογιστή, έχοντας πρόσβαση μέσω τερματικού. Στη συνέχεια, το μοντέλο πελάτη-διακομιστή αποτέλεσε μία εξέλιξη, έχοντας την εφαρμογή στην επιφάνεια εργασίας του πελάτη και τη βάση δεδομένων σε έναν εξυπηρετητή, με σκοπό να μοιραστεί και στους δύο η δυνατότητα επεξεργασίας. Αυτό αποτέλεσε μία σύνθετη αρχιτεκτονική που εμπεριέχει σέρβερς εφαρμογών και σέρβερ ιστού, με τη σύνδεση τους τελικούς χρήστες να συνδέονται μέσω δικτύου άμεσα στην βάση.

Ένα σύστημα διαχείρισης βάσεων δεδομένων γενικής χρήσης προσφέρει δημόσια διεπαφή προγραμματισμού εφαρμογών μαζί με έναν εναν πυρήνα πολλαπλών διεργασιών για επικοινωνία με την βάση, όπως SQL, ώστε να επιτρέπονται οι αλληλοεπιδράσεις γραπτών εφαρμογών με την βάση. Αντίθετα ενα σύστημα διαχείρισης βάσεων δεδομένων ειδικού σκοπού κάνει χρήση ιδιωτικής διεπαφής προγραμματισμού εφαρμογών και να είναι συνδεδεμένο αντίστοιχα προσαρμοσμένα με με μια καθολική εφαρμογή. Για παράδειγμα, ένα σύστημα ηλεκτρονικού ταχυδρομείου που εκτελεί πολλές από τις λειτουργίες ενός συστήματος διαχείρισης βάσεων δεδομένων γενικού σκοπού, όπως η εισαγωγή μηνυμάτων, η διαγραφή μηνυμάτων, ο χειρισμός συνημμένων, η αναζήτηση κατάλογος μπλοκαρίσματος, η συσχέτιση μηνυμάτων με μία διεύθυνση ηλεκτρονικού ταχυδρομείου κ.ο.κ. Παρ΄ όλα αυτά, υπάρχει περιορισμός σε ό,τι απαιτείται από τις άνωθεν λειτουργίες σε σχέση με τη διαχείριση ηλεκτρονικού ταχυδρομείου.

3.3.2 Εφαρμογή σύνδεσης με σύστημα διαχείρισης βάσης δεδομένων

Η σύνδεση με το σύστημα διαχείρισης βάσεων δεδομένων για να υπάρξει εξωτερική αλληλεπίδραση πραγματοποιείται μέσω εφαρμογής (Connolly, T.M. and Begg, C.E. [2014]). Το εύρος αυτό βρίσκεται μεταξύ εργαλείων βάσης δεδομένων όπου

χρήστες θέτουν ερωτήματα με τη γλώσσα SQL, μέχρι μία διαδικτυακή σελίδα που χρησιμοποιεί μία βάση δεδομένων, για αποθήκευση και αναζήτηση δεδομένων.

3.3.3 Διεπαφή Προγράμματος Εφαρμογής

Με χρήση διεπαφής προγραμματισμού εφαρμογών (API) είτε γλώσσα επερώτησης στη βάση δεδομένων, κωδικοποιούνται οι επαφές με τη βάση από τον προγραμματιστή. Η συγκεκριμένη διεπαφή προγραμματισμού εφαρμογών ή η επιλεγμένη γλώσσα θα πρέπει να υποστηρίζεται από το σύστημα διαχείρισης βάσεων δεδομένων, το οποίο είναι δυνατό έμμεσα, μέσω ενός προ-επεξεργαστή ή μίας διεπαφής προγραμματισμού εφαρμογών γεφύρωσης. Η ανεξαρτησία από τη βάση αποτελεί στόχο της διεπαφής προγραμματισμού εφαρμογών και το ODBC αποτελε διαδεδομένο παράδειγμα της ανεξαρτησίας αυτής. Άλλα παραδείγματα ανεξαρτησίας διεπαφών προγραμματισμού εφαρμογών αποτελούν το JDBC καθώς και το ADO.NET.

3.3.4 Γλώσσες Βάσεων Δεδομένων

Οι γλώσσες βάσεων δεδομένων αποτελούν γλώσσες οι οποίες με την ειδική χρήση τους εκτελούνε διάφορες από τις παρακάτω εργασίες (μερικές φορές χαρακτηρίζονται και ως υπο-γλώσσες) (Chapple, M. [22 April 2020])(International Business Machines (IBM) [27 October 2006]):

- Γλώσσα ελέγχου δεδομένων (**DCL**) - με αυτήν γίνεται ο έλεγχος πρόσβασης στις πληροφορίες.
- Γλώσσα ορισμού δεδομένων (**DLL**) - με αυτήν πραγματοποιείται διαχείριση στους τύπους δεδομένων όπως επίσης δημιουργούνται, τροποποιούνται ή διαγράφονται πίνακες και διαμορφώνονται οι σχέσεις των πινάκων.
- Γλώσσα χειρισμού δεδομένων (**DML**) - με αυτήν γίνεται εκτέλεση εργασιών, για παράδειγμα να εισάγουμε, να ενημερώσουμε ή να διαγράψουμε διάφορα δεδομένα.
- Γλώσσα ερωτήματος δεδομένων (**DQL**) - με αυτήν επιτρέπεται η αναζήτηση δεδομένων καθώς και ο υπολογισμός παραγόμενων πληροφοριών.

Μία γλώσσα βάσης δεδομένων μπορεί επίσης να περιλαμβάνει λειτουργίες όπως (Wagner, M. [2010]):

- Διαμόρφωση ειδικά για συστήματα διαχείρισης βάσεων δεδομένων και διαχείριση κινητήρα αποθήκευσης
- Υπολογισμοί για την τροποποίηση των αποτελεσμάτων ερωτημάτων, όπως καταμέτρηση, άθροισμα, μέσος όρος, ταξινόμηση, ομαδοποίηση και παραπομπή
- Επιβολή περιορισμών (π.χ. σε μια βάση δεδομένων αυτοκινήτων, επιτρέποντας μόνο έναν τύπο κινητήρα ανά αυτοκίνητο)
- Έκδοση διεπαφής προγραμματισμού εφαρμογών της γλώσσας ερωτήματος, για την ευκολία του προγραμματιστή

3.4 Αποθήκευση βάσεων δεδομένων

Η αποθήκευση βάσεων δεδομένων είναι ο περιέκτης της φυσικής υλοποίησης μίας βάσης δεδομένων. Περιλαμβάνει το εσωτερικό (φυσικό) επίπεδο της αρχιτεκτονικής της βάσης δεδομένων. Εμπεριέχει επιπλέον όλα τα απαραίτητα δεδομένα (π.χ. μεταδεδομένα, "πληροφορίες σχετικά με τις πληροφορίες" και εσωτερικές δομές δεδομένων) για την αναδιαμόρφωση σε εννοιολογικό επίπεδο καθώς και την αναδιαμόρφωση από τον εσωτερικό στο εξωτερικό επίπεδο όταν είναι αναγκαίο. Ο μηχανισμός βάσης δεδομένων είναι υπεύθυνος για την τοποθέτηση καθώς και τη μόνιμη αποθήκευση πληροφοριών, π.χ. "μηχανή αποθήκευσης". Αν και με τη βοήθεια λειτουργικού συστήματος παρέχεται πρόσβαση σε συστήματα διαχείρισης βάσεων δεδομένων (και χρησιμοποιώντας για ταξινόμηση αποθήκευσης το σύστημα αρχείων του λειτουργικού συστήματος που βρίσκονται), δύο ιδιότητες αποτελούν θεμέλιο για τη σωστή λειτουργία ενός συστήματος διαχείρισης βάσεων δεδομένων και για αυτό επιβλέπονται από τους προγραμματιστές της βάσης, να ρυθμίζονται οι παράμετροι και να αναπτύσσονται ιδιότητες αποθήκευσης. Ένα σύστημα διαχείρισης βάσεων δεδομένων, όταν είναι ενεργό, λειτουργεί πάντα με μία βάση δεδομένων που ποικίλει ο χώρος στον οποίο βρίσκεται (π.χ. εσωτερική μνήμη αποθήκευσης είτε εξωτερική αποθήκευση). Λόγω του πολύ μεγάλου μεγέθους, τόσο οι πληροφορίες της βάσης όσο και τα πρόσθετα δεδομένα που χρειάζονται για τη λειτουργία τους, κωδικοποιούνται στην μνήμη σε μορφή 0 ή 1 (bits). Τα δεδομένα συνήθως βρίσκονται στην αποθήκευση σε δομές που φαίνονται τελείως διαφορετικές από τον τρόπο εμφάνισης των δεδομένων σε εννοιολογικό και εξωτερικό επίπεδο, αλλά με τρόπους που προσπαθούν να βελτιστοποιήσουν (όσο το δυνατόν καλύτερα) την ανασυγκρότηση αυτών των επιπέδων όταν χρειάζονται από τους χρήστες και τα προγράμματα, καθώς και όπως για τον υπολογισμό πρόσθετων τύπων απαιτούμενων πληροφοριών από τα δεδομένα (π.χ. κατά την αναζήτηση της βάσης δεδομένων).

3.5 Ασφάλεια βάσεων δεδομένων

Η ασφάλεια στις βάσεις δεδομένων έχει σαν στόχο την προστασία όλων των πτυχών των πληροφοριών της βάσης δεδομένων, τόσο για τους ιδιοκτήτες όσο και για τους χρήστες της. Κυμαίνεται από την φύλαξη από σκόπιμες μη εξουσιοδοτημένες χρήσεις βάσεων δεδομένων έως ακούσιες προσβάσεις βάσης δεδομένων από υπάρξεις όπου δεν έχουνε άδεια πρόσβασης (π.χ. χρήστης ή προγράμματα υπολογιστών).

Ο έλεγχος πρόσβασης έχει σαν στόχο να διασφαλίζει ποιος (ένας χρήστης ή συγκεκριμένα προγράμματα υπολογιστών) επιτρέπεται να εισέρχεται σε ποιες πληροφορίες στη βάση δεδομένων. Τα δεδομένα έχουν συγκεκριμένα αντικείμενα βάσης δεδομένων (π.χ. διαφορετικά έγγραφα, ορισμένες εγγραφές, δομές δεδομένων), ορισμένες επεξεργασίες πάνω σε συγκεκριμένα αντικείμενα (π.χ. τύποι ερώτησης ή δεδομένες ερωτήσεις) ή χρήση συγκεκριμένων διαδρομών πρόσβασης στα προηγούμενα (π.χ. χρήση συγκεκριμένων ευρετηρίων ή διάφορες δομές δεδομένων για την εισαγωγή σε δεδομένα). Τα στοιχεία με τα οποία ελέγχεται η

πρόσβαση στις βάσεις δεδομένων ορίζονται από συγκεκριμένους αδειοδοτημένους (από τον κάτοχο της βάσης) υπαλλήλους που χρησιμοποιούν ειδικά ασφαλή συστήματα διαχείρισης βάσεων δεδομένων περιβάλλοντα.

Η διαχείριση της ασφάλειας του συστήματος διαχείρισης βάσεων δεδομένων γίνεται με διαφορετικούς τρόπους. Αρχικά, είτε σε ανάθεση ατόμων είτε άτομα μέσα σε ομάδες είτε (στα πιο σύνθετα μοντέλα), σε συνδυασμό αυτών, με τη χορήγηση των ανάλογων δικαιωμάτων. Εφόσον δεν έχει χορηγηθεί δικαίωμα πρόσβασης σε κάποιον άνθρωπο, αυτός δεν μπορεί να επιβλέψει ή να ενημερώσει τη βάση. Με τη χρήση κωδικών πρόσβασης από το χρήστη, επιτρέπεται η επεξεργασία σε όλη τη βάση δεδομένων είτε σε ένα υποσύνολο αυτής. Το τελευταίο ονομάζεται "subschemas". Παραδείγματος χάριν, μία βάση δεδομένων που αφορά εργαζομένους μίας εταιρίας, εμπεριέχει όλες τις προσωπικές πληροφορίες που αφορούν έναν μεμονωμένο υπάλληλό της. Παρόλα αυτά, ανάλογα τις ομάδες χρηστών που υπάρχουν στην εταιρεία, διανέμονται και οι κατάλληλες εξουσιοδοτήσεις. Το οικονομικό τμήμα έχει πρόσβαση σε πληροφορίες μισθοδοσίας και άλλα τμήματα έχουν πρόσβαση σε εργατικές και ιατρικές πληροφορίες υπαλλήλων. Αν το σύστημα διαχείρισης βάσεων δεδομένων παρέχει έναν τρόπο για να εισάγετε και να ενημερώσετε διαδραστικά τη βάση δεδομένων, καθώς και να την ερωτήσετε, αυτή η δυνατότητα επιτρέπει τη διαχείριση προσωπικών βάσεων δεδομένων.

Η ασφάλεια δεδομένων ασχολείται γενικά με την προστασία συγκεκριμένων τμημάτων δεδομένων, τόσο φυσικών (δηλαδή από διαφθορά, καταστροφή ή αφαίρεση), είτε τμήματα αυτών που οδηγούν σε σημαντικές πληροφορίες (π.χ., εξετάζοντας τα bits που περιλαμβάνουν, μπορεί να καταλήξει κανείς σε συγκεκριμένους έγκυρους αριθμούς πιστωτικών καρτών).

Αλλαγές και προσβάσεις στα αρχεία καταγραφής που έχουν πρόσβαση στα οποία χαρακτηριστικά, τι άλλαξε και πότε άλλαξε. Στις υπηρεσίες που καταγράφουν δεδομένα επιτρέπεται να έχουν διαχείριση της βάσης και να διατηρούνε σε ένα αρχείο προσβάσεις και αλλαγές στη βάση. Ενώ η καταγραφή αλλαγών μπορεί να γίνει και από την ίδια τη βάση, συχνά γίνεται μέσω του κώδικα επιπέδου εφαρμογής. Παρακολουθώντας τα αρχεία καταγραφής γίνεται μία επιχείρηση για εντοπισμούς παραβιάσεων ασφαλείας.

3.6 Συναλλαγές και ταυτοχρονισμός

Προκειμένου να επιτύχουμε ένα επίπεδο που ανεχόμαστε τα σφάλματα και να έχουμε ακέραια τα δεδομένα μετά από σημαντικά λάθη, χρησιμοποιούμε συναλλαγές που βασίζονται σε βάσεις δεδομένων. Μία συναλλαγή με βάση δεδομένων είναι μια μονάδα εργασίας, που συνήθως περιλαμβάνει έναν αριθμό λειτουργιών μέσω μίας βάσης δεδομένων (π.χ. ανάγνωση ενός αντικειμένου βάσης δεδομένων, γραφή, απόκτηση κλειδώματος και άλλα), μία αφαίρεση που υποστηρίζεται στη βάση δεδομένων και επίσης σε άλλα συστήματα. Κάθε συναλλαγή έχει σαφώς καθορισμένα όρια ως προς το ποια προγράμματα ή εκτελέσεις κώδικα συμπεριλαμβάνονται σε αυτή τη συναλλαγή (καθορίζεται από τον προγραμματιστή της συναλλαγής μέσω εντολών ειδικής συναλλαγής).

Με το αρκτικόλεξο ACID περιγράφουμε τα απόλυτα χαρακτηριστικά για να

πραγματοποιηθούν συναλλαγές με βάση δεδομένων: (A)ατομικότητα, (C)συνέπεια, (I)απομόνωση και (D)ανθεκτικότητα.

3.6.1 Αισιόδοξος έλεγχος ταυτοχρονισμού

Ο αισιόδοξος έλεγχος ταυτοχρονισμού (Optimistic concurrency control-OCC) είναι μια μέθοδος ελέγχου ταυτόχρονης λειτουργίας που εφαρμόζεται σε συστήματα συναλλαγών, όπως συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων και μνήμη συναλλαγών λογισμικού. Με τον αισιόδοξο έλεγχο ταυτοχρονισμού γίνεται η υπόθεση ολοκλήρωσης ταυτόχρονων συναλλαγών δίχως να παρεμβάλλεται η μια στην άλλη. Όταν χρησιμοποιείται, οι συναλλαγές χειρίζονται τα δεδομένα χωρίς να αποκτούν πρόσβαση σε αυτά. Αρχικά πριν δεσμευτούν τα δεδομένα γίνεται επαλήθευση για το αν κάποια συναλλαγή έχει κάνει τροποποίηση δεδομένων από τις πληροφορίες που έχει επεξεργαστεί. Αν βγει αληθής ο έλεγχος, τότε η συναλλαγή διαπραγμάτευσης επαναφέρεται και μπορεί να ξαναρχίσει (Rohit, P. [2003]). Ο αισιόδοξος έλεγχος ταυτόχρονης συμπεριφοράς προτάθηκε αρχικά από τον H.T. Kung και John T. Robinson. (Pettersson, H.T. and Robinson, J.T. [1981])

Ο αισιόδοξος έλεγχος ταυτοχρονισμού χρησιμοποιείται γενικά σε περιβάλλοντα με χαμηλό περιεχόμενο. Όταν οι συγκρούσεις είναι σπάνιες, οι συναλλαγές μπορούν να ολοκληρωθούν χωρίς τη δαπάνη της διαχείρισης κλειδαριών και χωρίς να έχουν συναλλαγές να περιμένουν να κλείσουν οι κλειδαριές άλλων συναλλαγών, οδηγώντας σε υψηλότερη απόδοση από άλλες μεθόδους ελέγχου ταυτόχρονης συναλλαγής. Ωστόσο, αν ο ισχυρισμός για πόρους δεδομένων είναι συχνός, το κόστος επανειλημμένης επανέναρξης των συναλλαγών έχει σημαντική επίπτωση στην απόδοση. Γενικά, θεωρείται πως άλλες μέθοδοι ελέγχου ταυτόχρονης λειτουργίας έχουν καλύτερες επιδόσεις υπό αυτές τις συνθήκες. Ωστόσο, οι "άπαισιόδοξες" μέθοδοι κλειδώματος μπορούν επίσης να έχουν ανεπαρκή απόδοση επειδή η ασφάλιση μπορεί να περιορίσει δραστικά την αποτελεσματική ανταμοιβή ακόμη και όταν αποφεύγονται τα αδιέξοδα.

3.6.1.1 Φάσεις ελέγχου αισιοδοξίας

Συγκεκριμένα, οι αισιόδοξες συναλλαγές ελέγχου περιλαμβάνουν αυτές τις φάσεις:

- Αρχή : Καταγράφει ένα χρονικό σημάδι για την έναρξη της συναλλαγής.
- Τροποποίηση : Διαβάζει τις τιμές βάσης δεδομένων και γράφει εικαστικά τις αλλαγές.
- Επικύρωση : Ελέγχει εάν άλλες συναλλαγές έχουν τροποποιήσει τα δεδομένα που έχει χρησιμοποιήσει η συναλλαγή αυτή (ανάγνωση ή γραφή). Αυτό περιλαμβάνει συναλλαγές που ολοκληρώθηκαν μετά την ώρα έναρξης της συναλλαγής και προαιρετικά, συναλλαγές που εξακολουθούν να ισχύουν κατά το χρόνο επικύρωσης.
- Ανάληψη / επιστροφή : Εάν δεν υπάρχει σύγκρουση, θέτει σε ισχύ όλες οι αλλαγές. Σε αντίθετη περίπτωση προσπαθεί να βρει λύση. Στις περισσότερες

περιπτώσεις διακόπτει τη συναλλαγή, ενώ παρ΄ όλα αυτά έχουν βρεθεί και άλλοι τρόποι που μπορεί να επιλυθεί το πρόβλημα. Πρέπει να ληφθεί μέριμνα για την αποφυγή ενός σφάλματος TOCTTOU(Time-of-check to time-of-use), ιδιαίτερα εάν αυτή η φάση και η προηγούμενη δεν εκτελούνται ως μία μόνο ατομική λειτουργία.

3.6.1.2 Χρήση στον ιστό

Η ανιθαγενής φύση του πρωτοκόλλου μεταφοράς υπερκειμένου (HTTP) καθιστά το κλείδωμα ανέφικτο για τις διεπαφές χρήστη του διαδικτύου. Είναι συνηθισμένο για τους χρήστες να επεξεργάζεται μια εγγραφή και μετά να φεύγει δίχως να χρησιμοποιεί συνδέσμους "Ακύρωσης" ή "Αποσύνδεσης". Εάν υπάρχει κλείδωμα επεξεργασίας, κανένας άλλος χρήστης δεν μπορεί να επεξεργαστεί την ίδια εγγραφή και οφείλει να περιμένει να τελειώσει ο χρόνος επεξεργασίας του πρώτου χρήστη.

Το πρωτόκολλο μεταφοράς υπερκειμένου περιέχει ένα είδος μεθόδου αισιόδοξου ελέγχου ταυτοχρονισμού: Με τη μέθοδο GET επιστρέφεται ένα ETag και με τα επόμενα PUTs χρησιμοποιείται η τιμή ETag στους τίτλους If-Match. Ενώ το πρώτο PUT θα επιτύχει, το δεύτερο δεν θα το καταφέρει καθώς η τιμή στο If-Match βασίζεται στην πρώτη έκδοση του πόρου. (Nielsen, H.F. and LaLiberte, D. [1999])

Κάποια συστήματα που διαχειρίζονται βάσεις δεδομένων έχουν ενσωματωμένο τον αισιόδοξο έλεγχο ταυτοχρονισμού στον κώδικά τους (χωρίς να χρειάζεται άλλος κώδικας εφαρμογής). Για τα υπόλοιπα συστήματα, η εφαρμογή έχει έναν κώδικα αισιόδοξου ελέγχου ταυτοχρονισμού έξω από τη βάση με σκοπό την αποφυγή αναμονής είτε την αντικατάσταση εγγραφών. Όταν υπάρξει τέτοια περίπτωση, περιλαμβάνεται στην φόρμα ένα επιπλέον κρυφό πεδίο έχοντα το πρωτότυπο αρχείο μαζί με σήμανση χρόνου και αριθμό κατάταξης. Στην διαδικασία της υποβολής του γίνεται σύγκριση με πληροφορίες από την βάση. Αν υπάρχει διαφορά καλείται ο αλγόριθμος που επιλύει τέτοιες διενέξεις.

3.6.1.3 Σύστημα **Vector**

Σε αντίθεση με ορισμένα συστήματα, το Vector χρησιμοποιεί το μοντέλο "αισιόδοξου ελέγχου ταυτότητας". Σε αυτό το μοντέλο, οι συναλλαγές επιτρέπεται να πραγματοποιούν αλλαγές στα δεδομένα χωρίς να ελέγχουν για πιθανές συγκρούσεις που προκαλούνται από άλλες συναλλαγές που έχουν διαπραχθεί στο μεταξύ. Οι συγκρούσεις ελέγχονται μόνο όταν μία συναλλαγή δεσμεύεται. Αυτό δεν αναφέρεται μόνο στις συγκρούσεις που προκαλούνται από την αλλαγή των ίδιων δεδομένων, αλλά και για να περιοριστούν οι παραβιάσεις που συμβαίνουν μόνο ως αποτέλεσμα δύο ταυτόχρονων συναλλαγών που διαπράττονται (για παράδειγμα, δύο ταυτόχρονες συναλλαγές που εισάγουν μία εγγραφή με το ίδιο πεδίο UNIQUE).

Σε σύγκριση με άλλες προσεγγίσεις, ο αισιόδοξος έλεγχος ταυτότητας αποφεύγει να κλειδώνει τα αρχεία για ταυτόχρονες συναλλαγές που εξελίσσονται ακόμα. Αυτό έχει ως αποτέλεσμα καλές επιδόσεις σε περιπτώσεις όπου οι συναλλαγές συνήθως δεν έρχονται σε σύγκρουση μεταξύ τους. Η πιο ορατή πτυχή του αισιόδοξου ελέγχου ταυτόχρονης λειτουργίας όπου μερικές φορές προκαλεί προβλήματα στους

χρήστες που είναι εξοικειωμένοι με άλλες λύσεις, είναι ότι οι συγκρούσεις εντοπίζονται μόνο κατά τη διάρκεια του δεσμευμένου χρόνου για την εκτέλεση της συναλλαγής. Η λογική της εφαρμογής πρέπει να λαμβάνει υπόψη αυτό.

Κεφάλαιο 4

UPPAAL

4.1 Εισαγωγή

Το UPPAAL αποτελεί μία εργαλειοθήκη για την επαλήθευση συστημάτων σε πραγματικό χρόνο που ανέπτυξαν από κοινού το Πανεπιστήμιο της Ουψάλα (Uppsala) όπως επίσης και το Πανεπιστήμιο του Άαλμποργκ (Aalborg). Έχει πολλές εφαρμογές όπως σε περιπτωσιολογικές μελέτες που κυμαίνονται από σύνολα κανόνων επικοινωνίας έως και εφαρμογές πολυμέσων. Το πρόγραμμα δημιουργήθηκε για τον έλεγχο συστημάτων που είναι διαμορφωμένα ως συνδεδεμένα χρονισμένα αυτόματα, επεκταμένα με μεταβλητές ακέραιων αριθμών, δομημένους τύπους στοιχείων, καθορισμένες από τους χρήστες συναρτήσεις καθώς και συγχρονισμούς καναλιών.

Η πρώτη έκδοση του UPPAAL κυκλοφόρησε στα μέσα του 1990 και από τότε βρίσκεται σε διαρκή ανάπτυξη (Larsen, K.G. and Pettersson, P. and Yi, W. [October 1997]). Οι βελτιώσεις και η ανάπτυξή του έχουν ως στόχο να δημιουργήσουν δομές δεδομένων, μερική μείωση του κώδικα, μία διανεμημένη έκδοση της UPPAAL, καθοδηγούμενη με ελάχιστο κόστος προσβασιμότητας, εργάστηκαν πάνω σε UML Statecharts, με τεχνικές επιτάχυνσης και νέες δομές δεδομένων και μειώσεις μνήμης. Η έκδοση 4.0 (Behrmann, G. and Bouyer, P. and Larsen, K.G. and Radek Pelánek [September 2005]) φέρνει μειώσεις συμμετρίας, η γενικευμένη μέθοδος "σκούπισμα-γραμμών", νέες μεθόδους αφαίρεσης καθορισμένες από το χειριστή, λειτουργίες στην επικρατούσα τάση. Το UPPAAL έχει χρησιμοποιηθεί επίσης πάνω σε Ph.D. διατριβές. Το περιβάλλον χρήστη είναι φτιαγμένο με την Java ενώ η μηχανή επαλήθευσης έχει κωδικοποιηθεί σε C++. Ο ιστότοπος από τον οποίο μπορούμε να το εγκαταστήσουμε είναι το: http://www.uppaal.com/.

4.2 Χρονισμένα Αυτόματα

Ο έλεγχος μοντέλου UPPAAL είναι βασισμένο στον ορισμό των χρονισμένων αυτομάτων (Timed Automata) (Alur, R. and Dill, D.L. [1990]) και η γλώσσα που μοντελοποιείται παρέχει επιπλέον γνωρίσματα όπως οριοθετημένες μεταβλητές. Η γλώσσα που χρησιμοποιούμε για να κάνουμε ερωτήσεις στην UPPAAL και για να

γίνει ορισμός των ποικίλων χαρακτηριστικών που οφείλουμε να εξεταστούν, είναι ένα υποσύνολο του TCTL (Timed Computation Tree Logic) (Alur, R. and Courcoubetis, C. and Dill, D.L. [1990]) ή σε ελεύθερη μετάφραση Χρονισμένη Λογική Υπολογιστικού Δένδρου). Σε αυτή την ενότητα παρουσιάζουμε τις γλώσσες μοντελοποίησης και αναζήτησης της UPPAAL και δίνουμε μία διαισθητική εξήγηση του χρόνου σε χρονισμένα αυτόματα.

4.3 Η γλώσσα μοντελοποίησης

4.3.1 Δίκτυα χρονισμένων αυτομάτων

Ένα χρονισμένο αυτόματο είναι μία μηχανή που περιέχει πεπερασμένες καταστάσεις, επεκταμένη με μεταβλητές που μετρούν τον χρόνο. Χρησιμοποιεί ένα πρότυπο πυκνού-χρόνου με χρονικές μεταβλητές που αντιστοιχούν σε πραγματικές μεταβλητές. Υπάρχει συγχρονισμός μεταξύ των ρολογιών. Το UPPAAL μοντελοποιεί ένα δίκτυο παράλληλων χρονισμένων αυτομάτων για να δημιουργήσει ένα σύστημα. Το μοντέλο έχει περαιτέρω επεκτάσεις με διακριτές μεταβλητές που είναι οριοθετημένες και αποτελούν ένα μέρος της κατάστασης που βρίσκεται. Όπως ακριβώς γίνεται και στις γλώσσες προγραμματισμού, οι μεταβλητές αυτές έχουν χαρακτηριστικά όπως: ανάγνωση, γραφή και αριθμητικές πράξεις μεταξύ τους. Για να οριστεί μια κατάσταση στο σύστημα χρειάζονται οι τοποθεσίες του συνόλου των αυτομάτων, τις τιμές των ρολογιών των πεπερασμένων παραμέτρων. Τα αυτόματα ενεργοποιούν ακμές (άλλες φορές χαρακτηρίζονται μεταβάσεις) μεμονωμένες ή συγχρονισμένες με άλλα αυτόματα , με σκοπό τη μετάβαση σε μία νέα κατάσταση.

Παραπάνω δόθηκαν οι βασικοί ορισμοί για να συντάξουμε χρονισμένα αυτόματα. Παρακάτω, δε θα εμβαθύνουμε σε πιο σύνθετες επεκτάσεις της UPPAAL, δηλαδή χρησιμοποιώντας επείγουσες και αφοσιωμένες τοποθεσίες. Έχοντας κατά νου τα ακόλουθα:C αποτελεί ένα σύνολο ρολογιών και B(C) αποτελεί το σύνολο των συζευγμάτων σε απλές συνθήκες της φόρμας $x \bowtie c$ ή $x-y \bowtie c$, όπου $x,y \in C, c \in N$ και $\bowtie \in \{<,\leq,=,\geq,>\}$. Το χρονισμένο αυτόματο αποτελεί έναν πεπερασμένο κατευθυνόμενο γράφο, αναβαθμισμένο με ιδιότητες και επανεκκινήσεις μη αρνητικών ρολογιών που παίρνουν πραγματικές τιμές.

Ορισμός 1 (Χρονισμένα Αυτόματα) Ένα χρονισμένο αυτόματο είναι ενα σύνολο (L, l_0, C, A, E, I), όπου το L αποτελεί ένα σύνολο θέσεων, $l_0 \in L$ αποτελεί την αρχική θέση, C αποτελεί το σύνολο όλων των ρολογιών, το A αποτελεί ένα σύνολο ενεργειών, συνεργασιών και εσωτερικής τ-δράσης, $E \subseteq L \times A \times B(C) \times 2^C \times L$ αποτελεί ένα σύνολο ακμών μεταξύ τοποθεσιών με μία ενέργεια, μία συνθήκη και ένα σύνολο ρολογιών που πρέπει να επαναρυθμιστούν και το $I : L \longrightarrow B(C)$ εκχωρεί τις σταθερές σε τοποθεσίες. □

Στη συνέχεια θα ορίσουμε τη έννοια του χρονισμένου αυτόματου. Οι τιμές των ρολογιών αποτελούν τη συνάρτηση $u : C \longrightarrow \mathbb{R}_{\geq 0}$. Ορίζουμε το $\mathbb{R}^C$ να είναι το σύνολο όλων των αποτιμήσεων ενός ρολογιού. Έστω $u_0(x) = 0$ για όλα τα $x \in C$.

Κάνοντας μία ελαφριά κατάχρηση του συμβολισμού θεωρώντας τις συνθήκες και τις σταθερές ως σύνολα των αποτιμήσεων των ρολογιών, γράφοντας $u \in I(l)$ σημαίνει ότι η u ικανοποιεί την $I(l)$.

Ορισμός 2 (Σημασιολογία Χρονισμένων Αυτομάτων) Ορίζουμε ένα χρονισμένο αυτόματο (L, l_0, C, A, E, I). Η σημασιολογία ορίζεται ως ένα μεταβατικό σύστημα $\langle S, s_0, \longrightarrow \rangle$, όπου το $S \subseteq L \times \mathbb{R}^C$ είναι το σύνολο των καταστάσεων, το $s_0 = (l_0, u_0)$ είναι η κατάσταση από την οποία ξεκινάει και το $\longrightarrow \subseteq S \times (\mathbb{R}_{\geq 0} \cup A) \times S$ είναι η μεταβατική σχέση έτσι ώστε:

- $(l, u) \xrightarrow{d} (l, u + d)$ όταν $\forall d' : 0 \leq d' \leq d \Longrightarrow u + d' \in I(l)$ και
- $(l, u) \xrightarrow{a} (l', u')$ όταν έχουμε $e = (l, a, g, r, l') \in E$ με σκοπό το $u \in g, u' = [r \mapsto 0]u$ και $u' \in I(l')$, όταν το $d \in \mathbb{R}_{\geq 0}, u + d$ χαρτογραφεί όλα τα ρολόγια $x \in C$ με την τιμή $u(x) + d$ και $[r \to 0]u$ υποδηλώνει την τιμή του ρολογιού που χαρτογραφεί όλα τα ρολόγια από το r έως το 0 και συμπίπτει με το u ανάμεσα στο διάστημα $C \backslash r$. □

Τα χρονισμένα αυτόματα αποτελούν σύνθεση από ένα δίκτυο χρονισμένων αυτομάτων χρησιμοποιώντας ρολόγια και ενέργειας από κοινό σύνολο και κατασκευάζεται από n χρονισμένα αυτόματα $A_i = (L_i, l_i^0, C, A, E_i, I_i), 1 \leq i \leq n$. Ένα διάνυσμα τοποθεσίας αποτελεί ένα διάνυσμα $\overline{l} = (l_1, ..., l_n)$. Κάνοντας σύνθεση αμετάβλητων συναρτήσεων σε μία κοινή συνάρτηση με διανύσματα θέσης $I(\overline{l}) = \wedge_i I_i(l_i)$. Συμβολίζουμε $\overline{l}[l_i'/l_i]$ για να δηλώσουμε το διάνυσμα σύμφωνα με το οποίο το κάθε στοιχείο στην τοποθεσία i (l_i) του $\overline{l}$ μεταφέρεται από το l_i'. Παρακάτω γίνεται ο ορισμός της έννοιας ενός δικτύου χρονισμένων αυτομάτων.

Ορισμός 3 (Σημασιολογία ενός Δικτύου Χρονισμένων Αυτομάτων) Ορίζουμε $A_i = (L_i, l_i^0, C, A, E_i, I_i)$ ως ένα δίκτυο με n χρονισμένα αυτόματα. Ορίζουμε $\overline{l_0} = (l_1^0, ..., l_n^0)$ ως το αρχικό διάνυσμα θέσης. Η εννοιολογία χαρακτηρίζεται ως ένα μεταβατικό σύστημα $\langle S, s_0, \longrightarrow \rangle$ όπου το $S = (L_1 \times ... \times L_n) \times \mathbb{R}^C$ εμπεριέχει όλες τις δυνατές καταστάσεις, $s_0 = (\overline{l_0}, u_0)$ χαρακτηρίζει την αρχική κατάσταση και $\longrightarrow \subseteq S \times S$ ορίζεται ως η μεταβατική σχέση που εμπεριέχει τις παρακάτω ιδιότητες:

- $(\overline{l}, u) \xrightarrow{d} (\overline{l}, u + d)$ όταν $\forall d' : 0 \leq d' \leq d \Longrightarrow u + d' \in I(\overline{l})$.
- $(\overline{l}, u) \xrightarrow{a} (\overline{l}[l_i'/l_i], u')$ εάν υπάρχει $l_i \xrightarrow{\tau g r} l_i'$ έτσι ώστε $u \in g, u' = [r \mapsto 0]u$ και $u' \in I(\overline{l}[l_i'/l_i])$.
- $(\overline{l}, u) \xrightarrow{a} (\overline{l}[l_j'/l_j, l_i'/l_i], u')$ εάν υπάρχει $l_i \xrightarrow{c?g_i r_i} l_i'$ και $l_j \xrightarrow{c!g_j r_j} l_j'$έτσι ώστε $u \in (g_i \wedge g_j), u' = [r_i \cup r_j \mapsto 0]u$ και $u' \in I(\overline{l}[l_j'/l_j, l_i'/l_i])$. □

4.3.2 Χρονισμένα Αυτόματα Στην **UPPAAL**

Η γλώσσα μοντελοποίησης της UPPAAL επεκτείνει τα χρονισμένα αυτόματα με τις ακόλουθες πρόσθετες λειτουργίες:

- Πρότυπα : Τα πρότυπα των αυτομάτων ορίζονται με ένα σύνολο παραμέτρων που μπορεί να είναι διαφορετικού τύπου (π.χ. int, chan). Οι παραπάνω μεταβλητές διαμορφώνονται αναλόγως το όρισμα που υπάρχει όταν πραγματοποιείται η διαδικασία δήλωσης.

- Σταθερές : Οι μεταβλητές που δεν μπορούν να μεταβληθούν χαρακτηρίζονται ως const name value. Οι αμετάβλητες είναι απαράλλακτες όταν κατασκευάζονται και οφείλουν να έχουν μία ακέραια τιμή.

- Οριοθετημένες ακέραιες μεταβλητές : Οι οριοθετημένες ακέραιες μεταβλητές δηλώνονται ως int[min,max] name, όπου το min και το max είναι η χαμηλότερη και η υψηλότερη τιμή αντιστοίχως. Περιορισμοί, σταθερές και αναθέσεις μπορούν να περιέχουν εκφράσεις που κυμαίνονται μέσα στις οριοθετημένες ακέραιες μεταβλητές. Τα όρια ελέγχονται κατά την επαλήθευση και παραβιάζοντας ένα όριο, οδηγεί σε μία μη έγκυρη κατάσταση που απορρίπτεται (κατά την εκτέλεση). Όταν δεν υπάρχει κάποιο όριο από το χρήστη, τότε γίνεται χρήση των προεπιλεγμένων ορίων από τον αριθμό -32768 έως τον αριθμό 32768.

- Δυαδικός Συγχρονισμός : Τα δυαδικά κανάλια που έχουν συγχρονισμό μεταξύ τους χαρακτηρίζονται ως chan c. Μία ακμή που έχει επισημανθεί με c! επιτρέπει συγχρονισμό με μία άλλη ακμή που έχει επισημανθεί με την ετικέτα c?. Ένα ζεύγος συγχρονισμού είναι επιλεγμένο μη αιτιοκρατικά εάν διάφοροι συνδυασμοί επιτρέπονται.

- Κανάλια εκπομπής : Τα κανάλια εκπομπής γράφονται ως broadcast chan c. Όταν επιτρέπεται συγχρονισμός εκπομπής, ο μεταδότης c! συγχρονίζεται με έναν επιθυμητό πλήθος δεκτών c?.Κάθε δέκτης που μπορεί να συγχρονιστεί με την εκάστοτε κατάσταση, οφείλει να το υλοποιήσει. Εάν δεν υπάρχουν δέκτες, τότε ο μεταδότης έχει ακόμα την δυνατότητα να υλοποιήσει το c!, δηλαδή η αποστολή εκπομπών, δεν σταματάει.

- Επείγον συγχρονισμός : Τα επείγοντα κανάλια συγχρονισμού δηλώνονται γράφοντας πριν τη δήλωση καναλιού, την λέξη-κλειδί urgent. Δεν πρέπει να υπάρξουν καθυστερήσεις σε περίπτωση που ο συγχρονισμός μετάβασης σε ένα επείγον κανάλι είναι ενεργοποιημένος. Ακμές που χρησιμοποιούν τα επείγοντα κανάλια για συγχρονισμό δεν μπορούν να έχουν χρονικούς περιορισμούς, δηλ. δεν υπάρχουν περιορισμοί στα ρολόγια τους.

- Επείγουσες θέσεις : Οι επείγουσες θέσεις είναι εννοιολογικά ισοδύναμες με την προσθήκη πρόσθετου ρολογιού x, όπου μηδενίζεται με τις συνολικές ακμές που έρχονται σε αυτό και έχει αμετάβλητο $x \leq 0$ στη θέση. Ως εκ τούτου, ο χρόνος δεν συνεχίζει όταν το αυτόματο είναι σε επείγουσα θέση.

- Δεσμευμένες θέσεις : Οι δεσμευμένες θέσεις έχουν ακόμη πιο πολλούς περιορισμούς όταν εκτελούνται από τις επείγουσες θέσεις. Μία κατάσταση έχει δεσμευτεί όταν κάποια από τις θέσεις της κατάστασης έχει δεσμευτεί. Μία δεσμευμένη κατάσταση δεν έχει το δικαίωμα να αργοπορήσει και στην επερχόμενη μετάβαση, οφείλει να έχει μία ακμή που εξέρχεται σε τουλάχιστον μία από τις δεσμευμένες θέσεις.

- Λίστες : Οι λίστες χρησιμοποιούνται για ρολόγια, κανάλια, σταθερές και ακέραιες μεταβλητές. Ορίζονται με την προσάρτηση ενός μεγέθους στο όνομα της παραμέτρου, π.χ. chan c[4]; clock a[3]; int[3,5] u[7];.

- Αρχικοποιητές : Οι αρχικοποιητές υπάρχουν για την αρχικοποιηθούν ακέραιες μεταβλητές και λίστες με ακέραιες μεταβλητές. Παραδείγματος χάριν, int i = 2; είτε int i[3]={1, 2, 3}.

- Τύποι εγγραφών : Ο τύπος εγγράφου ορίζεται με χρήση της δομής struct, ακριβώς με τον ίδιο τρόπο που δηλώνεται στη C.

- Προσαρμοσμένοι τύποι : Οι προσαρμοσμένοι τύποι δηλώνονται χρησιμοποιώντας το typedef, όπως και στη C. Έχουμε τη δυνατότητα να ορίσουμε οποιουσδήποτε προσαρμοσμένους τύπους από άλλους θεμελιακούς τύπους, όπως οι καταγραφές.

- Λειτουργίες χρήστη : Με χρήση προτύπων γίνεται ο καθορισμός των λειτουργιών χρήστη τόσο παγκόσμια όσο και τοπικά. Οι τοπικές λειτουργίες χρησιμοποιούν μεταβλητές προτύπων. Το συντακτικό είναι σχεδόν ίδιο με αυτό της γλώσσας C, εκτός από το ότι δεν υπάρχει δείκτης. Μόνο για ορίσουμε μεταβλητές χρησιμοποιούμε ίδια μορφή με την C++.

4.3.3 Εκφράσεις στο **UPPAAL**

Από τα ρολόγια έως και τις ακέραιες μεταβλητές χρησιμοποιούνται οι εκφράσεις στο UPPAAL. Εκφράσεις χρησιμοποιούνται με τις ακόλουθες ετικέτες:

- Επιλογή : Μία επιλεγμένη ετικέτα περιέχει μία λίστα με όνομα που χωρίζει με κόμματα: πληκτρολογήστε εκφράσεις όπου το όνομα είναι μεταβλητό όνομα και τύπος είναι ένας ορισμένος τύπος (ενσωματωμένος ή προσαρμοσμένος). Με τη χρήση μίας σχετικής ακμής έχουμε πρόσβαση στις μεταβλητές αυτές και οι τελευταίες λαμβάνουν μη-ντετερμινιστικές ανάλογα με τον τύπο τους.

- Περιορισμός : Οι περιορισμοί αποτελούν ιδιαίτερες εκφράσεις που ικανοποιούν τις παρακάτω προϋποθέσεις: όταν έχουμε τύπους δεδομένων αληθείας αποτελεί μία ενέργεια δίχως επιπτώσεις. Χρησιμοποιείται αποκλειστικά σε μεταβλητές ρολογιών, ακέραιων μεταβλητών καθώς και σταθερών (ή μίξη των προηγούμενων τύπων). Η σύγκριση των ρολογιών για τις διαφορές τους γίνεται μόνο με ακέραιες εκφράσεις. Οι περιορισμοί πάνω στα ρολόγια είναι ουσιαστικά συζεύξεις (επιτρέπονται οι διαχωρισμοί σε ακέραιες συνθήκες). Ένας περιορισμός μπορεί να καλέσει μία συνάρτηση χωρίς επιπτώσεις που

επιστρέφει έναν τύπο δεδομένων αληθείας(**boolean**), μολονότι οι χρονικοί περιορισμοί δεν έχουν υποστήριξη σε τέτοιου είδους λειτουργίες.

- Συγχρονισμός : Μία ετικέτα συγχρονισμού είναι υπό τη μορφή **Expression!** ή με την έκφραση **Expression?** είτε είναι μία κενή ετικέτα. Η έκφραση πρέπει να είναι μία ενέργεια χωρίς επιπτώσεις, εκτιμάται σε ένα κανάλι ενώ έχει αναφορά αποκλειστικά σε ακέραιους αριθμούς, αμετάβλητες και κανάλια.

- Ενημέρωση : Μία ετικέτα ενημέρωσης είναι μία λίστα εκφράσεων, διαχωρισμένη με κόμμα, με παράπλευρη επίδραση. Οι εκφράσεις οφείλουν να έχουν αναφορά μόνο σε ρολόγια, ακέραιες παραμέτρους και σταθερές. Πρέπει να εκχωρούνται αποκλειστικά ακέραιες τιμές στα ρολόγια. Επιπλέον, έχουνε την δυνατότητα να καλέσουν συναρτήσεις.

- Αμετάβλητο : Ένα αμετάβλητο είναι έκφραση που επιτυγχάνει τις ακόλουθες συνθήκες: είναι μία ενέργεια χωρίς επιπτώσεις. Αναφέρεται μόνο για ρολόγια, ακέραιες μεταβλητές και σταθερές. Είναι ένας συνδυασμός από συνθήκες που έχουνε μορφή $x < e$ ή $x <= e$ όταν το x αναπαριστά ρολόι αναφοράς και το e ισούται με έναν ακέραιο αριθμό. Ένα αμετάβλητο μπορεί να καλέσει μία συνάρτηση χωρίς επιπτώσεις που επιστρέφει έναν τύπο δεδομένων αληθείας(**boolean**), μολονότι τα χρονικά όρια δεν έχουν υποστήριξη σε τέτοιου είδους λειτουργίες.

4.4 Η γλώσσα επερώτησης στο **UPPAAL**

Ο κύριος σκοπός ενός μοντέλου-ελεγκτή είναι να επαληθεύσει το μοντέλο με μία προδιαγραφή των απαιτήσεων. Όπως και το μοντέλο, η προδιαγραφή των απαιτήσεων πρέπει να εκφραστεί σε μία, τυπικά καθορισμένη και μηχανικά αναγνώσιμη, γλώσσα. Πολλές παρόμοιες λογικές είναι υπαρκτές σε επιστημονικές βιβλιογραφίες και το UPPAAL κάνει χρήση μίας ευκολονόητης έκδοσης της χρονισμένης λογικής υπολογιστικού δένδρου. Όπως γίνεται στη χρονισμένη λογική υπολογιστικού δένδρου, η γλώσσα επερώτησης χρησιμοποιεί φόρμουλες μονοπατιού και φόρμουλες κατάστασης. Οι φόρμουλες κατάστασης περιγράφουν μεμονωμένες καταστάσεις ενώ οι φόρμουλες μονοπατιού προσδιορίζουν ποσοτικά μονοπάτια ή ίχνη του μοντέλου. Οι φόρμουλες μονοπατιού μπορούν να ταξινομηθούν με βάση την προσβασιμότητα, την ασφάλεια και τη ζωτικότητα. Το Σχήμα 4.1 παρουσιάζει τους διάφορους τύπους διαδρομής που χρησιμοποιεί το UPPAAL και παρουσιάζεται πιο κάτω, έπειτα από την εξήγηση των διάφορων τύπων διαδρομών. Κάθε τύπος περιγράφεται παρακάτω:

Φόρμουλα Κατάστασης : Μία φόρμουλα κατάστασης είναι μία έκφραση που χρησιμοποιείται για να αξιολογηθεί κάποια θέση, δίχως να χρειάζεται να εξεταστούν τα χαρακτηριστικά του μοντέλου. Παραδείγματος χάρη, αυτό θα μπορούσε να είναι μία απλή έκφραση, όπως $\mathrm{i} == 7$ όπου αυτό είναι αληθές σε μία κατάσταση κάθε φορά που το i είναι ίσο με 7. Η σύνταξη της φόρμουλας κατάστασης είναι ένα υπερσύνολο της σύνταξης των περιορισμών, δηλαδή η φόρμουλα κατάστασης

είναι μία έκφραση χωρίς επιπτώσεις, αλλά σε αντίθεση με τους περιορισμούς, η χρήση των διαχωρισμών δεν απαγορεύεται. Είναι επίσης δυνατό να δοκιμαστεί εάν μία συγκεκριμένη διαδικασία σε μία δεδομένη τοποθεσία χρησιμοποιεί μία έκφραση στη μορφή P.l, όταν το P χαρακτηρίζεται σαν μία διαδικασία και το l είναι μία θέση.

Στο UPPAAL, το αδιέξοδο εκφράζεται χρησιμοποιώντας έναν ειδικό τύπο κατάστασης (αν και αυτό δεν είναι αυστηρά μία φόρμουλα κατάστασης). Ο τύπος αυτό αποτελείται από τη λέξη-κλειδί 'deadlock' και ικανοποιείται πάντοτε για όλες τις καταστάσεις που χαρακτηρίζονται αδιέξοδες. Μία κατάσταση χαρακτηρίζεται ως μία κατάσταση αδιεξόδου όταν δεν υπάρχουν μεταβάσεις για να γίνουν εξερχόμενες ενέργειες τόσο από την ίδια την κατάσταση που βρισκόμαστε όσο και από καθυστερημένους διαδόχους. Επειδή το UPPAAL έχει ακόμα αρκετούς περιορισμούς, ο τύπος κατάστασης αδιεξόδου χρησιμοποιείται μόνο με φόρμουλες που παρέχουν πρόσβαση και διαδρομές που δεν μεταβάλλονται όπως αναλύεται στη συνέχεια.

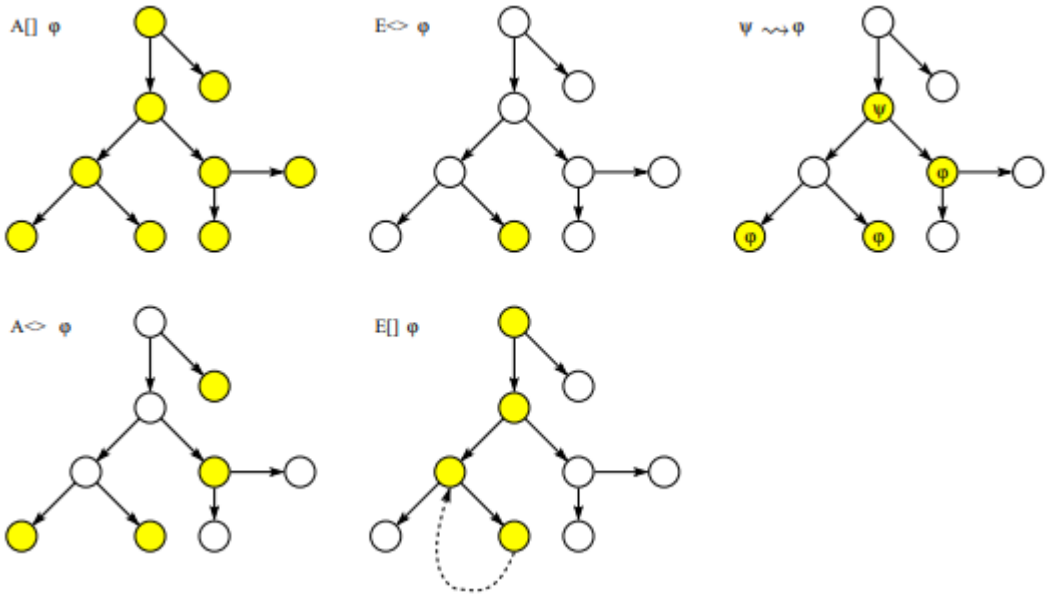
Ιδιότητες προσβασιμότητας : Η πιο απλή μορφή ιδιοτήτων αποτελούν οι ιδιότητες προσβασιμότητας. Θέτουν ερωτήσεις για το εάν μία δεδομένη φόρμουλα καταστάσεων, φ, μπορεί να ικανοποιηθεί από κάθε δυνατή κατάσταση. Ένας άλλος τρόπος για να δηλωθεί αυτό είναι: Υπάρχει μία πορεία, ξεκινώντας από μία αρχική θέση, ώστε τελικά το φ να ικανοποιείται σε αυτό το μονοπάτι.
Απαραίτητο εργαλείο τόσο για τον σχεδιασμό όσο και για τον έλεγχο νομιμότητας ενός μοντέλου αποτελούν οι ιδιότητες προσβασιμότητας. Παραδείγματος χάριν, δημιουργώντας ένα πρότυπο πρωτοκόλλου επικοινωνίας μεταξύ ενός αποστολέα και ενός παραλήπτη, έχει νόημα η ερώτηση εάν υπάρχει η δυνατότητα για τον αποστολέα να στείλει ένα μήνυμα είτε εάν μπορεί κάποιο μήνυμα να ληφθεί από τον παραλήπτη. Η ορθότητα του πρωτοκόλλου δεν επαληθεύεται μόνο με τις παραπάνω ιδιότητες, δηλαδή ότι κάθε μήνυμα τελικά θα παραδοθεί κάποια στιγμή, αλλά γίνεται η βασική επικύρωση του προτύπου.
Εκφράζουμε ότι κάποια κατάσταση που ικανοποιεί την φ θα πρέπει να είναι προσβάσιμη χρησιμοποιώντας φόρμουλα μονοπατιού $E\diamond\varphi$. Στο UPPAAL, συντάσσεται αυτή η ιδιότητα με την ακόλουθη έκφραση: $E <> \varphi$.

Ιδιότητες ασφαλείας : Οι ιδιότητες ασφαλείας είναι στη μορφή:"κάτι κακό δεν θα συμβεί ποτέ". Παραδείγματος χάριν, σε ένα εργοστάσιο παραγωγής ηλεκτρικού ρεύματος με πυρηνική ενέργεια, δύο από τις ιδιότητες ασφαλείας είναι οι εξής: να παραμένει πάντοτε η θερμότητα λειτουργίας χαμηλότερη από μία μέγιστη τιμή και να μην καταρρεύσει ποτέ το εργοστάσιο. Μία παραλλαγή αυτής της κατάστασης είναι αυτή: "Κάποιο γεγονός με βάση τις πιθανότητες δεν θα γίνει ουδέποτε". Παραδείγματος χάρη, όταν παίζετε ένα παιχνίδι, μία επιβεβαιωμένη κατάσταση είναι μία κατάσταση στην οποία μπορούμε ακόμα να κερδίσουμε το παιχνίδι, επομένως, πιθανώς δεν θα το χάσουμε.

Στο UPPAAL, αυτές οι ιδιότητες είναι διατυπωμένες με θετικό τρόπο, π.χ. κάτι καλό είναι αμετάβλητα αληθές. Έστω ότι η φ είναι μία φόρμουλα κατάστασης. Δηλώνουμε ότι η φ πρέπει να είναι αληθινή σε όλες τις διαθέσιμες καταστάσεις

με τις φόρμουλες μονοπατιού $A\Box\varphi$ (παρατηρούμε ότι : $A\Box\varphi = \neg E \diamond \neg\varphi$), ενώ το $E\Box\varphi$ λέει ότι πρέπει να υπάρχει ένα μέγιστο μονοπάτι έτσι ώστε η φ να είναι πάντα αληθής. Ένα μέγιστο μονοπάτι είναι μία διαδρομή που είτε είναι άπειρη είτε βρίσκεται στη τελευταία κατάσταση και δεν έχει εξερχόμενες μεταβάσεις. Στο UPPAAL, αυτό γράφεται $A[]\varphi$ και $E[]\varphi$ αντίστοιχα.

Ιδιότητες ζωτικότητας : Οι ιδιότητες ζωτικότητας έχουν την μορφή: κάποιο γεγονός στο τέλος θα πραγματοποιηθεί, π.χ. κάθε φορά που πατάτε το πλήκτρο ενεργοποίησης του τηλεκοντρόλ της τηλεόρασης, τότε η οθόνη οφείλει να ανάψει. Άλλο παράδειγμα που σχετίζεται με πρωτόκολλο επικοινωνίας είναι ότι όλα τα μηνύματα που έχουν σταλθεί, στο τέλος οφείλουν να έχουν ληφθεί.
Στην απλή του μορφή, η ζωτικότητα εκφράζεται με τον τύπο διαδρομής $A\diamond\varphi$, που σημαίνει ότι το φ κάποια στιγμή θα ικανοποιηθεί (παρατηρούμε ότι : $A\diamond\varphi = \neg E\Box\neg\varphi$). Η πιο χρήσιμη μορφή είναι τα άκρα ή η απόκριση ιδιότητα, η οποία γράφεται $\varphi \rightsquigarrow \psi$ το οποίο διαβάζεται ως όποτε το φ ικανοποιείται, τότε τελικά και το ψ θα ικανοποιηθεί, π.χ. κάθε φορά που αποστέλλεται ένα μήνυμα, τότε τελικά θα ληφθεί (Οι ειδικοί της χρονισμένης λογικής υπολογιστικού δένδρου θα αναγνωρίσουν ότι το $\varphi \rightsquigarrow \psi$ είναι ισοδύναμο με το $A\Box(\varphi \Longrightarrow A\diamond\psi)$). Στο UPPAAL δηλώνουμε αυτές τις ιδιότητες ως εξής: $A <> \varphi$ όπως και $\varphi --> \psi$, αντίστοιχα.



Σχήμα 4.1

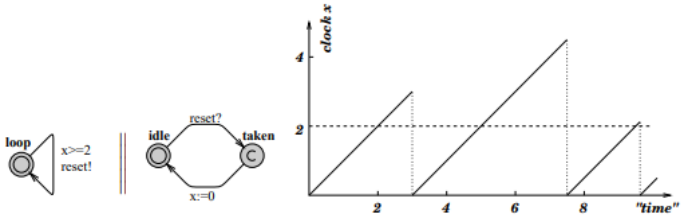
4.5 Κατανοώντας το χρόνο

Σταθερές και Περιορισμοί : Το UPPAAL χρησιμοποιεί ένα μοντέλο συνεχούς χρόνου. Με την βοήθεια ενός απλού παραδείγματος που κάνει χρήση έναν παρατηρητή μπορούμε να αντιληφθούμε την έννοια του χρόνου για το UPPAAL. Κανονικά, ένας παρατηρητής είναι ένα πρόσθετο αυτόματο που είναι υπεύθυνο για την ανίχνευση γεγονότων χωρίς να αλλάζει το παρατηρούμενο σύστημα. Στο προηγούμενο παράδειγμα, όταν επαναφέρουμε το ρολόι στο μηδέν ($(x := 0)$) παρομοιάζεται με έναν παρατηρητή για να γίνει πιο απλή η κατανόησή του.

Στο Σχήμα 4.2 παρουσιάζεται το πρώτο μοντέλο μαζί με τον παρατηρητή που έχει. Υπάρχουνε δύο αυτόματα που τρέχουν σε παράλληλη σειρά. Το πρώτο αυτόματο έχει μία κατάσταση που επιστρέφει στον εαυτό της με συνθήκη $x >= 2$, όπου το x είναι ένα ρολόι, που συγχρονίζεται με το κανάλι reset στο δεύτερο αυτόματο. Το δεύτερο αυτόματο, ο παρατηρητής, ανιχνεύει πότε έχει ικανοποιηθεί η ακμή του πρώτου αυτόματου με την τοποθεσία taken και στη συνέχεια έχει μία ακμή που επιστρέφει στο idle που επαναφέρει το ρολόι x. Μετακινήσαμε την επαναφορά του ρολογιού x από τη κατάσταση που επιστρέφει στον εαυτό της στον παρατηρητή, μόνο για να ελέγξουμε τι συμβαίνει στη μετάβαση πριν από την επαναφορά. Τελικά, καταλήγουμε στο συμπέρασμα ότι ο κόμβος taken έχει δέσμευση χρόνου (γράφεται με c) για να μην υπάρξει καμία καθυστέρηση όταν βρίσκεται σε αυτήν την κατάσταση.

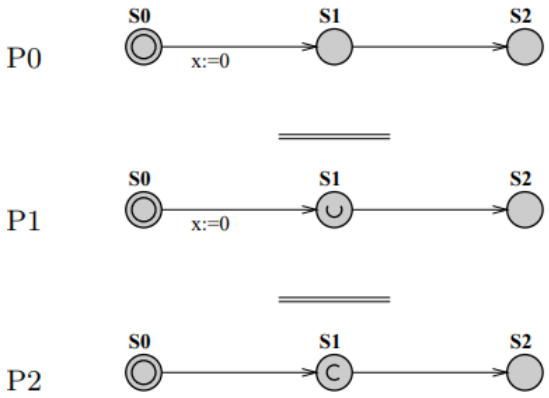
Τα ακόλουθα χαρακτηριστικά επαληθεύονται πλήρως στο UPPAAL. Με την υπόθεση ότι υπάρχει ένα αυτόματο Obs που παρατηρεί, έχουμε:

- $A[]$ **Obs.taken imply** $x >= 2$: Το ρολόι x θα επαναφέρεται κάθε φορά που το x είναι πάνω από 2. Άπο αυτό το ερώτημα συμπεραίνουμε ότι για όλες τις καταστάσεις που είναι προσβάσιμες, όταν ήμαστε στην κατάσταση Obs.taken, έχουμε $x >= 2$.
- $E <>$ **Obs.idle and** $x > 3$: Με αυτήν την έκφραση εννοείται ότι υπάρχει πιθανότητα να φτάσουμε σε μία προσβάσιμη θέση στην οποία το αυτόματο Obs βρίσκεται στη θέση idle ενώ ταυτόχρονα το x θα είναι μεγαλύτερο από 3. Βασικά κάνουμε τον έλεγχο καθυστέρησης τουλάχιστον 3 μονάδων ώρας ανάμεσα στις επαναφορές. Το αποτέλεσμα θα ήταν το ίδιο και για μεγαλύτερες τιμές, όπως το 30000, αφού δεν υπάρχουν σταθερές σε αυτό το μοντέλο.



Σχήμα 4.2

Δεσμευμένες και επείγουσες καταστάσεις : Τρεις είναι οι διαφορετικοί τύποι για μία κατάσταση στο UPPAAL: κανονικές καταστάσεις που έχουνε ή δεν έχουνε σταθερές (π.χ., όπως στο παραπάνω παράδειγμα με $x <= 2$), επείγουσες καταστάσεις και δεσμευμένες καταστάσεις. Το Σχήμα 4.3 απεικονίζει τρία αυτόματα για να τονίσει τη διαφορά μεταξύ τους. Η θέση σημειωμένη με u χαρακτηρίζεται επείγουσα (urgent) και η θέση σημειωμένη με c χαρακτηρίζεται δεσμευμένη (committed). Τα ρολόγια λειτουργούν τοπικά στα αυτόματα, δηλαδή ο μηδενισμός του x στο $P0$ δεν επηρεάζει το x στο $P1$.



Σχήμα 4.3

Για να κατανοήσουμε τη διαφορά μεταξύ των κανονικών καταστάσεων και των επειγουσών καταστάσεων, μπορούμε να παρατηρήσουμε ότι διατηρούνται οι ακόλουθες ιδιότητες:

- $E <> P0.S1$ **and** $P0.x > 0$: Είναι πιθανό να περιμένει με στην κατάσταση $S1$ του αυτομάτου $P0$.
- $A[]P1.S1$ **imply** $P1.x == 0$: Δεν είναι πιθανό να περιμένει με στην κατάσταση $S1$ του αυτομάτου $P1$.

Μία επείγουσα κατάσταση είναι ίση με μία θέση με εισερχόμενες ακμές που μηδενίζουν κάποιο ρολόι y και χαρακτηρίζονται με τη σχέση $y <= 0$. Ο χρόνος μπορεί να μην προχωράει σε μία επείγουσα κατάσταση, αλλά επιτρέπονται παρεμβολές με κανονικές καταστάσεις.

Μία δεσμευμένη κατάσταση εμπεριέχει περισσότερες οριοθετήσεις: για όλες τις καταστάσεις όπου το $P2.S1$ παραμένει ενεργό (όπως στο προηγούμενο παράδειγμά), η μόνη δυνατή μετακίνηση είναι η ακμή που εξέρχεται από το $P2.S1$. Μία θέση χαρακτηρίζεται δεσμευμένη όταν έχει δεσμευμένη τοποθεσία: καθυστερήσεις δεν επιτρέπονται και η δεσμευμένη θέση οφείλει να παραμείνει σε κατάσταση μετάβασης (είτε μία από τις δεσμευμένες καταστάσεις εφόσον υπάρχουν αρκετές).

4.6 Επισκόπηση της Εργαλειοθήκης του **UPPAAL**

4.6.1 Γραφικό περιβάλλον της **Java**

Η σκέψη για την δημιουργία αυτού του εργαλείου είναι η μοντελοποίηση ενός συστήματος με χρονισμένα αυτόματα. χρησιμοποιώντας ένα γραφικό επεξεργαστή, προσομοιώνοντας τον για να επιβεβαιώσει ότι συμπεριφέρεται όπως έχει προβλεφθεί, και τέλος για να επιβεβαιώνει ότι είναι σωστό, σεβόμενος το σύνολο των ιδιοτήτων που πρέπει να ισχύουν. Το γραφικό περιβάλλον χρήστη (GUI) της Java αντικατοπτρίζει την σκέψη αυτή. Διαιρείται σε 3 τμήματα: τον συντάκτη, τον προσομοιωτή και τον επαληθευτή, που είναι διαθέσιμα από τις 3 καρτέλες του προγράμματος.

4.6.1.1 Ο Συντάκτης

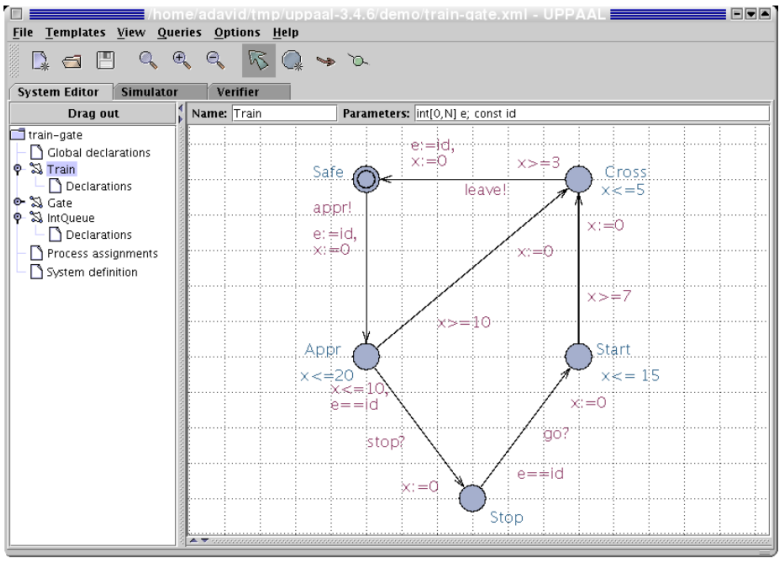
Ένα σύστημα χαρακτηρίζεται ως ένα δίκτυο χρονισμένων αυτομάτων, που ορίζονται διαδικασίες (**processes**) στο εργαλείο και λειτουργούν παράλληλα. Μία διαδικασία παράγεται από ένα παραμετροποιήσιμο πρότυπο. Ο συντάκτης διαιρείται σε δύο επιπλέον τμήματα: ένα πλαίσιο δέντρου για να αποκτήσουμε πρόσβαση στα διαφορετικά πρότυπα και δηλώσεις όπως και ένα πρόγραμμα επεξεργασίας κειμένου/ καμβά. Το Σχήμα 4.4 δείχνει τον επεξεργαστή με ένα παράδειγμα πύλης αμαξοστοιχίας του τμήματος 4. Οι θέσεις είναι επισημασμένες με ποικίλα ονόματα ενώ οι σταθερές όπως και οι ακμές λειτουργούν με περιοριστικές συνθήκες (π.χ. $e == id$), συγχρονισμό (π.χ., $go?$) καθώς και αναθέσεις (για παράδειγμα, $x := 0$).

Σχήμα 4.4 : Το αυτόματο του παραδείγματος της πύλης τρένων. Έχοντας ενεργοποιήσει το κουμπί επιλογής (**select**) που βρίσκεται στα εργαλεία, ο χρήστης μπορεί να επιλέγει θέσεις και ακμές ή να επεξεργάζεται ετικέτες. Με τις υπόλοιπες επιλογές μπορεί να προσθέσει καταστάσεις, ακμές και κορυφές στις ακμές, όπου τα τελευταία ονομάζονται καρφιά (**nails**). Δημιουργώντας μία νέα κατάσταση δεν υπάρχει προεπιλεγμένο όνομα. Μέσα σε δύο πεδία κειμένου επιτρέπεται στο χρήστη να ορίσει όνομα σε μία κατάσταση καθώς και στις παραμέτρους της. Μία συντόμευση είναι η εξής: Πιέζοντας τη ροδέλα του ποντικιού πάνω στον καμβά προσθέτεις νέα στοιχεία, καταστάσεις, ακμές είτε καρφιά αντίστοιχα.

Το δέντρο στο αριστερό μέρος παρέχει προσπέλαση σε διάφορα μέρη του συστήματος αναπαράστασης:

Συνολικές δηλώσεις : Περιέχει συνολικές ακέραιες μεταβλητές, ρολόγια, συγχρονισμό καναλιών και σταθερές.

Πρότυπα : Το **Train**, **Gate**, και το **IntQueue** είναι διαφορετικά, παραμετροποιήσιμα χρονισμένα αυτόματα. Ένα πρότυπο έχει τοπικές καταγραφές παραμέτρων,



Σχήμα 4.4

καναλιών και σταθερών.

Αναθέσεις διαδικασιών : Τα πρότυπα εμφανίζονται σε διαδικασίες. Η ενότητα επεξεργασίας διαδικασιών περιέχει δηλώσεις για αυτές τις περιπτώσεις.

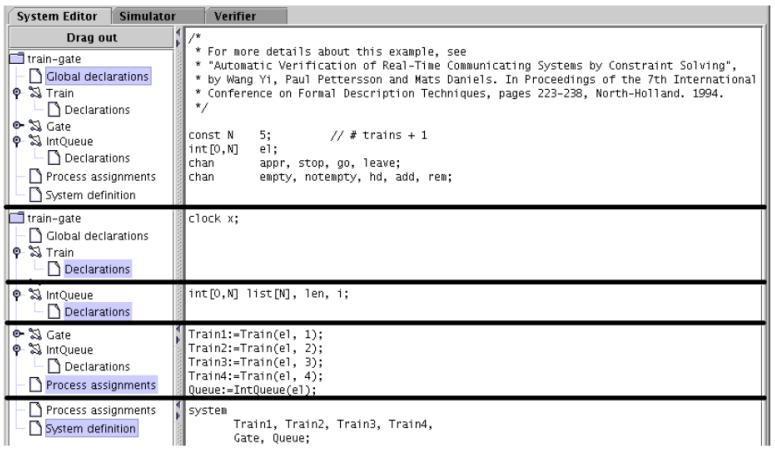
Ορισμός συστήματος : Ο κατάλογος με τις διαδικασίες του συστήματος.

Με το εργαλείο "βοήθεια συστήματος" παρέχεται βοήθεια σύνταξης ετικετών και δηλώσεων. Οι τοπικές και οι συνολικές δηλώσεις εμφανίζονται στο Σχήμα 4.5. Η γραφική σύνταξη του καμβά είναι άμεσα εμπνευσμένη από τον τρόπο που περιγράφονται τα χρονισμένα αυτόματα.

4.6.1.2 Ο Προσομοιωτής

Ο προσομοιωτής έχει 3 διαφορετικές μεθόδους λειτουργίας: ένας χρήστης έχει την δυνατότητα να δουλέψει τα αυτόματα χειροκίνητα, επιλέγοντας να πραγματοποιήσει συγκεκριμένες μεταβάσεις. Η τυχαία λειτουργία όπου αφήνει τα αυτόματα να λειτουργήσουν μόνα τους ή ο χρήστης μπορεί να περάσει από ένα μονοπάτι (όπου το εισάγει ή το επαληθεύει ο επαληθευτής) έτσι ώστε να παρακολουθεί εάν ορισμένες καταστάσεις είναι εφικτές. Το Σχήμα 4.6 δείχνει τον προσομοιωτή και χωρίζεται σε 4 μέρη:

Το τμήμα ελέγχου : Το τμήμα ελέγχου χρησιμοποιείται για να επιλέξει και να ενεργοποιήσει μεταβάσεις, να περάσει από ένα μονοπάτι και να μεταβάλλει την τυχαία προσομοίωση.



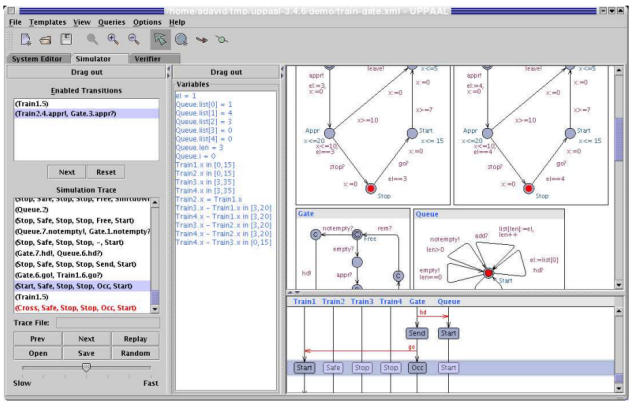
Σχήμα 4.5

Η προβολή μεταβλητών : Η απεικόνιση παραμέτρων παρουσιάζει τις τιμές των παραμέτρων και τους περιορισμούς των ρολογιών. Το UPPAAL δεν εμφανίζει συγκεκριμένες θέσεις με πραγματικές τιμές ρολογιών. Δεδομένου ότι υπάρχουν απεριόριστες από αυτές τις καταστάσεις, το UPPAAL παρουσιάζει σύνολα συγκεκριμένων θέσεων, γνωστές ως συμβολικές καταστάσεις. Οι συνολικές καταστάσεις σε συμβολική κατάσταση έχουν ακριβώς το ίδιο διάνυσμα θέσης και τιμές για τις παραμέτρους τους. Οι τιμές που μπορεί να πάρει ένα ρολόι διέπονται από ένα σύνολο κανόνων. Τα ρολόγια που είναι αποδεκτά στις συμβολικές καταστάσεις αποτελούν αυτά που ικανοποιούν όλα τα χαρακτηριστικά.

Η προβολή του συστήματος : Μέσα από τη προβολή συστήματος βλέπουμε όλα τα άμεσα ενεργοποιημένα αυτόματα καθώς και τις ενεργές θέσεις της κατάστασης που βρισκόμαστε εκείνη τη χρονική στιγμή.

Το διάγραμμα αλληλουχίας μηνυμάτων : Το διάγραμμα αλληλουχίας μηνυμάτων δείχνει τους συγχρονισμούς μεταξύ των διαφορετικών διαδικασιών καθώς και τις ενεργές θέσεις σε κάθε βήμα.

Σχήμα 4.6 :Προβολή της καρτέλας "προσομοίωσης"(Simulator) για το παράδειγμα πύλης τρένων. Η απόφαση για το εάν να επιλέξουμε μία μετάβαση, καθορίζει την ερμηνεία του περιοριστικού συστήματος για το μεταβλητό πλαίσιο. Όταν δεν επιλέγουμε μετάβαση, τότε το περιοριστικό σύστημα παρουσιάζει όλες τις πιθανές τιμές των ρολογιών που μπορούν να διαμορφωθούν κατά μήκος ενός μονοπατιού. Όταν μία μετάβαση επιλέγεται, τότε παρουσιάζονται μόνο οι τιμές του ρολογιού με βάση τις οποίες μπορεί να επιτευχθεί μία μετάβαση. Στο σύστημα βοήθειας παρουσιάζονται όλες οι συντομεύσεις πληκτρολογίου για να επιτευχθεί πλοήγηση στο προσομοιωτή δίχως ποντίκι.



Σχήμα 4.6

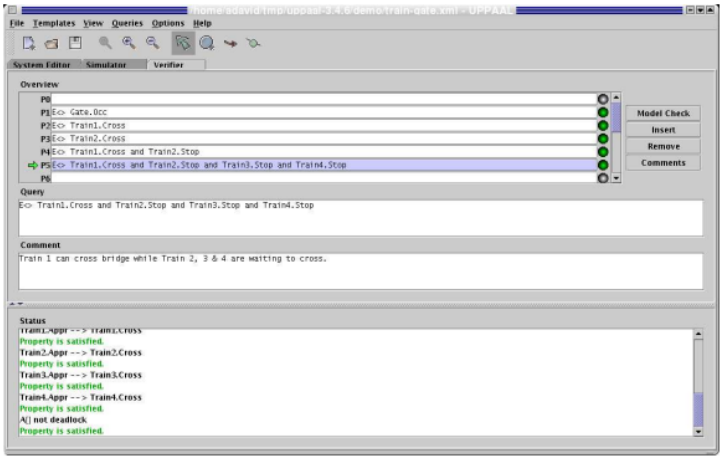
4.6.1.3 Ο Επαληθευτής

Η καρτέλα του επαληθευτή φαίνεται στο σχήμα 4.7. Οι ιδιότητες είναι επιλέξιμες στη λίστα "Επισκόπηση" (Overview). Ο χρήστης μπορεί να ελέγξει μία ή περισσότερες ιδιότητες του μοντέλου (μερικές ιδιότητες μόνο αν δεν δημιουργείται ίχνος), να προσθέσει ή να καταργήσει χαρακτηριστικά και να αλλάξει την εμφάνιση που προβάλλει τα χαρακτηριστικά μέσα στη λίστα. Μόλις επιλέγεται ένα χαρακτηριστικό, μπορούμε να διαμορφώσουμε τον ορισμό του (π.χ., $E <> Train1.Cross$ μαζί με $Train2.Stop$...) είτε παρατηρήσεις που τεκμηριώνουν τι σημαίνει ανεπίσημα η ιδιότητα αυτή. Το πλαίσιο "Κατάστασης" (Status) από κάτω παρουσιάζει την σύνδεση με το διακομιστή.

Μόλις ενεργοποιείται η κατασκευή μονοπατιών και ο ελεγκτής του μοντέλου εντοπίζει ένα μονοπάτι, τότε ο χρήστης ερωτάται αν θέλει να την εισαγάγει στον προσομοιωτή. Οι ικανοποιημένες ιδιότητες σημειώνονται με πράσινο χρώμα και οι παραβιασμένες σημειώνονται με κόκκινο χρώμα. Σε περίπτωση που έχει επιλεγεί στο μενού επιλογών είτε υπερβολική προσέγγιση (over approximation) είτε ανεκτική προσέγγιση (under approximation), τότε μπορεί η επαλήθευση να είναι αβέβαιη, αναλόγως την προσέγγιση που χρησιμοποιείται. Όταν υπάρξει αυτή η πιθανότητα, τα χαρακτηριστικά μαρκάρονται με κίτρινο χρώμα.

4.6.2 Ο αυτόνομος Επαληθευτής

Όταν εκτελείτε μεγάλες εργασίες επαλήθευσης, είναι συχνά δύσχρηστη η εκτέλεση αυτών από το εσωτερικό του γραφικού περιβάλλον χρήστη. Σε τέτοιες περιπτώσεις, ο επαληθευτής που χρησιμοποιεί γραμμές εντολών και χαρακτηρίζεται ως "verifyta" είναι πιο κατάλληλος. Επιπλέον, με τη χρήση του, εκτελείται πιο εύκολα η επαλήθευση σε απομακρυσμένο μηχάνημα UNIX ελευθερώνοντας σημαντικό ποσοστό μνήμης από στο σύστημα. Δέχεται εντολές γραμμής εντολών, χρησιμοποιώντας τόσες επιλογές, όσες έχει διαθέσιμες το γραφικό περιβάλλον χρήστη.


File Templates View Queries Options Help
System Editor Simulator Verifier
Overview
P0
P1 E<> Gate.Occ
P2 E<> Train1.Cross
P3 E<> Train2.Cross
P4 E<> Train1.Cross and Train2.Stop
P5 E<> Train1.Cross and Train2.Stop and Train3.Stop and Train4.Stop
P6
Model Check
Insert
Remove
Comments
Query
E<> Train1.Cross and Train2.Stop and Train3.Stop and Train4.Stop
Comment
Train 1 can cross bridge while Train 2, 3 & 4 are waiting to cross.
Status
Train2.Appr --> Train2.Cross
Property is satisfied.
Train3.Appr --> Train3.Cross
Property is satisfied.
Train4.Appr --> Train4.Cross
Property is satisfied.
A[] not deadlock
Property is satisfied.

Σχήμα 4.7

Κεφάλαιο 5

Επαλήθευση κλειδώματος δύο φάσεων στο **UPPAAL**

5.1 Η έννοια της συναλλαγής

Οι λογικά σχετικές λειτουργίες σε μια συναλλαγή μπορεί να περιλαμβάνουν λειτουργίες βάσης δεδομένων (λειτουργίες ανάγνωσης που διαβάζουν δεδομένα από τη βάση δεδομένων και λειτουργίες εγγραφής που τροποποιούν δεδομένα στη βάση δεδομένων) και άλλους υπολογισμούς που δεν αλληλεπιδρούν με τη βάση δεδομένων. Οι λειτουργίες ανάγνωσης και εγγραφής είναι ατομικές λειτουργίες, δηλ. Δεν συμβαίνουν ενδιάμεσο φύλλο σε αυτές τις λειτουργίες. Μια λειτουργία ανάγνωσης ή εγγραφής μπορεί να συσχετιστεί με ένα υπόθετο ότι όλες οι πλειάδες που ικανοποιούν το υπόθετο διαβάζονται ή τροποποιούνται. Σε αυτό το άρθρο δεν θεωρούμε λειτουργίες που βασίζονται σε κατηγορηματικά.

Μία συναλλαγή σχετίζεται με θεμελιακά στοιχεία διαχείρισης συναλλαγών. Το στοιχείο "Έναρξη" ενημερώνει τον διαχειριστή συναλλαγών για την έναρξη μιας συναλλαγής, ενώ τα στοιχεία "Εκτέλεση" και "Ματαίωση" υποδεικνύουν τον τερματισμό της συναλλαγής όταν απελευθερώνονται όλοι οι πόροι του συστήματος που διαθέτει η συναλλαγή. Όταν πραγματοποιείται μια συναλλαγή, οι αλλαγές που πραγματοποιούνται από αυτήν τη συναλλαγή αποθηκεύονται μόνιμα στη βάση δεδομένων και γίνονται ορατές σε άλλες συναλλαγές. Όταν μια συναλλαγή διακόπτεται, οι αλλαγές που πραγματοποιούνται από αυτήν τη συναλλαγή αναιρούνται. Ενώ μπορεί να υπάρχουν και άλλα θεμελιακά στοιχεία ανάλογα με τον συγκεκριμένο διαχειριστή συναλλαγών. Η Έναρξη, η Εκτέλεση και η Ματαίωση είναι τα βασικά που καθορίζουν το όριο μιας συναλλαγής και συνεπώς, είναι τα πρωτόγονα που εξετάζει η προσέγγιση μοντελοποίησης. Σε ένα RTDBMS, οι συναλλαγές συνδέονται με προθεσμίες, που σημαίνει ότι οι συναλλαγές πρέπει να ολοκληρωθούν εντός των καθορισμένων μονάδων ώρας μετά την έναρξη της.

5.2 Έλεγχος ταυτότητας

Ένα σύστημα διαχείρισης συναλλαγών αποτρέπει την ασυνέπεια των δεδομένων που προκαλείται από ταυτόχρονες συναλλαγές πρόσβασης στα ίδια δεδομένα μέσω ελέγχου ταυτότητας. Επικεντρωνόμαστε σε έναν τύπο ελέγχου ταυτόχρονης ταυτότητας, τον απαισιόδοξο έλεγχο ταυτότητας (PCC), ο οποίος εφαρμόζεται συνήθως σε σύγχρονα συστήματα βάσεων δεδομένων. (Elmasri, R.A. and Navathe, S.B. [2004])

Τα απαισιόδοξα πρωτόκολλα ελέγχου ταυτότητας χρησιμοποιούν τεχνικές κλειδώματος για την αποφυγή παρεμβολών από ταυτόχρονες συναλλαγές. Βασικά, μία συναλλαγή απαιτεί ένα αντίστοιχο κλείδωμα πριν αποκτήσει πρόσβαση στα δεδομένα και να αποδεσμεύσει το κλείδωμα μετά τη χρήση των δεδομένων. Ο διαχειριστής ελέγχου ταυτότητας (CCM) λαμβάνει αιτήματα κλειδώματος και ξεκλειδώματος δεδομένων και αποφασίζει ποιες συναλλαγές θα πρέπει να ξεκλειδώνονται, να περιμένουν ή να ακυρώνονται, σύμφωνα με τον επιλεγμένο αλγόριθμο ανάλυσης. Μεταξύ της οικογένειας των προτεινόμενων αλγορίθμων απαισιόδοξων ελέγχων ταυτότητας, ένα από τα πιο ευρέως χρησιμοποιούμενα είναι το αυστηρό κλείδωμα δύο φάσεων (2PL). Βασιζόμενοι στο αυστηρό κλείδωμα δύο φάσεων, μία συναλλαγή οφείλει να έχει κλείδωμα εγγραφής πριν την εγγραφή της σε ένα αντικείμενο πληροφοριών και οφείλει να έχει ένα κλείδωμα ανάγνωσης είτε ένα κλείδωμα εγγραφής πριν να διαβάσει οτιδήποτε από ένα αντικείμενο πληροφοριών. Εάν τα δεδομένα είναι ήδη κλειδωμένα για ανάγνωση, η αυστηρή συναλλαγή μπορεί να εξακολουθεί να έχει άλλο κλείδωμα ανάγνωσης, αλλά δεν μπορεί να αποκτήσει κλείδωμα εγγραφής. Εάν οι πληροφορίες βρίσκονται κλειδωμένες σε μία εγγραφή, δεν γίνεται να δοθούν σε άλλες συναλλαγές με κανένα κλείδωμα. Οι συναλλαγές που δεν έχουν κλειδαριά ούτε για ανάγνωση ούτε για εγγραφή, βρίσκονται σε μία ουρά αναμονής. Με το που μία συναλλαγή ανοίγει κάποιες πληροφορίες, οι κλειδαριά ανάγνωσης και εγγραφής αντίστοιχα που έχει αποδεσμεύεται και δίνεται το δικαίωμα για ξεκλείδωμα στην ακριβώς ακόλουθη συναλλαγή της ουράς αναμονής. Βασικά, μία συναλλαγή γίνεται σε δύο φάσεις που είναι διαδοχικές. Πρώτα υπάρχει μία αναπτυξιακή φάση και μετά μία συρρικνωτική φάση. Στην αναπτυξιακή φάση, η συναλλαγή έχει τη δυνατότητα να ζητάει κλειδαριές, ενώ στη συρρικνωτική φάση η συναλλαγή έχει τη δυνατότητα να αποδεσμεύει κλειδαριές. Στο αυστηρό κλείδωμα δύο φάσεων, η φάση συρρίκνωσης εμφανίζεται όταν η συναλλαγή εκτελείται ή διακόπτεται.

Άλλοι αλγόριθμοι απαισιόδοξων ελέγχων ταυτότητας διαφέρουν από το αυστηρό κλείδωμα δύο φάσεων σχετικά με τους τύπους κλειδαριών, την απόφαση για διένεξη κλειδώματος, τη χρονική στιγμή απόκτησης / απελευθέρωσης κλειδαριών, κ.λπ. (Elmasri, R.A. and Navathe, S.B. [2004]). Το κλείδωμα δύο φάσεων υψηλής προτεραιότητας (2PL-HP) (Abbott, R. and Garcia-Molina, H. [Mar 1988]) επιτρέπει σε συναλλαγές με υψηλότερη προτεραιότητα να κλειδώνουν δεδομένα που είναι ήδη κλειδωμένα από άλλη συναλλαγή με χαμηλότερη προτεραιότητα και να την ακυρώνουν. Διαφορετικοί αλγόριθμοι απαισιόδοξων ελέγχων ταυτότητας μπορούν επίσης να σχεδιαστούν για να αποκλείσουν διαφορετικούς τύπους παρεμβολών, προσαρμόζοντας τα σημεία κλειδώματος και ξεκλειδώματος.

5.3 Μοντελοποίηση ελέγχου επικαιρότητας συναλλαγής

Σε αυτήν την ενότητα περιγράφουμε την προσέγγισή μας για τη μοντελοποίηση ενός συστήματος ταυτόχρονης συναλλαγής σε πραγματικό χρόνο που αποσυντίθεται σε ένα σύνολο συναλλαγών και του Διαχειριστή Ελέγχου Ταυτότητας (CCManager). Όλο το σύστημα γράφεται ως δίκτυο αυτόματων χρονισμένων, με το οποίο οποιαδήποτε συναλλαγή καθώς και ο διαχειριστής ελέγχου ταυτότητας αντίστοιχα μοντελοποιείται ως χρονικό αυτόματο. Τυπικά, ένα σύστημα ταυτόχρονης συναλλαγής σε πραγματικό χρόνο ορίζεται ως εξής:

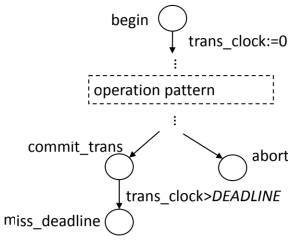
Ορισμός 4 (Σύστημα ταυτόχρονης συναλλαγής σε πραγματικό χρόνο) Ένα σύστημα ταυτόχρονης συναλλαγής N_S σε πραγματικό χρόνο ορίζεται από την ακόλουθη παράλληλη σύνθεση:

$$N_S := A_0 \parallel A_1 \parallel ... \parallel A_{n-1} \parallel A_{CCManager}$$

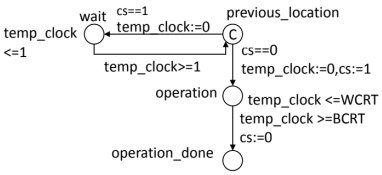
όπου A_{n-1} είναι το χρονικό αυτόματο της συναλλαγής T_{n-1} και $A_{CCManager}$ είναι το χρονικό αυτόματο του διαχειριστή ελέγχου ταυτότητας.

Σε κάθε συναλλαγή T_i εκχωρείται μία σχετική προθεσμία. Η βασική απαίτηση σχετικά με τις ιδιότητες σε πραγματικό χρόνο που πρέπει να πληροί το σύστημα μοντελοποίησης είναι η τήρηση των προθεσμιών όλων των συναλλαγών.

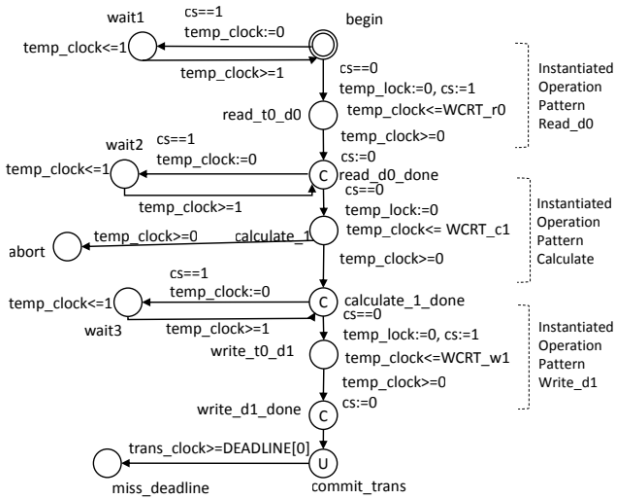
Η προσέγγιση μοντελοποίησης περιλαμβάνει ένα σύνολο χρονομετρημένων σκελετών αυτόματων για μοντελοποίηση των βασικών δομών A_i και $A_{CCManager}$, και ένα σύνολο παραμετρικών μοτίβων για τη μοντελοποίηση των λειτουργιών εντός των συναλλαγών. Οι σκελετοί υποτίθεται ότι πρέπει να προσαρμοστούν και να εμπλουτιστούν σε σχέση με το συγκεκριμένο σχεδιασμό του συστήματος, όπως ο επιλεγμένος αλγόριθμος απαισιόδοξου ελέγχου ταυτότητας. Τα μοτίβα, αντίθετα, μπορούν να τεκμηριωθούν και να επαναχρησιμοποιηθούν με μικρή αλλαγή, για να εμπλουτίσουν τον σκελετό ως βασικές μονάδες μοντελοποίησης. Οι προτεινόμενοι σκελετοί και μοτίβα παρουσιάζονται στις ακόλουθες ενότητες.



Σχήμα 5.1: Βασικό αυτόματο για μία συναλλαγή



Σχήμα 5.2: Πρότυπο λειτουργίας συναλλαγής



Σχήμα 5.3: Χρονικό αυτόματο για T_0 χωρίς CC

5.4 Βασικός σκελετός συναλλαγής

Τα πρωτόγονα διαχείρισης συναλλαγών (Έναρξη, Εκτέλεση και Αφαίρεση) και οι λειτουργίες (Ανάγνωση, Εγγραφή και Υπολογισμός), αποτελούν τη βάση του σκελετού αυτόματων για βασικές συναλλαγές. Όπως φαίνεται στο Σχ. 5.1, στο σκελετό συναλλαγών μας, οι τοποθεσίες Έναρξη, ***commit_trans*** (η λέξη commit είναι δεσμευμένη λέξη στο εργαλείο UPPAAL) και η Ματαίωση αντιπροσωπεύει τα αντίστοιχα πρωτόγονα. Μία συναλλαγή μπορεί να περιέχει πολλαπλές λειτουργίες, καθεμιά από τις οποίες μοντελοποιείται ως ένα παράδειγμα του "τρόπου λειτουργίας" στο αυτόματο. Εφόσον σκοπεύουμε να επαληθεύσουμε την επικαιρότητα της συναλλαγής, μία μεταβλητή ρολογιού ***trans_clock*** ορίζεται για το αυτόματο και επαναφέρεται στο μηδέν μόλις ξεκινήσει η συναλλαγή. Όταν πραγματοποιείται η συναλλαγή, η τιμή του ***trans_clock*** συγκρίνεται με το *DEADLINE* της συναλλαγής, η οποία είναι μία τιμή που καθορίζεται από τον σχεδιαστή. Εάν το ***trans_clock*** είναι μεγαλύτερο από το *DEADLINE*, θα πραγματοποιηθεί μετάβαση στην τοποθεσία ***miss_deadline***.

Το ατομικό μοτίβο λειτουργίας ανάγνωσης/εγγραφής ορίζεται στο Σχ. 5.2. Συνδεδεμένο σε προηγούμενη τοποθεσία, αυτό το μοτίβο περιέχει τοποθεσίες όπως ***operation*** και ***operation_done***, στις οποίες η λέξη "***operation***" πρέπει να αντικατασταθεί από την πραγματική λειτουργία, για παράδειγμα "***read_t1_d1***" (*T1* ανάγνωση *D1*). Η παράμετρος ***cs*** ορίζεται ως κρίσιμη ενότητα(critical section) που ορίζει τους πόρους που θα διαθέσει η CPU εξασφαλίζοντας ατομική συμπεριφορά. Η τοποθεσία "αναμονή" διαμορφώνει τη συμπεριφορά της αναμονής για τον επεξεργαστή. Όταν το ***cs*** ισούται με 0, το αυτόματο λαμβάνει τη CPU, ορίζει το ***cs*** σε 1 και εκτελεί τη λειτουργία. Πριν τη μετάβαση στη κατάσταση ***location_done***, όπου δηλώνει το τέλος της διαδικασίας, το ***cs*** μηδενίζεται ξανά. Μία μεταβλητή ρολογιού

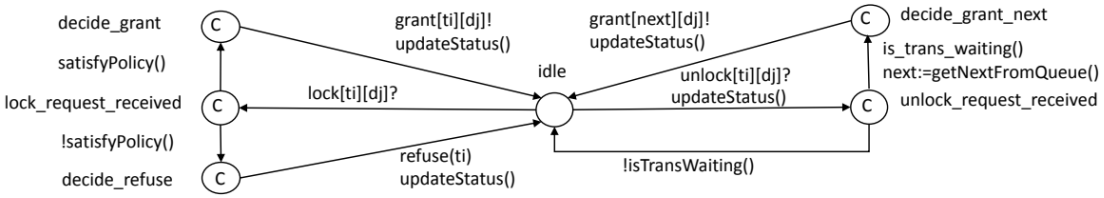
temp_clock ορίζεται για να εντοπίσει τον χρόνο που αφιερώνεται στη λειτουργία. Το *WCRT* και το *BCRT* είναι παράμετροι που καθορίζονται από τον σχεδιαστή, που αντιπροσωπεύουν τον χειρότερο και τον καλύτερο χρόνο απόκρισης, αντίστοιχα. Το αμετάβλητο *temp_clock* <= *WCRT* στη λειτουργία τοποθεσίας περιορίζει ότι η εκτέλεση της λειτουργίας διαρκεί στις περισσότερες μονάδες χρόνου *WCRT*. Ο περιορισμός *temp_clock* >= *BCRT* περιορίζει ότι η εκτέλεση διαρκεί τουλάχιστον μονάδες χρόνου *BCRT*. Για μη ατομικές λειτουργίες (υπολογισμοί), το μοτίβο είναι παρόμοιο με αυτό για ατομικές λειτουργίες, αλλά χωρίς αλλαγή της τιμής του *cs*.

Για ατομικές λειτουργίες ανάγνωσης και εγγραφής, οι χρόνοι απόκρισης είναι ίσοι με τους χρόνους εκτέλεσης. Για μη ατομικές λειτουργίες, οι χρόνοι απόκρισης θα μπορούσαν να προκύψουν από τον σχεδιαστή από τη συγκεκριμένη πολιτική προγραμματισμού και τους περιορισμούς χρονισμού, χρησιμοποιώντας για παράδειγμα τεχνικές ανάλυσης προγραμματισμού.

Η μοντελοποίηση μίας συναλλαγής είναι η αρχικοποίηση των προτύπων λειτουργίας και η σύνθεση των αρχικοποιημένων προτύπων με τον σκελετό. Για παράδειγμα, η αυτοματοποίηση της συναλλαγής T_0 φαίνεται στο Σχ. 4. Οι βασικές λειτουργίες εμπεριέχουν τις θέσεις **Read_d0**, **Calculate** καθώς και **Write_d1**. Αυτές οι βασικές λειτουργίες, όπως θα δείξουμε στη συνέχεια, μπορούν να χρησιμοποιηθούν ξανά και ξανά για να οριστεί μοντελοποίηση διαφορετικών συστημάτων CC.

5.5 Σκελετοί και σχέδια ελέγχου ταυτότητας

Ο απαισιόδοξος ταυτόχρονος έλεγχος χρησιμοποιεί μηχανισμούς κλειδώματος που απαιτούν αλληλεπίδραση μεταξύ του διαχειριστή ελέγχου ταυτότητας και των συναλλαγών. Παρουσιάζουμε σκελετούς για το διαχειριστή ελέγχου ταυτότητας, καθώς και μοτίβα για το σκελετό συναλλαγών που διαμορφώνουν τέτοιες αλληλεπιδράσεις.



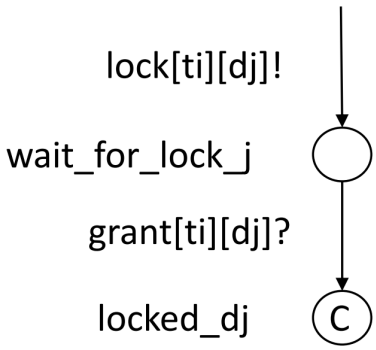
Σχήμα 5.4: Σκελετός Αυτόματου για έναν διαχειριστή απαισιόδοξου ελέγχου ταυτότητας

Ο σκελετός του αυτόματου για έναν διαχειριστή απαισιόδοξου ελέγχου ταυτότητας φαίνεται στο Σχ. 5.4. Με το που λάβει αίτημα κλειδώματος το αυτόματο με χρήση της θέσης *lock[ti][dj]*, εκτελεί τη μετάβαση από τη κατάσταση *idle* στη κατάσταση *lock_request_received*. Μία συνάρτηση που καθορίζεται από το χρήστη και εφαρμόζει τον αλγόριθμο ανάλυσης, *satisfy_policy()* ορίζεται ως προστατευτικό στις μεταβάσεις από το *lock_request_received*. Εάν το *satisfy_policy()* επιστρέψει

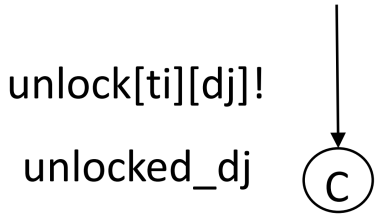
αληθινό, το αυτόματο μετακινείται στην ***decide_grant*** και στη συνέχεια, στέλνει αμέσως το σήμα ***grant[ti][dj]!*** στη συναλλαγή T_i. Κατά τη διάρκεια της μετάβασης, το αυτόματο ενδέχεται να χρειαστεί να ενημερώσει την κατάσταση των συναλλαγών και των κλειδαριών, χρησιμοποιώντας μια λειτουργία που καθορίζεται από το χρήστη ***update_status()***.

Όταν ο διαχειριστής απαισιόδοξου ελέγχου ταυτότητας λαμβάνει ένα αίτημα ξεκλειδώματος μέσω του ***unlock[ti][dj]***, ενημερώνει την κατάσταση των δεδομένων και τη συναλλαγή και μετακινείται στο ***unlock_request_received***. Οι φύλακες που επιτυγχάνουν μεταβάσεις από την θέση αυτή, κάνουν τον έλεγχο για να διαπιστώσουν εάν κάποια άλλη συναλλαγή περιμένει να κλειδώσει τις πληροφορίες, σύμφωνα με τη συνάρτηση που τίθεται από το χρήστη ***is_trans_waiting()***. Εάν το αποτέλεσμα της συνάρτησης αυτής υπολογιστεί θετικό, το αυτόματο αποστέλλει ένα μήνυμα μέσω της κατάστασης ***grant[next][dj]!*** στην επόμενη συναλλαγή και κάνει τις ανάλογες ενημερώσεις στη κατάσταση.

Ο σκελετός συναλλαγής πρέπει να επεκταθεί για να μοντελοποιηθεί η αλληλεπίδραση με τον διαχειριστή απαισιόδοξου ελέγχου ταυτότητας. Ένα σχέδιο κλειδώματος και ένα μοτίβο ξεκλειδώματος εισάγονται στα Σχ. 5.5 και Σχ. 5.6 αντίστοιχα. Εφόσον η συναλλαγή αποστέλλει ένα αίτημα μέσω της θέσης ***lock[ti][dj]***, περιμένει στην θέση ***wait_for_lock_j*** μέχρι να λάβει την επιβεβαίωση ***grant[ti][dj]***. Μπορούν να εισαχθούν στα βασικά αυτόματα δεδομένα συναλλαγών σε συγκεκριμένες θέσεις ανάλογα με τον επιλεγμένο αλγόριθμο απαισιόδοξου ελέγχου ταυτότητας. Για παράδειγμα, η χρήση αυστηρών 2PL κλειδαριών εγγραφής απελευθερώνονται μετά την ολοκλήρωση της συναλλαγής ή τη διακοπή. Για να ολοκληρωθεί η μοντελοποίηση με χρήση του αλγορίθμου απαισιόδοξου ελέγχου ταυτότητας, ο σχεδιασμός των αυτομάτων για διάφορα κλειδώματα εγγραφής πρέπει να εισαχθεί μετά τη θέση δέσμευσης ή ματαίωσης στο αυτόματο συναλλαγής.



Σχήμα 5.5: Μοτίβο κλειδώματος για τη συναλλαγή αυτόματου



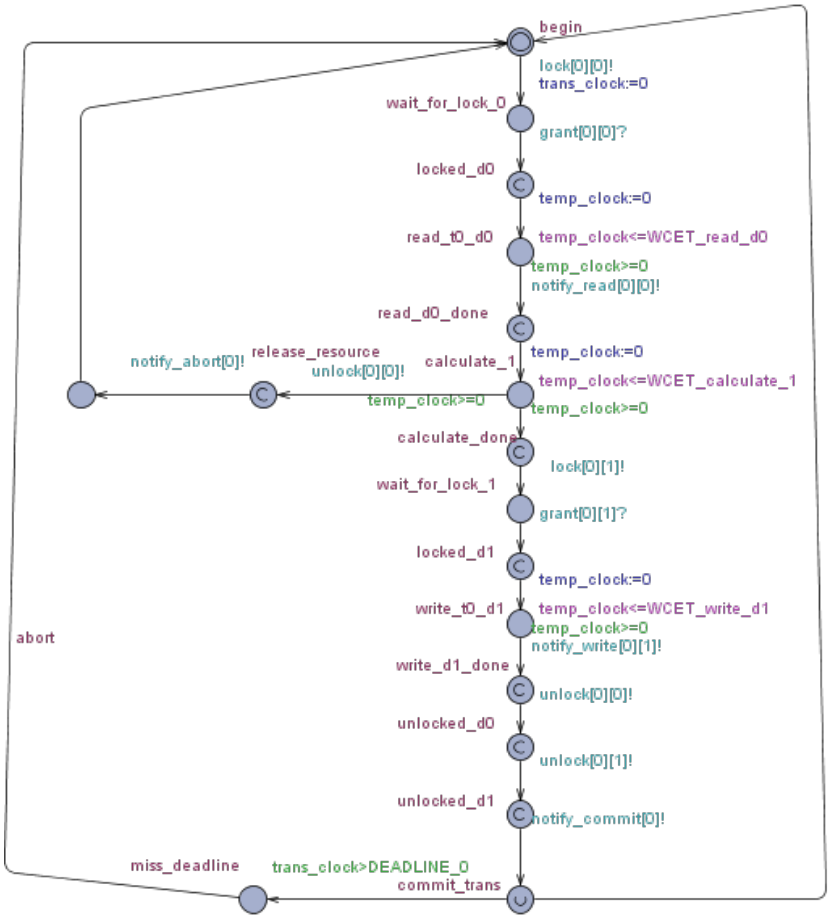
Σχήμα 5.6: Μοτίβο ξεκλειδώματος για την συναλλαγή αυτόματου

5.6 Έλεγχος Μοντέλου επικαιρότητας υπό αυστηρό κλείδωμα δύο φάσεων

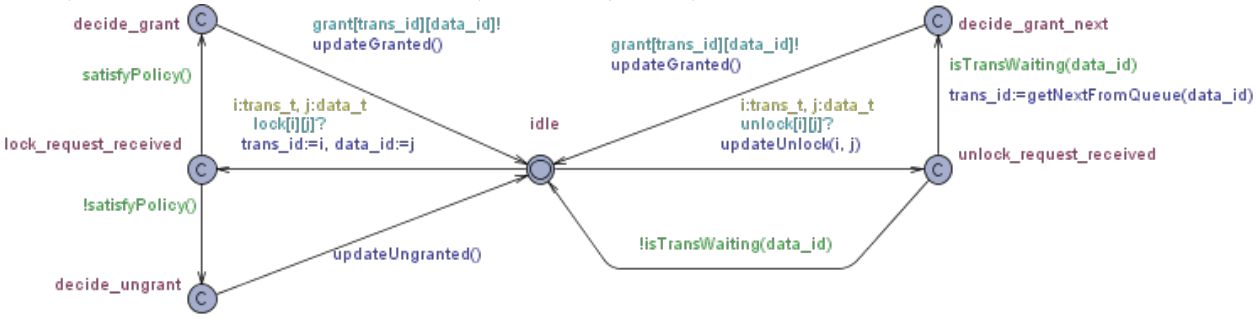
Απεικονίζουμε την προσέγγιση μοντελοποίησης με μοντελοποίηση ενός συστήματος ταυτόχρονης συναλλαγής που εφαρμόζει τον αυστηρό αλγόριθμο κλειδώματος δύο φάσεων που παρουσιάζεται παραπάνω. Ας εξετάσουμε τις δύο συναλλαγές στα Προγράμματα 1.1 και 1.2. Το T_0 περιέχει λειτουργίες r_0^0 και w_0^1, ενώ το T_1 περιέχει λειτουργίες w_0^1 και w_1^1. Ας υποθέσουμε ότι ο χρόνος που ξοδεύεται σε κάθε μεμονωμένη λειτουργία ανάγνωσης και εγγραφής στη χειρότερη περίπτωση είναι 1 μονάδα χρόνου και ο χρόνος για τον υπολογισμό σε η χειρότερη περίπτωση είναι 2 μονάδες χρόνου. Οι καλύτεροι χρόνοι εκτέλεσης υποθέσεων θεωρείται ότι είναι 0 για απλότητα. Οι προθεσμίες των T_0 και T_1 είναι 8 και 5 χρονικές μονάδες αντίστοιχα.

Το μοντέλο του T_0 φαίνεται στο Σχ. 5.7. Ο βασικός σκελετός συναλλαγών χωρίς CC συμβάλλει στις θέσεις ***begin***, ***commit_trans***, ***abort*** και ***miss_deadline***. Ο βασικός σχεδιασμός επεκτείνεται με λειτουργίες που συμπεριλαμβάνονται στη συναλλαγή, αλλά και με τις αλληλεπιδράσεις κλειδώματος και ξεκλειδώματος με το διαχειριστή ελέγχου ταυτότητας. Όπως φαίνεται στο Σχ. 5.7, η συναλλαγή προσπαθεί πρώτα να αποκτήσει ένα κλείδωμα ανάγνωσης του D_0, το οποίο διαμορφώνεται από ένα τυποποιημένο μοτίβο κλειδώματος **Readlock_d0**. Στη συνέχεια ακολουθεί η λειτουργία r_0^0, μοντελοποιημένη με ένα μοτίβο λειτουργίας **Read_d0**. Μετά το r_0^0, η συναλλαγή εκτελεί υπολογισμό, αποκτά ένα γράψιμο κλειδώματος D_0 και εκτελεί τη λειτουργία w_0^1. Πριν από τη διαπραγμάτευση, η συναλλαγή απελευθερώνει κλειδώματα χρησιμοποιώντας μοτίβα ξεκλειδώματος. Οι αρχικές καταστάσεις λειτουργίας **Read_d0**, **Calculate** καθώς και **Write_d1** χρησιμοποιούνται και στο μοντέλο του T_0 χωρίς CC όπως απεικονίζεται στο Σχ. 5.3. Το χρονικό αυτόματο του T_1 διαμορφώνεται με παρόμοιο τρόπο.

Το Σχήμα 5.7 απεικονίζει το χρονικό αυτόματο για το διαχειριστή ελέγχου ταυτότητας χρησιμοποιώντας τον επιλεγμένο αλγόριθμο κλειδώματος δύο φάσεων. Το βασικό μοντέλο απαισιόδοξου ελέγχου ταυτότητας αναπτύσσεται για να ενσωματώνει αναγνωστικά κλειδώματα (readlock) και κλειδώματα εγγραφής(writelock). Μοντελοποιώντας το σύστημα, στον πραγματικό αλγόριθμο φαίνονται οι συναρτήσεις που θέτει ο χρήστης. Η θέση ***satisfyPolicy()*** επιλέγει για το ποια συναλλαγή είναι ικανή να πάρει κάποιο κλείδωμα, με γνώμονα την τωρινή φάση των κλειδωμένων πληροφοριών. Οι λειτουργίες ***updateGranted()***, ***updateUngranted()*** και ***updateUnlock()*** ενημερώνουν την κατάσταση των συναλλαγών και των δεδομένων αφού ληφθούν μέτρα κλειδώματος, άρνησης ή ξεκλειδώματος, αντίστοιχα. Η συνάρτηση ***isTranswaiting()*** ελέγχει εάν υπάρχει κάποια συναλλαγή περιμένει στην ουρά για κλείδωμα δεδομένων που είχαν προηγουμένως κλειδωθεί. Η συνάρτηση ***getNextFromQueue()*** ανακτά την επόμενη συναλλαγή από την ουρά αναμονής. Η εφαρμογή αυτών των λειτουργιών παρατίθεται στο Πρόγραμμα 1.3.



Σχήμα 5.7: Το χρονικό αυτόματο για τη συναλλαγή T_0 χρησιμοποιώντας 2PL



Σχήμα 5.8: Το χρονικό αυτόματο για τον διαχειριστή ελέγχου ταυτότητας στο κλείδωμα δύο φάσεων

```
int trans_id, data_id;
bool locked[data_t];
int lockdata[trans_t][data_t];
int queue[data_t][trans_t];
int len[data_t];

void enqueue(int t, int d)
{
    queue[d][len[d]]=t;
    len[d]++;
}

void dequeue(int d)
{
    if(len[d]>0) {
        int i;
        for(i=0;i<len[d];i++)
        {
            queue[d][i]=queue[d][i+1];
        }
        queue[d][len[d]]=0;
        len[d]--;
    }
}

bool satisfyPolicy() {
    if(!locked[data_id])
        return true;
    return false;
}

void updateGranted()    {
    locked[data_id]=true;
    lockdata[trans_id][data_id]=1;
 }

 void updateUngranted()    {
     enqueue(trans_id, data_id);
 }

 void updateUnlock(int t, int d) {
     locked[d]=false;
     lockdata[t][d]=0;
     trans_id=t;
     data_id=d;
     //dequeue(d);
 }
```

```
bool isTransWaiting(int d) {
    if(len[d]>0)
        return true;
    return false;
}

int getNextFromQueue(int d) {
    int t=queue[d][0];
    dequeue(d);
    return t;
}
```

Πρόγραμμα 1.Λειτουργίες που καθορίζονται από τον χρήστη στο μοντέλο διαχειριστή ελέγχου ταυτότητας για κλείδωμα δύο φάσεων

5.7 Επαλήθευση επικαιρότητας

Έχοντας ένα δίκτυο χρονισμένων αυτομάτων για το σύστημα που μοντελοποιήθηκε, είμαστε σε θέση να ελέγξουμε την επικαιρότητα των συναλλαγών μέσω του ελεγκτή μοντέλων UPPAAL. Η ιδιότητα της επικαιρότητας, που απαιτεί κάθε συναλλαγή να πληροί την προθεσμία της, μπορεί να οριστεί ως μια ιδιότητα ασφαλείας όπου οι τοποθεσίες miss_deadline δεν είναι προσβάσιμες.

Οι προδιαγραφές παρατίθενται στο Σχήμα 5.9. Ορίζοντας $S1$ και $S2$ τις καθορίζουν την επικαιρότητα των T_0 και T_1, αντίστοιχα. Η επαλήθευση από τον ελεγκτή μοντέλου UPPAAL αποδεικνύει ότι το $S1$ είναι ικανοποιημένο. Ωστόσο, η επαλήθευση του $S2$ αποτυγχάνει, υποδεικνύοντας ότι το T_1 ενδέχεται να χάνει την προθεσμία του. Ο ελεγκτής μοντέλου παρέχει ένα ίχνος που οδηγεί στη λήξη της προθεσμίας. Από το ίχνος συνειδητοποιούμε ότι ο T_1 προσπαθούσε να κλειδώσει το D_1 πριν από το w_1^1, ενώ το D_1 ήταν ήδη κλειδωμένο από το T_0 πριν από το r_1^1 μέχρι το T_0 να δεσμευτεί. Ένας τόσο μεγάλος χρόνος μπλοκαρίσματος που εισήχθη από τον έλεγχο ταυτότητας προκάλεσε την παραβίαση της επικαιρότητας του T_1.

Σχήμα 5.9: Προσδιορισμός ιδιοτήτων και αποτελέσματα επαλήθευσης

```
A[] not Transaction1.miss_deadline
A[] not Transaction2.miss_deadline
A[] not (Transaction1.miss_deadline or Transaction2.miss_deadline)
```

5.8 Μοντέλο ελέγχου απομόνωσης συναλλαγής

Στην προηγούμενη ενότητα, η συναλλαγή T_1 χάνει την προθεσμία της λόγω του χρόνου αποκλεισμού που εισήγαγε έλεγχος ταυτόχρονης λειτουργίας 2PL, ο οποίος στοχεύει στην επίτευξη απομόνωσης. Διαφορετικά, θα μπορούσαμε να επιλέξουμε έναν λιγότερο οριοθετημένο μηχανισμό CC με στόχο χαμηλότερο βαθμό απομόνωσης, με τον οποίο υπάρχει βελτίωση στην επικαιρότητα. Προκειμένου να επιτευχθεί ελεγχόμενη χαλάρωση της απομόνωσης, είναι σημαντικό να κατανοήσουμε τις παραλλαγές απομόνωσης και να ανακαλύψουμε τρόπους για την επαλήθευση της επιλεγμένης παραλλαγής απομόνωσης. Σε αυτήν την ενότητα, θυμόμαστε πρώτα μία επισκόπηση των υπαρχόντων επιπέδων απομόνωσης. Με βάση τους ορισμούς των επιπέδων απομόνωσης, παρουσιάζουμε ένα μοντέλο παρατηρητή και τον σκελετό του αυτόματου που θα μπορούσε να συντεθεί στο δίκτυο αυτόματων συστημάτων ταυτόχρονης συναλλαγής που προτάθηκε στην προηγούμενη ενότητα.

5.8.1 Μία επισκόπηση της απομόνωσης

Η απομόνωση αναφέρεται στην ιδιότητα ότι η εκτέλεση μιας συναλλαγής δεν παρεμποδίζεται από άλλες συναλλαγές που εκτελούνται ταυτόχρονα (Adya, A. and Liskov, B. and O'Neil, P. [2000]). Δεδομένου ότι η πλήρης απομόνωση οδηγεί σε υποβάθμιση της απόδοσης και δεν είναι πάντα απαραίτητη, η χαλάρωση της απομόνωσης έχει εισαχθεί τόσο από τη βιομηχανία όσο και από τον ακαδημαϊκό χώρο. Μεγάλο μέρος των εμπορικών DBMS ενσωματώνουν τα επίπεδα απομόνωσης που ορίστηκαν από τον κανονισμό ANSI / ISO SQL92, τα οποία χωρίζονται σε κατηγορίες που είναι οι εξής: η σειριοποιησιμότητα συγκρούσεων (SERIALIZABILITY) (η μεγαλύτερη και πιο αυστηρή απομόνωση), επαναλήψιμες αναγνώσεις (REPEATABLE READS), δεσμευμένες αναγνώσεις (READ COMMITTED) και τέλος οι αδέσμευτες αναγνώσεις (READ UNCOMMITTED) (η πιο χαλαρή μέθοδος απομόνωση).

Ο Adya A. και άλλοι (Adya, A. and Liskov, B. and O'Neil, P. [2000]) έχουν γενικεύσει αυτά τα επίπεδα απομόνωσης και παρείχαν σαφείς ορισμούς χρησιμοποιώντας την έννοια των φαινομένων. Ένα φαινόμενο είναι ένας είδος συμπεριφοράς που μπορεί να οδηγήσει σε ασυνεπή δεδομένα και μπορεί να χαρακτηριστεί από τις άμεσες συγκρούσεις δύο συναλλαγών που εκτελούνται. Τα επίπεδα απομόνωσης ορίζονται με βάση τα φαινόμενα που πρέπει να αποφεύγονται σε κάθε επίπεδο.

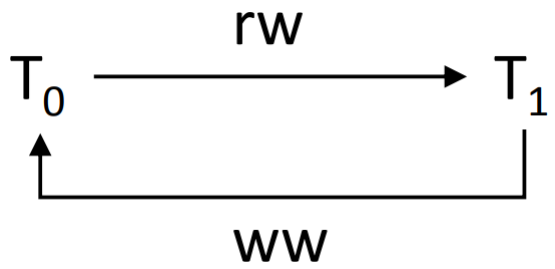
Οι άμεσες συγκρούσεις δύο δεσμευμένων συναλλαγών, χωρίς να λαμβάνονται υπόψη οι βασισμένες σε κατηγορίες πράξεις, ορίζονται ως εξής «Adya, A. and Liskov, B. and O'Neil, P. [2000]»:

Άμεση εξάρτηση ανάγνωσης : Η συναλλαγή T_j έχει άμεση εξάρτηση ανάγνωσης από την T_i, εάν η T_j διαβάζει x δεδομένα μετά από όταν η T_i γράφει x δεδομένα(πριν άλλες συναλλαγές γράψουν x).
Άμεση εξάρτηση εγγραφής : Η συναλλαγή T_j έχει άμεση εξάρτηση εγγραφής από την T_i, εάν η T_j γράφει x δεδομένα μετά από όταν η T_i γράφει x δεδομένα(πριν άλλες συναλλαγές γράψουν x).
Άμεση ανεξαρτησία : Η συναλλαγή T_j έχει άμεση ανεξαρτησία από την T_i, εάν η T_j γράφει x δεδομένα μετά από όταν η T_i διαβάζει x δεδομένα(πριν άλλες συναλλαγές

γράψουν x).

Με βάση τις άμεσες διενέξεις μπορεί να κατασκευαστεί ένα Γράφημα Άμεσης Σειριοποίησης (DSG). Κάθε συναλλαγή αντιπροσωπεύεται από έναν κόμβο σε ένα γράφημα άμεσης σειριοποίησης. Ένα άκρο $T_i \xrightarrow{wr} T_j$ αντιπροσωπεύει ότι το T_j άμεσα διαβάζεται εξαρτάται από το T_i. Η άμεση εξάρτηση από την εγγραφή και η άμεση ανάρτηση χαρακτηρίζονται ως $T_i \xrightarrow{ww} T_j$ και $Ti \xrightarrow{rw} Tj$ αντίστοιχα. Για παράδειγμα, μία πιθανή εκτέλεση συναλλαγών που περιλαμβάνει T_0 και T_1 μπορεί να δηλωθεί ως εξής: $< r_0^0, w_1^0, w_1^1, w_0^1 >$. Το DSG αυτής της εκτέλεσης συναλλαγής φαίνεται στο Σχ. 10.

Χρησιμοποιώντας αυτήν την αναπαράσταση, τα φαινόμενα ορίζονται στον Πίνακα 2 (Adya, A. and Liskov, B. and O'Neil, P. [2000]). Σύμφωνα με τον ορισμό, το DSG στο Σχ. 10 εμφανίζει το φαινόμενο $G2$ επειδή περιέχει μια ανάρτηση $T_0 \xrightarrow{rw} T_1$ μέσα σε έναν κύκλο.



Σχήμα 5.10: Το DSG μίας εκτέλεσης συναλλαγής $< r_0^0, w_1^0, w_1^1, w_0^1 >$

Πίνακας 2. Φαινόμενα που ορίζονται από τον Adya και άλλους

Φαινόμενο	Ορισμός
$G0$: Κύκλοι εγγραφών	Το DSG της εκτέλεσης συναλλαγής περιέχει έναν κατευθυνόμενο κύκλο που αποτελείται εξ ολοκλήρου από άκρα εξαρτημένα από εγγραφή.
$G1a$: Ματαιωμένες αναγνώσεις	Η εκτέλεση περιλαμβάνει μία δεσμευμένη συναλλαγή T_1 και μία ακυρωμένη συναλλαγή T_2 και ο T_1 διαβάζει τα δεδομένα που τροποποιήθηκαν από τον T_2.
$G1b$: Ενδιάμεσες αναγνώσεις	Η εκτέλεση περιλαμβάνει μία δεσμευμένη συναλλαγή T_1 που διαβάζει μία τροποποίηση του T_2 και αυτή η τροποποίηση δεν είναι η τελική τροποποίηση του T_2.
$G1c$: Κυκλική ροή πληροφοριών	Το DSG της εκτέλεσης περιέχει έναν κατευθυνόμενο κύκλο που περιλαμβάνει οποιεσδήποτε εξαρτημένες ακμές (αλλά όχι άκρες ανεξαρτησίας στον κύκλο)
$G2$: Κύκλοι ανεξαρτησίας	Το DSG της εκτέλεσης συναλλαγής περιέχει έναν κατευθυνόμενο κύκλο που αποτελείται από ένα ή περισσότερα άκρα ανεξαρτησίας.

Το επίπεδο απομόνωσης ορίζεται σαν την ιδιότητα να αποφύγουμε συγκεκριμένα υποσύνολα καθορισμένων φαινομένων. Για παράδειγμα, το επίπεδο SERIALIZABLE αποκλείει όλα τα προαναφερθέντα φαινόμενα, ενώ το επίπεδο READ COMMITTED αποκλείει μόνο τα $G0$ και $G1$. Επομένως, η εκτέλεση $< r_0^0, w_1^0, w_1^1, w_0^1 >$ που αντιπροσωπεύεται από το DSG στο Σχ. 5.10 παραβιάζει την απομόνωση SERIALIZABLE. Για να επιτύχουμε επαλήθευση συγκεκριμένου επιπέδου απομόνωσης, οφείλουμε να κάνουμε επαλήθευση απόκλισης των φαινομένων από τα άλλα επίπεδα με την ύπαρξη του υποκείμενου μηχανισμού ελέγχου ταυτότητας.

5.9 Ένα μοντέλο παρατηρητή για απομόνωση

Προκειμένου να εξακριβωθεί η ικανοποίηση ενός επιπέδου απομόνωσης, πρέπει να επαληθευτεί η απουσία των αντίστοιχων φαινομένων. Επομένως, θα εισάγουμε ένα αυτόματο παρατηρητή απομόνωσης μέσα στο μοντέλο μας για την καταγραφή των φαινομένων.

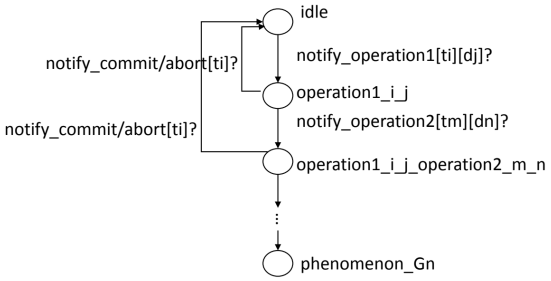
Ορισμός 5 (Επίπεδο Απομόνωσης) Υποθέτοντας ότι ένα σύστημα ταυτόχρονης συναλλαγής σε πραγματικό χρόνο N_S σκοπεύει να επιτύχει ένα επιλεγμένο επίπεδο απομόνωσης που αποκλείει k φαινόμενα, τότε μπορούμε να ορίσουμε το N_S ως εξής:

$$N_S := A_0 \parallel ... \parallel A_n \parallel A_{CCManager} \parallel O_0 \parallel ... \parallel O_{k-1}$$

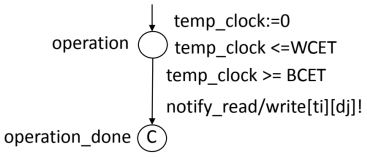
όπου A_n είναι το χρονικό αυτόματο της συναλλαγής T_n, $A_{CCManager}$ είναι το χρονικό αυτόματο του διαχειριστή ελέγχου ταυτότητας και O_{k-1} είναι το χρονικό αυτόματο που παρατηρεί το φαινόμενο G_k που δεν επιτρέπεται από το επιλεγμένο επίπεδο απομόνωσης.

Ο σκελετός του αυτόματου για έναν παρατηρητή απομόνωσης περιγράφεται στην Εικ. 5.11. Η αφετηρία του αυτόματου βρίσκεται στη κατάσταση αδράνειας(*idle*) και καταλήγει στη κατάσταση του φαινομένου G_n(*phenomenon_Gn*) όταν υλοποιηθούν όλες οι δυνατές μετατοπίσεις. Κάθε θέση μεταξύ του *idle* και του *phenomenon_Gn* είναι μία υπακολουθία της ακολουθίας λειτουργίας που ορίζει το *phenomenon_Gn*. Χωρίς βλάβη της γενικότητας, ορίζουμε το G_n ως την ακολουθία $< op_i^j, op_m^n, ... >$. Στο Σχήμα 11, όταν η συναλλαγή T_i ολοκληρώνει με επιτυχία τη λειτουργία op_i^j (ανάγνωση ή εγγραφή D_j), το αυτόματο του παρατηρητή ειδοποιείται μέσω του καναλιού *notify_operation1[ti][dj]* και πηγαίνει τη μετάβαση στη τοποθεσία *operation1_i_j*. Στη συνέχεια, όταν το T_m ολοκληρώσει με επιτυχία το op_m^n, το αυτόματο πραγματοποιεί τη μετάβαση από τη τοποθεσία *operation1_i_j* στη τοποθεσία *operation1_i_j_operation2_m_n*. Δεδομένου ότι το παρατηρούμενο φαινόμενο ξεκινά με μία λειτουργία της συναλλαγής T_i, το τέλος της T_i σημαίνει επίσης το τέλος της παρατήρησης. Συνεπώς, σε περίπτωση εκτέλεσης είτε διακοπής της συναλλαγής T_i, το αυτόματο του παρατηρητή δέχεται μέσω μίας ενημέρωσης της θέσης *notify_commit/abort[ti]* και επανέρχεται στη θέση αδράνειας. Τέτοια συμπεριφορά ειδοποίησης-μετάβασης επαναλαμβάνεται έως ότου

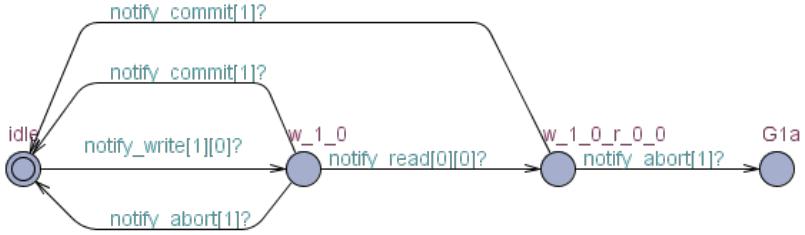
ο παρατηρητής φτάσει στο *phenomenon_Gn*, υποδεικνύοντας την ύπαρξη του G_n και επομένως την παραβίαση του επιθυμητού επιπέδου απομόνωσης.



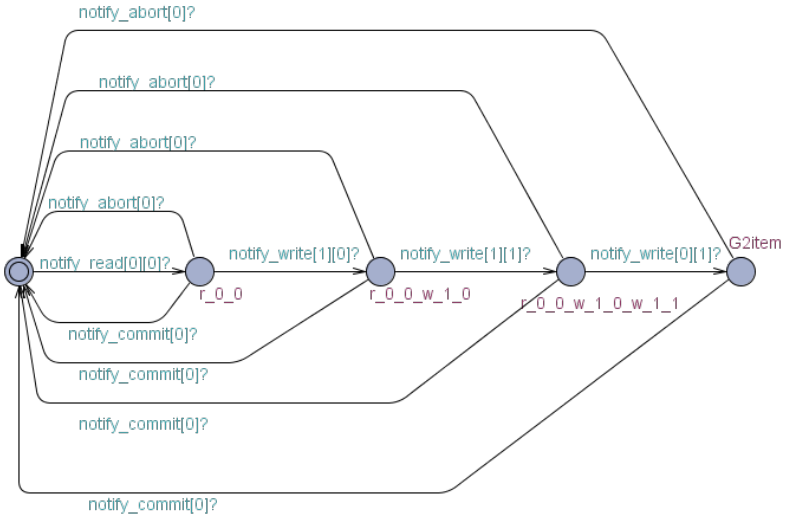
Σχήμα 5.11: Πρότυπο αυτόματου για έναν παρατηρητή απομόνωσης



Σχήμα 5.12: Επεκταμένο πρότυπο λειτουργίας για παρατηρητή απομόνωσης σε έναν βασικό σκελετό συναλλαγών



Σχήμα 5.13: Παρατηρητές απομόνωσης $G1a$

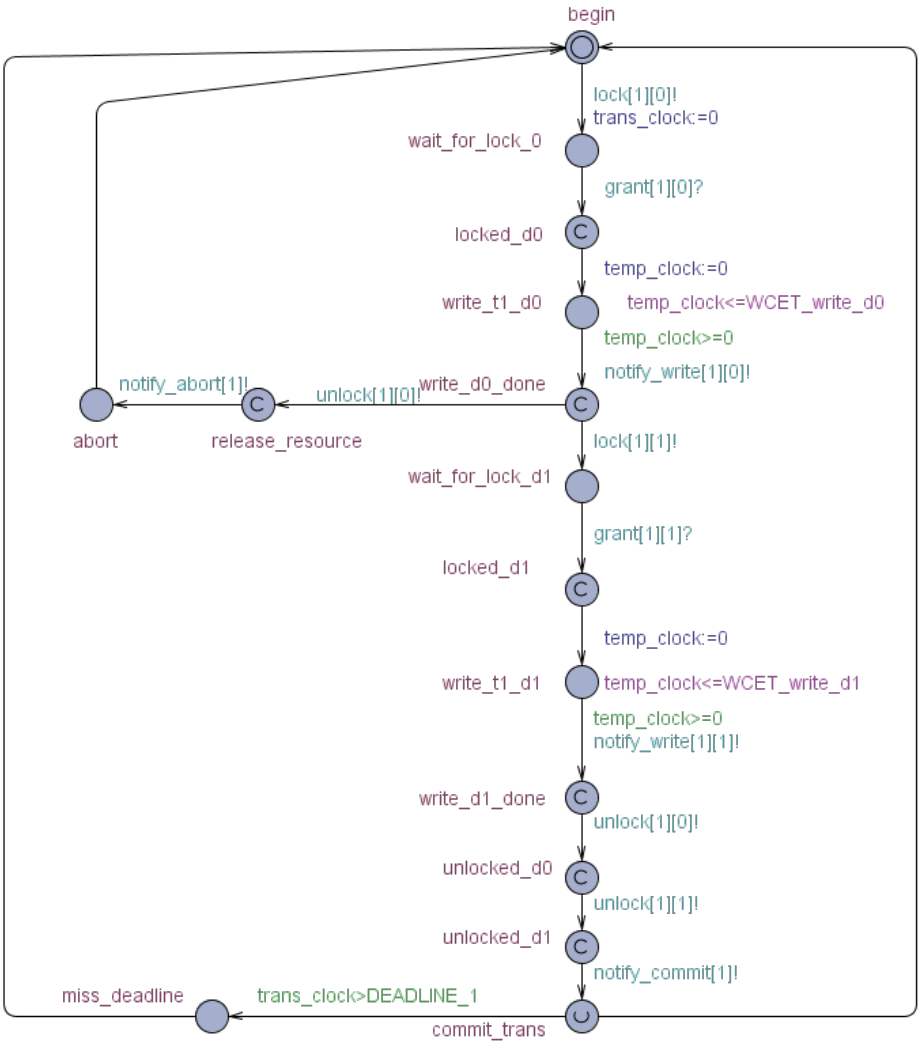


Σχήμα 5.14: Παρατηρητές απομόνωσης για $G2$

Κατά συνέπεια, ο σκελετός συναλλαγών και τα μοτίβα που εισήχθησαν στο προηγούμενο κεφάλαιο πρέπει να επεκταθούν ώστε να ενσωματώσουν τις ειδοποιήσεις. Στο βασικό μοντέλο συναλλαγής που παρουσιάζεται στο Σχήμα 5.1,η θέση ***notify_commit[ti]!*** μαζί με τη θέση ***notify_abort[ti]!*** οφείλουν να ενταχθούν στις μεταβάσεις που καταλήγουν στις θέσεις ***commit_trans*** όπως και ***commit_abort*** αντίστοιχα. Το μοτίβο λειτουργίας που ορίζεται στο σχήμα 5.2 επεκτείνεται με τις τοποθεσίες ***notify_operation[ti][dj]!*** (***notify_read[ti][dj]*** ή ***notify_write[ti][dj]***) και ***operation_done***, όπως φαίνεται στο Σχήμα 5.12.

5.9.1 Συμπεράσματα για την επαλήθευση (απομόνωση) του μοντέλου

Χρησιμοποιώντας το σκελετό του παρατηρητή απομόνωσης και τα μοτίβα που εισάγονται παραπάνω, μπορούμε να επαληθεύσουμε ότι το παράδειγμα του κεφαλαίου 4 επιτυγχάνει πλήρη απομόνωση, δηλαδή σε επίπεδο SERIALIZABLE απομόνωσης. Προκειμένου το τεκμηριωθεί αυτό, οφείλουμε να δείξουμε ότι ούτε ένα από τα φαινόμενα $G0$, $G1$ ($G1a$, $G1b$ και $G1c$) και $G2$ δεν μπορεί να προκύψει. Ως συνέπεια του ίδιου του ορισμού, το $G0$ επιβεβαιώνεται σε περίπτωση που υπάρχει βρόχος αλληλεξάρτησης εγγραφών ανάμεσα στις T_0 και T_1, ο οποίος είναι απίθανο να συμβεί, έχοντας λάβει υπόψη μας τις λειτουργίες που έχουν οι δύο συναλλαγές. Παρομοίως, τα $G1b$ και $G1c$ δεν πρόκειται να συμβούν, ως συνέπεια του ίδιου του ορισμού. Ως εκ τούτου, προκειμένου να επαληθεύσουμε ότι πληρείται το επίπεδο απομόνωσης SERIALIZABLE, πρέπει να αποδείξουμε την απουσία της $G1a$ και της $G2$. Μεταξύ αυτών, $G1a$(Ματαιωμένη Εγγραφή) μπορεί να περιγραφεί ως η ακολουθία $< w_1^0, r_0^0, a_1 >$, στην οποία το a_1 υποδηλώνει τη ματαίωση του T_1. Το $G2$ (Ανεξάρτητοι Κύκλοι) μπορεί να περιγραφεί ως η ακολουθία $< r_0^0, w_1^0, w_1^1, w_0^1 >$. Τα αυτόματα παρατήρησης για τα $G1a$ και $G2$ εμφανίζονται στα Σχήματα 5.13 και 5.14 αντίστοιχα. Το λεπτομερές μοντέλο UPPAAL του T_0 είναι το ίδιο όπως φαίνεται στο Σχήμα 5.7 καθώς και το μοντέλο του διαχειριστή ελέγχου ταυτότητας είναι ίδιο με το παραπάνω που εμφανίζεται στο Σχήμα 5.8. Το T_1 μοντέλο εμφανίζεται παρακάτω στο Σχήμα 5.15. Τα αποτελέσματα της επαλήθευσης στο Σχήμα 5.16 δείχνουν ότι ούτε η τοποθεσία $G1a$ ούτε η τοποθεσία $G2$ είναι προσβάσιμη, πράγμα που σημαίνει ότι το επαληθευμένο σύστημα επιτυγχάνει SERIALIZABLE επίπεδο απομόνωσης.



Σχήμα 5.15: μοντέλο UPPAAL της T_1 χρησιμοποιώντας κλείδωμα δύο φάσεων

```
A[] not IsolationObserverG1a.G1a
A[] not IsolationObserverG2.G2item
A[] not deadlock
A[] not Transaction1.miss_deadline
A[] not Transaction2.miss_deadline
A[] not (Transaction1.miss_deadline or Transaction2.miss_deadline)
```

Σχήμα 5.16: Αποτελέσματα επαλήθευσης με χρήση κλείδωμα δύο φάσεων

5.10 Μελλοντικές επεκτάσεις

Με βάση την παρούσα πτυχιακή εργασία, θα μπορούσαν να υπάρξουν βελτιώσεις στο ήδη υπάρχον μοντέλο (ανάπτυξη περισσότερων καταστάσεων μετάβασης) καθώς και η δημιουργία διαφορετικών μοντέλων κλειδώματος ανάγνωσης και εγγραφής πληροφοριών. Είναι αναγκαία η δημιουργία μοντέλων επαλήθευσης όπως μοντέλα γράφων σειριοποιησιμότητας, μοντέλα χρονοσημάτων, μοντέλα ελέγχου ταυτοχρονισμού πολλαπλών εκδοχών και μοντέλα ελέγχου ταυτοχρονισμού με ευρετήρια. Τέλος, θα μπορούσε να γίνει η δημιουργία πειραματικών μοντέλων στο περιβάλλον UPPAAL όπως αυτή που βρίσκεται στο έγγραφο Nyström, D. and Nolin, M. and Tešanović, A. and Nordström, C. and Hansson, J. [2004] για να αποδειχθεί έμπρακτα η αποτελεσματικότητά τους.

Βιβλιογραφία

Abbott, R. and Garcia-Molina, H. Scheduling real-time transactions. *SIGMOD Rec.*, 17(1):71–81, Mar 1988.

Adya, A. and Liskov, B. and O'Neil, P. Generalized isolation level definitions. *In: Data Engineering*, 16th International Conference:67–78, 2000.

Alur, R. and Courcoubetis, C. and Dill, D. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

Alur, R. and Courcoubetis, C. and Dill, D.L. Model-checking for real-time systems. *In 5th Symposium on Logic in Computer Science (LICS'90)*, pages 414–425, 1990.

Alur, R. and Dill, D.L. Automata for modeling real-time systems. *In Proc. of Int. Colloquium on Algorithms, Languages, and Programming*, 443 of LNCS:322–335, 1990.

Alur, R. and Dill, D.L. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

Bachman, C.W. The Programmer as Navigator. *Communications of the ACM*, 16 (11):653–658, 1973. doi: 10.1145/355611.362534.

Bartoletti, M. and Pompianu, L. An empirical analysis of smart contracts: platforms, applications and design patterns,. 2017. URL `https://arxiv.org/pdf/1703.06322.pdf`.

Behrmann, G. and Bouyer, P. and Larsen, K.G. and Radek Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *International Journal on Software Tools for Technology Transfer*, September 2005.

Beynon-Davies, P. *Database Systems.* Palgrave Macmillan, 3rd edition, 2003. ISBN 978-1-4039-1601-3.

Cai, S. Modeling Real-time Transactions in UPPAAL. *Tech. Rep.*, April 2015. URL `http://www.es.mdh.se/publications/3911`.

CapGemini. Getting from Hype to Reality. 2017. URL `https://www.capgemini.com/consulting-de/wp-content/uploads/sites/32/2017/08/smart_contracts_paper_long_0.pdf`.

Carbonnelle, P. TOPDB Top Database index. *pypl.github.io.*, 2020.

Chapple, M. The Fundamentals of SQL. 22 April 2020. URL `https://www.lifewire.com/sql-fundamentals-1019780`.

Connolly, T.M. and Begg, C.E. *Database Systems – A Practical Approach to Design Implementation and Management.* Palgrave Macmillan, 6th edition, 2014. ISBN 978-1-2920-6118-4.

Datta, A. and Son, S. A study of concurrency control in real-time, active database systems. *Knowledge and Data Engineering, IEEE Transactions*, 14(3):465–484, May 2002.

Elmasri, R.A. and Navathe, S.B. *Fundamentals of Database Systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 4th edition, 2004.

Gerd Behrmann, Alexandre David, and Kim G. Larsen. A Tutorial on Uppaal 4.0. Updated November 28, 2006. URL `https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf`.

International Business Machines (IBM). Structured Query Language (SQL). 27 October 2006.

Kao, B. and Garcia-Molina, H. An overview of real-time database systems. *Real Time Computing*, pages 261–282, Springer Berlin Heidelberg,1994.

Kehrli, J. Blockchain 2.0 - from bitcoin transactions to smart contract applications. November 2016.

Kot, M. Modeling real-time database concurrency control protocol two-phase-locking in UPPAAL. *Computer Science and Information Technology, International Multiconference*, pages 673–678, 2008.

Kot, M. Modeling selected real-time database concurrency control protocols in UPPAAL. *Innovations in Systems and Software Engineering*, 5(2):129–138, 2009.

Larsen, K.G. and Pettersson, P. and Yi, W. Uppaal in a nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, October 1997.

Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, Aquinas Hobor. Making Smart Contracts Smarter. URL `https://eprint.iacr.org/2016/633.pdf`.

Nelson, A.F. and Nelson, W.H.M. *Database Systems.* Prentice Hall, 2001. ISBN 978-0-2017-4130-8.

Nielsen, H.F. and LaLiberte, D. Editing the Web - Detecting the Lost Update Problem Using Unreserved Checkout. *W3C Note*, 10 May 1999.

Nyström, D. and Nolin, M. and Tešanović, A. and Nordström, C. and Hansson, J. Pessimistic Concurrency Control and Versioning to Support Database Pointers in Real-Time Databases. pages 261–270, 2004. doi: 10.1109/ECRTS.2004.24.

Pettersson, H.T. and Robinson, J.T. On Optimistic Methods for Concurrency Control. *ACM Transactions on Database Systems*, 1981.

Pettersson, P. Modelling and verification of real-time systems using timed automata: theory and practice. *Department of Computer systems, Univ.*, 1999.

Rohit, P. *Common Data Access Issues. Expert One-on-One J2EE Design and Development. Wrox Press.* 2003. ISBN 978-0-7645-4385-2.

Sha, L. and Rajkumar, R. and Son, S.H. and et al. A real-time locking protocol. *Computers, IEEE Transactions*, 40(7):793–800, 1991.

Tsitchizris, Dionysios C. and Lochovsky, Fred H. *Data Models. Prentice–Hall.* 1982. ISBN 978-0-1319-6428-0.

Ullman, J. and Widom, J. *A First Course in Database Systems. Prentice–Hall.* 1997. ISBN 978-0-1386-1337-2.

Wagner, M. *SQL/XML:2006 – Evaluierung der Standardkonformität ausgewählter Datenbanksysteme.* Diplomica Verlag, 2010. ISBN 978-3-8366-9609-8.

Webster, M. Administration – Definition of administration by Merriam-Webster. a. URL `https://www.merriam-webster.com/dictionary/administration`.

Webster, M. Retrieval – Definition of retrieval by Merriam-Webster. b. URL `https://www.merriam-webster.com/dictionary/retrieval`.

Webster, M. Update – Definition of update by Merriam-Webster. c. URL `https://www.merriam-webster.com/dictionary/update`.

Xiong, M. and Ramamritham, K. Specification and analysis of transactions in real-time active databases. *Real-Time Database and Information Systems: Research Advances*, vol. 420:327–351, Springer US(2009).