Ατομική Διπλωματική Εργασία

ΜΟΝΤΕΛΟΠΟΙΗΣΗ, ΕΞΟΡΥΞΗ ΚΑΙ ΕΥΘΥΓΡΑΜΜΙΣΗ ΔΙΕΡΓΑΣΙΩΝ

Χαρά Θεοχάρους

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μοντελοποίηση, Εξόρυξη και Ευθυγράμμιση Διεργασιών

Χαρά Θεοχάρους

Επιβλέπων Καθηγητής
Γιάννης Δημόπουλος

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

Μάιος 2022

Ευχαριστίες

Ιδιαίτερες ευχαριστίες θα ήθελα να εκφράσω προς τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας, Δρ. Γιάννη Δημόπουλο, καθηγητή του Τμήματος Πληροφορικής, για την καθοριστική καθοδήγηση και βοήθεια που προσέφερε κατά την διάρκεια εκπόνησης της εργασίας. Επίσης, ευχαριστώ την οικογένεια και τους φίλους και συμφοιτητές μου για την πολύτιμη στήριξη τους.

Περίληψη

Οι στόχοι αυτής της διπλωματικής είναι δύο και αναφέρονται στη χρήση τεχνικών Τεχνητής Νοημοσύνης σε προβλήματα εξόρυξης διεργασιών *(process mining)*. Πρώτα γίνεται μια παρουσίαση της γλώσσας DECLARE, με την οποία είναι δυνατή η μοντελοποίηση διεργασιών. Κατόπιν μελετάται το πρόβλημα της ευθυγράμμισης ενός μοντέλου διεργασίας με ένα αρχείο καταγραφής συμβάντων στις γλώσσες PDDL και Clingo.

Η DECLARE είναι μια δηλωτική γλώσσα βασισμένη σε LTL, με την οποία μπορούμε να πετύχουμε αυτόματη ανακάλυψη δηλωτικών μοντέλων, που θα αξιοποιηθούν για εξόρυξη διεργασιών. Σε ένα δηλωτικό μοντέλο η συμπεριφορά μιας διεργασίας περιγράφεται ως ένα σύνολο κανόνων που πρέπει να ικανοποιούνται κατά τη διάρκεια εκτέλεσης της διεργασίας.

Η PDDL *(Planning Domain Definition Language)* είναι μια επίσημη γλώσσα αναπαράστασης γνώσης που έχει σχεδιαστεί για να εκφράζει μοντέλα δράσης *(planning models)*. Ο Προγραμματισμός Δράσης *(Planning)* είναι ο κλάδος της Τεχνητής Νοημοσύνης που επιδιώκει να αυτοματοποιήσει την αιτιολογία *(reasoning)* που διαμορφώνει ένα σχέδιο για την επίτευξη ενός δεδομένου στόχου σε μια δεδομένη κατάσταση.

Η Clingo είναι μια λογική γλώσσα προγραμματισμού συνόλου απαντήσεων *(Answer Set Programming)*. Αποτελεί μια απλή και ισχυρή γλώσσα μοντελοποίησης για την περιγραφή προβλημάτων ως λογικά προγράμματα, και τον υπολογισμό συνόλων απαντήσεων που αντιπροσωπεύουν λύσεις στο δεδομένο πρόβλημα.

Τα γενικά συμπεράσματα της μελέτης επικεντρώνονται στα πλεονεκτήματα της γλώσσας DECLARE για την μοντελοποίηση διεργασιών, έναντι των διαδικαστικών μοντέλων. Σχετικά με την σύγκριση των PDDL και Clingo για το πρόβλημα της ευθυγράμμισης, δεν φαίνεται να υπερτερεί κάποια έναντι της άλλης σε θέμα χρόνου εκτέλεσης.

Περιεχόμενα

Κεφάλαιο **1** **Εισαγωγή**..... **1**

1.1 Εξόρυξη διεργασιών 1

1.2 Δίκτυα Petri 3

1.3 Αρχεία καταγραφής συμβάντων 5

1.4 Έλεγχος συμμόρφωσης 6

1.5 Δηλωτικά μοντέλα 8

1.6 Γραμμική χρονική λογική 8

1.7 Προβλήματα δράσης και PDDL 12

1.8 Προγραμματισμός συνόλου απαντήσεων 16

Κεφάλαιο **2** **Δηλωτικά μοντέλα και DECLARE**..... **19**

2.1 Εισαγωγή 19

2.2 Πρότυπα 20

2.3 Προσέγγιση 26

2.4 Προηγμένες τεχνικές εξόρυξης 28

2.5 Εργαλεία 30

2.6 Παραδείγματα 37

Κεφάλαιο **3** **Ευθυγράμμιση Διεργασιών**..... **50**

3.1 Ευθυγράμμιση διεργασιών σε PDDL 50

3.2 Ευθυγράμμιση διεργασιών σε Clingo 61

Κεφάλαιο **4** **Συμπεράσματα** **67**

4.1 Δηλωτικά και διαδικαστικά μοντέλα 67

4.2 Ευθυγράμμιση διεργασιών σε PDDL και Clingo 67

Β ι β λ ι ο γ ρ α φ ί α …… **69**

Π α ρ ά ρ τ η μ α Α……..……..……..… **Α-1**

Π α ρ ά ρ τ η μ α Β…… **Β-5**

Π α ρ ά ρ τ η μ α Γ…… **Γ-7**

Π α ρ ά ρ τ η μ α Δ……..……..……..… **Δ-13**

Π α ρ ά ρ τ η μ α Ε…… **Ε-16**

Π α ρ ά ρ τ η μ α ΣΤ……**ΣΤ-18**

Κεφάλαιο 1

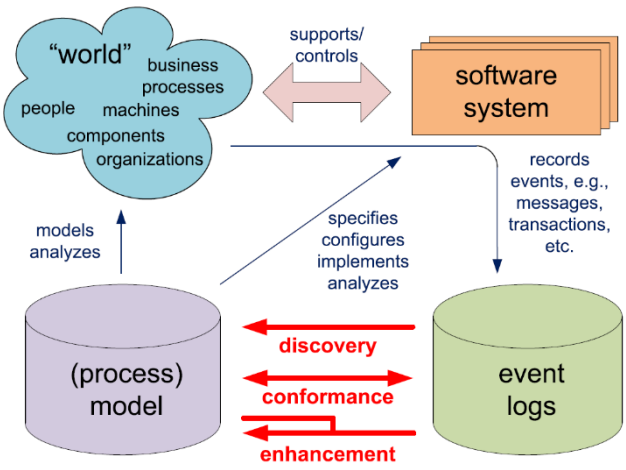
Εισαγωγή

1.1 Εισαγωγή στην εξόρυξη διεργασιών 1
1.2 Δίκτυα Petri 3
1.3 Αρχεία καταγραφής συμβάντων 5
1.4 Έλεγχος συμμόρφωσης 6
1.5 Δηλωτικά μοντέλα 8
1.6 Γραμμική χρονική λογική 8
1.7 Προβλήματα δράσης PDDL 12
1.8 Προγραμματισμός συνόλου απαντήσεων 16

1.1 Εισαγωγή στην εξόρυξη διεργασιών

Η εξόρυξη διεργασιών είναι ένας σχετικά νέος ερευνητικός κλάδος που βρίσκεται μεταξύ της μηχανικής μάθησης *(machine learning)* και της εξόρυξης δεδομένων *(data mining)* από τη μια πλευρά και της μοντελοποίησης και ανάλυσης διεργασιών *(process modelling and analysing)* από την άλλη. Η ιδέα της εξόρυξης διεργασιών είναι να ανακαλύπτει, να παρακολουθεί και να βελτιώνει πραγματικές διεργασίες εξάγοντας γνώση από αρχεία καταγραφής συμβάντων *(event logs)* που είναι άμεσα διαθέσιμα στα σημερινά συστήματα.

Το Σχήμα 1.1 δείχνει ότι η εξόρυξη διεργασιών δημιουργεί δεσμούς μεταξύ των πραγματικών *(world)* διεργασιών και των δεδομένων τους *(event logs)* από τη μια πλευρά, και των μοντέλων διεργασιών *(process model)* από την άλλη. Τα σημερινά συστήματα λογισμικού *(software systems)* καταγράφουν τεράστια ποσότητα γεγονότων, δηλαδή δημιουργούνται τα αρχεία καταγραφής συμβάντων, τα οποία μπορούν να χρησιμοποιηθούν για τη διεξαγωγή τριών τύπων εξόρυξης διεργασιών. Ο πρώτος τύπος εξόρυξης διεργασιών είναι η ανακάλυψη *(discovery)*. Μια τεχνική ανακάλυψης παίρνει ένα αρχείο καταγραφής συμβάντων και παράγει ένα μοντέλο χωρίς να χρησιμοποιεί εκ των προτέρων πληροφορίες. Ο δεύτερος τύπος εξόρυξης διεργασιών είναι η συμμόρφωση *(conformance)*, κατά την οποία ένα υπάρχον μοντέλο διεργασίας συγκρίνεται με ένα αρχείο καταγραφής συμβάντων της ίδιας διεργασίας. Ο έλεγχος συμμόρφωσης μπορεί να χρησιμοποιηθεί για να ελέγξει εάν η πραγματικότητα, όπως καταγράφεται στο αρχείο καταγραφής, συμμορφώνεται με το μοντέλο και αντίστροφα. Ο τρίτος τύπος εξόρυξης διεργασιών είναι η βελτίωση *(enhancement)*, όπου η ιδέα είναι να επεκταθεί ή να βελτιωθεί ένα υπάρχον μοντέλο διεργασίας χρησιμοποιώντας πληροφορίες σχετικά με την πραγματική διαδικασία που καταγράφονται σε κάποιο αρχείο καταγραφής συμβάντων, ώστε να αντικατοπτρίζει καλύτερα την πραγματικότητα.



Σχήμα 1.1 Εξόρυξη διεργασιών [1]

1.2 Δίκτυα Petri

Τα δίκτυα Petri *(Petri nets)* είναι η παλαιότερη και περισσότερο διερευνημένη γλώσσα μοντελοποίησης διεργασιών. Ένα δίκτυο Petri είναι ένα διμερές γράφημα που αποτελείται από θέσεις *(places)* και μεταβάσεις *(transitions)*. Η δομή του δικτύου είναι στατική, αλλά, υπό τον κανόνα πυροδότησης *(firing rule)*, οι μάρκες *(tokens)* μπορούν να ρέουν μέσω του δικτύου. Η κατάσταση ενός δικτυού Petri καθορίζεται από την κατανομή των μαρκών στις θέσεις του δικτύου και αναφέρεται ως σήμανση *(marking)*. Ακολουθούν αυστηρότεροι ορισμοί.

Ορισμός 1 (Δίκτυο Petri) [1]. Ένα δίκτυο Petri είναι μια τριάδα $N = (P, T, F)$ όπου

- P είναι ένα πεπερασμένο σύνολο θέσεων,
- T είναι ένα πεπερασμένο σύνολο μεταβάσεων τέτοιο ώστε $P \cap T = \emptyset$, και
- $F \subseteq (P \times T) \cup (T \times P)$ είναι ένα σύνολο κατευθυνόμενων τόξων που ονομάζεται σχέση ροής *(flow relation)* μεταξύ θέσεων και μεταβάσεων.

Ορισμός 2 (Σημασμένο Δίκτυο Petri) [1,2]. Ένα σημασμένο *(marked)* δίκτυο Petri είναι ένα ζεύγος (N, M) όπου

- $N = (P, T, F)$ είναι ένα δίκτυο Petri και
- $M \in B(P)$ είναι ένα πολυσύνολο του P που δηλώνει τη σήμανση του δικτύου, ή με ένα διαφορετικό ορισμό η σήμανση είναι μια συνάρτηση $m: P \to \mathbb{N}$.

Δεδομένου ενός δικτύου $N = (P, T, F)$, ως $\bullet t$ συμβολίζεται το σύνολο των θέσεων εισόδου *(input places)* της μετάβασης $t \in T$, που αποτελείται από τις θέσεις $p \in P$ για τις οποίες υπάρχει κατευθυνόμενο τόξο από το p στο t, δηλαδή $(p, t) \in F$.
Παρομοίως, ως $t \bullet$ συμβολίζεται το σύνολο των θέσεων εξόδου *(input places)* της μετάβασης $t \in T$, που αποτελείται από τις θέσεις $p \in P$ για τις οποίες υπάρχει κατευθυνόμενο τόξο από το t στο p.

Μία μετάβαση t είναι ενεργοποιημένη κατά τη σήμανση m αν κάθε θέση εισόδου της t περιέχει τουλάχιστον μία μάρκα, δηλαδή $\forall\, p \in \bullet t, m(p) > 0$.

Μία μετάβαση t μπορεί να πυροδοτήσει κατά τη σήμανση m αν είναι ενεργοποιημένη. Ως αποτέλεσμα της πυροδότησης, μία μάρκα θα καταναλωθεί από κάθε θέση εισόδου της t και μία θα παραχθεί σε κάθε θέση εξόδου της t. Δηλαδή, σύμφωνα με τον κανόνα πυροδότησης, η πυροδότηση της t κατά τη σήμανση m θα οδηγήσει στη σήμανση m' όπου

$$m'(p) = \begin{cases} m(p) - 1, \alpha\nu\ p \in\ \bullet t \backslash t \bullet \\ m(p) + 1, \alpha\nu\ p \in t \bullet \backslash \bullet t \\ m(p), \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

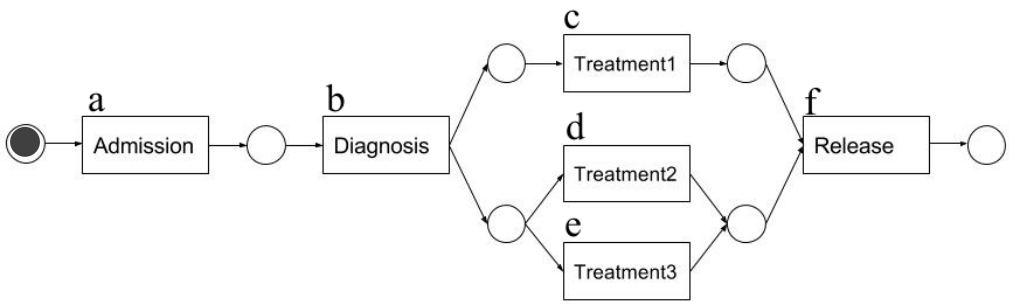
Ορισμός 3 (Δίκτυο Petri με ετικέτες) [2]. Ένα δίκτυο Petri με ετικέτες *(labelled Petri Net)* είναι μια πλειάδα (N, A, l, m_i, m_f) όπου

- $N = (P, T, F)$ είναι ένα δίκτυο Petri,
- A είναι το σύνολο των ετικετών δραστηριότητας,
- $l : T \nrightarrow$ A είναι μια συνάρτηση που συσχετίζει μια ετικέτα με ορισμένες μεταβάσεις στο T, και
- τα m_i και m_f είναι η αρχική και η τελική σήμανση.

Η συνάρτηση l δεν χρειάζεται να είναι ολική, καθώς ορισμένες μεταβάσεις δεν σχετίζονται με ετικέτα. Τέτοιες μεταβάσεις δεν αντιπροσωπεύουν πραγματικές δραστηριότητες σε διεργασίες, αλλά η εισαγωγή τους είναι μερικές φορές απαραίτητη για την ορθή αναπαράσταση της συμπεριφοράς της διεργασίας. Εκείνες οι μεταβάσεις του N που δεν σχετίζονται με καμία ετικέτα, δηλαδή δεν ανήκουν στο πεδίο ορισμού της l, συμβολίζονται με $Inv(N) \subset T$ *(inv από invisible).*

Στο Σχήμα 1.2 βλέπουμε ένα παράδειγμα ενός δικτύου Petri. Το δίκτυο αυτό μοντελοποιεί τη διεργασία της φροντίδας ενός ασθενή κατά την οποία μετά την Εισαγωγή πραγματοποιείται Διάγνωση, μετά την οποία πραγματοποιείται η Θεραπεία1 παράλληλα είτε με τη Θεραπεία2 είτε με τη Θεραπεία3, και τέλος ο ασθενής απελευθερώνεται. Οι θέσεις του δικτύου απεικονίζονται με τους επτά λευκούς κύκλους του σχήματος. Τα a, b, c, d, e, και f αποτελούν τις μεταβάσεις του δικτύου που απεικονίζονται με ορθογώνια και αντιστοιχούν στις ετικέτες Admission, Diagnosis, Treatment1, Treatment2, Treatment3, και Release. Αν το δίκτυο χρειαζόταν να κάποια invisible μετάβαση, αυτή θα απεικονιζόταν στο δίκτυο με ένα μαύρο τετράγωνο. Επίσης, βλέπουμε ότι η αρχική σήμανση του δικτύου αποτελείται από μία μάρκα στην αρχική

θέση του δικτύου, που απεικονίζεται με μαύρο κύκλο. Αν εφαρμόσουμε διαδοχικά τον κανόνα πυροδότησης μεταβάσεων, θα καταλήξουμε στη σήμανση που θα αποτελείται από μία μάρκα στην τελική θέση του δικτύου.



Σχήμα 1.2 Δίκτυο Petri για διεργασία φροντίδας ασθενή [3]

1.3 Αρχεία καταγραφής συμβάντων

Ένα αρχείο καταγραφής συμβάντων δείχνει ένα παράδειγμα συμπεριφοράς μιας διεργασίας. Αποτελείται από ένα σύνολο στιγμιότυπων της διεργασίας *(process instances)* και κάθε στιγμιότυπο διεργασίας ή αλλιώς ίχνος *(trace)* περιγράφεται από μια ακολουθία γεγονότων/συμβάντων *(events)*. Κάθε συμβάν συνοδεύεται από μια χρονική σήμανση *(timestamp)* για να υποδείξει την ώρα που αυτό έχει καταγραφεί, και συχνά άλλα πρόσθετα δεδομένα. Ακολουθεί αυστηρότερος ορισμός.

Ορισμός 4 (Αρχείο καταγραφής συμβάντων) [2]. Έστω ένα labelled δίκτυο Petri (N, A, l, m_i, m_f) όπου $N = (P, T, F)$, E το σύμπαν των γεγονότων και το T' το σύμπαν των χρονικών σημάνσεων. Ένα ίχνος *(trace)* $\varphi \subset E$ είναι ένα πεπερασμένο σύνολο γεγονότων. Ένα αρχείο καταγραφής συμβάντων L για το N είναι μια πλειάδα $(L, \lambda_N, \lambda_T)$ που αποτελείται από

- ένα σύνολο $L \subset 2^E$ ιχνών,
- μια συνάρτηση $\lambda_N : E \to A$, η οποία συσχετίζει κάθε γεγονός $e \in E$ με μια δραστηριότητα $\lambda_N(e) \in A$, και
- μια συνάρτηση $\lambda_T : E \to T'$, η οποία συσχετίζει κάθε συμβάν $e \in E$ με μια χρονική σήμανση $\lambda_T(e) \in T'$.

1.4 Έλεγχος συμμόρφωσης

Ο έλεγχος συμμόρφωσης *(conformance checking)* είναι το πρόβλημα της επαλήθευσης εάν οι πραγματικές εκτελέσεις διεργασιών, οι οποίες καταγράφονται σε αρχεία καταγραφής συμβάντων, είναι συμβατές με ένα ανακαλυφθέν μοντέλο που κωδικοποιεί τους περιορισμούς της διεργασίας, και μπορεί να χρησιμοποιηθεί για τη βελτίωση της υποκείμενης διαδικασίας.

Στο πλαίσιο του ελέγχου συμμόρφωσης, οι τεχνικές που βασίζονται στην ευθυγράμμιση μπορούν να εντοπίσουν ακριβώς πού παρατηρούνται αποκλίσεις κτίζοντας μια ευθυγράμμιση του αρχείου καταγραφής και του μοντέλου. Για να κτιστεί η ευθυγράμμιση τα συμβάντα στο αρχείο καταγραφής συμβάντων πρέπει να συσχετιστούν με μεταβάσεις στο μοντέλο και αντίστροφα. Συγκεκριμένα, πρέπει να συσχετίσουμε τις κινήσεις στο αρχείο καταγραφής με τις κινήσεις στο μοντέλο. Ωστόσο, ενδέχεται ορισμένες από τις κινήσεις στο αρχείο καταγραφής να μην μπορούν να μιμηθούν κινήσεις από το μοντέλο και το αντίστροφο, και αναπαρίστανται με ≫. Ακολουθεί αυστηρότερος ορισμός.

Ορισμός 5 (Κινήσεις Ευθυγράμμισης) [2]. Έστω ένα labelled δίκτυο Petri $N = (P, T, F, A, l, m_i, m_f)$ και ένα αρχείο καταγραφής συμβάντων $(L, \lambda_N, \lambda_T)$. Μία νόμιμη κίνηση ευθυγράμμισης για το N αναπαρίσταται από το ζεύγος $(S_L, S_M) \in (E \cup \{\gg\} \times T \cup \{\gg\})\backslash\{(\gg, \gg)\}$ και είναι

- κίνηση στο αρχείο καταγραφής αν $S_L \neq \gg$ και $S_M = \gg$
- κίνηση στο μοντέλο αν $S_L = \gg$ και $S_M \in T$
- σύγχρονη κίνηση αν $S_L \neq \gg$, $S_M \in T\backslash Inv(N)$ και $\lambda_N(S_L) = l(S_M)$

Μία ευθυγράμμιση είναι μία ακολουθία από κινήσεις ευθυγράμμισης που ορίζεται ως

Ορισμός 6 (Ευθυγράμμιση) [2]. Έστω ένα labelled δίκτυο Petri $N = (P, T, F, A, l, m_i, m_f)$, ένα αρχείο καταγραφής συμβάντων $(L, \lambda_N, \lambda_T)$, Γ_N το σύμπαν των κινήσεων ευθυγράμμισης για το N και το $(L, \lambda_N, \lambda_T)$, και $\varphi \in L$ ένα ίχνος. Η

ακολουθία $\gamma \in \Gamma_N^*$ είναι μια ευθυγράμμιση των N και φ αν, αγνοώντας όλες τις εμφανίσεις του ≫

- η προβολή στο πρώτο στοιχείο του γ δίνει μια ακολουθία $\sigma' \in E^*$ τέτοια ώστε
  - $e \in \sigma' \Leftrightarrow$ e ∈ φ, και
  - $\forall\, e' \in \sigma'.e'' \in prefix(\sigma', e') \Rightarrow \lambda T\,(e'') \leq \lambda T\,(e')$
- η προβολή στο δεύτερο στοιχείο του γ δίνει μια ακολουθία $\sigma'' \in T^*$ τέτοια ώστε $m_i \xrightarrow{\sigma''} m_f$

Πολλές ευθυγραμμίσεις είναι δυνατές για το ίδιο ίχνος. Ωστόσο, στοχεύουμε να βρούμε μια πλήρη ευθυγράμμιση των φ και N με ελάχιστο κόστος απόκλισης. Για να ορίσουμε τη σοβαρότητα μιας απόκλισης, εισάγουμε μια συνάρτηση κόστους για ευθυγραμμίσεις. Οποιαδήποτε ευθυγράμμιση με το χαμηλότερο δυνατό κόστος ονομάζεται βέλτιστη ευθυγράμμιση.

Ορισμός 7 (Συνάρτηση Κόστους) [2]. Έστω ένα labelled δίκτυο Petri $N = (P, T, F, A, l, m_i, m_f)$ και $\varphi \in L$ ένα αρχείο καταγραφής συμβάντων. Υποθέτοντας ως Γ_N το σύνολο όλων των νόμιμων κινήσεων ευθυγράμμισης για το N και το $(L, \lambda_N, \lambda_T)$, μια συνάρτηση κόστους κ αναθέτει ένα μη αρνητικό κόστος σε κάθε νόμιμη κίνηση: $\Gamma_N \to \mathbb{N}_0$. Το κόστος μιας ευθυγράμμισης $\gamma \in \Gamma_N$ μεταξύ φ και N υπολογίζεται ως το άθροισμα του κόστους όλων των συστατικών κινήσεων: $K(\gamma) = \sum_{(S_L, S_M) \in \gamma} \kappa(S_L, S_M)$. Η στοίχιση γ είναι μια βέλτιστη στοίχιση εάν, για οποιαδήποτε στοίχιση γ' των N και φ, $K(\gamma) \leq K(\gamma')$.

Οι κλασικές τεχνικές που βασίζονται στην ευθυγράμμιση βασίζονται στην υπόθεση της τέλειας γνώσης της σειράς με την οποία οι δραστηριότητες της διαδικασίας εκτελέστηκαν στην πραγματικότητα. Ωστόσο, η εμπειρία δείχνει ότι, λόγω σφαλμάτων και ανακριβειών καταγραφής, δεν είναι πάντα δυνατό να προσδιοριστεί η ακριβής σειρά με την οποία εκτελέστηκαν ορισμένες δραστηριότητες. Στο Κεφάλαιο 3 θα μελετηθεί μια τεχνική που βασίζεται σε ευθυγράμμιση, όπου αφαιρείται η υπόθεση τέλειας γνώσης της σειράς εκτέλεσης.

1.5 Δηλωτικά μοντέλα

Οι κλασικές τεχνικές ανακάλυψης μοντέλων διεργασιών έχουν δύο σημαντικά μειονεκτήματα. Τα παραγόμενα μοντέλα τείνουν να είναι μεγάλα και πολύπλοκα *(spaghetti-like models)*, ειδικά σε ευέλικτα περιβάλλοντα όπου οι εκτελέσεις διεργασιών περιλαμβάνουν πολλαπλές εναλλακτικές λύσεις, αφού οι παραδοσιακές τεχνικές ανακάλυψης κατασκευάζουν διαδικαστικά μοντέλα *(procedural models)* που δείχνουν ρητά όλες τις πιθανές συμπεριφορές. Επιπλέον, οι τεχνικές αυτές προσφέρουν περιορισμένες δυνατότητες καθοδήγησης της διαδικασίας εξόρυξης προς συγκεκριμένες ιδιότητες ενδιαφέροντος.

Αυτά τα προβλήματα μπορούν να λυθούν με την ανακάλυψη δηλωτικών μοντέλων *(declarative models)* που θα αναλυθούν στο Κεφάλαιο 2, τα οποία βασίζονται στη σημασιολογία της Γραμμικής Χρονικής Λογικής.

1.6 Γραμμική Χρονική Λογική

1.6.1 Τροπικές λογικές

Όπως αναφέρεται στο [4], οι τροπικές λογικές *(modal logics)* χρησιμοποιούνται στις περιπτώσεις που μια δήλωση δεν χρειάζεται να είναι απολύτως αληθής ή ψευδής, και επομένως πρέπει να γίνουν πιο λεπτές διακρίσεις από το «αληθές» ή «ψευδές». Έτσι, η τροπική λογική διέκρινε τις δηλώσεις μεταξύ αυτών που είναι αναγκαστικά αληθείς και εκείνων που είναι πιθανώς αληθείς.

Για παράδειγμα, το 1 + 1 = 2, ως δήλωση για τους φυσικούς αριθμούς του δεκαδικού συστήματος αρίθμησης, είναι αναγκαστικά αληθές λόγω του τρόπου με τον οποίο ορίζονται οι έννοιες. Αλλά οποιαδήποτε ιστορική δήλωση όπως «ο Ναπολέων έχασε τη μάχη του Βατερλώ» είναι μόνο πιθανώς αληθινή. Αν οι συνθήκες ήταν διαφορετικές, το αποτέλεσμα της μάχης μπορεί να ήταν διαφορετικό.

1.6.2 Χρονικές λογικές

Οι χρονικές λογικές *(temporal logics)* είναι ειδική κατηγορία τροπικών λογικών, όπου το «αναγκαστικά» ερμηνεύεται ως «πάντα» *(globally)* και το «ενδεχομένως» ερμηνεύεται ως «τελικά» *(future)* [4]. Η γραμμική χρονική λογική αποτελεί μια χρονική λογική.

1.6.3 Γραμμική χρονική λογική

1.6.3.1 Εισαγωγή

Στην γραμμική χρονική λογική *(Linear Temporal Logic – LTL)* μπορούν επίσης να υπάρξουν προτάσεις που θα είναι αληθείς την επόμενη χρονική στιγμή *(next)*, και προτάσεις που θα είναι αληθείς μέχρι να γίνουν αληθείς κάποιες άλλες *(until)*. Με τη χρήση των τεσσάρων αυτών τελεστών (η σημασιολογία των οποίων συνοψίζεται στον Πίνακα 2.1), η Γραμμική Χρονική Λογική μπορεί μεταξύ άλλων να εκφράσει ιδιότητες ασφάλειας (τίποτε 'κακό' δεν θα συμβεί, π.χ. βλέπε (1)), ζωτικότητας (κάτι 'καλό' τελικά θα συμβεί, π.χ. βλέπε (2)), και ανταπόκρισης (π.χ. βλέπε (3)), και έχει αποδειχθεί ότι είναι η προτιμώμενη λογική για την επαλήθευση προγραμμάτων.

Τελεστής	**Σημασιολογία**	
○φ	Next	Το φ θα συμβεί στην επόμενη χρονική στιγμή.
□φ	Globally	Το φ θα συμβαίνει σε κάθε μελλοντική χρονική στιγμή (συμπεριλαμβανομένης της τρέχουσας).
◊φ	Future	Το φ θα συμβεί σε κάποια μελλοντική χρονική στιγμή (συμπεριλαμβανομένης της τρέχουσας).
φ U ψ	Until	Το φ θα συμβαίνει τουλάχιστον μέχρι να συμβεί το ψ. Το ψ θα συμβεί στην τρέχουσα ή σε μελλοντική χρονική στιγμή.

Πίνακας 2.1 Σημασιολογία τελεστών LTL

Ακολουθούν παραδείγματα χρήσης των τελεστών της Γραμμικής Χρονικής Λογικής

(1) □ ¬damage Ποτέ δεν θα υπάρξει βλάβη στο αεροπλάνο.

(2) ◊ dest Το αεροπλάνο κάποτε θα φτάσει στον προορισμό

του.

(3)	□ (take_off ⇒ ○ flying)	Κάθε φορά που το αεροπλάνο απογειώνεται, την επόμενη στιγμή βρίσκεται στον αέρα.
(4)	flying U landed	Το αεροπλάνο βρίσκεται στον αέρα μέχρι να προσγειωθεί.

1.6.3.2 *Σύνταξη*

Πιο αυστηρά, η Προτασιακή Γραμμική Χρονική Λογική (PLTL) ορίζεται ως το μικρότερο σύνολο ιδιοτήτων που παράγονται ως εξής [5]:

$$\Phi ::= p \mid \neg\Phi \mid \Phi \vee \Phi \mid X\,\Phi \mid \Phi\, U\, \Phi$$

Δηλαδή:

- Κάθε ατομική πρόταση p είναι ιδιότητα
- Αν οι Φ και Ψ είναι ιδιότητες, τότε και οι $\neg\Phi$ και $\Phi \vee \Psi$ είναι ιδιότητες
- Αν η Φ είναι ιδιότητα, τότε και η $X\,\Phi$ (*next*) είναι ιδιότητα
- Αν οι Φ και Ψ είναι ιδιότητες, τότε και η $\Phi\, U\, \Psi$ (*until*) είναι ιδιότητα

Οι τελεστές F(*future)* και G(*globally)* μπορούν να εκφραστούν συναρτήσει των υπόλοιπων τελεστών. Συγκεκριμένα, $F\,\Phi \equiv true\ U\ \Phi$ και $G\,\Phi \equiv \neg F\,\neg\Phi$.

1.6.3.3 Σημασιολογική ερμηνεία

Η σημασιολογική ερμηνεία της PLTL δίνεται σε σχέση με τις δομές Kripke.
Μια δομή Kripke ορίζεται ως μια πλειάδα M = (S, R, I, Label), όπου

- S είναι ένα αριθμήσιμο σύνολο από καταστάσεις,
- I ⊆ S είναι το σύνολο των αρχικών καταστάσεων,
- R ⊆ S × S είναι μία σχέση μεταβάσεων, όπου (s, s') ∈ R αν υπάρχει μετάβαση από την κατάσταση s στην κατάσταση s', και
- Label : S → 2^{AP} είναι μια συνάρτηση η οποία συνδέει κάθε κατάσταση με τις ατομικές προτάσεις τις οποίες ικανοποιεί. [5]

Έστω μια κατάσταση s ∈ S. Τότε, Label(s) είναι το σύνολο των ατομικών προτάσεων που ισχύουν στην κατάσταση s. Ονομάζουμε μια ακολουθία από καταστάσεις $s_0\ s_1\ s_2\ldots$

μονοπάτι αν s_0 είναι μια αρχική κατάσταση και $(s_i, s_{i+1}) \in R$ για κάθε $i \geq 0$. Επίσης, αν $s = s_0\ s_1\ s_2\ldots$, τότε s_i είναι το μονοπάτι $s_i\ s_{i+1}\ s_{i+2}\ldots$.

Στο Σχήμα 1.3 φαίνεται η αναπαράσταση της πιο κάτω δομής Kripke, που αναφέρεται στην κίνηση ενός ελατηρίου:

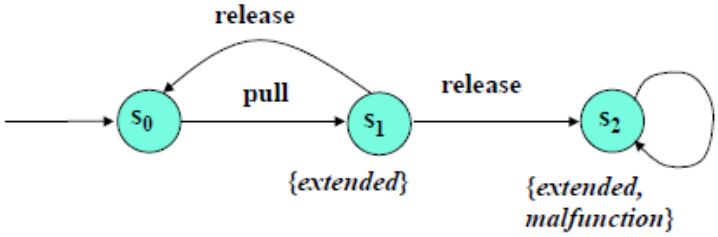
$(S = \{s_0, s_1, s_2\},$

$I = \{s_0\},$

$R = \{(s_0, s_1), (s_1, s_0), (s_1, s_2), (s_2, s_2)\},$

$Label(s_1) = \{extended\}$, $Label(s_2) = \{extended, malfunction\}$).

Τα $p_1 = s_0\ s_1\ s_0\ s_1\ s_0\ s_1\ s_0\ \ldots$, $p_2 = s_0\ s_1\ s_2\ s_2\ s_2\ s_2\ s_2\ \ldots$, $p_3 = s_0\ s_1\ s_0\ s_1\ s_2\ s_2\ s_2\ \ldots$ αποτελούν κάποια παραδείγματα μονοπατιών της δομής.



Σχήμα 1.3 Παράδειγμα δομής Kripke

Έστω μονοπάτι $s = s_0\ s_1\ s_2\ldots$, και ιδιότητα Φ. Ορίζουμε τη σχέση |= όπου

$s \models \Phi$ αν και μόνο αν η ιδιότητα Φ ικανοποιείται στο μονοπάτι s

ως εξής:

- $s \models p$ αν και μόνο αν $p \in Label(s0)$
- $s \models \neg\Phi$ αν και μόνο αν δεν ισχύει ότι $s \models \Phi$
- $s \models \Phi \vee \Psi$ αν και μόνο αν $(s \models \Phi)$ ή $(s \models \Psi)$
- $s \models X\ \Phi$ αν και μόνο αν $s_1 \models \Phi$
- $s \models \Phi\ U\ \Psi$ αν και μόνο αν υπάρχει $j \geq 0$ τέτοιο ώστε $s_j \models \Psi$ και για κάθε $0 \leq k < j$ ισχύει $s_k \models \Phi$. [5]

Για παράδειγμα, αν θεωρήσουμε το μονοπάτι $p_3 = s_0\ s_1\ s_0\ s_1\ s_2\ s_2\ s_2\ \ldots$ της δομής που φαίνεται στο Σχήμα 1.3, η ιδιότητα $p_3 \models F\ G\ extended$ ικανοποιείται, επειδή στο μονοπάτι p_3 υπάρχει μελλοντική κατάσταση που σε όλες τις καταστάσεις μετά από αυτήν ισχύει το *extended*.

Έστω δομή Kripke M και ιδιότητα Φ. M |= Φ αν και μόνο αν όλα τα μονοπάτια που ξεκινούν από κάθε αρχική κατάσταση της Kripke δομής M ικανοποιούν την ιδιότητα Φ. [5]

Για παράδειγμα, αν M η δομή που φαίνεται στο Σχήμα 1.3, η ιδιότητα *S |= F G extended* δεν ικανοποιείται, επειδή υπάρχει μονοπάτι $p_1 = s_0\ s_1\ s_0\ s_1\ s_0\ s_1\ s_0\ \ldots$ στο οποίο δεν υπάρχει κατάσταση που σε όλες τις καταστάσεις μετά από αυτήν ισχύει το *extended.*

1.7 Προβλήματα δράσης και PDDL

1.7.1 Προβλήματα δράσης

Ο Προγραμματισμός Δράσης *(Planning)* είναι ο κλάδος της Τεχνητής Νοημοσύνης που επιδιώκει να αυτοματοποιήσει την αιτιολογία *(reasoning)* που διαμορφώνει ένα σχέδιο δράσεων για την επίτευξη ενός δεδομένου στόχου σε μια δεδομένη κατάσταση. Ένα σύστημα δράσης λαμβάνει ως είσοδο ένα μοντέλο της αρχικής κατάστασης, τις διαθέσιμες ενέργειες για την αλλαγή της και τη συνθήκη στόχου, για την παραγωγή ενός σχεδίου που αποτελείται από εκείνες τις ενέργειες που θα επιτύχουν τον στόχο όταν εκτελεστούν από την αρχική κατάσταση.

1.7.2 Η γλώσσα PDDL

Η PDDL *(Planning Domain Definition Language)* είναι μια επίσημη γλώσσα αναπαράστασης γνώσης που έχει σχεδιαστεί για να εκφράζει μοντέλα δράσης. Αν και δεν είναι η μόνη γλώσσα μοντελοποίησης για τον προγραμματισμό δράσης, η PDDL είναι εκ των πραγμάτων η τυπική γλώσσα εισόδου πολλών συστημάτων προγραμματισμού δράσης και έχουν προκύψει αρκετές παραλλαγές της γλώσσας που καταγράφουν προβλήματα δράσης διαφορετικής φύσης και πολυπλοκότητας. [6]

Με την PDDL ο ορισμός ενός προβλήματος δράσης χωρίζεται σε δύο μέρη/αρχεία με επέκταση ".pddl": το *domain* και το *problem*. Η δομή των δύο αρχείων φαίνεται στο Σχήμα 1.4 και Σχήμα 1.5 αντίστοιχα.

Το *domain* είναι ένα γενικό μοντέλο των σχετικών πτυχών του κόσμου στον οποίο σχεδιάζουμε. Αρχικά ορίζουμε ποια χαρακτηριστικά *(requirements)* της PDDL χρησιμοποιεί το *domain*, τα οποία συνήθως δείχνουν τι είδους πρόβλημα θέλουμε να κωδικοποιήσουμε. Για τα πιο απλά προβλήματα χρησιμοποιούμε το χαρακτηριστικό *:strips*, και υπάρχουν πολλά άλλα χαρακτηριστικά [7] μεταξύ των οποίων είναι τα *:typing* που επιτρέπει τύπους αντικειμένων, *:universal-preconditions* που επιτρέπει τη χρήση του *forall*, και *:conditional-effects* που επιτρέπει τη χρήση του *when*, για κωδικοποίηση πιο σύνθετων προβλημάτων. Ακολούθως, το *domain* μπορεί να ορίζει κάποιους τύπους αντικειμένων *(objects)* του κόσμου, ορίζει τις μεταβλητές κατάστασης *(predicates)*, δηλαδή γεγονότα που μπορούν να είναι αληθή ή ψευδή, και τις δράσεις *(actions)*. Για κάθε δράση υπάρχουν παράμετροι αντικειμένων *(parameters)* που εμπλέκονται στην εκτέλεση της, οι προϋποθέσεις *(preconditions)* που πρέπει να πληρούνται για την εκτέλεσή της, δηλαδή η κατάσταση στην οποία πρέπει να βρίσκονται τα αντικείμενα που εμπλέκονται, και το αποτέλεσμα *(effect)* που επιφέρει η δράση στις καταστάσεις των αντικειμένων όταν εκτελεστεί.

```
(define (domain <όνομα domain>)
    (:requirements <χαρακτηριστικά της PDDL>)
    (:objects <αντικείμενα>)
    (:predicates <μεταβλητές κατάστασης>)
    (:action <όνομα δράσης 1>
            :parameters (<παράμετροι δράσης 1>)
            :precondition (<προϋποθέσεις δράσης 1>)
            :effect (<αποτέλεσμα δράσης 1>)
    )
)
```

Σχήμα 1.4 Δομή domain αρχείου

Το *problem* είναι ένα συγκεκριμένο παράδειγμα προβλήματος στον κόσμο που ορίζεται στο *domain*, και καθορίζει αντικείμενα *(objects)* του κόσμου, και από πού ξεκινάμε και τι πρέπει να επιτύχουμε, δηλαδή την αρχική *(init)* και τελική επιθυμητή *(goal)* κατάσταση των αντικειμένων.

```
(define (problem <όνομα problem>)
    (:domain <όνομα domain>)
    (:objects <αντικείμενα>)
    (:init <αρχική κατάσταση αντικειμένων>)
    (:goal <τελική κατάσταση αντικειμένων>)
            )
)
```

Σχήμα 1.5 Δομή problem αρχείου

Ακολουθεί ένα απλό παράδειγμα που κωδικοποιεί το εξής πρόβλημα:

> Δεδομένου ενός συνόλου τριών κύβων A, B, και C που βρίσκονται αρχικά σε ένα τραπέζι, ο στόχος είναι να δημιουργηθεί ένας πύργος με τους κύβους, στον οποίο ο C είναι πάνω στον A, ο A πάνω στον B, και ο B πάνω στο τραπέζι.

Δηλαδή θέλουμε να βρούμε τις πιθανές ακολουθίες βημάτων μετακίνησης των κύβων που θα μας οδηγήσουν από την αρχική κατάσταση των κύβων στον στόχο. Η κωδικοποίηση σε PDDL φαίνεται στο Σχήμα 1.6.

```
(define (domain blocks-world)
        (:requirements :strips)
        (:objects block)
        (:predicates
                (on ?x ?y - block)
                (clear ?x - block)
        )
        (:action move
                :parameters
                        (?b ?x ?y - block)
```

```
            :precondition
                (and (on ?b ?x) (clear ?b))
            :effect
                (and
                    (not (on ?b ?x)) (clear ?x))
                    (when (clear ?y)
                        (and (on ?b ?y) (not (clear ?y)))
                    )
                    (when (not (clear ?y))
                        (on ?b table)
                    )
                )
        )
)

(define (problem blocks-problem)
        (:domain blocks-world)
        (:objects A B C table - block)
        (:init
            (on A table) (clear A)
            (on B table) (clear B)
            (on C table) (clear C)
        )
        (:goal
            (and (on C A) (on A B))
        )
)
```

Σχήμα 1.6 Παράδειγμα κωδικοποίησης PDDL για το πρόβλημα των κύβων

Στο *domain* βλέπουμε ότι στον κόσμο υπάρχουν αντικείμενα τύπου block, και στο *problem* ορίζονται τα αντικείμενα A, B, C και table τύπου block. Το predicate (on ?x ?y - block) συμβολίζει ότι το block x βρίσκεται πάνω στο block y, ενώ το (clear ?x - block)

ότι δεν υπάρχει κάποιο αντικείμενο πάνω στο block x. Στο πρόβλημα αυτό υπάρχει μία δράση στην οποία συμμετέχουν 3 block, τα b, x, και y, για τα οποία ισχύει ότι αρχικά το b βρίσκεται πάνω στο x. Βλέπουμε ότι οι καταστάσεις των αντικειμένων που δηλώνουμε στο *precondition*, το *effect*, και το *goal* μπορούν να ενωθούν με σύζευξη *(and)*, ενώ επίσης υπάρχει και δυνατότητα άρνησης *(not)* μίας κατάστασης. Στην περίπτωση που *(when)* πάνω στο block y δεν βρίσκεται κάποιο block, τότε το b θα μετακινηθεί πάνω στο y, αλλιώς το b θα μετακινηθεί στο τραπέζι.

Δίνοντας την κωδικοποίηση των *domain* και *problem* σε έναν planner, θα μας δώσει ως απάντηση όλες τις δυνατές ακολουθίες κινήσεων που θα μας οδηγήσουν στον στόχο, αν εφαρμοστούν πάνω στην αρχική κατάσταση που ορίσαμε. Μια πιθανή ακολουθία κινήσεων που αποτελεί λύση στο πρόβλημα μας φαίνεται στο Σχήμα 1.7, όπου πρώτα μετακινούμε το A από το τραπέζι στο B, και ακολούθως το C από το τραπέζι στο A. Αυτή η λύση αποτελείται από τον ελάχιστο αριθμό βημάτων που θα μπορούσαν να γίνουν για αυτό το πρόβλημα. Η PDDL παρέχει μετρικές *(metrics)* για ελαχιστοποίηση του κόστους, ανάλογα με το πως το κόστος ορίζεται σε ένα πρόβλημα, τις οποίες θα δούμε στην υλοποίηση του προβλήματος ευθυγράμμισης διεργασιών σε PDDL στο Κεφάλαιο 3.

(move A table B)
(move C table A)

Σχήμα 1.7 Βέλτιστη λύση στο πρόβλημα των κύβων

1.8 Προγραμματισμός συνόλου απαντήσεων

Ο Προγραμματισμός Συνόλου Απαντήσεων [8] *(Answer Set Programming - ASP)* είναι μια μορφή δηλωτικού προγραμματισμού *(declarative programming)* και λογικού προγραμματισμού που ασχολείται με δύσκολα (κυρίως NP-hard) προβλήματα εύρεσης. Είναι βασισμένος στα *stable model semantics* του λογικού προγραμματισμού και χρησιμοποιεί κανόνες που μοιάζουν με κανόνες Prolog, αλλά μέσω διαφορετικών υπολογιστικών μεθόδων. Η διαφορά του ASP από άλλες μορφές λογικού προγραμματισμού βρίσκεται στην αναπαράσταση των λύσεων, αφού χρησιμοποιεί

σύνολα απαντήσεων *(answer sets)* για να αναπαριστά τις λύσεις, ενώ οι υπόλοιπες μορφές λογικού προγραμματισμού χρησιμοποιούν σύνολα αντικαταστάσεων *(answer substitutions)*. Η κύρια ιδέα του ASP είναι η δημιουργία ενός προγράμματος στο οποίο περιγράφεται ένα πρόβλημα χωρίς να χρησιμοποιηθούν μεταβλητές, δηλαδή το πρόγραμμα αποτελείται από περιορισμούς και σταθερές.

Πιο αυστηρά, το λογικό πρόγραμμα P σε ένα σύνολο ατόμων *(atoms)* A, είναι ένα πεπερασμένο σύνολο κανόνων *(rules)*. Ένας κανόνας r έχει τη μορφή

$$a_0 \leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n.,$$

όπου $0 \leq m \leq n$ και κάθε $a_i \in A$ είναι ένα άτομο για $0 \leq i \leq n$.

Ο κανόνας ορίζει ότι αν ισχύουν τα άτομα $a_1, \dots, a_m$ και δεν γνωρίζουμε ότι ισχύουν τα άτομα $a_{m+1}, \dots, a_n$ *(default negation)*, τότε ισχύει το άτομο a_0.

Επίσης ορίζουμε ως

- $head(r)\ h = a_0$
- $body(r) = \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\}$
- $body(r)^+ = \{a_1, \dots, a_m\}$
- $body(r)^- = \{a_{m+1}, \dots, a_n\}$
- $atom(P) = \bigcup_{r \in P}(\{head(r)\} \cup body(r)^+ \cup body(r)^-)$
- $body(P) = \{body(r) \mid r \in P\}$

Ένα πρόγραμμα P είναι θετικό αν $body(r) = \emptyset$ για όλα τα $r \in P$.

Ένα σύνολο ατόμων X είναι κλειστό κάτω από ένα θετικό πρόγραμμα P εάν και μόνο εάν για οποιοδήποτε $r \in P$, $body(r) \subseteq$ X όποτε το $body(r)^+ \subseteq$ X. Το X αντιστοιχεί σε ένα μοντέλο του P. Το μικρότερο σύνολο ατόμων που είναι κλειστό κάτω από ένα θετικό πρόγραμμα P συμβολίζεται με $C_n(P)$, και ονομάζεται ευσταθές μοντέλο *(stable model)* του P.

Δηλαδή, το ευσταθές μοντέλο ενός προγράμματος είναι το μικρότερο σύνολο συμπερασμάτων που μπορώ να εξάγω από τα συμπεράσματα του προγράμματος.

Ακολουθεί ο τρόπος με τον οποίο γράφουμε έναν κανόνα. Για παράδειγμα ο κανόνας

```
p(X) :- q(X).
```

που σε ένα πρόγραμμα με σταθερές τις {a, b, c} αντιστοιχεί στους τρείς κανόνες

```
p(a) :- q(a). p(b) :- q(b). p(c) :- q(c).
```

ορίζει ότι αν για κάποιο αντικείμενο ισχύει το q, τότε ισχύει και το r.

Εκτός από την μορφή των κανόνων που είδαμε πιο πάνω, υπάρχουν επίσης πιο σύνθετοι κανόνες, όπως οι κανόνες ακεραιότητας *(integrity)*, επιλογής *(choice)*, και συνάθροισης *(aggregates)* που θα χρησιμοποιηθούν και στην κωδικοποίηση του προβλήματος της ευθυγράμμισης σε Clingo στο Κεφάλαιο 3.

Ένας κανόνας ακεραιότητας μπορεί να είναι ο κανόνας

```
:- q(X), not p(X).
```

και ορίζει ότι στο σύνολο απαντήσεων δεν μπορεί να υπάρχει κάποιο αντικείμενο για το οποίο ισχύει το q και δεν γνωρίζουμε ότι ισχύει το p.

Ένας κανόνας επιλογής μπορεί να είναι ο κανόνας

```
{ p(X) } 1 :- q(Y).
```

και ορίζει ότι για κάθε αντικείμενο για το οποίο ισχύει το q μπορεί να ισχύει ή να μην ισχύει το p (για κάθε X θα προστεθεί στο σύνολο απαντήσεων το p το πολύ μία φορά).

Ένας κανόνας συνάθροισης μπορεί να είναι ο κανόνας

```
p(Y) :- r(Y), 2 #count { X : q(X) } 7.
```

και ορίζει ότι αν το πλήθος των αντικειμένων που ικανοποιούν το q είναι μεταξύ 2 και 7, και επίσης το αντικείμενο Y ικανοποιεί το r, τότε το Y ικανοποιεί το p.

Αντίστοιχα με το #count, υπάρχουν και άλλες συναρτήσεις όπως η #sum που υπολογίζει το άθροισμα και η #max που βρίσκει μέγιστο αριθμό.

Τέλος, οι δηλώσεις βελτιστοποίησης *(optimization statements)* εκφράζουν συναρτήσεις για ελαχιστοποίηση (ή μεγιστοποίηση) κόστους. Μία δήλωση βελτιστοποίησης μπορεί να είναι η

```
#minimize { X : q(X) }
```

και ορίζει ότι θέλουμε τις λύσεις με τον ελάχιστο αριθμό αντικειμένων X που ικανοποιούν το q.

Κεφάλαιο 2

Δηλωτικά μοντέλα και DECLARE

2.1 Εισαγωγή 19
2.2 Πρότυπα 20
2.3 Προσέγγιση 26
2.4 Προηγμένες τεχνικές εξόρυξης 28
2.5 Εργαλεία 30
2.6 Παραδείγματα 37

2.1 Εισαγωγή

Σε ένα δηλωτικό μοντέλο η συμπεριφορά μιας διεργασίας περιγράφεται ως ένα συμπαγές σύνολο κανόνων/περιορισμών που πρέπει να ικανοποιούνται καθ' όλη τη διάρκεια της εκτέλεσης της διεργασίας. Με τη δηλωτική γλώσσα DECLARE, που προτάθηκε από τους Pesic και Van der Aalst [9,10], μπορούμε να πετύχουμε αυτόματη ανακάλυψη δηλωτικών μοντέλων μέσω του εργαλείου εξόρυξης διεργασιών ProM, το οποίο θα δούμε στην ενότητα 2.5.1. Η DECLARE υπερτερεί από τις προσεγγίσεις τύπου Apriori, όπως η εξόρυξη ακολουθιών *(sequence mining)* και η εξόρυξη επεισοδίων *(episode mining)*, που μπορούν να ανακαλύψουν τοπικά μοτίβα σε ένα αρχείο καταγραφής, αλλά όχι ένα συνολικό μοντέλο, ούτε κανόνες που αντιπροσωπεύουν αρνητικές συμπεριφορές (τι δεν πρέπει να συμβεί) και επιλογές. Η προσέγγιση που ακολουθεί μπορεί επίσης να εφαρμοστεί όταν υπάρχουν μόνο θετικές ερμηνείες/στιγμιότυπα (τι πρέπει να συμβεί), κάτι που αποτελεί πλεονέκτημα σε σχέση με άλλες τεχνικές για την ανακάλυψη

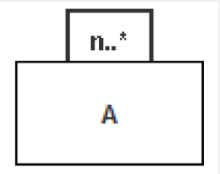
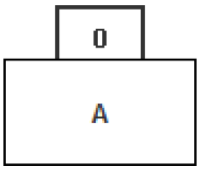
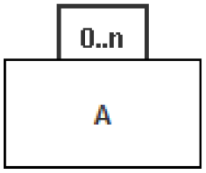
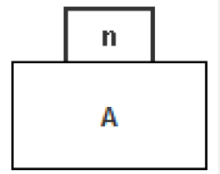
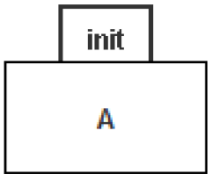
δηλωτικών διεργασιών που χρησιμοποιούν την επέκταση του λογικού προγραμματισμού SCIFF, καθώς τα αρχεία καταγραφής σπάνια επισημαίνουν ξεκάθαρα αρνητικές πληροφορίες.

2.2 Πρότυπα

Ένα μοντέλο DECLARE αποτελείται από ένα σύνολο περιορισμών *(constraint-based)* οι οποίοι βασίζονται σε πρότυπα. Τα πρότυπα είναι αφηρημένες οντότητες που ορίζουν παραμετροποιημένες κατηγορίες ιδιοτήτων. Είναι κατανοητά προς τον χρήστη λόγω της γραφικής τους αναπαράστασης, αλλά παράλληλα έχουν επίσημη σημασιολογία προκειμένου να είναι επαληθεύσιμα και εκτελέσιμα, αφού η σημασιολογία τους καθορίζεται μέσω τύπων LTL. Κάθε περιορισμός κληρονομεί τη γραφική αναπαράσταση και τη σημασιολογία από το πρότυπό του. Τα πρότυπα της DECLARE ταξινομούνται σε τέσσερις ομάδες: ύπαρξη (existence), σχέση (relation), αρνητική σχέση (negative relation) και επιλογή (choice), που περιγράφονται πιο κάτω.

Τα πρότυπα ύπαρξης, σύμφωνα με το [11], περιλαμβάνουν μόνο ένα γεγονός (μοναδιαία σχέση) και καθορίζουν την πληθικότητα ή τη θέση ενός συμβάντος σε ένα στιγμιότυπο διεργασίας. Η γραφική σημειογραφία και η σημασιολογία LTL κάθε προτύπου ύπαρξης φαίνονται στον Πίνακα 2.2., όπου έστω A ένα συμβάν.

Πρότυπο	**Σημασιολογία LTL**	**Γραφική σημειογραφία**	**Εξήγηση**
existence(1,A)	◊A		Το A θα συμβεί κάποια στιγμή
existence(2,A)	◊(A^○(existence(1,A)))		Το A θα συμβεί κάποια στιγμή, και από την επόμενη στιγμή το A θα συμβεί ξανά κάποια στιγμή, δηλαδή πρέπει να εμφανίζεται τουλάχιστον 2 φορές σε

			ένα στιγμιότυπο διεργασίας
… existence(n,A)	… ◊(A^○(existence(n-1,A)))		Το Α πρέπει να εμφανίζεται τουλάχιστον n φορές σε ένα στιγμιότυπο διεργασίας
absence(A)	¬existence(1,A)		Το Α δεν πρέπει να εμφανίζεται σε ένα στιγμιότυπο διεργασίας
absence(2,A)	¬existence(2,A)		Το Α δεν πρέπει να εμφανίζεται 2 φορές, δηλαδή πρέπει να εμφανίζεται το πολύ 1 φορά, σε ένα στιγμιότυπο διεργασίας
… absence(n+1,A)	… ¬existence(n+1,A)		Το Α δεν πρέπει να εμφανίζεται n+1 φορές, δηλαδή πρέπει να εμφανίζεται το πολύ n φορές, σε ένα στιγμιότυπο διεργασίας
exactly(n,A)	existence(n,A) ^ absence(n+1,A)		Το Α πρέπει να εμφανίζεται ακριβώς n φορές σε ένα στιγμιότυπο διεργασίας
init(A)	A		Κάθε στιγμιότυπο διεργασίας πρέπει να ξεκινά με το Α

Πίνακας 2.2 Πρότυπα ύπαρξης

Ακολουθούν παραδείγματα χρήσης των προτύπων ύπαρξης

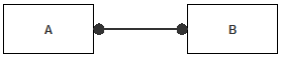
(5) existence(1,take_off) Το αεροπλάνο θα απογειωθεί τουλάχιστον μία φορά.

(6) absence(1,damage) Ποτέ δεν θα υπάρξει βλάβη στο αεροπλάνο.

(7) exactly(2,take_off) Το αεροπλάνο θα απογειωθεί ακριβώς δύο φορές.

(8) init(landed) Το αεροπλάνο βρίσκεται αρχικά προσγειωμένο.

Τα πρότυπα σχέσης, σύμφωνα με το [11], ορίζουν μια εξάρτηση μεταξύ δύο γεγονότων (δυαδική σχέση). Η γραφική σημειογραφία και η σημασιολογία LTL κάθε προτύπου σχέσης φαίνονται στον Πίνακα 2.3., όπου έστω A και B δύο συμβάντα. Τα πρότυπα alternate response, alternate precedence και alternate succession ενισχύουν τα πρότυπα response, precedence και succession απαιτώντας ότι τα συμβάντα πρέπει να εναλλάσσονται χωρίς επαναλήψεις αυτών των γεγονότων στο μεταξύ. Τα πρότυπα chain response, chain precedence και chain succession ενισχύουν τα πρότυπα response, precedence και succession απαιτώντας ότι οι εμφανίσεις των δύο γεγονότων (A και B) βρίσκονται η μία δίπλα στην άλλη.

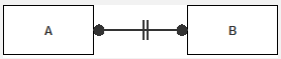
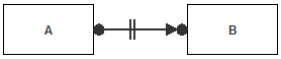
Πρότυπο	**Σημασιολογία LTL**	**Γραφική σημειογραφία**	**Εξήγηση**
responded existence(A,B)	$\Diamond A \Rightarrow \Diamond B$		Εάν συμβεί το A, τότε το B θα πρέπει επίσης να συμβεί (είτε πριν είτε μετά το A)
co-existence(A,B)	$\Diamond A \Leftrightarrow \Diamond B$		Εάν συμβεί ένα από τα A ή B, θα πρέπει να συμβεί και το άλλο
response(A,B)	$\Box(A \Rightarrow \Diamond B)$		Οποιαδήποτε στιγμή, όταν συμβαίνει το A, το B θα πρέπει τελικά να συμβεί μετά το A
precedence(A,B)	$(\neg B \text{ U } A) \vee \Box(\neg B)$		Δεν θα συμβεί το B μέχρι να συμβεί το A ή δεν θα συμβεί ποτέ το

			Β, δηλαδή το Β πρέπει να συμβαίνει μόνο εάν το Α έχει προηγηθεί
succession(A,B)	response(A,B) ∧ precedence(A,B)		Ισχύουν οι σχέσεις response(A,B) και precedence(A,B)
alternate response(A,B)	□(A ⇒ ○(¬A U B))		Οποιαδήποτε στιγμή, όταν συμβαίνει το Α, τότε από την επόμενη στιγμή δεν θα ξανασυμβεί το Α μέχρι να συμβεί το Β
alternate precedence(A,B)	precedence(A,B) ∧ □(B ⇒ ○(precedence(A,B)))		Το Β πρέπει να συμβαίνει μόνο εάν το Α έχει προηγηθεί, και κάθε στιγμή που συμβαίνει το Β, το Β πρέπει να συμβαίνει μόνο εάν το Α έχει προηγηθεί.
alternate succession(A,B)	alternate response(A,B) ∧ alternate precedence(A,B)		Ισχύουν οι σχέσεις alternate response(A,B) και alternate precedence(A,B)
chain response(A,B)	□(A ⇒ ○B)		Οποιαδήποτε στιγμή, όταν συμβαίνει το Α, την επόμενη χρονική στιγμή το Β θα πρέπει να συμβεί
chain precedence(A,B)	□(○B ⇒ A)		Οποιαδήποτε στιγμή, όταν την επόμενη στιγμή συμβεί το Β,

			τότε αυτή τη στιγμή πρέπει να συμβεί το A
chain succession(A,B)	□(A ⇔ ○B)		Οποιαδήποτε στιγμή, όταν συμβεί το A, τότε την επόμενη στιγμή θα συμβεί το B, και αντίστροφα

Πίνακας 2.3 Πρότυπα σχέσης

Τα πρότυπα αρνητικής σχέσης, σύμφωνα με το [11], ορίζουν μια αρνητική εξάρτηση μεταξύ δύο γεγονότων (δυαδική σχέση). Η γραφική σημειογραφία και η σημασιολογία LTL κάθε προτύπου αρνητικής σχέσης φαίνονται στον Πίνακα 2.4.

Πρότυπο	**Σημασιολογία LTL**	**Γραφική σημειογραφία**	**Εξήγηση**
not co-existence(A,B)	¬(◊A^◊B)		Τα A και B δεν μπορούν να συμβούν στο ίδιο στιγμιότυπο διεργασίας
not succession(A,B)	□(A ⇒ ¬ (◊B))		Οποιαδήποτε εμφάνιση του A δεν μπορεί να ακολουθηθεί τελικά από το B
not chain succession(A,B)	□(A ⇒ ○(¬B)		Το A δεν μπορεί να ακολουθηθεί απευθείας από το B

Πίνακας 2.4 Πρότυπα αρνητικής σχέσης

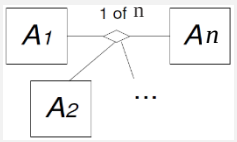
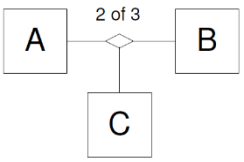
Ακολουθούν παραδείγματα χρήσης μερικών προτύπων σχέσης και αρνητικής σχέσης

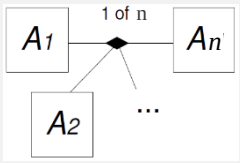
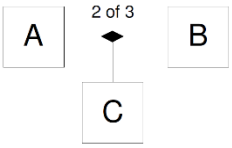
(9) response(take_off,landed) Όταν το αεροπλάνο απογειώνεται, κάποτε στο μέλλον θα προσγειωθεί.

(10) precedence(take_off,flying) Το αεροπλάνο βρίσκεται στον αέρα εάν προηγουμένως έχει απογειωθεί.

(11) not chain succession(damage,dest) Όταν υπάρχει βλάβη στο αεροπλάνο, δεν μπορεί την επόμενη στιγμή να βρίσκεται στον προορισμό του.

Τα πρότυπα επιλογής, σύμφωνα με το [12], καθορίζουν ότι πρέπει να γίνει επιλογή μεταξύ κάποιων γεγονότων. Η γραφική σημειογραφία και η σημασιολογία LTL κάθε προτύπου επιλογής φαίνονται στον Πίνακα 2.5.

Πρότυπο	**Σημασιολογία LTL**	**Γραφική σημειογραφία**	**Εξήγηση**
1 of 2(A,B)	$\Diamond A \vee \Diamond B$		Πρέπει να συμβεί τουλάχιστον 1 γεγονός μεταξύ των A και B
1 of 3(A,B,C)	$\Diamond A \vee \Diamond B \vee \Diamond C$		Πρέπει να συμβεί τουλάχιστον 1 γεγονός μεταξύ των A,B,C
… 1 of n(A_1,...,A_n)	… $\Diamond A_1 \vee ... \vee \Diamond A_n$		Πρέπει να συμβεί τουλάχιστον 1 γεγονός μεταξύ των A_1,...,A_n
2 of 3(A,B,C)	$(\Diamond A \wedge \Diamond B) \vee$ $(\Diamond B \wedge \Diamond C) \vee$ $(\Diamond A \wedge \Diamond C)$		Πρέπει να συμβούν τουλάχιστον 2 γεγονότα από τα A,B,C
exclusive 1 of 2(A,B)	$(\Diamond A \wedge \neg \Diamond B) \vee$ $(\neg \Diamond A \wedge \Diamond B)$		Πρέπει να συμβεί ακριβώς 1 γεγονός μεταξύ των A και B

exclusive 1 of 3 (A,B,C)	(◊A∧¬◊B∧¬◊C) ∨ (¬◊A∧◊B∧¬◊C) ∨ (¬◊A∧¬◊B∧◊C)		Πρέπει να συμβεί ακριβώς 1 γεγονός μεταξύ των A,B,C
… exclusive 1 of n(A_1...,A_n)	… …		Πρέπει να συμβεί ακριβώς 1 γεγονός μεταξύ των A_1,...,A_n
exclusive 2 of 3 (A,B,C)	(◊A ∧ ◊B ∧ ¬◊C) ∨ (◊A ∧ ¬◊B ∧ ◊C) ∨ (¬◊A ∧ ◊B ∧ ◊C)		Πρέπει να συμβούν ακριβώς 2 γεγονότα μεταξύ των A,B,C

Πίνακας 2.5 Πρότυπα επιλογής

Ακολουθούν παραδείγματα χρήσης μερικών προτύπων επιλογής

(12) 1 of 2(landed,dest) Το αεροπλάνο είναι προσγειωμένο ή βρίσκεται στον προορισμό του (ή και τα δύο).

(13) exclusive 1 of 2(flying,landed) Το αεροπλάνο είτε βρίσκεται στον αέρα είτε είναι προσγειωμένο (όχι και τα δύο).

(14) 2 of 3(landed,dest,damage) Το αεροπλάνο είναι προσγειωμένο και στον προορισμό του, ή είναι προσγειωμένο και έχει βλάβη, ή είναι στον προορισμό του και έχε βλάβη.

2.3 Προσέγγιση

Σε αυτή την ενότητα αρχικά εισάγεται ένας αλγόριθμος για την ανακάλυψη μοντέλων DECLARE, και στη συνέχεια θα επεκταθούμε σε ορισμένες πρόσθετες παραμέτρους, όπως αυτά περιγράφονται στο άρθρο [11], ώστε να ληφθούν υπόψη η χρονική πολυπλοκότητα του αλγορίθμου και ο θόρυβος στα αρχεία καταγραφής, και να μπορεί να εφαρμοστεί ο πιο πάνω αλγόριθμος σε πραγματικά αρχεία καταγραφής.

2.3.1 Αλγόριθμος ανακάλυψης μοντέλων DECLARE

Για την ανακάλυψη μοντέλων DECLARE χρειαζόμαστε ένα σύνολο T από πρότυπα DECLARE και ένα αρχείο καταγραφής W. Το πρώτο βήμα του αλγορίθμου ανακάλυψης είναι να δημιουργήσουμε ένα μοντέλο $D_{candidates}$ που αποτελείται από υποψήφιους περιορισμούς DECLARE. Έστω E = $\{e_1, \ldots, e_n\}$ το σύμπαν των γεγονότων (κλάσεις γεγονότων, όλα τα διαφορετικά είδη γεγονότων) που ανήκουν στο W. Το $D_{candidates}$ δημιουργείται με την εισαγωγή κάθε προτύπου $t(a_1, \ldots, a_k)$ στο T με όλες τις πιθανές διαθέσεις *(dispositions)* μήκους k των n κλάσεων συμβάντων $e_1, \ldots, e_n$. Κάθε πρότυπο με k παραμέτρους παράγει n^k περιορισμούς στο μοντέλο έτσι ώστε ο αριθμός των περιορισμών στο $D_{candidates}$ να είναι $h = \sum_{t \epsilon T} n^{k_t}$, όπου k_t είναι ο αριθμός των παραμέτρων του t. Στο δεύτερο βήμα του αλγορίθμου, το $D_{candidates}$ μεταφράζεται σε ένα LTL μοντέλο $L_{candidates}$, που αποτελείται από μια λίστα LTL κανόνων που αντιπροσωπεύουν το $D_{candidates}$. Κάθε κανόνας στο $L_{candidates}$ ελέγχεται για να αποφασίσουμε εάν ικανοποιείται στο W ή όχι. Εάν δεν ικανοποιείται, αφαιρείται από το μοντέλο. Στο τέλος της φάσης ελέγχου, είναι διαθέσιμο ένα φιλτραρισμένο LTL μοντέλο L που περιλαμβάνει τους υπόλοιπους κανόνες LTL. Τέλος, το μοντέλο L μεταφράζεται σε ένα μοντέλο DECLARE D που είναι το αποτέλεσμα της διαδικασίας ανακάλυψης.

2.3.2 Ποσοστό συμβάντων

Η παράμετρος Ποσοστό Συμβάντων *(Percentage of Events - PoE)* μπορεί να χρησιμοποιηθεί για να αποφευχθεί η ανακάλυψη περιορισμών που αναφέρονται σε κλάσεις συμβάντων που σπάνια εμφανίζονται στο αρχείο καταγραφής. Αυτή η παράμετρος καθορίζει το ποσοστό των κλάσεων γεγονότων που θα χρησιμοποιηθούν για τη δημιουργία των υποψηφίων περιορισμών. Για παράδειγμα, εάν PoE = 50% οι περιορισμοί που ανακαλύφθηκαν θα περιλαμβάνουν μόνο το 50% των κλάσεων συμβάντων στο αρχείο καταγραφής (των πιο συχνών). Αυτή η παράμετρος έχει επίσης θετική επίδραση στη χρονική πολυπλοκότητα του αλγορίθμου.

2.3.3 Ποσοστό στιγμιότυπων

Η παράμετρος Ποσοστό Στιγμιότυπων *(Percentage of Instances - PoI)* μπορεί να χρησιμοποιηθεί για να καθοριστεί ότι ένας περιορισμός DECLARE μπορεί να ανακαλυφθεί ακόμα και αν δεν ισχύει για όλες τις περιπτώσεις διεργασίας του αρχείου καταγραφής. Για παράδειγμα, εάν PoI = 80%, θα ανακαλυφθεί ένας περιορισμός εάν τουλάχιστον το 80% των στιγμιότυπων διεργασίας ικανοποιούν τον περιορισμό. Αυτή η παράμετρος είναι χρήσιμη σε περίπτωση θορυβωδών αρχείων καταγραφής, όπου οι κανόνες παραβιάζονται σε εξαιρετικές περιπτώσεις, αλλά ισχύουν για τις περισσότερες περιπτώσεις.

2.4 Προηγμένες τεχνικές εξόρυξης

Σε αυτήν την ενότητα εισάγονται δύο προηγμένες τεχνικές για την υποστήριξη της ανακάλυψης των μοντέλων DECLARE: η Περικομμένη Σημασιολογία *(Truncated Semantics)* και η Ανίχνευση Κενότητας *(Vacuity Detection)* για τύπους LTL, όπως αυτές περιγράφονται στο άρθρο [11].

2.4.1 Περικομμένη σημασιολογία

Στον βασικό αλγόριθμο, ένας περιορισμός ανακαλύπτεται εάν ικανοποιείται σε ένα δεδομένο ποσοστό των περιπτώσεων διεργασίας. Ωστόσο, συχνά τα διαθέσιμα αρχεία καταγραφής εξάγονται από μεγαλύτερα αρχεία καταγραφής και τα στιγμιότυπα διεργασίας είναι προθέματα μεγαλύτερων στιγμιότυπων διεργασίας. Αυτό επηρεάζει τους περιορισμούς που ανακαλύφθηκαν, δηλαδή ορισμένοι περιορισμοί παραμένουν ανεξερεύνητοι.

Σε ένα περικομμένο *(truncated)* στιγμιότυπο, η τιμή αλήθειας ενός τύπου LTL μπορεί να είναι μη οριστική (δηλαδή, να παραβιάζεται προσωρινά ή να ικανοποιείται προσωρινά). Για την αντιμετώπιση αυτού του προβλήματος, εισάγεται η ισχυρή σημασιολογία και η αδύναμη σημασιολογία για τύπους LTL. Στην παραδοσιακή σημασιολογία LTL (που

ονομάζεται ουδέτερη σημασιολογία), ένας τύπος αξιολογείται ως αληθής εάν ικανοποιείται προσωρινά και αξιολογείται ως ψευδής εάν παραβιάζεται προσωρινά.

Χρησιμοποιώντας την ασθενή σημασιολογία αντί για την ουδέτερη, είναι δυνατό να αποδυναμωθεί το κριτήριο αποδοχής που χρησιμοποιείται για το φιλτράρισμα της λίστας των υποψήφιων περιορισμών στον αλγόριθμο ανακάλυψης, που οδηγεί σε αυξημένο αριθμό περιορισμών που ανακαλύφθηκαν.

Η ισχυρή σημασιολογία συνεπάγεται την ουδέτερη σημασιολογία, η οποία συνεπάγεται την ασθενή σημασιολογία. Από αυτό το θεώρημα προκύπτει ότι χρησιμοποιώντας την ισχυρή σημασιολογία έχουμε τη μέγιστη αξιοπιστία για την ικανοποίηση ενός περιορισμού. Από την άλλη, χρησιμοποιώντας την ασθενή σημασιολογία, έχουμε μεγαλύτερη ευελιξία στη διαδικασία ανακάλυψης. Η ασθενής σημασιολογία LTL φαίνεται να δίνει πιο σημαντικά αποτελέσματα όταν χρησιμοποιούνται αρχεία καταγραφής που περιέχουν περικομμένα στιγμιότυπα διεργασίας.

Για παράδειγμα, με ασθενή σημασιολογία LTL, ο περιορισμός "κάθε αίτημα τελικά επιβεβαιώνεται" ικανοποιείται σε ένα στιγμιότυπο διεργασίας που περιέχει μόνο ένα αίτημα και δεν περιέχει επιβεβαιώσεις, ενώ με ισχυρή σημασιολογία δεν ικανοποιείται.

2.4.2 Ανίχνευση κενότητας

Ένας περιορισμός ικανοποιείται εν κενώ *(vacuously satisfied)*, εάν ο περιορισμός δεν είναι πραγματικά ‘ενεργοποιημένος’. Στο πλαίσιο της ανακάλυψης, ενδιαφέροντες μάρτυρες *(interesting witnesses)* είναι οι περιπτώσεις διεργασίας όπου ένας περιορισμός ικανοποιείται όχι εν κενώ. Για να συνδυάσουμε την ανακάλυψη περιορισμών όχι εν κενώ με τα οφέλη της ασθενής σημασιολογίας LTL, ονομάζουμε ένα (περικομμένο) στιγμιότυπο διεργασίας ενδιαφέροντα μάρτυρα για έναν περιορισμό φ αν χρησιμοποιώντας την ασθενή σημασιολογία LTL, το στιγμιότυπο διεργασίας ικανοποιεί το φ, και χρησιμοποιώντας την ουδέτερη σημασιολογία LTL, κάποιο πρόθεμα του (περικομμένου) στιγμιότυπου διεργασίας ικανοποιεί το φ όχι εν κενώ.

Ένα τέτοιο παράδειγμα περιορισμού που ικανοποιείται εν κενώ δίνεται από τον περιορισμό "κάθε αίτημα τελικά επιβεβαιώνεται" σε ένα στιγμιότυπο που δεν περιέχει αιτήματα. Ένα περικομμένο στιγμιότυπο διεργασίας είναι ενδιαφέροντας μάρτυρας για τον περιορισμό "κάθε αίτημα τελικά επιβεβαιώνεται" εάν περιέχει τουλάχιστον ένα αίτημα ακολουθούμενο από μια επιβεβαίωση.

2.5 Εργαλεία

Σε αυτή την ενότητα θα δούμε τα εργαλεία ProM και RuM που θα μας βοηθήσουν σε επόμενη (2.6) ενότητα στην ανακάλυψη και παρακολούθηση δηλωτικών μοντέλων.

2.5.1 ProM

Το ProM (Process Mining) [13] είναι ένα εργαλείο εξόρυξης διεργασιών, και παρέχει μεταξύ άλλων εργαλεία *(plugins)* ανακάλυψης και επεξεργασίας δηλωτικών μοντέλων. Ακολουθούν τα βήματα μετατροπής ενός αρχείου σε αρχείο καταγραφής συμβάντων και ανακάλυψης δηλωτικού μοντέλου με τα αντίστοιχα plugins. Περισσότερα plugins του ProM και μια γενικότερη εισαγωγή στην εξόρυξη διεργασιών αναλύονται στο διαδικτυακό μάθημα [14].

Ανοίγοντας το πρόγραμμα ProM υπάρχει η επιλογή *import* όπου μπορούμε να εισάγουμε αρχεία *csv* ή *xes*. Έστω ότι εισάγουμε το αρχείο online_orders_event_log.csv, που περιέχει τα γεγονότα που καταγράφηκαν για τη διεργασία διαδικτυακών παραγγελιών μιας ιστοσελίδας για 6 διαφορετικά στιγμιότυπα, και τη χρονική στιγμή που ολοκληρώθηκε κάθε γεγονός, όπως φαίνονται στον Πίνακα 2.6.

case	*event*	*completeTime*
1	Order	27/10/2021 14:08
1	Pay online	27/10/2021 14:10
1	Receive	27/10/2021 14:45
2	Order	27/10/2021 14:10
2	Pay online	27/10/2021 14:11
2	Receive	27/10/2021 15:46
2	Order	27/10/2021 19:10
2	Pay online	27/10/2021 19:11
2	Receive	27/10/2021 19:46
3	Order	27/10/2021 14:30
3	Receive	27/10/2021 14:50
3	Pay in cash	27/10/2021 15:52
4	Order	27/10/2021 15:00
4	Receive	27/10/2021 15:35
4	Pay in cash	27/10/2021 15:36
4	Order	27/10/2021 20:00
4	Receive	27/10/2021 20:35
4	Pay in cash	27/10/2021 20:36
5	Order	27/10/2021 15:10
5	Pay online	27/10/2021 15:11
5	Receive	27/10/2021 15:45
5	Order	27/10/2021 20:39
5	Receive	27/10/2021 21:30
5	Pay in cash	27/10/2021 21:31
6	Order	27/10/2021 20:10
6	Receive	27/10/2021 20:20
6	Pay in cash	27/10/2021 20:21
6	Order	27/10/2021 20:30
6	Pay online	27/10/2021 20:31
6	Receive	27/10/2021 21:03

Πίνακας 2.6 Τεχνητό αρχείο καταγραφής συμβάντων για το τη διεργασία διαδικτυακών παραγγελιών

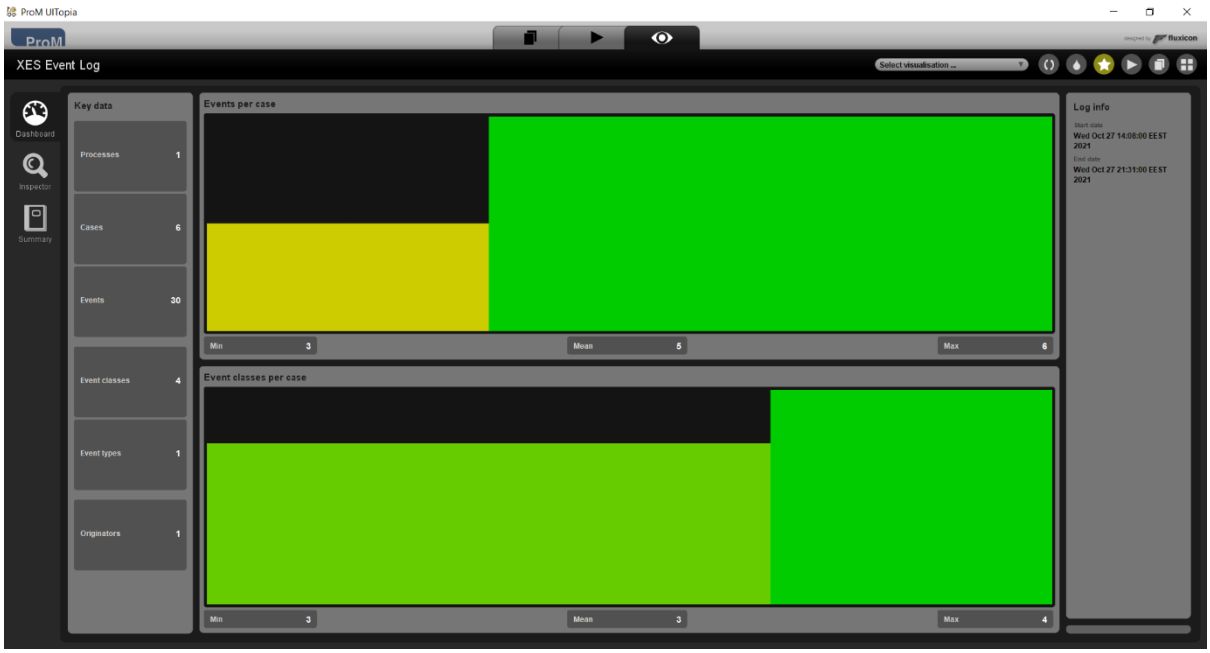
Εισάγοντας το πιο πάνω αρχείο, το ProM δίνει τη δυνατότητα μετατροπής του αρχείου μέσω του *Convert CSV to XES*, το οποίο μετατρέπει το αρχείο csv σε αρχείο καταγραφής

συμβάντων. Το αρχείο αυτό περιέχει τα δεδομένα του Πίνακα 2.6, αλλά σε μορφή *xes*. Στο Σχήμα 2.1 φαίνεται το περιεχόμενο του *xes* αρχείου για το πρώτο στιγμιότυπο (trace) που περιέχει τρία γεγονότα (event) που χαρακτηρίζονται από το όνομά (name) τους και τη χρονική στιγμή (timestamp) που συνέβησαν.

```
<trace>
	<string key="concept:name" value="1"/>
	<event>
		<string key="concept:name" value="Order"/>
		<string key="lifecycle:transition" value="complete"/>
		<date key="time:timestamp" value="2021-10-27T14:08:00.000+03:00"/>
	</event>
	<event>
		<string key="concept:name" value="Pay online"/>
		<string key="lifecycle:transition" value="complete"/>
		<date key="time:timestamp" value="2021-10-27T14:10:00.000+03:00"/>
	</event>
	<event>
		<string key="concept:name" value="Receive"/>
		<string key="lifecycle:transition" value="complete"/>
		<date key="time:timestamp" value="2021-10-27T14:45:00.000+03:00"/>
	</event>
</trace>
```

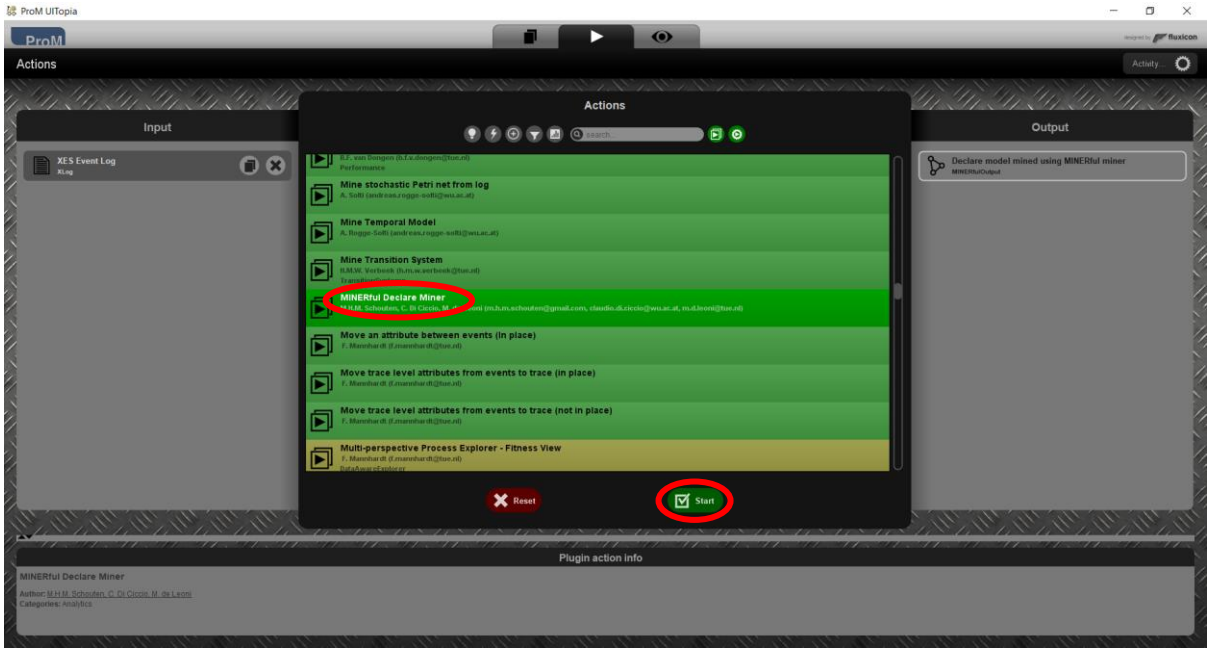
Σχήμα 2.1 Περιεχόμενο *xes* αρχείου για ένα στιγμιότυπο της διεργασίας διαδικτυακών παραγγελιών

Μετά την μετατροπή φαίνονται στην οθόνη του ProM μερικές πληροφορίες σχετικά με τη διεργασία, όπως φαίνεται στο Σχήμα 2.2. Οι σημαντικότερες πληροφορίες που μπορούμε να δούμε είναι ο αριθμός των στιγμιότυπων, των γεγονότων και των κλάσεων γεγονότων (διαφορετικά είδη γεγονότων) στον πίνακα στα αριστερά της οθόνης. Στο πάνω γράφημα βλέπουμε τον ελάχιστο, το μέσο, και το μέγιστο αριθμό γεγονότων ανά στιγμιότυπο διεργασίας και στο κάτω τον ελάχιστο, το μέσο, και το μέγιστο αριθμό διαφορετικών γεγονότων ανά στιγμιότυπο διεργασίας.



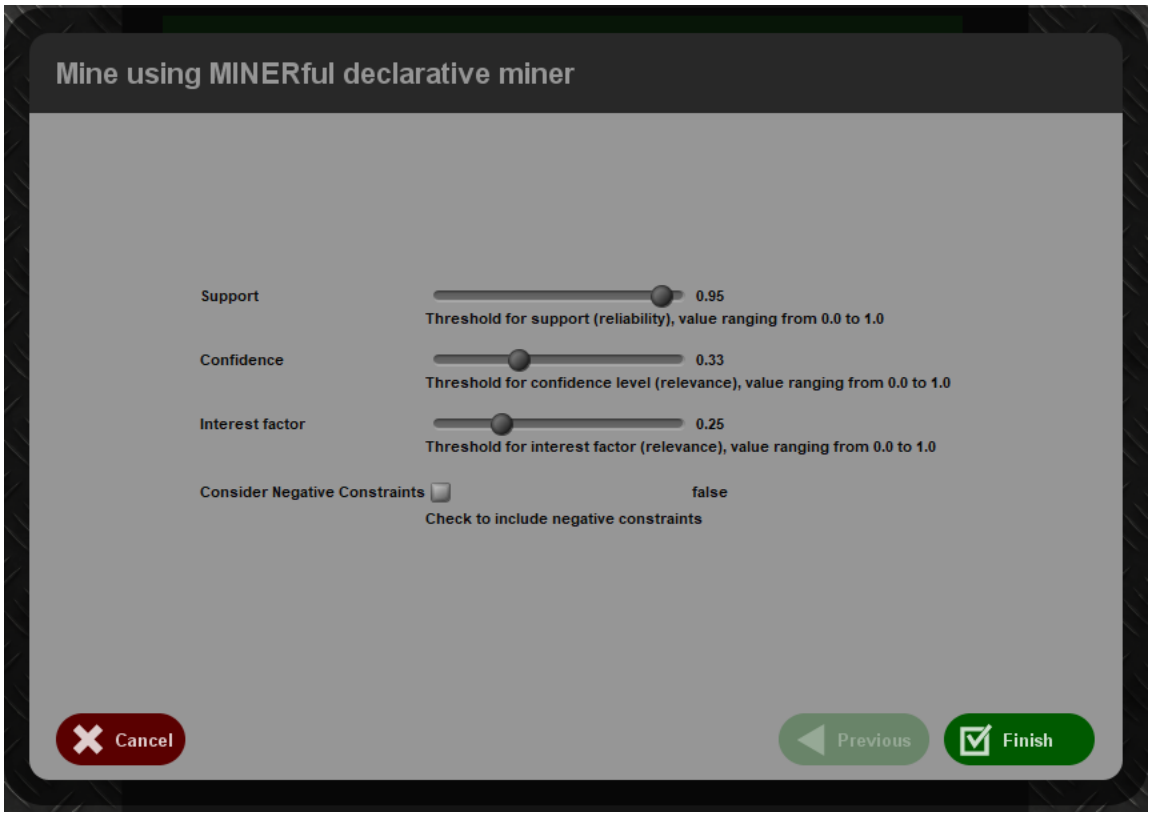
Σχήμα 2.2 Πληροφορίες για τη διεργασία διαδικτυακών παραγγελιών

Έχοντας επιλεγμένο το αρχείο καταγραφής και πατώντας το *Action button*, φαίνονται τα plugins που μπορούν να χρησιμοποιηθούν με είσοδο το αρχείο αυτό, όπως φαίνεται στο Σχήμα 2.3.



Σχήμα 2.3 Actions για αρχείο καταγραφής

Αν επιλέξουμε το plugin *MINERful Declare Miner*, Start (όπως φαίνεται στο Σχήμα 2.3) και στη συνέχεια Finish, θα παραχθεί ένα δηλωτικό μοντέλο (όπως φαίνεται στο Σχήμα 2.5) βάσει του αρχείου καταγραφής, με τις προκαθορισμένες παραμέτρους (όπως φαίνονται στο Σχήμα 2.4).



Σχήμα 2.4 Παράμετροι πριν την ανακάλυψη με το MINERful declarative miner

Οι παράμετροι που φαίνονται στο Σχήμα 2.4 μπορούν να ρυθμιστούν από τον χρήστη πριν ή και μετά την ανακάλυψη του μοντέλου.

Η παράμετρος υποστήριξη *(support)* είναι ο αριθμός των εκπληρωμένων ενεργοποιήσεων *(activations)* των περιορισμών διαιρούμενος είτε με τον αριθμό των ιχνών στο αρχείο καταγραφής, είτε τον αριθμό των εμφανίσεων των ενεργοποιήσεων [15]. Για παράδειγμα στον περιορισμό «κάθε υποβολή παραγγελίας ακολουθείται από παραλαβή της παραγγελίας», κάθε υποβολή παραγγελίας είναι η ενεργοποίηση, και θα

εκπληρωθεί ή θα παραβιαστεί ανάλογα με το αν θα ταιριάξει με κάποια παραλαβή της παραγγελίας (στόχος) ή όχι.

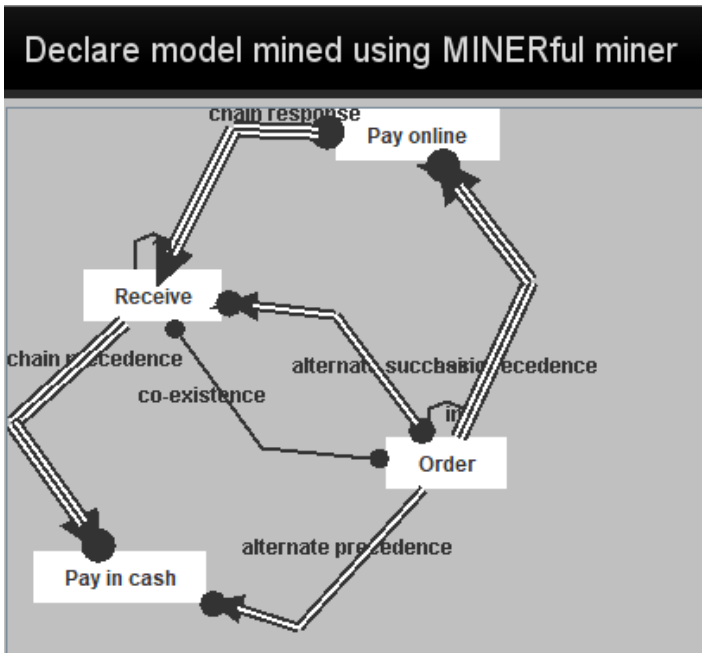
Η παράμετρος εμπιστοσύνης *(confidence)* είναι το γινόμενο της υποστήριξης και του κλάσματος των ιχνών στο αρχείο καταγραφής όπου είτε εμφανίζεται η δραστηριότητα (περιορισμοί ύπαρξης), είτε λαμβάνει χώρα η ενεργοποίηση (περιορισμοί σχέσης). [15]

Η παράμετρος συντελεστή ενδιαφέροντος *(interest factor)* είναι το γινόμενο της εμπιστοσύνης και το κλάσμα των ιχνών στο αρχείο καταγραφής όπου είτε εμφανίζεται η περιορισμένη δραστηριότητα (περιορισμοί ύπαρξης), είτε εμφανίζεται ο στόχος (περιορισμοί σχέσης). [15]

Άρα μειώνοντας τον αριθμό στις παραμέτρους αυτές, χαλαρώνουμε τους περιορισμούς που πρέπει να πληροί το μοντέλο που θα ανακαλυφθεί.

Σχετικά με την επιλογή για αρνητικούς περιορισμούς (consider negative constraints), όταν η διεργασία χαρακτηρίζεται από μέρη με άκαμπτη δομή, το μοντέλο που ανακαλύφθηκε μπορεί να περιέχει τεράστιο αριθμό αρνητικών περιορισμών. Ως εκ τούτου, παρέχεται η επιλογή να μην ληφθούν υπόψη αρνητικοί περιορισμοί, αυξάνοντας έτσι την αναγνωσιμότητα των μοντέλων που ανακαλύφθηκαν.

Το δηλωτικό μοντέλο που ανακαλύφθηκε θα σχολιαστεί περεταίρω στην ενότητα 2.6.



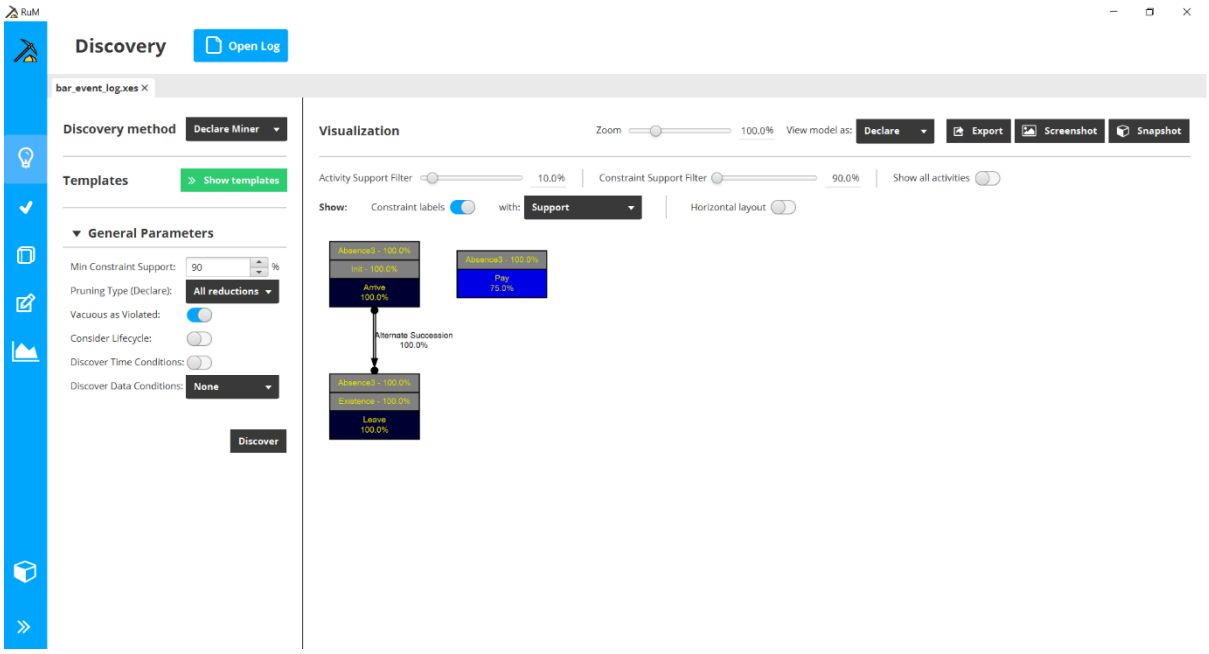
Σχήμα 2.5 Δηλωτικό μοντέλο από το MINERful declarative miner

2.5.2 RuM

Το RuM (Rule mining) [16] είναι ένα εργαλείο που παρέχει δυνατότητες ανακάλυψης δηλωτικών μοντέλων, ελέγχου συμμόρφωσης, δημιουργίας αρχείων καταγραφής, επεξεργασίας και παρακολούθησης δηλωτικών μοντέλων. Τα βήματα για περισσότερες λειτουργίες του RuM είναι διαθέσιμες στον οδηγό [17].

Στο Σχήμα 2.6 φαίνεται το δηλωτικό μοντέλο που ανακαλύφθηκε με τη μέθοδο Declare Miner μέσω του RuM (με τις προκαθορισμένες παραμέτρους) βάσει του αρχείου καταγραφής που περιέχει τα δεδομένα του Πίνακα 2.6 που είδαμε πιο πάνω. Προσφέρονται επίσης δυνατότητες για αλλαγή της μεθόδου ανακάληψης, επιλογή των προτύπων που θα χρησιμοποιηθούν στο μοντέλο, ρύθμισης της παραμέτρου υποστήριξης (που είδαμε στην ενότητα 2.5.1) και άλλων παραμέτρων σχετικά με την ανακάλυψη και την γραφική αναπαράσταση του μοντέλου.

Το δηλωτικό μοντέλο που ανακαλύφθηκε, όπως και μοντέλα που θα ανακαλυφθούν με άλλες μεθόδους, θα σχολιαστεί περεταίρω στην ενότητα 2.6.



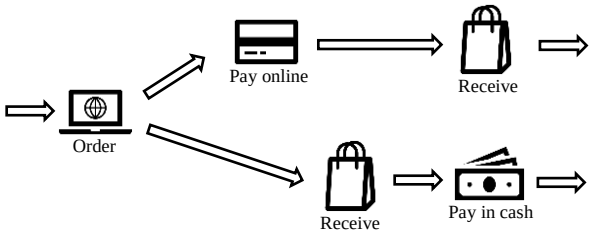
Σχήμα 2.6 Δηλωτικό μοντέλο από το RuM και παράμετροι ανακάλυψης

2.6 Παραδείγματα

Σε αυτή την ενότητα θα σχολιαστούν τα αποτελέσματα που δίνουν τα διαφορετικά εργαλεία σε δύο παραδείγματα απλουστευμένων διεργασιών.

2.6.1 Διεργασία για διαδικτυακές παραγγελίες

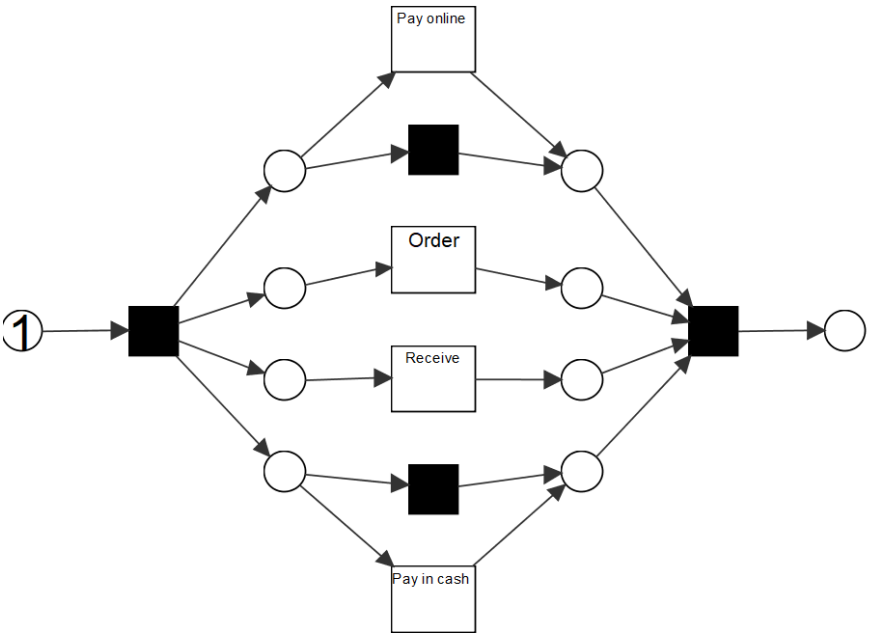
Το πρώτο παράδειγμα αφορά τις διαδικτυακές παραγγελίες ενός ηλεκτρονικού καταστήματος. Όπως μπορούμε να δούμε στο Σχήμα 2.7, κατά τη διεργασία μιας διαδικτυακής παραγγελίας, αφού έχει γίνει επιλογή της επιθυμητής παραγγελίας του πελάτη, υπάρχει δυνατότητα διαδικτυακής πληρωμής και ακολούθως παραλαβής της παραγγελίας από τον πελάτη, ή δυνατότητα πληρωμής με μετρητά εφόσον έχει γίνει η παραλαβή της παραγγελίας από τον πελάτη. Για απλούστευση θεωρούμε ότι ένας πελάτης μπορεί να κάνει νέα παραγγελία εφόσον έχει παραλάβει πιθανές προηγούμενες παραγγελίες του.



Σχήμα 2.7 Διεργασία διαδικτυακών παραγγελιών

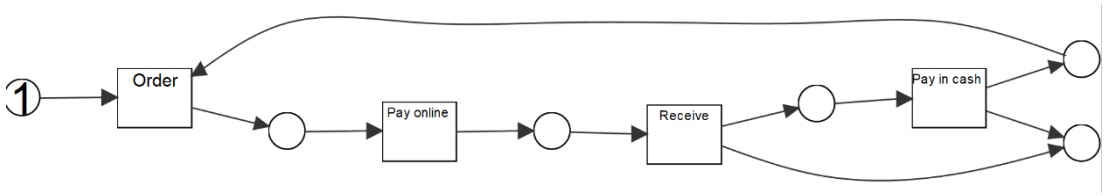
Αρχικά, έχω δημιουργήσει ένα αρχείο csv με τα περιεχόμενα που φαίνονται στον Πίνακα 2.6. Η μετατροπή του σε αρχείο καταγραφής συμβάντων έγινε με το plugin *Convert CSV to XES*, όπως περιεγράφηκε στην ενότητα 2.5.1.

Το Petri net του σχήματος 2.8 που παράχθηκε με το ProM plugin *Mine Petri net with Inductive Miner* βάσει του αρχείου καταγραφής, δεν είναι αντιπροσωπευτικό για την διεργασία που περιεγράφηκε πιο πάνω. Βλέπουμε ότι ορθά η παραγγελία και η παραλαβή της είναι αναγκαίο να συμβούν σε κάθε στιγμιότυπο. Όμως λανθασμένα είναι πιθανόν να μην συμβεί ούτε πληρωμή με μετρητά ούτε διαδικτυακή πληρωμή. Επίσης, δεν καθορίζεται η χρονική σειρά που επιτρέπεται να συμβούν τα γεγονότα.



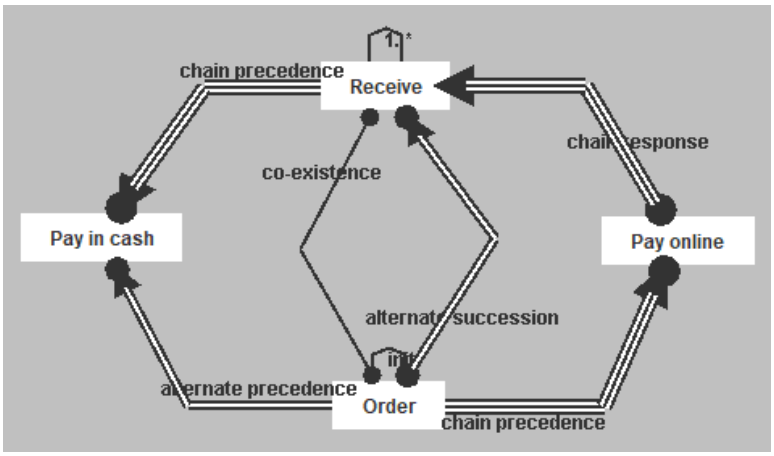
Σχήμα 2.8 Petri net για διεργασία διαδικτυακών παραγγελιών από *Inductive Miner*

Στο Σχήμα 2.9 βλέπουμε το Petri net που παράχθηκε με το ProM plugin *Alpha Miner* βάσει του αρχείου καταγραφής, που επίσης δεν είναι αντιπροσωπευτικό για την διεργασία που περιεγράφηκε πιο πάνω. Σε αντίθεση με το προηγούμενο μοντέλο στο Σχήμα 2.8, στο μοντέλο στο Σχήμα 2.9 καθορίζεται κάποια χρονική σειρά που επιτρέπεται να συμβούν τα γεγονότα. Όμως λανθασμένα είναι πιθανόν να συμβεί και πληρωμή με μετρητά και διαδικτυακή πληρωμή, αφού βάσει αυτού του μοντέλου η διαδικτυακή πληρωμή είναι υποχρεωτική.



Σχήμα 2.9 Petri net για διεργασία διαδικτυακών παραγγελιών από *Alpha Miner*

Το δηλωτικό μοντέλο στο Σχήμα 2.10 που παράχθηκε με το ProM plugin *MINERful Declare Miner* φαίνεται να είναι ορθό βάσει της περιγραφής της διεργασίας που δόθηκε πιο πάνω, και η εξήγηση δίνεται στον Πίνακα 2.7 για κάθε υπαρξιακή και σχεσιακή δήλωση ξεχωριστά.



Σχήμα 2.10 Δηλωτικό μοντέλο για διεργασία διαδικτυακών παραγγελιών από το ProM

DECLARE	**LTL Σημασιολογία**	**Εξήγηση**
init (Order)	Order	Κάθε στιγμιότυπο της διεργασίας ξεκινά με την επιλογή της παραγγελίας
existence (1, Receive)	◊Receive	Κάθε στιγμιότυπο της διεργασίας περιέχει παραλαβή της παραγγελίας
co-existence (Order, Receive)	◊Order ⇔ ◊Receive	Όταν γίνεται παραγγελία γίνεται και παραλαβή, και αντίστροφα
alternate succession (Order, Receive)	□(Order ⇒ O(¬Order U Receive)) ^ ((¬Receive U Order) ν □(¬ Receive))) ^ □(Receive ⇒ O((¬Receive U Order) ν □(¬ Receive))) **απόδειξη κάτω από τον πίνακα*	Όταν γίνει παραγγελία θα ακολουθήσει παραλαβή, και παραλαβή γίνεται αν έχει προηγηθεί παραγγελία, χωρίς την μεσολάβηση άλλης παραγγελίας ή παραλαβής και στις δύο προϋποθέσεις
chain precedence (Order, Pay online)	□(O Pay online ⇒ Order)	Η διαδικτυακή πληρωμή γίνεται αν το ακριβώς προηγούμενο βήμα ήταν η παραγγελία
chain response (Pay online, Receive)	□(Pay online ⇒ O Receive)	Όταν γίνει διαδικτυακή πληρωμή, το αμέσως επόμενο βήμα είναι η παραλαβή της παραγγελίας
alternate precedence (Order, Pay in cash)	((¬Pay in cash U Order) ν □(¬Pay in cash)) ^ □(Pay in cash ⇒ O((¬Pay in cash U Order) ν □(¬ Pay in cash))))	Η πληρωμή με μετρητά γίνεται αν έχει προηγηθεί παραγγελία, χωρίς την μεσολάβηση άλλης παραγγελίας ή πληρωμής με μετρητά

	**απόδειξη κάτω από τον πίνακα*	
chain precedence (Receive, Pay in cash)	□(O Pay in cash ⇒ Receive)	Η πληρωμή με μετρητά γίνεται αν το ακριβώς προηγούμενο βήμα ήταν η παραλαβή της παραγγελίας

<u>* Απόδειξη 1</u>

alternate succession (Order, Receive)

= alternate response (Order, Receive) ^ alternate precedence (Order, Receive)

= □(Order ⇒ O(¬Order U Receive)) ^ precedence(Order, Receive) ^ □(Receive ⇒ O(precedence(Order, Receive)))

= □(Order ⇒ O(¬Order U Receive)) ^ ((¬Receive U Order) ν □(¬ Receive))) ^ □(Receive ⇒ O((¬Receive U Order) ν □(¬ Receive)))

<u>* Απόδειξη 2</u>

alternate precedence (Order, Pay in cash)

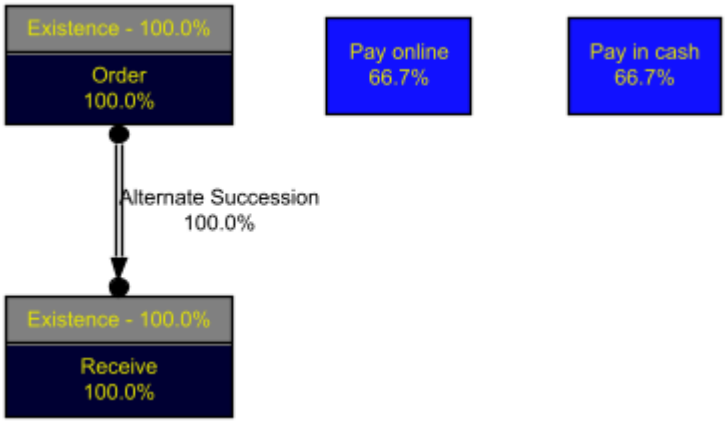
= precedence(Order, Pay in cash) ^ □(Pay in cash ⇒O(precedence(Order, Pay in cash)))

= ((¬Pay in cash U Order) ν □(¬ Pay in cash)) ^ □(Pay in cash ⇒ O((¬Pay in cash U Order) ν □(¬ Pay in cash)))

Πίνακας 2.7 Μετάφραση σε LTL για διεργασία διαδικτυακών παραγγελιών

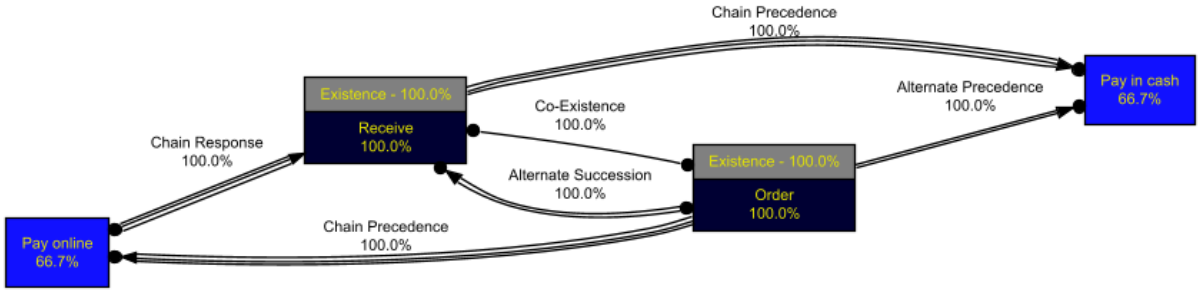
Για την μετάφραση κάθε υπαρξιακής και σχεσιακής δήλωσης σε σημασιολογία LTL, το ProM plugin Declare to LTL με είσοδο το δηλωτικό μοντέλο δεν ανταποκρίθηκε με κάποιο αποτέλεσμα.

Στη συνέχεια θα δούμε τη διαδικασία ανακάλυψης δηλωτικού μοντέλου με το εργαλείο RuM. Το δηλωτικό μοντέλο στο Σχήμα 2.11 που παράχθηκε με την προκαθορισμένη μέθοδο ανακάλυψης Declare Miner του RuM δεν είναι αντιπροσωπευτικό για την διεργασία διαδικτυακών παραγγελιών. Έχει ανακαλυφθεί ορθά η ύπαρξη των γεγονότων Order και Receive και η σχέση alternate succession μεταξύ, όμως δεν έχει ανακαλυφθεί καμιά σχέση των γεγονότων Pay online και Pay in cash με τα υπόλοιπα γεγονότα.



Σχήμα 2.11 Δηλωτικό μοντέλο για διεργασία διαδικτυακών παραγγελιών από το RuM με τη μέθοδο Declare Miner

Το δηλωτικό μοντέλο που παράχθηκε στο RuM αλλάζοντας την μέθοδο ανακάλυψης (Discovery method) από Declare Miner σε MINERful, φαίνεται στο Σχήμα 2.12 και είναι αντίστοιχο με το μοντέλο που παράχθηκε με το ProM plugin MINERful Declare Miner για την διεργασία διαδικτυακών παραγγελιών (Σχήμα 2.10).



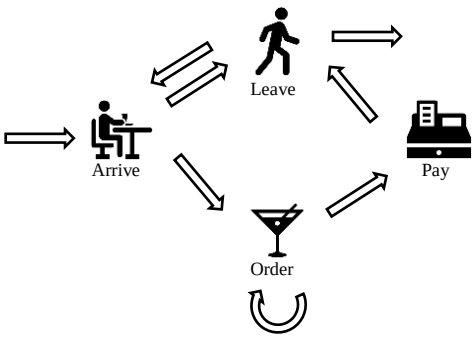
Σχήμα 2.12 Δηλωτικό μοντέλο για διεργασία διαδικτυακών παραγγελιών από το RuM με τη μέθοδο MINERful

Συμπερασματικά, με αυτό το παράδειγμα είδαμε το πλεονέκτημα που έχουν τα δηλωτικά μοντέλα έναντι των διαδικαστικών. Τα δηλωτικά μοντέλα έδειξαν να έχουν μεγαλύτερη εκφραστικότητα, επειδή μπορούν να αναπαραστήσουν πολλές χρονικές σχέσεις μεταξύ δύο γεγονότων, σε αντίθεση με τα δίκτυα Petri που είχαν αδυναμίες στη μοντελοποίηση της διεργασίας του συγκεκριμένου παραδείγματος, που φαίνεται να

οφείλεται στο γεγονός ότι η διεργασία δεν ολοκληρώνεται πάντα με το ίδιο γεγονός, και αυτό εξαρτάται από το γεγονός που θα ακολουθήσει το γεγονός *Order*.

2.6.2 Διεργασία για επισκέψεις πελατών σε μπαρ

Το δεύτερο παράδειγμα αφορά επισκέψεις πελατών σε μπαρ. Όπως μπορούμε να δούμε στο Σχήμα 2.13, κατά τη διεργασία μιας επίσκεψης σε μπαρ, όταν ο πελάτης φτάσει στο χώρο μπορεί ακολούθως να φύγει χωρίς να παραγγείλει τίποτα, ή κατ' επανάληψη να παραγγείλει ότι επιθυμεί και ακολούθως να πληρώσει και να φύγει. Επίσης μπορεί, αφού έχει φύγει, να ξαναπάει στο μπαρ και να ακολουθήσει τα επόμενα βήματα όπως πρίν, ή να μην ξαναπάει.



Σχήμα 2.13 Διεργασία επίσκεψης σε μπαρ

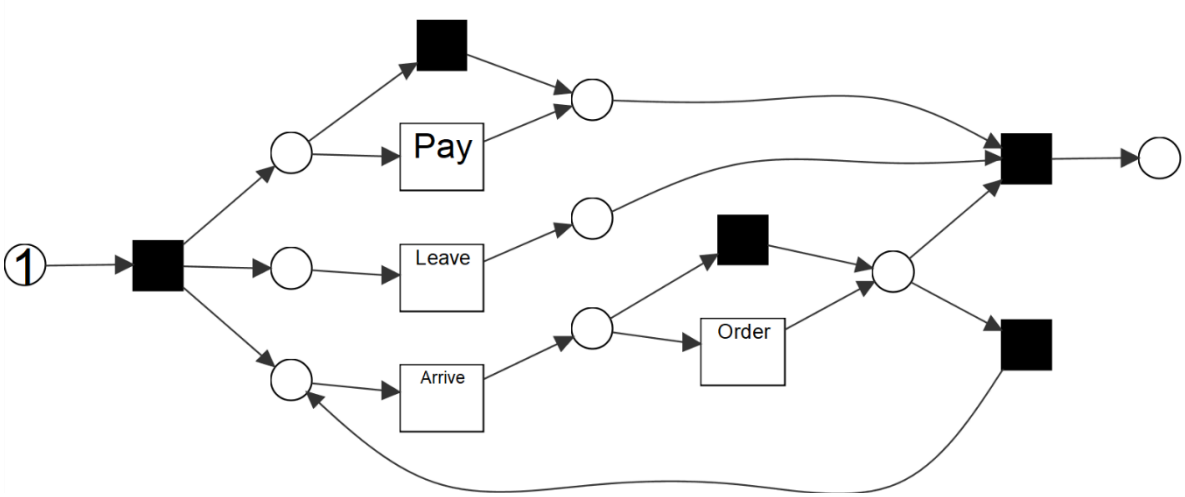
case	*event*	*completeTime*
1	Arrive	01/11/2021 22:08
1	Leave	01/11/2021 22:10
2	Arrive	01/11/2021 22:11
2	Order	01/11/2021 22:20
2	Pay	01/11/2021 23:00
2	Leave	01/11/2021 23:01
3	Arrive	01/11/2021 22:15
3	Order	01/11/2021 22:21
3	Order	01/11/2021 22:40
3	Pay	01/11/2021 23:10

3	Leave	01/11/2021 23:11
4	Arrive	01/11/2021 22:22
4	Order	01/11/2021 22:30
4	Order	01/11/2021 22:35
4	Order	01/11/2021 22:39
4	Pay	01/11/2021 23:31
4	Leave	01/11/2021 23:32
4	Arrive	01/11/2021 23:35
4	Order	01/11/2021 23:47
4	Pay	01/11/2021 23:58
4	Leave	01/11/2021 23:59

Πίνακας 2.8 Τεχνητό αρχείο καταγραφής συμβάντων για το τη διεργασία επίσκεψης σε μπαρ

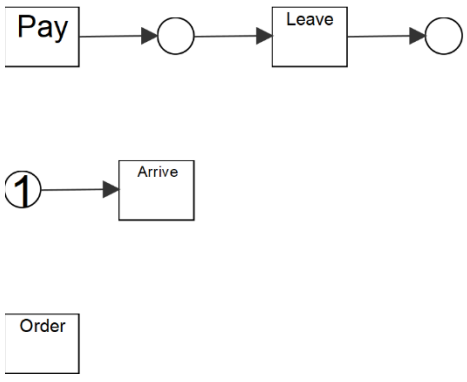
Αρχικά, έχω δημιουργήσει ένα αρχείο csv με τα περιεχόμενα που φαίνονται στον Πίνακα 2.8. Η μετατροπή του σε αρχείο καταγραφής συμβάντων έγινε με το plugin *Convert CSV to XES*, όπως περιεγράφηκε στην ενότητα 2.5.1.

Το Petri net στο Σχήμα 2.14 που παράχθηκε με το ProM plugin *Mine Petri net with Inductive Miner* βάσει του αρχείου καταγραφής, δεν είναι αντιπροσωπευτικό για την διεργασία που περιεγράφηκε πιο πάνω. Το μοντέλο αυτό επιτρέπει στιγμιότυπο που δεν περιέχει πληρωμή αλλά περιέχει παραγγελία, και αντίστροφα, κάτι που δεν είναι επιθυμητό. Επίσης, δεν επιτρέπονται πολλαπλές παραγγελίες αν δεν μεσολαβήσει άφιξη, και δεν καθορίζεται η χρονική σειρά που επιτρέπεται να συμβούν τα γεγονότα.



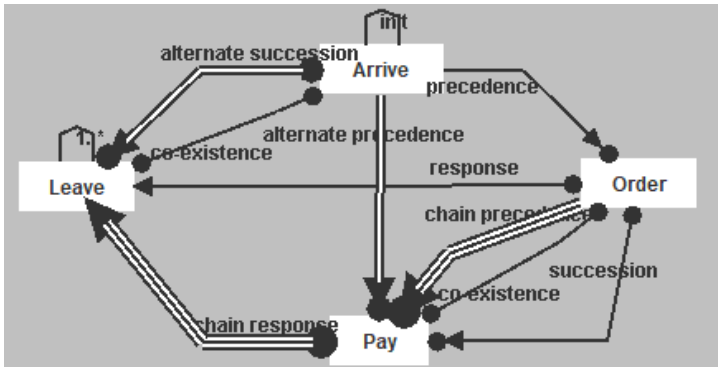
Σχήμα 2.14 Petri net για διεργασία επισκέψεων σε μπαρ από *Inductive Miner*

Στο Σχήμα 2.15 βλέπουμε το Petri net που παράχθηκε με το ProM plugin *Alpha Miner* βάσει του αρχείου καταγραφής, που επίσης δεν είναι αντιπροσωπευτικό για την διεργασία που περιεγράφηκε πιο πάνω. Βλέπουμε ότι ορθά το πρώτο γεγονός στο μοντέλο είναι η άφιξη του πελάτη, και τα δύο τελευταία είναι η πληρωμή και η αποχώρηση του πελάτη από το μπαρ, όμως δεν ήταν εφικτή η ένωση αυτών των τμημάτων του μοντέλου και του γεγονότος της παραγγελίας.



Σχήμα 2.15 Petri net για διεργασία επισκέψεων σε μπαρ από *Alpha Miner*

Το δηλωτικό μοντέλο στο Σχήμα 2.16 που παράχθηκε με το ProM plugin *MINERful Declare Miner* φαίνεται να είναι ορθό βάσει της περιγραφής της διεργασίας που δόθηκε πιο πάνω, και η εξήγηση δίνεται στον Πίνακα 2.9 για κάθε υπαρξιακή και σχεσιακή δήλωση ξεχωριστά.



Σχήμα 2.16 Δηλωτικό μοντέλο για διεργασία επίσκεψης σε μπαρ από το ProM

DECLARE	**LTL Σημασιολογία**	**Εξήγηση**
init (Arrive)	Arrive	Κάθε στιγμιότυπο της διεργασίας ξεκινά με την άφιξη στον χώρο
existence (1, Leave)	◊Leave	Κάθε στιγμιότυπο της διεργασίας περιέχει αποχώρηση από το χώρο
co-existence (Arrive, Leave)	◊Arrive ⇔ ◊Leave	Όταν γίνεται άφιξη γίνεται και αποχώρηση, και αντίστροφα
alternate succession (Arrive, Leave)	□(Arrive ⇒ O(¬Arrive U Leave)) ^ ((¬Leave U Arrive) ν □(¬ Leave))) ^ □(Leave ⇒ O((¬Leave U Arrive) ν □(¬ Leave))) **απόδειξη κάτω από τον πίνακα*	Όταν γίνει άφιξη θα ακολουθήσει αποχώρηση, και αποχώρηση γίνεται αν έχει προηγηθεί άφιξη, χωρίς την μεσολάβηση άλλης άφιξης ή αποχώρησης και στις δύο προϋποθέσεις
precedence (Arrive, Order)	(¬Order U Arrive) ν □(¬ Order)	Η παραγγελία γίνεται αν έχει προηγηθεί άφιξη

co-existence (Order, Pay)	◊Order ⇔ ◊Pay	Όταν γίνεται παραγγελία γίνεται και πληρωμή, και αντίστροφα
chain precedence (Order, Pay)	□(O Pay ⇒ Order)	Η πληρωμή γίνεται αν το ακριβώς προηγούμενο βήμα ήταν η παραγγελία
succession (Order, Pay)	□(Order ⇒ ◊Pay) ^ (¬Pay U Order) ν □(¬ Pay) **απόδειξη κάτω από τον πίνακα*	Όταν γίνει παραγγελία θα ακολουθήσει πληρωμή, και πληρωμή γίνεται αν έχει προηγηθεί παραγγελία
alternate precedence (Arrive, Pay)	((¬Pay U Arrive) ν □(¬ Pay)) ^ □(Pay ⇒ O((¬Pay U Arrive) ν □(¬ Pay))) **απόδειξη κάτω από τον πίνακα*	Η πληρωμή γίνεται αν έχει προηγηθεί άφιξη, χωρίς την μεσολάβηση άλλης άφιξης ή πληρωμής
response (Order, Leave)	□(Order ⇒ ◊Leave)	Όταν γίνει παραγγελία, θα γίνει μελλοντικά αποχώρηση
chain response (Pay, Leave)	□(Pay ⇒ O Leave)	Όταν γίνει πληρωμή, το αμέσως επόμενο βήμα είναι η αποχώρηση

<u>* Απόδειξη 3</u>

alternate succession (Arrive, Leave)

= alternate response (Arrive, Leave) ^ alternate precedence (Arrive, Leave)

= □(Arrive ⇒ O(¬Arrive U Leave)) ^ precedence(Arrive, Leave) ^ □(Leave ⇒ O(precedence(Arrive, Leave)))

= □(Arrive ⇒ O(¬Arrive U Leave)) ^ ((¬Leave U Arrive) ν □(¬ Leave))) ^ □(Leave ⇒ O((¬Leave U Arrive) ν □(¬ Leave)))

<u>* Απόδειξη 4</u>

succession (Order, Pay)

= response (Order, Pay) ^ precedence (Order, Pay)

= □(Order ⇒ ◊Pay) ^ (¬Pay U Order) ν □(¬ Pay)

<u>* Απόδειξη 5</u>

alternate precedence (Arrive, Pay)

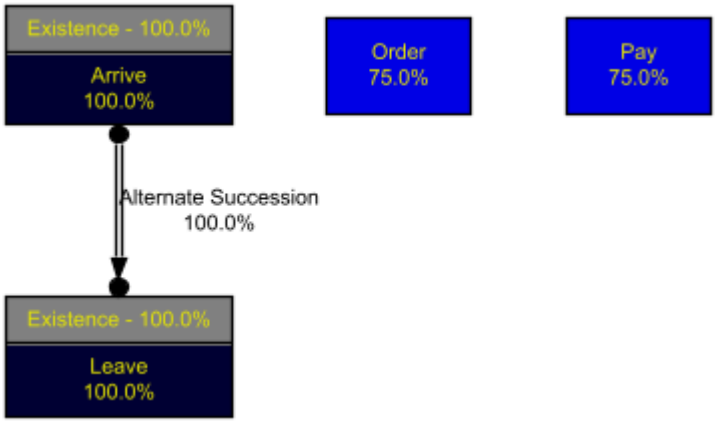
= precedence (Arrive, Pay) ^ □(Pay ⇒ O(precedence(Arrive, Pay)))

= ((¬Pay U Arrive) ν □(¬ Pay)) ^ □(Pay ⇒ O((¬Pay U Arrive) ν □(¬ Pay)))

Πίνακας 2.9 Μετάφραση σε LTL για διεργασία επίσκεψης σε μπαρ

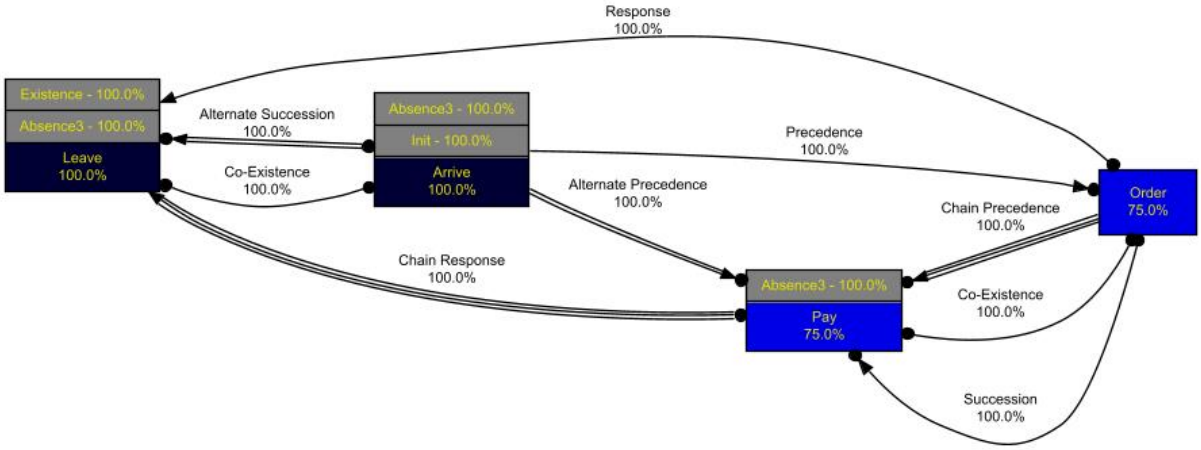
Για την μετάφραση κάθε υπαρξιακής και σχεσιακής δήλωσης σε σημασιολογία LTL, το ProM plugin *Declare to LTL* με είσοδο το δηλωτικό μοντέλο δεν ανταποκρίθηκε με κάποιο αποτέλεσμα.

Στη συνέχεια θα δούμε τη διαδικασία ανακάλυψης δηλωτικού μοντέλου με το εργαλείο RuM. Το δηλωτικό μοντέλο στο Σχήμα 2.17 που παράχθηκε με την προκαθορισμένη μέθοδο ανακάλυψης Declare Miner του RuM δεν είναι αντιπροσωπευτικό για την διεργασία παραγγελιών σε μπαρ. Έχει ανακαλυφθεί ορθά η ύπαρξη των γεγονότων Arrive και Leave και η σχέση alternate succession μεταξύ τους, όμως δεν έχει ανακαλυφθεί καμιά σχέση των γεγονότων Order και Pay με τα υπόλοιπα γεγονότα.



Σχήμα 2.17 Δηλωτικό μοντέλο για διεργασία επίσκεψης σε μπαρ από το RuM με τη μέθοδο Declare Miner

Το δηλωτικό μοντέλο στο Σχήμα 2.18 που παράχθηκε στο RuM αλλάζοντας την μέθοδο ανακάλυψης (Discovery method) από Declare Miner σε MINERful, είναι αντίστοιχο με το μοντέλο που παράχθηκε με το ProM plugin MINERful Declare Miner για την διεργασία παραγγελιών σε μπαρ.



Σχήμα 2.18 Δηλωτικό μοντέλο για διεργασία επίσκεψης σε μπαρ από το RuM με τη μέθοδο MINERful

Κεφάλαιο 3

Ευθυγράμμιση Διεργασιών

3.1 Ευθυγράμμιση διεργασιών σε PDDL 50
3.2 Ευθυγράμμιση διεργασιών σε Clingo 61

3.1 Ευθυγράμμιση Διεργασιών σε PDDL

3.1.1 Εισαγωγή

Σε αυτήν την ενότητα θα μελετήσουμε την τεχνική ελέγχου συμμόρφωσης που βασίζεται σε ευθυγράμμιση η οποία περιγράφεται στο άρθρο [2], όπου αφαιρείται η υπόθεση τέλειας γνώσης της σειράς εκτέλεσης των συμβάντων, δηλαδή το αρχείο καταγραφής συμβάντων περιέχει μερικά συμβάντα που συνέβησαν την ίδια χρονική στιγμή. Η τεχνική μετατρέπει το πρόβλημα του ελέγχου συμμόρφωσης βάσει ευθυγράμμισης σε πρόβλημα προγραμματισμού κωδικοποιημένο σε PDDL, για το οποίο μπορεί να βρεθεί μια σωστή λύση σε πεπερασμένο χρονικό διάστημα για προβλήματα μικρού μεγέθους.

3.1.2 Περιγραφή υλοποίησης

Η μοντελοποίηση ενός προβλήματος σε PDDL αποτελείται από δύο αρχεία, το domain και το problem. Θα ξεκινήσουμε με σχολιασμό του αρχείου domain της υλοποίησης του προβλήματος της ευθυγράμμισης στην γλώσσα PDDL, που βρίσκεται στο Παράρτημα Α.

```
(:types transition place event num)
```

Η εντολή αυτή δηλώνει τους τύπους που θα χρησιμοποιηθούν για την κωδικοποίηση, οι οποίοι είναι οι

- place, για τις θέσεις του δικτύου Petri
- transition, για τις μεταβάσεις του δικτύου Petri
- event, για τα συμβάντα του αρχείου καταγραφής και
- num, για τις διακριτές χρονικές στιγμές που συνέβησαν τα γεγονότα

```
(:predicates
    (token ?p - place)
    (aligned ?e - event)
    (ptarc ?p ?t)
    (tparc ?t ?p)
    (event_time ?e - event ?t - num)
    (smaller ?t1 ?t2 - num)
    (associated ?t - transition ?e - event)
)
```

Εδώ δηλώνονται τα boolean predicates που θα χρησιμοποιηθούν, τα οποία είναι

- (token ?p - place), αληθές αν το p περιέχει μία μάρκα την τρέχουσα στιγμή
- (aligned ?e - event), αληθές αν το e έχει ευθυγραμμιστεί μέχρι την τρέχουσα στιγμή
- (ptarc ?p - place ?t - transition), αληθές αν υπάρχει τόξο από το p στο t
- (tparc ? t - transition ?p - place), αληθές αν υπάρχει τόξο από το t στο p
- (event_time ?e - event ?t - num), αληθές αν τη χρονική στιγμή t συνέβη το e
- (smaller ?t1 ?t2 - num), αληθές αν η χρονική στιγμή t1 προηγείται της t2
- (associated ?t - transition ?e - event), αληθές αν το t στο μοντέλο αντιστοιχεί με το e στο αρχείο καταγραφής

```
(:functions
    (move-model-cost ?t - transition)
    (move-log-cost ?e - event)
    (total-cost)
```

)

Επίσης θα χρειαστούν τρεις συναρτήσεις *(functions)* για να μετρήσουμε το κόστος της ευθυγράμμισης

- (move-model-cost ?t - transition), για το κόστος κάθε κίνησης στο μοντέλο
- (move-log-cost ?e - event), για το κόστος κάθε κίνησης στο αρχείο καταγραφής
- (total-cost), για το συνολικό κόστος της ευθυγράμμισης

Στο αρχείο προβλήματος που θα δούμε αργότερα, θα ορίσουμε την αρχική και τελική κατάσταση. Σκοπός είναι να φτάσουμε από την αρχική στην τελική κατάσταση με το λιγότερο συνολικό κόστος. Για να γίνει αυτό εισάγουμε ως δράσεις τα τρία είδη κινήσεων για ευθυγράμμιση.

```
(:action moveSync
   :parameters
      (?t - transition ?e - event ?t2 - num)
      :precondition
      (and
            (forall (?pa - place)
              (imply (ptarc ?pa ?t)
                (token ?pa)
            )
         )
                (associated ?t ?e)
                (event_time ?e ?t2)
                (not (aligned ?e))
            (forall (?ea - event)
                 (forall (?t1 - num)
            (imply (and (event_time ?ea ?t1) (smaller ?t1 ?t2))
                (aligned ?ea)
            )
                 )
         )
```

```
)
```

Αρχίζουμε με τη δράση moveSync για τη σύγχρονη κίνηση πάνω στη μετάβαση t και το συμβάν e. Για να γίνει η κίνηση, πρέπει να τηρούνται οι προϋποθέσεις *(preconditions)* της δράσης. Η μετάβαση t πρέπει να είναι ενεργοποιημένη, δηλαδή να υπάρχουν μάρκες σε όλες τις θέσεις εισόδου της t (το *(imply (ptarc ?pa ?t)* ερμηνεύεται ως 'αν υπάρχει τόξο από το pa στο t'). Όλα τα συμβάντα που έγιναν σε χρόνο μικρότερο από τον χρόνο που έγινε το e που συνδέεται με το t πρέπει να είναι ευθυγραμμισμένα, ενώ το e όχι.

```
        :effect
        (and
                (forall (?pa - place)
            (when (ptarc ?pa ?t)
                (not (token ?pa))
            )
        )
        (forall (?pb - place)
            (when (tparc ?t ?pb)
                (token ?pb)
            )
        )
        (aligned ?e)
    )
)
```

Αποτέλεσμα *(effect)* της σύγχρονης κίνησης είναι να παραχθεί μία μάρκα σε κάθε θέση εξόδου του t και να ευθυγραμμιστεί το αντίστοιχό του συμβάν e.

```
(:action moveInTheModel
    :parameters
        (?t - transition)
        :precondition
            (and
```

```
        (forall (?pa - place)
        (imply (ptarc ?pa ?t)
            (token ?pa)
        )
    )
    )
```

Η δράση moveInTheModel αντιπροσωπεύει την κίνηση στο μοντέλο για τη μετάβαση t. Για να γίνει η κίνηση πρέπει η t να είναι ενεργοποιημένη (όπως περιεγράφηκε και στη σύγχρονη κίνηση).

```
    :effect
        (and
        (forall (?pa - place)
            (when (ptarc ?pa ?t)
                (not (token ?pa))
            )
        )
        (forall (?pb - place)
            (when (tparc ?t ?pb)
                (token ?pb)
            )
        )
        (increase (total-cost) (move-model-cost ?t))
        )
)
```

Αποτέλεσμα της κίνησης στο μοντέλο είναι να παραχθεί μία μάρκα σε κάθε θέση εξόδου του t και να αυξηθεί το συνολικό κόστος ανάλογα με το (move-model-cost t) που καθορίζεται στο αρχείο προβλήματος.

```
(:action moveInTheLog
    :parameters
```

```
(?e - event ?t2 - num)
:precondition
   (and
        (event_time ?e ?t2)
        (not (aligned ?e))
      (forall (?ea - event)
          (forall (?t1 - num)
      (imply (and (event_time ?ea ?t1) (smaller ?t1 ?t2))
        (aligned ?ea)
      )
           )
   )
   )
```

Η δράση moveInTheLog αντιπροσωπεύει την κίνηση στο αρχείο καταγραφής για το συμβάν e. Για να γίνει κίνηση στο αρχείο καταγραφής βάσει ενός event e πρέπει όλα τα event που έγιναν σε χρόνο μικρότερο από τον χρόνο που έγινε το e να είναι ευθυγραμμισμένα.

```
:effect
   (and
      (aligned ?e)
      (increase (total-cost) (move-log-cost ?e))
)
  )
)
```

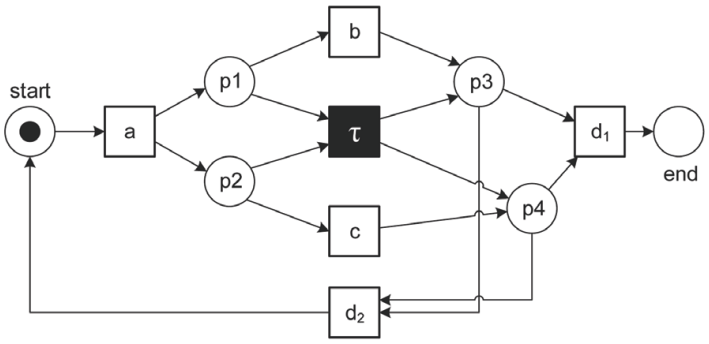
Τέλος, αποτέλεσμα της κίνησης στο αρχείο είναι να ευθυγραμμιστεί το event e και να αυξηθεί το συνολικό κόστος ανάλογα με το (move-log-cost e) που καθορίζεται στο αρχείο προβλήματος.

3.1.3 Αρχεία προβλημάτων και αποτελέσματα

Σε αυτή την ενότητα θα δούμε δύο αρχεία προβλημάτων ευθυγράμμισης διεργασιών στην γλώσσα PDDL και τα αποτελέσματα που δίνουν.

3.1.3.1 Πρόβλημα μικρού μεγέθους

Στην πρώτη κωδικοποίηση προβλήματος σε PDDL θα γίνει ευθυγράμμιση του δικτύου Petri που φαίνεται στο Σχήμα 3.1. με το εξής στιγμιότυπο αρχείου καταγραφής: $\varphi = \langle\{a, b\}, \{d, c\}\rangle$, όπου το συμβάν a αντιστοιχεί με την μετάβαση a, το b με την b, το c με την c, και το d με τις d1 και d2. Από το φ εξάγουμε ότι τα συμβάντα a και b έγιναν την χρονική στιγμή 1 και τα συμβάντα d και c τη χρονική στιγμή 2.



Σχήμα 3.1 Δίκτυο Petri για ευθυγράμμιση

```
(:objects
        a b c d1 d2 inv - transition
        st p1 p2 p3 p4 en - place
        e_a e_b e_c e_d  - event
        1 2 - num
)
```

Αρχικά δηλώνουμε τα αντικείμενα του δικτύου Petri και του στιγμιότυπου φ. Τα a, b, c d1, d2, και inv αποτελούν τις μεταβάσεις του δικτύου, τα st, p1, p2, p3, p4, και en αποτελούν τις θέσεις του δικτύου, τα e_a, e_b, e_c, και e_d αποτελούν τα συμβάντα του

στιγμιότυπου φ και 1, 2 είναι όλες οι διαφορετικές χρονικές στιγμές που εκτελέστηκαν τα συμβάντα του φ.

```
(:init
    (token st) (= (total-cost) 0)
    (= (move-model-cost a) 1) (= (move-model-cost b) 1)
    (= (move-model-cost c) 1) (= (move-model-cost d1) 1)
    (= (move-model-cost d2) 1) (= (move-model-cost inv) 0)
    (= (move-log-cost e_a) 1) (= (move-log-cost e_b) 1)
    (= (move-log-cost e_c) 1) (= (move-log-cost e_d) 1)
    (ptarc st a) (tparc a p1) (tparc a p2)
    (ptarc p1 b) (tparc b p3)
    (ptarc p1 inv) (ptarc p2 inv) (tparc inv p3) (tparc inv p4)
    (ptarc p2 c) (tparc c p4)
    (ptarc p3 d1) (ptarc p4 d1) (tparc d1 en)
    (ptarc p3 d2) (ptarc p4 d2) (tparc d2 st)
    (event_time e_a 1) (event_time e_b 1) (event_time e_c 2) (event_time e_d
2)
    (smaller 1 2)
    (associated a e_a) (associated b e_b) (associated c e_c)
    (associated d1 e_d) (associated d2 e_d)
  )
```

Εδώ βλέπουμε τις συνθήκες που ισχύουν κατά την αρχική κατάσταση (init). Πρέπει να δηλώσουμε την αρχική σήμανση του δικτύου, δηλαδή σε ποιες θέσεις υπάρχουν μάρκες αρχικά, να αρχικοποιήσουμε το συνολικό κόστος με 0 και το κόστος για τις σύγχρονες κινήσεις με 0 και για τις υπόλοιπες με 1. Επίσης, πρέπει να δείξουμε τη δομή του μοντέλου δηλώνοντας τα τόξα από τις μεταβάσεις στις θέσεις του δικτύου και αντίστροφα, πότε συνέβη κάθε συμβάν, τη σχέση μεταξύ των χρονικών στιγμών, και τις συνδέσεις μεταξύ των μεταβάσεων και των συμβάντων.

```
(:goal
    (and
```

```
            (token en) (not (token st)) (not (token p1))
            (not (token p2)) (not (token p3)) (not (token p4))
            (aligned e_a) (aligned e_b) (aligned e_c) (aligned e_d)
        )
    )
```

Στο goal πρέπει να δηλώσουμε την τελική κατάσταση, δηλαδή την τελική σήμανση του δικτύου (που πρέπει να υπάρχουν μάρκες μετά το πέρας της ευθυγράμμισης και που όχι), και ότι όλα τα συμβάντα του στιγμιότυπου πρέπει να ευθυγραμμιστούν.

```
    (:metric
        minimize (total-cost)
    )
```

Τέλος, με την μετρική minimize δηλώνουμε ότι θέλουμε να πάρουμε ως αποτέλεσμα τις δράσεις που θα μας οδηγήσουν από την αρχική στην τελική κατάσταση (δηλαδή θα ευθυγραμμίσουν το μοντέλο με το στιγμιότυπο φ) με το μικρότερο κόστος (δηλαδή προτιμούμε τις σύγχρονες κινήσεις, αφού έχουν κόστος 0).

Έστω ότι aligning-domain.pddl είναι το αρχείο που περιέχει την PDDL κωδικοποίηση που βρίσκεται στο Παράρτημα Α, και aligning-problem.pddl είναι το αρχείο που περιέχει το πρόβλημα που βρίσκεται στο Παράρτημα Β. Θα εκτελέσουμε το πρόγραμμα χρησιμοποιώντας τον planner FAST-DOWNWARD [18], τρέχοντας την εντολή ./fast-downward.py --alias seq-sat-lama-2011 --search-time-limit 60 aligning-domain.pddl aligning-problem.pddl.

Ο planner έδωσε ως αποτέλεσμα τέσσερις διαφορετικές λύσεις.

```
(moveinthemodel a)
(moveinthemodel c)
(movesync b e_b 1)
(moveinthemodel d1)
(moveinthelog e_a 1)
```

(moveinthelog e_c 2)
(moveinthelog e_d 2)
; cost = 6 (general cost)

Αυτή είναι η πρώτη λύση, η οποία αποτελείται από 7 κινήσεις. Αρχικά γίνεται κίνηση στο μοντέλο στην μετάβαση a, άρα παράγονται μάρκες στις θέσεις p1 και p2. Λόγω της μάρκας στο p2 μπορεί να γίνει και κίνηση στο μοντέλο στη μετάβαση c, άρα παράγεται μάρκα στη θέση p4. Μετά επιλέγεται να γίνει σύγχρονη κίνηση μεταξύ της μετάβασης b και του γεγονότος e_b. Αυτό είναι εφικτό επειδή υπάρχει μάρκα στη θέση p1 και δεν υπάρχουν γεγονότα στο στιγμιότυπο που να έχουν συμβεί πριν από το e_b ώστε να απαιτήσουμε να ήταν ευθυγραμμισμένα, έτσι παράγεται μία μάρκα στη θέση p3 και ευθυγραμμίζεται το συμβάν e_b. Έχοντας τώρα μάρκες στις θέσεις p3 και p4 είναι δυνατή η κίνηση στο μοντέλο στη μετάβαση d_1, παράγοντας έτσι μάρκα στην τελική θέση. Όμως για να ολοκληρωθεί η ευθυγράμμιση πρέπει να ευθυγραμμιστούν όλα τα συμβάντα του στιγμιότυπου. Επομένως με την κίνηση στο αρχείο στο e_a ευθυγραμμίζεται το συμβάν a, και αφού τώρα έχουν ευθυγραμμιστεί όλα τα συμβάντα που έγιναν σε χρόνο 1, μπορούν να ευθυγραμμιστούν και αυτά που έγιναν σε χρόνο 2 με τις δύο τελευταίες κινήσεις στο αρχείο. Αυτή η λύση περιλαμβάνει 3 κινήσεις στο μοντέλο και 3 στο αρχείο με κόστος 1 για την κάθε μία, και μία σύγχρονη κίνηση με μηδενικό κόστος, άρα έχει συνολικό κόστος 6.

(moveinthemodel a)
(moveinthemodel c)
(movesync b e_b 1)
(moveinthelog e_a 1)
(movesync d1 e_d 2)
(moveinthelog e_c 2)
; cost = 4 (general cost)

(moveinthemodel a)
(moveinthelog e_a 1)
(movesync b e_b 1)
(movesync c e_c 2)
(movesync d1 e_d 2)

; cost = 2 (general cost)

(movesync a e_a 1)
(movesync b e_b 1)
(movesync c e_c 2)
(movesync d1 e_d 2)
; cost = 0 (general cost)

Αυτές είναι οι υπόλοιπες τρεις λύσεις που έδωσε ο planner μέχρι να φτάσει στη βέλτιστη λύση που σε αυτή την περίπτωση έχει κόστος 0, αφού το μοντέλο και το στιγμιότυπο φ είναι σύμφωνα μεταξύ τους.

3.1.3.2 Πρόβλημα μεγαλύτερου μεγέθους

Αφού είδαμε ότι η κωδικοποίηση σε PDDL δίνει σωστά αποτελέσματα σε μικρά προβλήματα, θα θέλαμε τώρα να δούμε την απόδοση του προγράμματος σε μεγαλύτερα προβλήματα. Στη σελίδα [19] δίνονται τα αρχεία μοντέλων και αρχείων καταγραφής σε μορφή *pnml* και *xes* αντίστοιχα, που χρησιμοποιήθηκαν στο άρθρο [2] και είχαν παραχθεί με το εργαλείο PGL2 [20].

Με τη δημιουργία ενός προγράμματος στη γλώσσα *Python* έχω μετατρέψει τα αρχεία αυτά στη μορφή που πρέπει να έχει ένα αρχείο προβλήματος, όπως την είδαμε στην ενότητα 3.1.3.1. Το πρόγραμμα αυτό κάνει χρήση των βιβλιοθηκών *pm4py.objects.petri_net.importer* και *pm4py.objects.log.importer.xes* που παρέχουν συναρτήσεις για αναπαράσταση του μοντέλου και του αρχείου καταγραφής σε ειδικές δομές δεδομένων. Στη συνέχεια το πρόγραμμα δημιουργεί ένα αρχείο στο οποίο τυπώνονται τα δεδομένα αυτά, προσθέτοντας ενδιάμεσα την απαραίτητη κωδικοποίηση που απαιτεί η PDDL. Επίσης δημιουργεί και ένα αρχείο για την Clingo κωδικοποίηση που θα δούμε στην ενότητα 3.2.3.2.

Στο Παράρτημα Γ δίνεται το αρχείο προβλήματος για ευθυγράμμιση του μικρότερου μοντέλου d53.pnml που περιέχει 95 μεταβάσεις και 88 θέσεις, με το πρώτο στιγμιότυπο από τα 1,000 του αρχείου d53_PO_avg10_rad1.xes που περιέχει 27 συμβάντα.

Δυστυχώς, η PDDL κωδικοποίηση δεν μπορεί να τρέξει με είσοδο αρχείο προβλήματος μεγάλου μεγέθους με τον planner *FAST-DOWNWARD* λόγω του χρόνου εκτέλεσης. Ο planner *SymBA*-2* [21] είναι καταλληλότερος για αυτή την περίπτωση, όμως λόγω της απουσίας κάποιων βιβλιοθηκών δεν μπορεί επίσης να χρησιμοποιηθεί.

3.2 Ευθυγράμμιση Διεργασιών σε Clingo

3.2.1 Εισαγωγή

Θα θέλαμε να συγκρίνουμε την κωδικοποίηση σε PDDL που αναφέρθηκε στην προηγούμενη ενότητα με την κωδικοποίηση σε γλώσσα προγραμματισμού Clingo. Η Clingo είναι μια λογική γλώσσα προγραμματισμού συνόλου απαντήσεων *(Answer Set Programming)*. Αποτελεί μια απλή και ισχυρή γλώσσα μοντελοποίησης για την περιγραφή προβλημάτων ως λογικά προγράμματα, και τον υπολογισμό συνόλων απαντήσεων που αντιπροσωπεύουν λύσεις στο δεδομένο πρόβλημα.

Η μέθοδος με την οποία θα υλοποιηθεί το πρόγραμμα σε Clingo είναι η παρόμοια με αυτήν που περιεγράφηκε στο 3.1. Έχει γίνει όμως μια βελτιστοποίηση η οποία δίνει τη δυνατότητα να υπάρχουν περισσότερες από μία μάρκες στις θέσεις του δικτύου. Επίσης, έγινε προσθήκη ενός περιορισμού για το χρόνο μέσα στον οποίο θέλουμε να γίνουν οι κινήσεις της ευθυγράμμισης, π.χ. για time(0..3), οι κινήσεις πρέπει να γίνουν σε 4 το πολύ χρονικές στιγμές.

3.2.2 Περιγραφή υλοποίησης

Σε αυτήν την ενότητα θα σχολιάσουμε την υλοποίηση του προβλήματος της ευθυγράμμισης στην γλώσσα προγραμματισμού Clingo, που βρίσκεται στο Παράρτημα Δ.

```
:- place(P), #max { TS : time(TS) } = M, holds(P, Q, M+1), Q != 0, ptarc(P, _).
:- place(P), #max { TS : time(TS) } = M, holds(P, Q, M+1), Q != 1, not ptarc(P,
_).
```

```
:- #count { E : event(E) } = CE, #count { E,TS : aligned(E,TS) } = CA, CE !=
CA.
```

Το holds(P, Q, TS) είναι αληθές αν την χρονική στιγμή TS υπάρχουν Q token στο place P. Άρα με αυτές τις εντολές ορίζουμε ότι με το τέλος της κωδικοποίησης, δεν μπορούν να υπάρχουν θέσεις που δεν είναι τελικές και να περιέχουν μάρκα, ούτε να υπάρχουν τελικές θέσεις που δεν περιέχουν μία μάρκα. Επίσης, με την 3η εντολή ορίζουμε ότι όλα τα συμβάντα πρέπει να γίνουν align, μία φορά το κάθε ένα.

```
notenabled(T,TS):-ptarc(P,T),holds(P,Q,TS),Q<=0,place(P),trans(T),time(TS).
enabled(T,TS):-trans(T),time(TS),not notenabled(T,TS).
```

Εδώ ορίζουμε ως μη ενεργοποιημένη μια μετάβαση που έχει έστω και μια θέση εισόδου χωρίς μάρκα, και ως ενεργοποιημένη μια που δεν είναι μη ενεργοποιημένη.

```
already_aligned(E,TS):-event(E),aligned(E,TS2),time(TS),time(TS2),TS2<=TS.
notprevaligned(E,TS):-not
already_aligned(EP,TS),event_time(E,T),event_time(EP,TP),event(E),event(EP)
,TP<T,time(TS).
prevaligned(E,TS):-event(E),not notprevaligned(E,TS),not
already_aligned(E,TS),time(TS).
```

Εδώ ορίζουμε αν τα προηγούμενα συμβάντα ενός συμβάντος είναι ήδη ευθυγραμμισμένα (prevaligned), ελέγχοντας ότι κανένα από τα προηγούμενα αυτά συμβάντα είναι μη-ευθυγραμμισμένο.

```
{model_move(T,TS)}1:-enabled(T,TS).
{trace_move(E,TS)}1:-prevaligned(E,TS).
{sync_move(E,T,TS)}1:-enabled(T,TS),prevaligned(E,TS),associated(E,T).
```

Η κίνηση στο μοντέλο μπορεί να συμβεί αν η μετάβαση είναι ενεργοποιημένη. Η κίνηση στο στιγμιότυπο μπορεί να συμβεί αν όλα τα προηγούμενα συμβάντα είναι ευθυγραμμισμένα. Η σύγχρονη κίνηση μπορεί να συμβεί αν η μετάβαση είναι

ενεργοποιημένη και όλα τα προηγούμενα συμβάντα του συμβάντος που σχετίζεται με την T είναι ευθυγραμμισμένα.

```
:-sync_move(_,_,TS),model_move(_,TS).
:-sync_move(_,_,TS),trace_move(_,TS).
:-model_move(_,TS),trace_move(_,TS).
:-
model_move(T1,TS),model_move(T2,TS),ptarc(P,T1),ptarc(P,T2),place(P),T1!
=T2.
:-
sync_move(_,T1,TS),sync_move(_,T2,TS),ptarc(P,T1),ptarc(P,T2),place(P),T1!
=T2.
```

Με αυτές τις εντολές αποτρέπουμε τις ταυτόχρονες κινήσεις διαφορετικών τύπων, όπως και τις ταυτόχρονες κινήσεις στο μοντέλο (ή σύγχρονες) αν αφορούν μεταβάσεις που έχουν κοινή θέση εισόδου.

```
#minimize { C,T,TS : model_move(T, TS), move_model_cost(T, C) }.
#minimize { C,E,TS : trace_move(E, TS), move_log_cost(E, C) }.
```

Με τη χρήση του minimize ελαχιστοποιούμε το κόστος, δηλαδή τις κινήσεις στο μοντέλο και στο στιγμιότυπο.

```
aligned(E,TS+1):-trace_move(E,TS).
```

Μετά την κίνηση στο στιγμιότυπο, το αντίστοιχο συμβάν σημειώνεται ως ευθυγραμμισμένο.

```
consumed(P,TS):-model_move(T,TS),ptarc(P,T),place(P).
produced(P,TS):-model_move(T,TS),tparc(T,P),place(P).
```

Μετά την κίνηση στο μοντέλο στη μετάβαση T, σημειώνουμε τις θέσεις εισόδου του T ως consumed και τις θέσεις εξόδου ως produced.

```
consumed(P,TS):-sync_move(E,T,TS),ptarc(P,T),place(P).
produced(P,TS):-sync_move(E,T,TS),tparc(T,P),place(P).
aligned(E,TS+1):-sync_move(E,T,TS).
```

Η σύγχρονη κίνηση συνδυάζει τα αποτελέσματα της κίνησης στο στιγμιότυπο και της κίνησης στο μοντέλο.

```
holds(P,Q-1,TS+1):-consumed(P,TS),holds(P,Q,TS).
holds(P,Q+1,TS+1):-produced(P,TS),holds(P,Q,TS).
holds(P,Q,TS+1):-holds(P,Q,TS),not consumed(P,TS),not
produced(P,TS),place(P),time(TS).
```

Από όσες θέσεις είναι σημειωμένες ως consumed πρέπει να αφαιρεθεί μία μάρκα, και σε όσες είναι σημειωμένες ως produced να προστεθεί μία. Οι θέσεις που δεν είναι σημειωμένες ούτε με consumed ούτε με produced διατηρούν τον αριθμό των μαρκών που περιείχαν και για την επόμενη χρονική στιγμή.

3.2.3 Αρχεία προβλημάτων και αποτελέσματα

3.2.3.1 Πρόβλημα μικρού μεγέθους

Έστω ότι aligning.lp είναι το αρχείο που περιέχει την Clingo κωδικοποίηση που βρίσκεται στο Παράρτημα Δ, και net_small.lp είναι το αρχείο που περιέχει το πρόβλημα που βρίσκεται στο Παράρτημα Ε, ίδιο με αυτό που αναφέρθηκε στην ενότητα 3.1.3.1 αλλά κωδικοποιημένο σε Clingo αντί σε PDDL, με την προσθήκη ενός περιορισμού για το χρόνο μέσα στον οποίο θέλουμε να γίνουν οι κινήσεις της ευθυγράμμισης (time(0..3)).

Θα εκτελέσουμε το πρόγραμμα χρησιμοποιώντας το Anaconda Prompt (miniconda3), τρέχοντας την εντολή clingo aligning.lp 0 net_small.lp, που θα μας δώσει όλες τις πιθανές λύσεις και το κόστος τους. Η λύση που παίρνουμε φαίνεται στο Σχήμα 3.2 και είναι η ίδια με τη βέλτιστη λύση που δίνει το πρόγραμμα σε PDDL.

```
clingo version 5.5.1
Reading from aligning.lp ...
Solving...
Answer: 1
sync_move(e_a,a,0) sync_move(e_b,b,1) sync_move(e_c,c,2) sync_move(e_d,d1,3)
Optimization: 0
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 0
Calls        : 1
Time         : 0.015s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.016s
```

Σχήμα 3.2 Λύση για μικρό πρόβλημα σε Clingo

3.2.3.2 Πρόβλημα μεγαλύτερου μεγέθους

Ένα πρόβλημα που αντιμετωπίσαμε αρχικά στην κωδικοποίηση στη Clingo ήταν ο χρόνος εκτέλεσης που χρειαζόταν για να τερματίσει βρίσκοντας βέλτιστη λύση, στην περίπτωση που κάθε χρονική στιγμή επιτρέπεται μία μόνο κίνηση, όπου δεν ήταν ο αναμενόμενος. Ως εκ τούτου, επιτρέπονται ταυτόχρονες κινήσεις αν είναι ίδιου τύπου και δεν παραβιάζουν κάποιον άλλο κανόνα.

Με το πρόγραμμα στη γλώσσα *Python* που αναφέρθηκε επίσης στην ενότητα 3.1.3.2, έχω μετατρέψει τα αρχεία d53.pnml και το πρώτο στιγμιότυπο του d53_PO_avg10_rad1.xes και στη Clingo μορφή που πρέπει να έχει το αρχείο δεδομένων, και βρίσκεται στο Παράρτημα ΣΤ. Η μεταβλητή time(0..26) επιλέχθηκε να παίρνει τιμές μέχρι το 26 επειδή ο αριθμός των συμβάντων στο στιγμιότυπο είναι 27 και είναι μικρότερος από τον αριθμό των μεταβάσεων (95).

Η λύσεις που παίρνουμε φαίνονται στο Σχήμα 3.3. Ο χρόνος εκτέλεσης για αυτό το πρόβλημα ευθυγράμμισης είναι κατά μέσο όρο 1,5 δευτερόλεπτα.

Μπορούμε να δούμε ότι όντως δεν συμβαίνουν ταυτόχρονες κινήσεις διαφορετικού τύπου, ότι όλα τα γεγονότα ευθυγραμμίζονται, και οι σύγχρονες κινήσεις γίνονται μεταξύ μεταβάσεων και γεγονότων που σχετίζονται.

```
clingo version 5.5.1
Reading from aligning.lp ...
Solving...
Answer: 1
trace_move(e9,25) trace_move(e10,25) trace_move(e11,25) trace_move(e12,25) trace_move(e13,25) trace_move
(e14,25) trace_move(e15,25) trace_move(e16,25) trace_move(e17,25) trace_move(e18,25) trace_move(e19,26)
trace_move(e20,26) trace_move(e21,26) trace_move(e22,26) trace_move(e23,26) trace_move(e24,26) trace_mov
e(e25,26) trace_move(e26,26) sync_move(e0,a,24) sync_move(e1,af,24) sync_move(e2,ah,24) sync_move(e3,ai,
24) sync_move(e4,aj,24) sync_move(e5,ak,24) sync_move(e6,al,24) sync_move(e7,am,24) sync_move(e8,ao,24)
Optimization: 18
Answer: 2
trace_move(e0,24) trace_move(e1,24) trace_move(e2,24) trace_move(e3,24) trace_move(e4,24) trace_move(e5,
24) trace_move(e6,24) trace_move(e7,24) trace_move(e8,24) trace_move(e19,26) trace_move(e20,26) trace_mo
ve(e21,26) trace_move(e22,26) trace_move(e23,26) trace_move(e24,26) trace_move(e25,26) trace_move(e26,26
) sync_move(e9,an,25) sync_move(e10,ah,25) sync_move(e11,ai,25) sync_move(e12,aj,25) sync_move(e13,ak,25
) sync_move(e14,al,25) sync_move(e15,am,25) sync_move(e16,ao,25) sync_move(e17,an,25) sync_move(e18,ah,2
5)
Optimization: 17
Answer: 3
trace_move(e19,26) trace_move(e20,26) trace_move(e21,26) trace_move(e22,26) trace_move(e23,26) trace_mov
e(e24,26) trace_move(e25,26) trace_move(e26,26) sync_move(e0,a,24) sync_move(e1,af,24) sync_move(e2,ah,2
4) sync_move(e3,ai,24) sync_move(e4,aj,24) sync_move(e5,ak,24) sync_move(e6,al,24) sync_move(e7,am,24) s
ync_move(e8,ao,24) sync_move(e9,an,25) sync_move(e10,ah,25) sync_move(e11,ai,25) sync_move(e12,aj,25) sy
nc_move(e13,ak,25) sync_move(e14,al,25) sync_move(e15,am,25) sync_move(e16,ao,25) sync_move(e17,an,25) s
ync_move(e18,ah,25)
Optimization: 8
Answer: 4
sync_move(e0,a,24) sync_move(e1,af,24) sync_move(e2,ah,24) sync_move(e3,ai,24) sync_move(e4,aj,24) sync_
move(e5,ak,24) sync_move(e6,al,24) sync_move(e7,am,24) sync_move(e8,ao,24) sync_move(e9,an,25) sync_move
(e10,ah,25) sync_move(e11,ai,25) sync_move(e12,aj,25) sync_move(e13,ak,25) sync_move(e14,al,25) sync_mov
e(e15,am,25) sync_move(e16,ao,25) sync_move(e17,an,25) sync_move(e18,ah,25) sync_move(e19,ai,26) sync_mo
ve(e20,aj,26) sync_move(e21,ak,26) sync_move(e22,ag,26) sync_move(e23,b,26) sync_move(e24,aq,26) sync_mo
ve(e25,bu,26) sync_move(e26,ar,26)
Optimization: 0
OPTIMUM FOUND

Models       : 4
  Optimum    : yes
Optimization : 0
Calls        : 1
Time         : 1.537s (Solving: 0.20s 1st Model: 0.16s Unsat: 0.02s)
CPU Time     : 1.438s
```

Σχήμα 3.3 Λύση για μεγάλο πρόβλημα σε Clingo

Κεφάλαιο 4

Συμπεράσματα

4.1 Δηλωτικά και διαδικαστικά μοντέλα 67
4.2 Ευθυγράμμιση διεργασιών σε PDDL και Clingo 67

4.1 Δηλωτικά και διαδικαστικά μοντέλα

Από τα παραδείγματα ανακάληψης διαδικαστικών (δίκτυα Petri) και δηλωτικών μοντέλων (μοντέλα DECLARE) διεργασιών στο Κεφάλαιο 2, συμπεραίνουμε ότι τα δηλωτικά μοντέλα είναι πιο εκφραστικά σε σχέση με τα διαδικαστικά. Τα δηλωτικά μοντέλα μπορούν να αναπαραστήσουν πολλές χρονικές σχέσεις μεταξύ των γεγονότων, κρατώντας παράλληλα και την επίσημη σημασιολογία της Γραμμικής Χρονικής Λογικής. Η δημιουργία δικτύων Petri διέπετε από πιο αυστηρούς κανόνες, που σε κάποια παραδείγματα διεργασιών απέτρεψαν την εύρεση ενός αντιπροσωπευτικού μοντέλου για τη διεργασία, καθώς επίσης δεν δίνουν μεγάλες δυνατότητες στο χρήστη για προσανατολισμό της ανακάλυψης.

4.2 Ευθυγράμμιση Διεργασιών σε PDDL και Clingo

Στην εργασία αυτή δείξαμε ότι τόσο η γλώσσα PDDL όσο και το σύστημα Clingo έχουν την δυνατότητα να μοντελοποιήσουν προβλήματα ευθυγράμμισης διεργασιών. Η πειραματική μας αξιολόγηση όμως σε μεγάλα προβλήματα κατέδειξε ότι πιθανόν η

αποδοτικότητα τους σε χρόνους επίλυσης να μην είναι αυτή που χρειάζεται σε πρακτικές εφαρμογές. Για το λόγο αυτό, μελέτη των κωδικοποιήσεων αυτών χρειάζεται να επεκταθεί περαιτέρω.

Βιβλιογραφία

[1] W. v. d. Alst, Process Mining, Springer, 2016.

[2] M. de Leoni, G. Lanciano και A. Marella, «Aligning Partially-Ordered Process-Execution Traces and Models Using Automated Planning,» 2018.

[3] P. Dixit, A. Corvò, H. S. G. Caballero και B. Hompes, «Enabling Interactive Process Analysis with Process Mining and Visual Analytics,» 2017.

[4] M. Ben-Ari, Mathematical Logic for Computer Science, 3rd επιμ., Springer, 2012.

[5] M. Huth και M. Ryan, Logic in Computer Science, New York: Cambridge University Press, 2004.

[6] P. Haslum, N. Lipovetzky, D. Magazzeni και C. Muise, An Introduction to the Planning Domain Definition Language, Morgan & Claypool Publishers, 2019.

[7] «Requirements (PDDL 1.2),» [Ηλεκτρονικό]. Available: https://planning.wiki/ref/pddl/requirements.

[8] «Potassco, the Potsdam Answer Set Solving Collection,» [Ηλεκτρονικό]. Available: https://potassco.org/.

[9] M. Pesic και W. V. d. Alst, «A Declarative Approach for Flexible Business Processes Management,» 2006.

[10] M. Pesic, H. Schonenberg και W. V. d. Aalst, «DECLARE: Full Support for Loosely-Structured Processes».

[11] F. M. Maggi, A. J. Mooij και W. M. P. v. d. Aalst, «User-Guided Discovery of Declarative Process Models,» 2011.

[12] M. Pesic, Constraint-Based Workflow Management Systems: Shifting Controls to users, University Press Facilities, Eindhoven, 2008.

[13] «ProM,» [Ηλεκτρονικό]. Available: https://www.promtools.org/doku.php?id=prom610 .

[14] «Introduction to Process Mining with ProM,» Eindhoven University of Technology, Future Learn, [Ηλεκτρονικό]. Available: https://www.futurelearn.com/courses/process-mining.

[15] C. D. Ciccio, J. Mendling και M. d. Leoni, «Declarative process discovery with MINERful in ProM,» 2015.

[16] «RuM,» [Ηλεκτρονικό]. Available: https://rulemining.org/ .

[17] «RuM Tutorial,» ICPM, 2020. [Ηλεκτρονικό]. Available: https://rulemining.org/wp-content/uploads/2020/08/RuM_tutorial_ICPM2020.pdf .

[18] «Fast Downward,» [Ηλεκτρονικό]. Available: https://www.fast-downward.org/QuickStart

[19] [Ηλεκτρονικό]. Available: https://data.4tu.nl/articles/dataset/Models_and_Logs_used_in_the_paper_Aligning_Partially-Ordered_Process-Execution_Traces_and_Models_Using_Automated_Planning_accepted_for_ICAPS_2018/12709085/1.

[20] A. Burattin, «PLG2: Multiperspective Process Randomization with Online and Offline Simulations,» 2017

[21] P. Kissmann και S. Edelkamp, «SymBA∗:A Symbolic Bidirectional A∗ Planner,» 2014.

Παράρτημα Α

Σε αυτό το παράρτημα βρίσκεται το domain αρχείο της PDDL κωδικοποίησης για την ευθυγράμμιση διεργασιών με την μέθοδο που αναφέρεται στο Κεφάλαιο 3.

```
(define (domain aligning-domain)

   (:requirements :adl :action-costs)

   (:types transition place event num)

   (:predicates
         (token ?p - place)
         (aligned ?e - event)
         (ptarc ?p ?t)
         (tparc ?t ?p)
         (event_time ?e - event ?t - num)
         (smaller ?t1 ?t2 - num)
         (associated ?t - transition ?e - event)
   )

   (:functions
         (move-model-cost ?t - transition)
         (move-log-cost ?e - event)
         (total-cost)
   )

   (:action moveSync
```

```
:parameters
   (?t - transition ?e - event ?t2 - num)
   :precondition
   (and
         (forall (?pa - place)
         (imply (ptarc ?pa ?t)
            (token ?pa)
         )
      )
            (associated ?t ?e)
            (event_time ?e ?t2)
            (not (aligned ?e))
         (forall (?ea - event)
             (forall (?t1 - num)
         (imply (and (event_time ?ea ?t1) (smaller ?t1 ?t2))
            (aligned ?ea)
         )
              )
      )
      )
   :effect
   (and
            (forall (?pa - place)
         (when (ptarc ?pa ?t)
            (not (token ?pa))
         )
      )
      (forall (?pb - place)
         (when (tparc ?t ?pb)
            (token ?pb)
         )
      )
      (aligned ?e)
```

```
        )
)

(:action moveInTheModel
    :parameters
        (?t - transition)
        :precondition
            (and
                (forall (?pa - place)
                (imply (ptarc ?pa ?t)
                    (token ?pa)
                )
            )
            )
        :effect
            (and
            (forall (?pa - place)
                (when (ptarc ?pa ?t)
                    (not (token ?pa))
                )
            )
            (forall (?pb - place)
                (when (tparc ?t ?pb)
                    (token ?pb)
                )
            )
            (increase (total-cost) (move-model-cost ?t))
            )
)

(:action moveInTheLog
    :parameters
        (?e - event ?t2 - num)
```

```
        :precondition
           (and
                (event_time ?e ?t2)
                (not (aligned ?e))
              (forall (?ea - event)
                 (forall (?t1 - num)
              (imply (and (event_time ?ea ?t1) (smaller ?t1 ?t2))
                (aligned ?ea)
              )
                  )
           )
           )
        :effect
           (and
              (aligned ?e)
              (increase (total-cost) (move-log-cost ?e))
        )
)
```

Παράρτημα Β

Σε αυτό το παράρτημα βρίσκεται ένα μικρό problem αρχείο της PDDL κωδικοποίησης για την ευθυγράμμιση διεργασιών με την μέθοδο που αναφέρεται στο Κεφάλαιο 3.

```
(define (problem aligning-problem)

   (:domain aligning-domain)

   (:objects
         a b c d1 d2 inv - transition
         st p1 p2 p3 p4 en - place
         e_a e_b e_c e_d  - event
         1 2 - num
   )

   (:init
         (token st) (= (total-cost) 0)
         (= (move-model-cost a) 1) (= (move-model-cost b) 1)
         (= (move-model-cost c) 1) (= (move-model-cost d1) 1)
         (= (move-model-cost d2) 1) (= (move-model-cost inv) 0)
         (= (move-log-cost e_a) 1) (= (move-log-cost e_b) 1)
         (= (move-log-cost e_c) 1) (= (move-log-cost e_d) 1)
         (ptarc st a) (tparc a p1) (tparc a p2)
         (ptarc p1 b) (tparc b p3)
         (ptarc p1 inv) (ptarc p2 inv) (tparc inv p3) (tparc inv p4)
         (ptarc p2 c) (tparc c p4)
         (ptarc p3 d1) (ptarc p4 d1) (tparc d1 en)
```

```
        (ptarc p3 d2) (ptarc p4 d2) (tparc d2 st)
        (event_time e_a 1) (event_time e_b 1) (event_time e_c 2) (event_time e_d 2)
        (smaller 1 2)
        (associated a e_a) (associated b e_b) (associated c e_c)
        (associated d1 e_d) (associated d2 e_d)
    )

    (:goal
        (and
            (token en) (not (token st)) (not (token p1))
            (not (token p2)) (not (token p3)) (not (token p4))
            (aligned e_a) (aligned e_b) (aligned e_c) (aligned e_d)
        )
    )

    (:metric
        minimize (total-cost)
    )

)
```

Παράρτημα Γ

Σε αυτό το παράρτημα βρίσκεται ένα μεγάλο problem αρχείο της PDDL κωδικοποίησης για την ευθυγράμμιση διεργασιών με την μέθοδο που αναφέρεται στο Κεφάλαιο 3.

(define (problem aligning-problem)

(:domain aligning-domain)

(:objects

BX E BB H 352 BS K A BE 365 364 T BT 362 BN AW AY BK J AM M BZ BG O F I G AK BU 440 Y W BI AX S D AU BH BY AF AP BL X CB BM BR R BA AI AT BD BO 368 BF N CA AB C 358 AL 372 AE AR BV U AJ Z 359 AQ 363 CC 373 AV BP BW AA AZ V B Q L P AD AS AN 369 AC AG BJ BQ 353 441 AH BC AO - transition

399 356 354 357 395 418 435 408 385 397 366 398 384 407 361 388 390 344 412 416 346 355 382 349 429 409 342 380 422 343 414 377 391 371 375 403 436 381 438 437 400 420 432 405 260 374 426 439 423 419 427 347 431 383 410 386 345 433 401 413 434 348 392 411 360 389 367 430 415 428 406 387 259 402 404 376 393 350 425 351 394 421 379 396 378 424 417 370 - place

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 - event

1 2 3 - num

)

(:init

(token 259)

(= (total-cost) 0)

(= (move-model-cost BX) 1) (= (move-model-cost E) 1) (= (move-model-cost BB) 1) (= (move-model-cost H) 1) (= (move-model-cost BS) 1)

(= (move-model-cost K) 1) (= (move-model-cost A) 1) (= (move-model-cost BE) 1) (= (move-model-cost T) 1) (= (move-model-cost BT) 1)

(= (move-model-cost BN) 1) (= (move-model-cost AW) 1) (= (move-model-cost AY) 1) (= (move-model-cost BK) 1) (= (move-model-cost J) 1)

(= (move-model-cost AM) 1) (= (move-model-cost M) 1) (= (move-model-cost BZ) 1) (= (move-model-cost BG) 1) (= (move-model-cost O) 1)

(= (move-model-cost F) 1) (= (move-model-cost I) 1) (= (move-model-cost G) 1) (= (move-model-cost AK) 1) (= (move-model-cost BU) 1)

(= (move-model-cost Y) 1) (= (move-model-cost W) 1) (= (move-model-cost BI) 1) (= (move-model-cost AX) 1) (= (move-model-cost S) 1)

(= (move-model-cost D) 1) (= (move-model-cost AU) 1) (= (move-model-cost BH) 1) (= (move-model-cost BY) 1) (= (move-model-cost AF) 1)

(= (move-model-cost AP) 1) (= (move-model-cost BL) 1) (= (move-model-cost X) 1) (= (move-model-cost CB) 1) (= (move-model-cost BM) 1)

(= (move-model-cost BR) 1) (= (move-model-cost R) 1) (= (move-model-cost BA) 1) (= (move-model-cost AI) 1) (= (move-model-cost AT) 1)

(= (move-model-cost BD) 1) (= (move-model-cost BO) 1) (= (move-model-cost BF) 1) (= (move-model-cost N) 1) (= (move-model-cost CA) 1)

(= (move-model-cost AB) 1) (= (move-model-cost C) 1) (= (move-model-cost AL) 1) (= (move-model-cost AE) 1) (= (move-model-cost AR) 1)

(= (move-model-cost BV) 1) (= (move-model-cost U) 1) (= (move-model-cost AJ) 1) (= (move-model-cost Z) 1) (= (move-model-cost AQ) 1)

(= (move-model-cost CC) 1) (= (move-model-cost AV) 1) (= (move-model-cost BP) 1) (= (move-model-cost BW) 1) (= (move-model-cost AA) 1)

(= (move-model-cost AZ) 1) (= (move-model-cost V) 1) (= (move-model-cost B) 1) (= (move-model-cost Q) 1) (= (move-model-cost L) 1)

(= (move-model-cost P) 1) (= (move-model-cost AD) 1) (= (move-model-cost AS) 1) (= (move-model-cost AN) 1) (= (move-model-cost AC) 1)

(= (move-model-cost AG) 1) (= (move-model-cost BJ) 1) (= (move-model-cost BQ) 1) (= (move-model-cost AH) 1) (= (move-model-cost BC) 1)

(= (move-model-cost AO) 1)

(= (move-log-cost 0) 1) (= (move-log-cost 1) 1) (= (move-log-cost 2) 1) (= (move-log-cost 3) 1) (= (move-log-cost 4) 1)

(= (move-log-cost 5) 1) (= (move-log-cost 6) 1) (= (move-log-cost 7) 1) (= (move-log-cost 8) 1) (= (move-log-cost 9) 1)

(= (move-log-cost 10) 1) (= (move-log-cost 11) 1) (= (move-log-cost 12) 1) (= (move-log-cost 13) 1) (= (move-log-cost 14) 1)

(= (move-log-cost 15) 1) (= (move-log-cost 16) 1) (= (move-log-cost 17) 1) (= (move-log-cost 18) 1) (= (move-log-cost 19) 1)

(= (move-log-cost 20) 1) (= (move-log-cost 21) 1) (= (move-log-cost 22) 1) (= (move-log-cost 23) 1) (= (move-log-cost 24) 1)

(= (move-log-cost 25) 1) (= (move-log-cost 26) 1)

(tparc BU 367) (ptarc 351 AN) (tparc H 420) (tparc AC 437) (tparc AJ 377) (tparc 364 385) (ptarc 366 BV) (tparc AY 342) (ptarc 416 L) (ptarc 349 BE)

(ptarc 345 440) (tparc AA 433) (tparc AZ 375) (tparc 362 434) (ptarc 437 363) (ptarc 411 E) (tparc CB 404) (tparc 364 383) (ptarc 422 H) (ptarc 413 373)

(tparc W 356) (tparc BV 399) (tparc T 431) (tparc BM 390) (tparc 372 414) (tparc BJ 397) (tparc 440 344) (ptarc 370 AU) (ptarc 384 365) (ptarc 378 AJ)

(tparc BE 375) (ptarc 436 AC) (ptarc 379 AM) (tparc BO 384) (tparc CA 402) (ptarc 434 AB) (ptarc 428 F) (tparc 352 401) (ptarc 421 I) (ptarc 350 AO)

(tparc B 409) (ptarc 396 359) (ptarc 417 373) (ptarc 381 BN) (tparc 368 422) (ptarc 401 CA) (ptarc 374 BD) (ptarc 438 362) (ptarc 392 BQ) (ptarc 414 K)

(ptarc 399 BW) (tparc Q 354) (tparc BI 349) (ptarc 419 373) (ptarc 355 R) (tparc BQ 391) (tparc V 355) (ptarc 398 BK) (ptarc 356 Y) (tparc E 428)

(tparc BP 386) (ptarc 423 369) (ptarc 374 AW) (ptarc 361 B) (tparc BC 343) (ptarc 425 369) (ptarc 366 BU) (ptarc 393 359) (ptarc 357 X) (tparc BB 343)

(tparc AV 371) (ptarc 356 AE) (tparc R 361) (tparc AS 370) (tparc L 417) (tparc BG 349) (tparc K 415) (ptarc 348 BH) (ptarc 406 BZ) (tparc M 419)

(tparc G 361) (ptarc 348 BG) (ptarc 389 BL) (tparc BA 343) (ptarc 388 BM) (tparc 353 406) (ptarc 408 BY) (tparc 365 388) (tparc AG 361) (ptarc 342 BC)

(tparc BR 393) (ptarc 410 D) (ptarc 350 441) (tparc AT 367) (tparc BY 405) (tparc N 344) (ptarc 360 AF) (tparc AU 374) (ptarc 420 372) (tparc AL 379)

(tparc D 411) (tparc AP 351) (ptarc 345 O) (tparc AH 376) (tparc 364 381) (tparc AW 380) (tparc BZ 407) (ptarc 435 363) (tparc 363 439) (ptarc 375 AV)

(tparc AN 346) (ptarc 259 A) (tparc BN 382) (tparc 362 432) (ptarc 429 T) (tparc A 360) (tparc 358 392) (tparc AM 350) (tparc P 345) (tparc BX 408)

(ptarc 380 AX) (ptarc 405 352) (ptarc 371 AT) (ptarc 383 BO) (tparc 441 351) (ptarc 407 CC) (tparc 372 418) (ptarc 371 BJ) (ptarc 382 365) (tparc AK 347)

(tparc AQ 366) (ptarc 418 M) (ptarc 409 AQ) (tparc BT 396) (tparc 358 389) (tparc 358 395) (ptarc 374 AY) (ptarc 430 V) (ptarc 348 BI) (tparc X 354)

(ptarc 350 AP) (ptarc 439 Z) (tparc AI 378) (ptarc 427 G) (tparc AB 435) (ptarc 400 BX) (ptarc 355 W) (tparc 372 416) (ptarc 360 Q) (tparc S 429)

(ptarc 390 359) (ptarc 394 BT) (tparc 369 427) (ptarc 431 U) (ptarc 354 S) (ptarc 424 N) (ptarc 387 364) (tparc AX 375) (tparc BF 349) (ptarc 360 C)

(tparc 362 436) (ptarc 426 368) (ptarc 433 363) (tparc AR 260) (tparc 373 421) (ptarc 356 AD) (tparc BW 400) (ptarc 432 AA) (tparc U 430) (ptarc 395 BS)

(ptarc 403 CB) (tparc Y 438) (tparc F 426) (ptarc 412 J) (tparc AD 357) (ptarc 367 AR) (tparc C 410) (tparc O 425) (tparc 372 412) (ptarc 347 AG)

(ptarc 377 AK) (tparc AO 351) (ptarc 402 353) (tparc CC 367) (tparc J 413) (tparc I 423) (ptarc 366 AS) (tparc BL 387) (tparc AE 357) (tparc 359 398)

(ptarc 376 AI) (ptarc 386 365) (ptarc 391 BR) (tparc BD 348) (ptarc 348 BF) (tparc BH 349) (ptarc 346 AH) (ptarc 415 373) (ptarc 344 P) (tparc BK 370)

(ptarc 385 BP) (ptarc 347 AL) (ptarc 342 BA) (tparc AF 346) (tparc BS 394) (ptarc 342 BB) (tparc 352 403) (ptarc 397 358) (ptarc 404 353) (tparc 368 424)

(ptarc 343 AZ) (tparc Z 357)

(event_time 0,1) (event_time 1,1) (event_time 2,1) (event_time 3,1) (event_time 4,1) (event_time 5,1) (event_time 6,1) (event_time 7,1) (event_time 8,1)

(event_time 9,2) (event_time 10,2) (event_time 11,2) (event_time 12,2) (event_time 13,2) (event_time 14,2) (event_time 15,2) (event_time 16,2) (event_time 17,2) (event_time 18,2)

(event_time 19,3) (event_time 20,3) (event_time 21,3) (event_time 22,3) (event_time 23,3) (event_time 24,3) (event_time 25,3) (event_time 26,3)

(smaller 1,2) (smaller 2,3) (smaller 3,4)

(associated 0,A) (associated 1,AF) (associated 2,AH) (associated 3,AI) (associated 4,AJ)

(associated 5,AK) (associated 6,AL) (associated 7,AM) (associated 8,AO) (associated 9,AN)

(associated 10,AH) (associated 11,AI) (associated 12,AJ) (associated 13,AK) (associated 14,AL)

(associated 15,AM) (associated 16,AO) (associated 17,AN) (associated 18,AH) (associated 19,AI)

(associated 20,AJ) (associated 21,AK) (associated 22,AG) (associated 23,B) (associated 24,AQ)

(associated 25,BU) (associated 26,AR)

)

(:goal

(and

(not (token 399)) (not (token 356)) (not (token 354)) (not (token 357)) (not (token 395)) (not (token 418)) (not (token 435)) (not (token 408)) (not (token 385)) (not (token 397))

(not (token 366)) (not (token 398)) (not (token 384)) (not (token 407)) (not (token 361)) (not (token 388)) (not (token 390)) (not (token 344)) (not (token 412)) (not (token 416))

(not (token 346)) (not (token 355)) (not (token 382)) (not (token 349)) (not (token 429)) (not (token 409)) (not (token 342)) (not (token 380)) (not (token 422)) (not (token 343))

(not (token 414)) (not (token 377)) (not (token 391)) (not (token 371)) (not (token 375)) (not (token 403)) (not (token 436)) (not (token 381)) (not (token 438)) (not (token 437))

(not (token 400)) (not (token 420)) (not (token 432)) (not (token 405)) (token 260) (not (token 374)) (not (token 426)) (not (token 439)) (not (token 423)) (not (token 419))

(not (token 427)) (not (token 347)) (not (token 431)) (not (token 383)) (not (token 410)) (not (token 386)) (not (token 345)) (not (token 433)) (not (token 401)) (not (token 413))

(not (token 434)) (not (token 348)) (not (token 392)) (not (token 411)) (not (token 360)) (not (token 389)) (not (token 367)) (not (token 430)) (not (token 415)) (not (token 428))

(not (token 406)) (not (token 387)) (not (token 259)) (not (token 402)) (not (token 404)) (not (token 376)) (not (token 393)) (not (token 350)) (not (token 425)) (not (token 351))

(not (token 394)) (not (token 421)) (not (token 379)) (not (token 396)) (not (token 378)) (not (token 424)) (not (token 417)) (not (token 370))

(aligned 0) (aligned 1) (aligned 2) (aligned 3) (aligned 4) (aligned 5) (aligned 6) (aligned 7) (aligned 8) (aligned 9)

(aligned 10) (aligned 11) (aligned 12) (aligned 13) (aligned 14) (aligned 15) (aligned 16) (aligned 17) (aligned 18) (aligned 19)

(aligned 20) (aligned 21) (aligned 22) (aligned 23) (aligned 24) (aligned 25) (aligned 26)

)

)

(:metric

minimize (total-cost)

)

)

Παράρτημα Δ

Σε αυτό το παράρτημα βρίσκεται το αρχείο της Clingo κωδικοποίησης για την ευθυγράμμιση διεργασιών με την μέθοδο που αναφέρεται στο Κεφάλαιο 3.

```
%% Goals %%

% At the last timestamp end places have to hold 1 token
:- place(P), #max { TS : time(TS) } = M, holds(P, Q, M+1), Q != 0, ptarc(P, _).
:- place(P), #max { TS : time(TS) } = M, holds(P, Q, M+1), Q != 1, not ptarc(P, _).

% All events have to be aligned once
:- #count { E : event(E) } = CE, #count { E,TS : aligned(E,TS) } = CA, CE != CA.


%% Preconditions %%

% Transition is enabled when there is no input place not holding token
notenabled(T,TS):-ptarc(P,T),holds(P,Q,TS),Q<=0,place(P),trans(T),time(TS).
enabled(T,TS):-trans(T),time(TS),not notenabled(T,TS).

% An unaligned event's previous events are already aligned, when there is no event
occured before never aligned until current timestamp
already_aligned(E,TS):-event(E),aligned(E,TS2),time(TS),time(TS2),TS2<=TS.
notprevaligned(E,TS):-not
already_aligned(EP,TS),event_time(E,T),event_time(EP,TP),event(E),event(EP),TP<T,
time(TS).
```

```
prevaligned(E,TS):-event(E),not notprevaligned(E,TS),not
already_aligned(E,TS),time(TS).


%% Moves %%

% Model move may occured when a transition is enabled
{model_move(T,TS)}1:-enabled(T,TS).

% Trace move may occured when an event's previous events are already aligned
{trace_move(E,TS)}1:-prevaligned(E,TS).

% Sync move may occured when a transition is enabled and its associated event's
previous events are already aligned
{sync_move(E,T,TS)}1:-enabled(T,TS),prevaligned(E,TS),associated(E,T).

% Prevent simultaneous moves of different type
:-sync_move(_,_,TS),model_move(_,TS).
:-sync_move(_,_,TS),trace_move(_,TS).
:-model_move(_,TS),trace_move(_,TS).
% Prevent simultaneous fire of different transitions with common input place
:-model_move(T1,TS),model_move(T2,TS),ptarc(P,T1),ptarc(P,T2),place(P),T1!=T2.
:-sync_move(_,T1,TS),sync_move(_,T2,TS),ptarc(P,T1),ptarc(P,T2),place(P),T1!=T2.

% Minimize model and trace moves
#minimize { C,T,TS : model_move(T, TS), move_model_cost(T, C) }.
#minimize { C,E,TS : trace_move(E, TS), move_log_cost(E, C) }.


%% Effects %%

% After a trace move, the event is marked as aligned
aligned(E,TS+1):-trace_move(E,TS).
```

```
% After model move, a token is consumed in each input place and one is produced in each output place (transition fire)
consumed(P,TS):-model_move(T,TS),ptarc(P,T),place(P).
produced(P,TS):-model_move(T,TS),tparc(T,P),place(P).

% After a sync move, a token is consumed in each input place and one is produced in each output place and the associated event is marked as aligned
consumed(P,TS):-sync_move(E,T,TS),ptarc(P,T),place(P).
produced(P,TS):-sync_move(E,T,TS),tparc(T,P),place(P).
aligned(E,TS+1):-sync_move(E,T,TS).

holds(P,Q-1,TS+1):-consumed(P,TS),holds(P,Q,TS).
holds(P,Q+1,TS+1):-produced(P,TS),holds(P,Q,TS).
holds(P,Q,TS+1):-holds(P,Q,TS),not consumed(P,TS),not produced(P,TS),place(P),time(TS).


#show sync_move/3.
#show trace_move/2.
#show model_move/2.
```

Παράρτημα Ε

Σε αυτό το παράρτημα βρίσκεται ένα μικρό αρχείο δεδομένων της Clingo κωδικοποίησης για την ευθυγράμμιση διεργασιών με την μέθοδο που αναφέρεται στο Κεφάλαιο 3.

```
place(st;p1;p2;p3;p4;en).

holds(st,1,0).
holds(p1,0,0).
holds(p2,0,0).
holds(p3,0,0).
holds(p4,0,0).
holds(en,0,0).

trans(a;b;c;d1;d2;inv).
move_model_cost((a;b;c;d1;d2;inv),1).

ptarc(st,a).
tparc(a,(p1;p2)).
ptarc(p1,(b;inv)).
tparc(b,p3).
ptarc(p2,(c;inv)).
tparc(inv,(p3;p4)).
tparc(c,p4).
ptarc(p3,(d1;d2)).
ptarc(p4,(d1;d2)).
tparc(d1,en).
tparc(d2,st).
```

```
time(0..3).

event(e_a;e_b;e_c;e_d).
event_time((e_a;e_b),1).
event_time((e_c;e_d),2).
move_log_cost((e_a;e_b;e_c;e_d),1).

associated(e_a,a).
associated(e_b,b).
associated(e_c,c).
associated(e_d,(d1;d2)).
```

Παράρτημα ΣΤ

Σε αυτό το παράρτημα βρίσκεται ένα μεγάλο αρχείο δεδομένων της Clingo κωδικοποίησης για την ευθυγράμμιση διεργασιών με την μέθοδο που αναφέρεται στο Κεφάλαιο 3.

trans(BX;E;BB;H;3520;BS;K;A;BE;3650;3640;T;BT;3620;BN;AW;AY;BK;J;AM;M;BZ;BG;O;F;I;G;AK;BU;4400;Y;W;BI;AX;S;D;AU;BH;BY;AF;AP;BL;X;CB;BM;BR;R;BA;AI;AT;BD;BO;3680;BF;N;CA;AB;C;3580;AL;3720;AE;AR;BV;U;AJ;Z;3590;AQ;3630;CC;3730;AV;BP;BW;AA;AZ;V;B;Q;L;P;AD;AS;AN;3690;AC;AG;BJ;BQ;3530;4410;AH;BC;AO;).
place(399;356;354;357;395;418;435;408;385;397;366;398;384;407;361;388;390;344;412;416;346;355;382;349;429;409;342;380;422;343;414;377;391;371;375;403;436;381;438;437;400;420;432;405;260;374;426;439;423;419;427;347;431;383;410;386;345;433;401;413;434;348;392;411;360;389;367;430;415;428;406;387;259;402;404;376;393;350;425;351;394;421;379;396;378;424;417;370;).
event(0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23;24;25;26;).

time(0..26).

holds(259,0).
move_model_cost((BX;E;BB;H;BS;K;A;BE;T;BT;BN;AW;AY;BK;J;AM;M;BZ;BG;O;F;I;G;AK;BU;Y;W;BI;AX;S;D;AU;BH;BY;AF;AP;BL;X;CB;BM;BR;R;BA;AI;AT;BD;BO;BF;N;CA;AB;C;AL;AE;AR;BV;U;AJ;Z;AQ;CC;AV;BP;BW;AA;AZ;V;B;Q;L;P;AD;AS;AN;AC;AG;BJ;BQ;AH;BC;AO;),1).
move_log_cost((0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23;24;25;26;),1).

tparc(BU,367). ptarc(351,AN). tparc(H,420). tparc(AC,437). tparc(AJ,377). tparc(364,385). ptarc(366,BV). tparc(AY,342). ptarc(416,L). ptarc(349,BE). ptarc(345,440). tparc(AA,433). tparc(AZ,375). tparc(362,434). ptarc(437,363). ptarc(411,E). tparc(CB,404). tparc(364,383). ptarc(422,H). ptarc(413,373). tparc(W,356). tparc(BV,399). tparc(T,431). tparc(BM,390). tparc(372,414). tparc(BJ,397). tparc(440,344). ptarc(370,AU). ptarc(384,365). ptarc(378,AJ). tparc(BE,375). ptarc(436,AC). ptarc(379,AM). tparc(BO,384). tparc(CA,402). ptarc(434,AB). ptarc(428,F). tparc(352,401). ptarc(421,I). ptarc(350,AO). tparc(B,409). ptarc(396,359). ptarc(417,373). ptarc(381,BN). tparc(368,422). ptarc(401,CA). ptarc(374,BD). ptarc(438,362). ptarc(392,BQ). ptarc(414,K). ptarc(399,BW). tparc(Q,354). tparc(BI,349). ptarc(419,373). ptarc(355,R). tparc(BQ,391). tparc(V,355). ptarc(398,BK). ptarc(356,Y). tparc(E,428). tparc(BP,386). ptarc(423,369). ptarc(374,AW). ptarc(361,B). tparc(BC,343). ptarc(425,369). ptarc(366,BU). ptarc(393,359). ptarc(357,X). tparc(BB,343). tparc(AV,371). ptarc(356,AE). tparc(R,361). tparc(AS,370). tparc(L,417). tparc(BG,349). tparc(K,415). ptarc(348,BH). ptarc(406,BZ). tparc(M,419). tparc(G,361). ptarc(348,BG). ptarc(389,BL). tparc(BA,343). ptarc(388,BM). tparc(353,406). ptarc(408,BY). tparc(365,388). tparc(AG,361). ptarc(342,BC). tparc(BR,393). ptarc(410,D). ptarc(350,441). tparc(AT,367). tparc(BY,405). tparc(N,344). ptarc(360,AF). tparc(AU,374). ptarc(420,372). tparc(AL,379). tparc(D,411). tparc(AP,351). ptarc(345,O). tparc(AH,376). tparc(364,381). tparc(AW,380). tparc(BZ,407). ptarc(435,363). tparc(363,439). ptarc(375,AV). tparc(AN,346). ptarc(259,A). tparc(BN,382). tparc(362,432). ptarc(429,T). tparc(A,360). tparc(358,392). tparc(AM,350). tparc(P,345). tparc(BX,408). ptarc(380,AX). ptarc(405,352). ptarc(371,AT). ptarc(383,BO). tparc(441,351). ptarc(407,CC). tparc(372,418). ptarc(371,BJ). ptarc(382,365). tparc(AK,347). tparc(AQ,366). ptarc(418,M). ptarc(409,AQ). tparc(BT,396). tparc(358,389). tparc(358,395). ptarc(374,AY). ptarc(430,V). ptarc(348,BI). tparc(X,354). ptarc(350,AP). ptarc(439,Z). tparc(AI,378). ptarc(427,G). tparc(AB,435). ptarc(400,BX). ptarc(355,W). tparc(372,416). ptarc(360,Q). tparc(S,429). ptarc(390,359). ptarc(394,BT). tparc(369,427). ptarc(431,U). ptarc(354,S). ptarc(424,N). ptarc(387,364). tparc(AX,375). tparc(BF,349). ptarc(360,C).

tparc(362,436). ptarc(426,368). ptarc(433,363). tparc(AR,260). tparc(373,421).
ptarc(356,AD). tparc(BW,400). ptarc(432,AA). tparc(U,430). ptarc(395,BS).
ptarc(403,CB). tparc(Y,438). tparc(F,426). ptarc(412,J). tparc(AD,357). ptarc(367,AR).
tparc(C,410). tparc(O,425). tparc(372,412). ptarc(347,AG).
ptarc(377,AK). tparc(AO,351). ptarc(402,353). tparc(CC,367). tparc(J,413).
tparc(I,423). ptarc(366,AS). tparc(BL,387). tparc(AE,357). tparc(359,398).
ptarc(376,AI). ptarc(386,365). ptarc(391,BR). tparc(BD,348). ptarc(348,BF).
tparc(BH,349). ptarc(346,AH). ptarc(415,373). ptarc(344,P). tparc(BK,370).
ptarc(385,BP). ptarc(347,AL). ptarc(342,BA). tparc(AF,346). tparc(BS,394).
ptarc(342,BB). tparc(352,403). ptarc(397,358). ptarc(404,353). tparc(368,424).
ptarc(343,AZ). tparc(Z,357).

event_time(0,1). event_time(1,1). event_time(2,1). event_time(3,1). event_time(4,1).
event_time(5,1). event_time(6,1). event_time(7,1). event_time(8,1).
event_time(9,2). event_time(10,2). event_time(11,2). event_time(12,2).
event_time(13,2). event_time(14,2). event_time(15,2). event_time(16,2).
event_time(17,2). event_time(18,2).
event_time(19,3). event_time(20,3). event_time(21,3). event_time(22,3).
event_time(23,3). event_time(24,3). event_time(25,3). event_time(26,3).

associated(0,A). associated(1,AF). associated(2,AH). associated(3,AI).
associated(4,AJ).
associated(5,AK). associated(6,AL). associated(7,AM). associated(8,AO).
associated(9,AN).
associated(10,AH). associated(11,AI). associated(12,AJ). associated(13,AK).
associated(14,AL).
associated(15,AM). associated(16,AO). associated(17,AN). associated(18,AH).
associated(19,AI).
associated(20,AJ). associated(21,AK). associated(22,AG). associated(23,B).
associated(24,AQ).
associated(25,BU). associated(26,AR).