Ατομική Διπλωματική Εργασία

TERMINATION OF ONTOLOGICAL REASONING

Ανδρέας Νικολάου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Termination of Ontological Reasoning

Ανδρέας Νικολάου

Επιβλέπων Καθηγητής
Ανδρέας Πιερής

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

Μάιος 2024

Περίληψη

Η διπλωματική αυτή εργασία ασχολείται με την επίλυση του προβλήματος τερματισμού της διαδικασίας Semi-oblivious Chase για Simple Linear TGDs weak-acyclicity. Ένα Chase Procedure αποτελεί ένα σημαντικό εργαλείο για την επεξεργασία και την ανάλυση δεδομένων σε βάσεις δεδομένων, αλλά η δυνατότητα τερματισμού του είναι ένα κρίσιμο ζήτημα. Στόχος της εργασίας είναι η ανάπτυξη αλγόριθμου για τον εντοπισμό κρίσιμου κύκλου στον εξαρτημένο γράφο των TGDs, άρα να μπορεί να αποφασίσει αν το Chase Procedure θα τερματίσει ή όχι. Η εργασία παρουσιάζει μια λεπτομερή ανάλυση των χαρακτηριστικών των Simple Linear TGDs και του Weak-acyclicity. Η υλοποίηση του αλγορίθμου έγινε με τη χρήση της γλώσσας προγραμματισμού PHP, ενώ η πειραματική αξιολόγηση πραγματοποιήθηκε με τη χρήση ενός συνόλου δεδομένων TGDs. Τα αποτελέσματα των πειραμάτων έδειξαν ότι η υλοποίηση του αλγορίθμου έγινε σωστά και μπορεί να επεξεργαστεί τα δεδομένα γρήγορα, ακόμη και για μεγάλα σύνολα TGDs. Επιπλέον, τα αποτελέσματα επιβεβαίωσαν ότι η weak-acyclicity ιδιότητα είναι ένας σημαντικός παράγοντας για τον τερματισμό του Chase Procedure. Τέλος, η εργασία αυτή αναφέρεται σε μία επέκταση του αλγορίθμου σε μελλοντική εργασία, όπως η υλοποίηση του αλγορίθμου για Linear TGDs.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο και Στόχος Διπλωματικής Εργασίας

Το Chase Procedure [2] είναι ένα από τα πιο θεμελιώδη αλγοριθμικά εργαλεία στη θεωρία βάσεων δεδομένων. Έχοντας υπόψη το πρόβλημα του τερματισμού ενός Chase Pocedure που αναφέρεται στον έλεγχο αν, δεδομένου ενός συνόλου περιορισμών (TGDs) και μιας βάσης δεδομένων, η διαδικασία θα τερματίσει ή αν θα συνεχίσει επ' άπειρον. Συγκεκριμένα, αν η διαδικασία θα παράγει ένα πεπερασμένο αποτέλεσμα για κάθε πιθανή βάση δεδομένων.

Στόχος αυτής της Διπλωματικής Εργασίας είναι να υλοποιηθεί ο αλγόριθμος που ελέγχει τον τερματισμό ενός Chase Procedure για Simple Linear TGDs με την ιδιότητα weak-acyclicity, έτσι ώστε η διαδικασία να μην τρέχει επ' άπειρον [2]. Όπως και επίσης στόχος είναι να αξιολογηθεί ο αλγόριθμος αυτός όσον αφορά τους χρόνους απόδοσης, αναλύοντας ποια μέρη της διαδικασίας του αλγορίθμου απαιτούν τον περισσότερο χρόνο για να αποφασίσουν αν η διαδικασία θα τερματίσει ή όχι.

1.2 Οργάνωση Κειμένου

Η υφιστάμενη διπλωματική εργασία αποτελείται από έξι κεφάλαια.

Στο Κεφάλαιο 2 εξετάζουμε το γενικό πρόβλημα τερματισμού ενός Chase Procedure για TGDs, και παρουσιάζονται οι βασικές αρχές καθώς και ο ορισμός του Chase Procedure.

Στο Κεφάλαιο 3 επικεντρωνόμαστε στην περιγραφή του προβλήματος τερματισμού της διαδικασίας Semi-oblivious Chase. Ειδικότερα, παρουσιάζεται η ιδιότητα του weak-acyclicity για τα Simple Linear TGDs.

Στο Κεφάλαιο 4 περιγράφουμε τον τρόπο υλοποίησης του αλγορίθμου. Αρχικά, παρουσιάζεται η γενική προσέγγιση και τα εργαλεία ανάπτυξης που χρησιμοποιήθηκαν. Στη συνέχεια, γίνεται λεπτομερής περιγραφή της υλοποίησης, η οποία περιλαμβάνει τη μετατροπή των δεδομένων, τη δημιουργία του εξαρτημένου γράφου και τον εντοπισμό κρίσιμου κύκλου.

Στο Κεφάλαιο 5 περιλαμβάνεται η πειραματική αξιολόγηση του αλγορίθμου. Παρουσιάζονται τα αποτελέσματα των πειραμάτων και γίνεται σχολιασμός αυτών, με στόχο την ανάλυση της αποτελεσματικότητας και της ακρίβειας του αλγορίθμου σε διάφορα σύνολα δεδομένων, αλλά και για κάθε διαδικασία.

Τέλος, το Κεφάλαιο 6 συνοψίζει τα κύρια ευρήματα της εργασίας και προτείνει κατευθύνσεις για μελλοντική εργασία. Επιπλέον, αναφέρονται τα γενικά συμπεράσματα που προκύπτουν από την έρευνα και την υλοποίηση του αλγορίθμου.

Με αυτή τη δομή, η διπλωματική εργασία στοχεύει να παρουσιάσει με σαφήνεια και ολοκληρωμένο τρόπο τη μελέτη, την ανάπτυξη και την αξιολόγηση του αλγορίθμου για την επίλυση του προβλήματος τερματισμού της διαδικασίας Semi-oblivious Chase για Simple Linear TGDs.

1.3 Ορισμοί και Ακρωνύμια

Ορισμοί και Ακρωνύμια	Εξήγηση
CP	Chase Procedure – Διαδικασία Καταδίωξης
TGD	Tuple-generating dependencies – Εξαρτήσεις που δημιουργούν πλειάδες
SL	Simple Linear (TGDs)
Triggers	Σημαίνει να προκαλέσεις ή να ενεργοποιήσεις κάτι. Χρησιμοποιείται γενικά για να περιγράψει την ενέργεια που ξεκινά μια αλυσίδα γεγονότων ή διαδικασιών.
Null	Αναφέρεται σε μία ειδική τιμή που χρησιμοποιείται για να δηλώσει την "απουσία" μιας συγκεκριμένης τιμής. Αυτό σημαίνει ότι η τιμή είναι άγνωστη ή μη διαθέσιμη.

Πίνακας 1.1 – Ορισμοί και Ακρωνύμια

Κεφάλαιο 2

Chase Termination Problem

2.1 Chase Termination για TGDs

Το Chase Procedure αποτελεί ένα κρίσιμο αλγοριθμικό εργαλείο στην θεωρία των βάσεων δεδομένων.

Η κεντρική πρόκληση της διαδικασίας αυτής είναι ο τερματισμός της σε κάθε περίπτωση, δηλαδή, η διαπίστωση εάν ολοκληρώνεται όταν εφαρμόζεται με βάση συγκεκριμένες εξαρτήσεις που δημιουργούν πλειάδες (TGDs). Αυτό το πρόβλημα είναι γνωστό ότι δεν έχει γενική λύση. Επίσης, εξετάζουμε αν περιορισμένες κατηγορίες των TGD, που έχουν προταθεί για διαφορετικές χρήσεις όπως οντολογικά ερωτήματα, μπορούν να κάνουν το πρόβλημα διαχειρίσιμο. Αναλύουμε αποτελέσματα που εγγυώνται την ολοκλήρωση της διαδικασίας για κατηγορίες TGD βασισμένες στην έννοια της φύλαξης (guardedness).

2.2 Chase Procedure και TGD

Η Διαδικασία Καταδίωξης (Chase Procedure - CP), όπως προαναφέραμε, θεωρείται ως ένα από τα πιο θεμελιώδη αλγοριθμικά εργαλεία σε βάσεις δεδομένων. Έχει εφαρμοστεί με επιτυχία σε ένα ευρύ φάσμα προβλημάτων όπως, έλεγχος της λογικής επίπτωσης των περιορισμών, υπολογισμός λύσεων ανταλλαγής δεδομένων, απάντηση κάποιων ερωτημάτων (queries) υπό περιορισμούς κ.ά.

Η διαδικασία αυτή λαμβάνει ως είσοδο μια βάση δεδομένων D και ένα σύνολο Σ από περιορισμούς και, αν τερματίζεται (πράγμα που δεν είναι εγγυημένο), το αποτέλεσμά του είναι ένα πεπερασμένο σύνολο με όνομα D_Σ που έχει τις πιο κάτω ιδιότητες:

- Το D_Σ είναι *μοντέλο* του D και του Σ, δηλαδή περιέχει το D και ικανοποιεί τους περιορισμούς του Σ
- Το D_Σ είναι *καθολικό*, δηλαδή μπορεί να ενσωματωθεί ομομορφικά σε κάθε άλλο μοντέλο των D και Σ

Με άλλα λόγια, το Chase Procedure είναι ένα αλγοριθμικό εργαλείο για τον υπολογισμό καθολικών μοντέλων του D και του Σ, το οποίο μπορεί να θεωρηθεί ως εκπρόσωπος όλων των άλλων μοντέλων του D και του Σ. Πράγματι, πολλά βασικά προβλήματα βάσης δεδομένων, όπως τα παραπάνω, μπορούν να λυθούν με την απλή παρουσίαση ενός καθολικού μοντέλου.

Κατηγορία περιορισμών, που μπορεί να αντιμετωπιστεί με CP, είναι η κλάση των εξαρτήσεων που δημιουργούν πλειάδες (TGD), γνωστοί και ως «existential rules». Τα TGDs είναι προεκτάσεις της μορφής:

$$\forall X \forall Y(\varphi(X, Y) \rightarrow \exists Z\, \psi(Y, Z))$$

όπου φ(X, Y) είναι το σώμα (body), ψ(Y, Z) είναι η κεφαλή (head) και τα $\boldsymbol{\varphi}$, $\boldsymbol{\psi}$ είναι σύνδεσμοι ατόμων, που ουσιαστικά δηλώνουν ότι η παρουσία κάποιων πλειάδων σε μια περίπτωση συνεπάγεται την ύπαρξη κάποιων άλλων πλειάδων στην ίδια περίπτωση.

Δίνεται μια βάση δεδομένων D και ένα σύνολο Σ των TGD, το CP προσθέτει νέα άτομα στο D μέχρι το τελικό αποτέλεσμα να ικανοποιήσει το Σ.

Για παράδειγμα, θεωρούμε τη βάση δεδομένων D = {person(Bob)} και ένα TGD $\forall X(person(X) \rightarrow \exists Y\ hasFather(X, Y) \wedge person(Y))$, που υποστηρίζει ότι κάθε άτομο έχει έναν πατέρα που είναι επίσης άτομο. Το άτομο της βάσης δεδομένων ενεργοποιεί (triggers) το TGD και το CP θα προσθέσει στο D τα άτομα hasFather(Bob, z1) και person(z1) για να το ικανοποιήσουν, όπου z1 είναι ένα null που αντιπροσωπεύει κάποια άγνωστη τιμή. Ωστόσο, το νέο άτομο person(z1) ενεργοποιεί ξανά το TGD και το CP αναγκάζεται να προσθέσει τα άτομα hasFather(z1, z2), person(z2), όπου z2 είναι ένα νέο null. Πράγμα που οδηγεί στο να μη τερματίζει ποτέ, με το συγκεκριμένο παράδειγμα. Δηλαδή, το αποτέλεσμα του CP είναι:

$$D_\Sigma = \{person(Bob), hasFather(Bob, z1)\} \cup \bigcup_{i>0} \{person(zi), hasFather(zi, zi+1)\},$$

z1,z2,.. είναι nulls.

Κεφάλαιο 3

Περιγραφή Semi-oblivious Chase Termination Problem για SL

3.1 Semi-oblivious Chase

Στην διαδικασία Semi-oblivious Chase, λαμβάνονται υπόψη οι εφαρμογές TGD που έχουν ήδη γίνει. Εάν μια εφαρμογή ενός TGD έχει ήδη οδηγήσει στη δημιουργία νέας μεταβλητής (null), η διαδικασία Semi-oblivious δεν δημιουργεί εκ νέου τις ίδιες αυτές μεταβλητές. Αυτό μειώνει τον αριθμό των εφαρμογών κανόνων και αυξάνει τις πιθανότητες τερματισμού της διαδικασίας.

Σε αντίθεση με την διαδικασία Oblivious Chase που δημιουργεί νέες μεταβλητές (nulls) κάθε φορά που εφαρμόζεται το TGD, ανεξαρτήτως του αν έχει εφαρμοστεί στο παρελθόν. Γι αυτό και σε αυτή την διπλωματική εργασία θα ασχοληθούμε μόνο με την διαδικασία Semi-oblivious Chase.

3.2 Simple Linear TGDs

Ένα TGD είναι «Simple Linear» αν πληροί τις ακόλουθες προϋποθέσεις:

- Απλότητα (Simple): Στην αριστερή πλευρά του (body) δεν επαναλαμβάνονται οι ίδιες μεταβλητές.
- Γραμμικότητα (Linear): Η αριστερή πλευρά (body) αποτελείται από ένα μόνο άτομο.

3.3 Weak-Acyclicity

Weak-Acyclicity είναι μια ιδιότητα που εξασφαλίζει ότι το CP θα τερματίσει, εφόσον τα TGDs είναι SL. Η συνθήκη για τα Simple Linear είναι ικανή και αναγκαία, πράγμα που δεν ισχύει για τα Linear, αφού εάν το σύνολο των (Linear) TGDs δεν είναι weak-acyclic δεν σημαίνει απαραίτητα ότι το CP δεν θα τερματίσει, ενώ αν είναι weak-acyclic γνωρίζουμε ότι το CP θα τερματίσει σίγουρα. Με απλά λόγια για τα Linear TGDs η συνθήκη είναι ικανή και όχι αναγκαία, και γι αυτό τον λόγο χρησιμοποιείται μόνο σε Simple Linear TGDs.

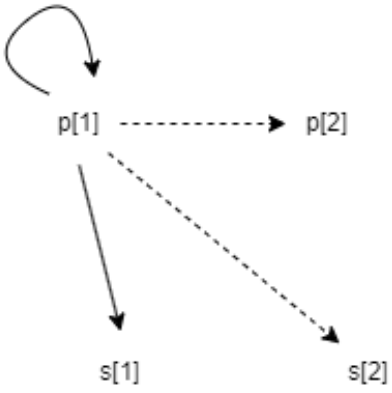
Για να ελέγξουμε αν ένα σύνολο TGDs είναι weak-acyclic, δημιουργούμε τον εξαρτημένο γράφο (dependency graph) των TGDs και εξετάζουμε αν υπάρχουν κύκλοι που περιέχουν ειδικές ακμές.

Δημιουργία του Εξαρτημένου Γράφου (Dependency Graph):

- Κανονικές Ακμές (Normal Edges): Μια κανονική ακμή τύπου (π, π΄) υπάρχει αν μια μεταβλητή από τη θέση π στην αριστερή πλευρά ενός TGD εμφανίζεται επίσης στη θέση π' στη δεξιά πλευρά του ίδιου TGD.
- Ειδικές Ακμές (Special Edges): Μια ειδική ακμή τύπου (π, π΄) υπάρχει αν μια μεταβλητή από τη θέση π στην αριστερή πλευρά ενός TGD οδηγεί στη δημιουργία μιας νέας μεταβλητής στη θέση π' στη δεξιά πλευρά του ίδιου TGD.

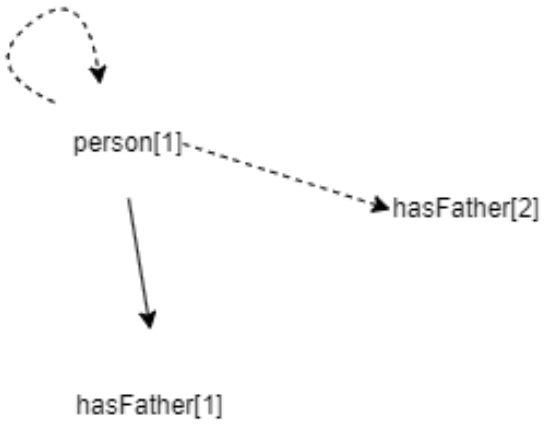
Ένα σύνολο TGDs είναι weak-acyclic αν κανένας κύκλος στον εξαρτημένο γράφο δεν περιέχει ειδική ακμή (special edge). Αν υπάρχει ένας τέτοιος κύκλος (κρίσιμος κύκλος), τότε το σύνολο TGDs δεν είναι weak-acyclic και η διαδικασία δεν τερματίζει.

Για παράδειγμα το TGD $p(X, Y) \rightarrow \exists Z\ s(X, Z), p(X, Z)$, είναι weak-acyclic αφού ο παρακάτω εξαρτημένος γράφος που το αφορά, δεν έχει κρίσιμο κύκλο.



Σχήμα 3.1 – Εξαρτημένος Γράφος (weak-acyclic)

Ενώ, το TGD person(X) → ∃Y hasFather(X, Y) ∧ person(Y), δεν είναι weak-acyclic αφού ο παρακάτω εξαρτημένος γράφος που το αφορά, έχει κρίσιμο κύκλο.



Σχήμα 3.2 – Εξαρτημένος Γράφος (non weak-acyclic)

Κεφάλαιο 4

Υλοποίηση Semi-oblivious Chase Termination Problem

4.1 Περιγραφή Τρόπου Υλοποίησης

Η υλοποίηση αυτή αφορά τον αλγόριθμο που ελέγχει αν ένα σύνολο από SL TGDs είναι weak-acyclic, εξασφαλίζοντας έτσι ότι η διαδικασία semi-oblivious chase θα τερματίσει. Για την υλοποίηση του αλγορίθμου αυτού χρησιμοποιήθηκε η γλώσσα προγραμματισμού PHP, όπως επίσης και HTML/CSS για την κόσμια εμφάνιση των αποτελεσμάτων στο web.

Αρχικά, διαβάζουμε τα TGDs από ένα αρχείο εισόδου, το οποίο επιλέγουμε από το interface. Με βάση αυτά τα TGDs, δημιουργούμε τον εξαρτημένο γράφο και ελέγχουμε κατά πόσο υπάρχει κρίσιμος κύκλος, δηλαδή κύκλος με ειδική ακμή. Εφόσον εντοπιστεί αυτό σημαίνει ότι το σύνολο των TGDs δεν είναι weak-acyclic και το CP δεν θα τερματίσει.

Τα ανάλογα μηνύματα όπως οι χρόνοι της κάθε διαδικασίας, ο αριθμός των TGDs, ο αριθμός των Predicates και το αν τερματίζει ή όχι, εμφανίζονται σε έναν πίνακα στο user interface με την βοήθεια των HTML και CSS.

4.2 Εργαλεία Ανάπτυξης

4.2.1 Εργαλείο VS Code

Το Visual Studio Code [6] αποτελεί ένα πλήρες ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που παρέχεται από τη Microsoft. Χρησιμοποιείται κυρίως από προγραμματιστές για την ανάπτυξη εφαρμογών, ιστοσελίδων και άλλων λογισμικών.

Χαρακτηρίζεται από την ευκολία χρήσης του και την υποστήριξη πολλών γλωσσών προγραμματισμού όπως Java, PHP, Python, και πολλές άλλες.

4.2.2 Εργαλείο Draw.io

Το Draw.io [3] είναι ένα διαδικτυακό λογισμικό διαγραμμάτων που επιτρέπει στους χρήστες να δημιουργούν διάφορους τύπους διαγραμμάτων όπως δίκτυα ροής, διαγράμματα UML, διαγράμματα σχέσεων οντοτήτων, γράφους, και πολλά άλλα.

Το εργαλείο υποστηρίζει πολλαπλές ενσωματώσεις, επιτρέποντας την αρμονική συνεργασία με άλλες πλατφόρμες όπως Google Drive, Microsoft OneDrive, SharePoint και ακόμη και GitHub. Αυτό το καθιστά εξαιρετικά ευέλικτο για χρήση και άμεσα προσβάσιμο.

4.2.3 Γλώσσα Προγραμματισμού PHP

Η PHP [5] είναι μια δημοφιλής γλώσσα προγραμματισμού. Είναι ιδιαίτερα γνωστή για την ικανότητά της να ενσωματώνεται σε κώδικα HTML, κάτι που την κάνει ιδιαίτερα εύχρηστη για την δημιουργία δυναμικών ιστοσελίδων.

Είναι υποστηρικτική σε πολλές βάσεις δεδομένων και έχει ένα πλούσιο σύνολο ενσωματωμένων λειτουργιών που διευκολύνουν την ανάπτυξη των εφαρμογών.

4.2.4 Γλώσσα Περιγραφής HTML

HTML είναι ένα ακρώνυμο για τις λέξεις HyperText Markup Language [4]. Είναι η γλώσσα που χρησιμοποιείται στη δημιουργία και στο σχεδιασμό εφαρμογών διαδικτύου. Η HTML

για την δόμηση του διαδικτυακού υλικού χρησιμοποιεί «tags» τα οποία ορίζουν επικεφαλίδες, παραγράφους, συνδέσμους, εικόνες κ.ά.

Κάθε έγγραφο HTML περιέχει μια σειρά από στοιχεία, τα οποία είναι τα δομικά στοιχεία όλων των ιστοσελίδων. Τα στοιχεία της HTML αντιπροσωπεύονται από ετικέτες που γράφονται χρησιμοποιώντας αγκύλες. Για παράδειγμα, μια παράγραφος δημιουργείται με την ετικέτα <p>, και ένας σύνδεσμος δημιουργείται με την ετικέτα <a>.

4.2.5 Γλώσσα Προγραμματισμού CSS

Η CSS (Cascading Style Sheets) [1] είναι μια απλή και ισχυρή γλώσσα που χρησιμοποιείται για να καθορίσει την εμφάνιση και τη διαμόρφωση των ιστοσελίδων. Προτάθηκε από το WorldWide Web Consortium (W3C) το 1996 και αποτελεί μια από τις βασικές τεχνολογίες του Παγκόσμιου Ιστού.

Η CSS επιτρέπει στους προγραμματιστές να ξεχωρίσουν την παρουσίαση μιας ιστοσελίδας από τη δομή της, καθορίζοντας πώς θα εμφανίζονται τα στοιχεία της HTML. Με την CSS, μπορούμε να ορίσουμε χρώματα, γραμματοσειρές, διαστήματα, και άλλες οπτικές ιδιότητες των στοιχείων της ιστοσελίδας.

Ένα από τα κύρια πλεονεκτήματα της CSS είναι η δυνατότητα επαναχρησιμοποίησης των στυλ. Αντί να χρειάζεται να γράφουμε τις ίδιες στιλιστικές πληροφορίες ξανά και ξανά σε κάθε σελίδα, μπορούμε να δημιουργήσουμε ένα εξωτερικό αρχείο CSS που περιέχει όλους τους κανόνες στυλ και να το παραπέμπουμε από τις σελίδες HTML. Αυτό καθιστά τη διαδικασία της ανάπτυξης και της συντήρησης των ιστοσελίδων πιο αποδοτική και οργανωμένη.

Σε αυτή την εργασία χρησιμοποιήθηκε για την εισαγωγή συγκεκριμένου στυλ στην HTML.

4.3 Υλοποίηση

Στην υλοποίηση της εργασίας αυτής, δημιουργήθηκαν δύο κλάσεις για να διευκολυνθεί η επεξεργασία των TGDs και η ανάλυση του Εξαρτημένου Γράφου. Η χρήση των αντικειμένων στις συναρτήσεις βελτίωσε την αναγνωσιμότητα, τη συντήρηση και την επεκτασιμότητα του κώδικα. Παρακάτω περιγράφονται οι κλάσεις που δημιουργήθηκαν:

1. Η κλάση TGD αναπαριστά ένα TGD, το οποίο περιλαμβάνει ένα σώμα και μια κεφαλή.

2. Η κλάση DependencyGraph περιέχει δύο πίνακες για την αποθήκευση των κανονικών ακμών και των ειδικών ακμών, όπως και τις παρακάτω συναρτήσεις:
   - Η συνάρτηση *addEdge()* προσθέτει μια κανονική ακμή στον γράφο.
   - Η συνάρτηση *addSpecialEdge()* προσθέτει μια ειδική ακμή στον γράφο.
   - Η συνάρτηση *getNumPredicates()* υπολογίζει και επιστρέφει τον αριθμό των μοναδικών predicates στον γράφο.
   - Η συνάρτηση *hasCriticalCycle()* ελέγχει αν υπάρχει κρίσιμος κύκλος στον γράφο, χρησιμοποιώντας την ιδιωτική μέθοδο *dfs()* για αναδρομική αναζήτηση βάθους.

4.3.1 Περιγραφή Υλοποίησης Μετατροπής Δεδομένων

Για την υλοποίηση της μετατροπής των δεδομένων οφείλεται η συνάρτηση *getTGDs()*, που έχει ως στόχο την ανάγνωση και επεξεργασία των TGDs από ένα αρχείο, και τη μετατροπή τους σε κατάλληλες δομές δεδομένων για περαιτέρω επεξεργασία από τον αλγόριθμο.

Η συνάρτηση αυτή διαβάζει το αρχείο γραμμή προς γραμμή, αναλύει το περιεχόμενο κάθε γραμμής εάν αυτό είναι έγκυρο και επιθυμητό. Αν περάσει από τους κατάλληλους ελέγχους, τότε αυτό σημαίνει ότι είναι ένα έγκυρο SL TGD και έτσι δημιουργείται αντικείμενο (object) τύπου TGD που αποθηκεύεται σε έναν πίνακα.

Πιο αναλυτικά:

Εφόσον είναι μία έγκυρη γραμμή (TGD), η γραμμή διαχωρίζεται σε δύο μέρη, την κεφαλή ($head_string) και το σώμα ($body_string).

Το σώμα του κανόνα ($body_string) καθαρίζεται από περιττά κενά (trim) και ελέγχεται αν περιέχει μόνο ένα ζεύγος παρενθέσεων και ότι δεν υπάρχει επανάληψη μεταβλητής, αυτό διασφαλίζει ότι είναι SL. Αν το σώμα πληροί τις προϋποθέσεις, αναλύεται σε predicate και args (ορίσματα). Οι θέσεις και οι μεταβλητές αποθηκεύονται σε ένα πίνακα ($bodyArr).

Η κεφαλή του κανόνα ($head_string) καθαρίζεται από περιττά κενά (trim) και διαχωρίζεται σε επιμέρους κανόνες χρησιμοποιώντας τη explode() και έτσι κάθε κανόνας αναλύεται σε

predicate και args. Όλα τα τυχόν predicates της κεφαλής με τα ορίσματα (args) τους αποθηκεύονται σε έναν πίνακα ($headArr).
Δημιουργία Αντικειμένου TGD:

Μετά την ανάλυση του σώματος και της κεφαλής, δημιουργείται το αντικείμενο TGD και αποθηκεύεται στον πίνακα με τα TGDs.

Η συνάρτηση επιστρέφει τον πίνακα με τα αντικείμενα TGDs.

Η συνάρτηση *getTGDs()* αποτελεί ένα κρίσιμο μέρος της διαδικασίας, καθώς εξασφαλίζει ότι τα δεδομένα TGDs αναλύονται σωστά και μετατρέπονται σε κατάλληλες δομές για περαιτέρω επεξεργασία από τον αλγόριθμο. Η χρήση ελέγχων και φιλτραρίσματος εξασφαλίζουν ότι μόνο έγκυρα SL TGDs επεξεργάζονται.

4.3.2 Περιγραφή Υλοποίησης Εξαρτημένου Γράφου

Η συνάρτηση *createDependencyGraph()* έχει ως στόχο τη δημιουργία του εξαρτημένου γράφου από τα δεδομένα TGDs. Ο Εξαρτημένος Γράφος αποτελείται από κόμβους που αντιπροσωπεύουν τις θέσεις των μεταβλητών στα predicates τους και από ακμές που αντιπροσωπεύουν τις εξαρτήσεις μεταξύ αυτών των θέσεων.

Η συνάρτηση ξεκινά με τη δημιουργία ενός νέου αντικειμένου τύπου DependencyGraph το οποίο θα χρησιμοποιηθεί για την αποθήκευση των κόμβων και των ακμών του γράφου. Η συνάρτηση διατρέχει κάθε αντικείμενο TGD του πίνακα που παίρνει ως είσοδο. Κάθε TGD γνωρίζουμε ότι περιέχει ένα σώμα και μια κεφαλή.

Για κάθε TGD, η συνάρτηση εξάγει το predicate και τα ορίσματα (args) του σώματος. Στη συνέχεια, η συνάρτηση διατρέχει κάθε άτομο από την κεφαλή του TGD. Η συνάρτηση δημιουργεί τις ακμές του γράφου με βάση τις εξαρτήσεις μεταξύ των μεταβλητών του σώματος και της κεφαλής. Τότε δημιουργούνται κανονικές ακμές από το σώμα στην κεφαλή. Επίσης, αν η κεφαλή περιέχει μια ειδική μεταβλητή που ξεκινά με "Ex", τότε δημιουργείται μια ειδική ακμή.

Μετά την επεξεργασία όλων των TGDs, η συνάρτηση επιστρέφει το αντικείμενο DependencyGraph που περιέχει όλες τις κανονικές και ειδικές ακμές που δημιουργήθηκαν.

Η συνάρτηση *createDependencyGraph()* είναι ένα σημαντικό μέρος της διαδικασίας, καθώς δημιουργεί τον εξαρτημένο γράφο που χρησιμοποιείται για τον εντοπισμό κρίσιμων κύκλων και την απόφαση για τον τερματισμό του CP. Η λογική της συνάρτησης εξασφαλίζει ότι όλες οι εξαρτήσεις μεταξύ των μεταβλητών αποθηκεύονται σωστά στον γράφο, επιτρέποντας έτσι την περαιτέρω ανάλυση και επεξεργασία.

4.3.3 Περιγραφή Υλοποίησης Εντοπισμού Κρίσιμου Κύκλου

Η συνάρτηση *hasCriticalCycle()* έχει ως στόχο τον εντοπισμό κρίσιμου κύκλου στον εξαρτημένο γράφο που δημιουργήθηκε από τα TGDs.

Η συνάρτηση αρχικοποιεί δύο δομές δεδομένων, έναν πίνακα που παρακολουθεί ποιους κόμβους έχει επισκεφθεί ($visited[]) και έναν πίνακα που παρακολουθεί ποιοι κόμβοι βρίσκονται στη στοίβα αναδρομής ($recStack[]). Επίσης, αρχικοποιεί και μια λογική μεταβλητή που υποδεικνύει αν έχει εντοπιστεί ειδική ακμή σε έναν τυχόν κύκλο ($hasSpecialEdge). Η συνάρτηση διατρέχει όλους τους κόμβους και αν κάποιο κόμβο δεν τον έχει επισκεφθεί ακόμα, ξεκινά μια αναδρομική αναζήτηση βάθους (DFS) από αυτόν τον κόμβο.

Η ιδιωτική συνάρτηση *dfs()* εκτελεί μια αναδρομική αναζήτηση βάθους για να εντοπίσει κρίσιμους κύκλους. Η συνάρτηση αυτή επισκέπτεται τον τρέχοντα κόμβο και τον προσθέτει στη στοίβα αναδρομής. Ελέγχει για self-loop σε ειδική ακμή και αν υπάρχει, ενημερώνει τη μεταβλητή $hasSpecialEdge και επιστρέφει «true». Διαφορετικά, ανακτά όλους τους γειτονικούς κόμβους από τις κανονικές και ειδικές ακμές, και τους διατρέχει. Αν ένα γειτονικό κόμβο δεν τον έχει επισκεφθεί, εκτελεί αναδρομή σε αυτόν τον κόμβο. Αν ένας γειτονικός κόμβος βρίσκεται ήδη στη στοίβα αναδρομής, υποδεικνύει την ύπαρξη κύκλου. Ελέγχει αν η ακμή είναι ειδική και ενημερώνει τη μεταβλητή $hasSpecialEdge και αφαιρεί τον τρέχοντα κόμβο από τη στοίβα αναδρομής μετά την ολοκλήρωση της εξερεύνησης.

Αν κατά τη διάρκεια της *dfs()* εντοπιστεί κύκλος με ειδική ακμή, τότε η συνάρτηση *hasCriticalCycle()* επιστρέφει «true». Αν δεν εντοπιστεί τέτοιος κύκλος, τότε επιστρέφει «false».

Κεφάλαιο 5

Πειραματική Αξιολόγηση Αλγορίθμου

5.1 Αποτελέσματα Πειραμάτων

Τα πειράματα διεξήχθησαν σε ένα σύνολο από αρχεία δεδομένων. Οι μετρήσεις που καταγράφηκαν περιλαμβάνουν τον αριθμό των TGDs, τον αριθμό των Predicates, τον Χρόνο Μετατροπής Δεδομένων (Parsing Time), τον Χρόνο Δημιουργίας του Εξαρτημένου Γράφου (Creation of Dependency Graph Time), τον Χρόνο Εντοπισμού ενός Κρίσιμου Κύκλου (Detection of a Critical Circle Time), και το αν η διαδικασία τερματίζει ή όχι (Terminates Or Not).

Ακολουθεί ο πίνακας με τα αποτελέσματα:

File	Number of TGDs	Number of Predicates	Parsing Time (msec)	Creation of Dependency Graph Time (msec)	Detection of a Critical Circle Time (msec)	Terminates Or Not
00001.txt	3140	2143	6.836	3.031	0.365	No
00002.txt	1740	929	4.336	1.974	0.702	Yes
00153.txt	182	134	1.497	0.476	0.091	No
00007.txt	237	121	0.602	0.219	0.105	Yes
00008.txt	237	121	0.632	0.232	0.11	Yes
00009.txt	237	121	0.615	0.22	0.107	Yes
00010.txt	237	121	1.358	0.687	0.241	Yes
00011.txt	237	121	0.841	0.229	0.107	Yes
00012.txt	471	271	1.173	0.36	0.185	Yes
00014.txt	3079	1501	6.562	3.123	0.399	No
00020.txt	2263	1012	5.516	2.403	0.352	No
00021.txt	2192	1026	4.693	2.372	0.32	No
00024.txt	3079	1501	6.601	3.195	0.502	No
00025.txt	1117	408	2.538	1.525	0.302	No
00049.txt	65	40	0.283	0.069	0.032	Yes
00050.txt	65	40	0.778	0.217	0.106	Yes
00055.txt	245	180	0.658	0.207	0.118	Yes
00057.txt	14	10	0.154	0.02	0.013	Yes
00071.txt	15	13	0.114	0.022	0.017	Yes
00151.txt	323	169	2.858	0.885	0.34	Yes

Πίνακας 5.1 – Αποτελέσματα Πειραμάτων

5.2 Σχολιασμός Αποτελεσμάτων

Παρατηρούμε ότι όσο μεγαλύτερος είναι ο αριθμός των TGDs και των Predicates, τόσο περισσότερο χρόνο χρειάζεται ο αλγόριθμος για να επεξεργαστεί τα δεδομένα και να δημιουργήσει τον εξαρτημένο γράφο. Αυτό είναι αναμενόμενο, καθώς η πολυπλοκότητα του γράφου αυξάνεται με τον αριθμό των TGDs και των Predicates.

Ο Χρόνος Μετατροπής Δεδομένων (Parsing Time) είναι γενικά ο μεγαλύτερος από τους τρεις χρόνους που καταγράφηκαν. Αυτό οφείλεται στο γεγονός ότι η διαδικασία περιλαμβάνει την ανάγνωση και την ανάλυση του αρχείου, καθώς και τη δημιουργία των εσωτερικών δομών δεδομένων για την αναπαράσταση των TGDs, καθώς και τον έλεγχο ότι είναι Simple Linear TGDs. Επίσης παρατηρούμε ότι ο χρόνος αυξάνεται με τον αριθμό των TGDs. Για παράδειγμα, το αρχείο «00001.txt» με 3140 TGDs χρειάστηκε 6.836 msec, ενώ το αρχείο «00057.txt» με 14 TGDs χρειάστηκε μόνο 0.154 msec. Αυτό δείχνει ότι η επεξεργασία των δεδομένων είναι γραμμική ως προς τον αριθμό των TGDs, πράγμα που είναι επίσης αναμενόμενο αφού ο αλγόριθμος περνά από κάθε γραμμή του αρχείου για να κάνει την μετατροπή.

Ο Χρόνος Δημιουργίας του Εξαρτημένου Γράφου αυξάνεται με τον αριθμό των TGDs και των predicates. Αυτό δείχνει ότι η πολυπλοκότητα της διαδικασίας δημιουργίας του γράφου είναι επίσης γραμμική ως προς τον αριθμό των TGDs και των Predicates.

Ο Χρόνος Εντοπισμού ενός Κρίσιμου Κύκλου είναι γενικά μικρός και αυξάνεται με την πολυπλοκότητα του γράφου. Αυτό δείχνει ότι ο αλγόριθμος είναι αποτελεσματικός στον εντοπισμό κρίσιμων κύκλων, ακόμη και για μεγάλα σύνολα δεδομένων.

Τα αποτελέσματα δείχνουν ότι τα αρχεία με μικρότερο αριθμό TGDs και Predicates έχουν μεγαλύτερη πιθανότητα να τερματίζουν, σε σχέση με τα μεγαλύτερα αρχεία.

Από τα αποτελέσματα των πειραμάτων μπορούμε να συμπεράνουμε ότι η υλοποίηση του αλγορίθμου είναι επιτυχημένη. Επιπλέον, ότι οι χρόνοι μετατροπής δεδομένων, δημιουργίας και επεξεργασίας του εξαρτημένου γράφου είναι γραμμικοί ως προς τον αριθμό των TGDs και των Predicates, γεγονός που δείχνει ότι ο αλγόριθμος είναι καλά σχεδιασμένος και κλιμακώνεται αποτελεσματικά. Τέλος, ότι ο χρόνος εντοπισμού ενός κρίσιμου κύκλου είναι μικρός, γεγονός που δείχνει ότι ο αλγόριθμος είναι αποτελεσματικός στην ανίχνευση

κρίσιμων κύκλων. Αυτό είναι σημαντικό, καθώς ο εντοπισμός κρίσιμων κύκλων είναι το κλειδί για τον προσδιορισμό του τερματισμού της διαδικασίας.

Κεφάλαιο 6

Συμπεράσματα

6.1 Μελλοντική Εργασία

Ο αλγόριθμος που υλοποιήθηκε σε αυτήν τη Διπλωματική Εργασία αντιμετωπίζει το πρόβλημα του τερματισμού ενός Chase Procedure για Simple Linear TGDs weak-acyclicity με επιτυχία. Ωστόσο, υπάρχουν αρκετές προοπτικές για μελλοντική εργασία που θα μπορούσαν να επεκτείνουν και να βελτιώσουν το έργο αυτό.

Ένας βασικός τομέας για μελλοντική έρευνα και ανάπτυξη είναι η υλοποίηση του αλγορίθμου και για Linear TGDs. Τα Linear TGDs είναι μια πιο γενική κατηγορία από τα Simple Linear TGDs, και η επεξεργασία τους παρουσιάζει περισσότερες προκλήσεις.

Ένα TGD είναι Linear αν πληροί απλά την γραμμικότητα (Linear), δηλαδή αν η αριστερή του πλευρά (body) αποτελείται από ένα μόνο άτομο, και σε αντίθεση με τα Simple Linear στην αριστερή πλευρά (body) ενός Linear μπορούν να επαναλαμβάνονται οι ίδιες μεταβλητές. Κάποια παραδείγματα Linear TGDs είναι:

- $p(X, X) \rightarrow \exists Z\ s(Z, X)$
- $a(X, Y, X) \rightarrow \exists Z\ b(Y, Z, X)$

Η προσέγγιση που προτείνεται για την υλοποίηση του αλγορίθμου για Linear TGDs περιλαμβάνει τη μετατροπή τους σε Simple Linear TGDs. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας έναν συγκεκριμένο αλγόριθμο μετατροπής. Μετά την υλοποίηση του αλγορίθμου μετατροπής θα είναι δυνατή η εφαρμογή του αλγορίθμου για SL TGDs weak-acyclicity.

Η επέκταση της υπάρχουσας εργασίας για να περιλαμβάνει και Linear TGDs αποτελεί ένα σημαντικό βήμα για την επίλυση του προβλήματος τερματισμού σε ένα ευρύτερο σύνολο TGDs. Αυτό θα καθιστούσε τον αλγόριθμο πιο ευέλικτο και χρήσιμο σε ένα μεγαλύτερο εύρος εφαρμογών, επιτρέποντας την καλύτερη κατανόηση και διαχείριση των δεδομένων στις βάσεις δεδομένων.

6.2 Συμπεράσματα

Η διπλωματική αυτή εργασία ασχολήθηκε με την υλοποίηση και την αξιολόγηση ενός αλγορίθμου για την επίλυση του προβλήματος τερματισμού της διαδικασίας semi-oblivious chase για Simple Linear TGDs weak-acyclicity. Τα κύρια συμπεράσματα που προέκυψαν από την έρευνα και τα πειράματα είναι τα εξής:

Ο αλγόριθμος που υλοποιήθηκε αποδείχθηκε αποτελεσματικός στην επεξεργασία και τον εντοπισμό κρίσιμων κύκλων σε μεγάλα σύνολα δεδομένων TGDs. Οι χρόνοι εκτέλεσης για την επεξεργασία των TGDs, τη δημιουργία του εξαρτημένου γράφου και τον εντοπισμό κρίσιμου κύκλου ήταν γραμμικοί ως προς τον αριθμό των TGDs και των predicates.

Τα αποτελέσματα των πειραμάτων έδειξαν ότι ο χρόνος εντοπισμού κρίσιμων κύκλων ήταν μικρός επιβεβαιώνοντας την αποτελεσματικότητα του αλγορίθμου. Επιπλέον, τα πειράματα επιβεβαίωσαν ότι το weak-acyclicity είναι ένας σημαντικός παράγοντας για τον τερματισμό ενός CP.

Βιβλιογραφία

[1] "Cascading Style Sheets", Bert Bos, 2021. [Ηλεκτρονικό]. Διαθέσιμο από: https://www.w3.org/Style/CSS/Overview.en.html

[2] Marco Calautti & Georg Gottlob & Andreas Pieris. "Chase Termination for Guarded Existential Rules". PDF File.

[3] "draw.io". [Ηλεκτρονικό]. Διαθέσιμο από: https://www.drawio.com/

[4] "HTML: HyperText Markup Language". [Ηλεκτρονικό]. Διαθέσιμο από: https://developer.mozilla.org/en-US/docs/Web/HTML

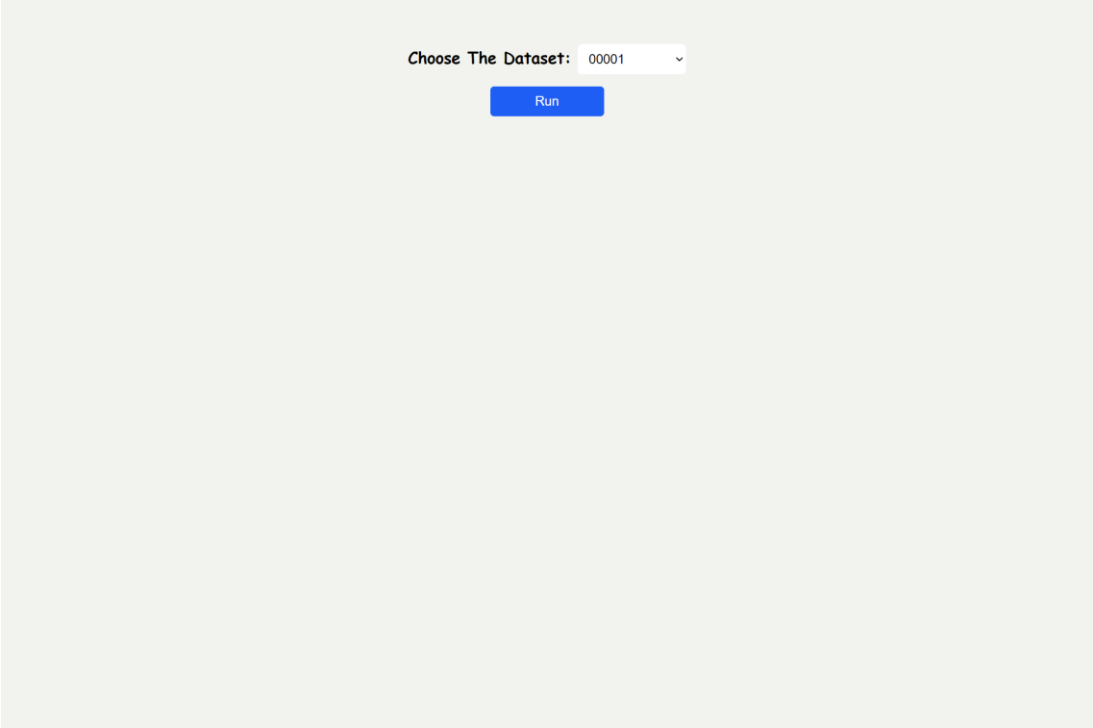
[5] "PHP: What is PHP? - Manual". [Ηλεκτρονικό]. Διαθέσιμο από: https://www.php.net/manual/en/intro-whatis.php

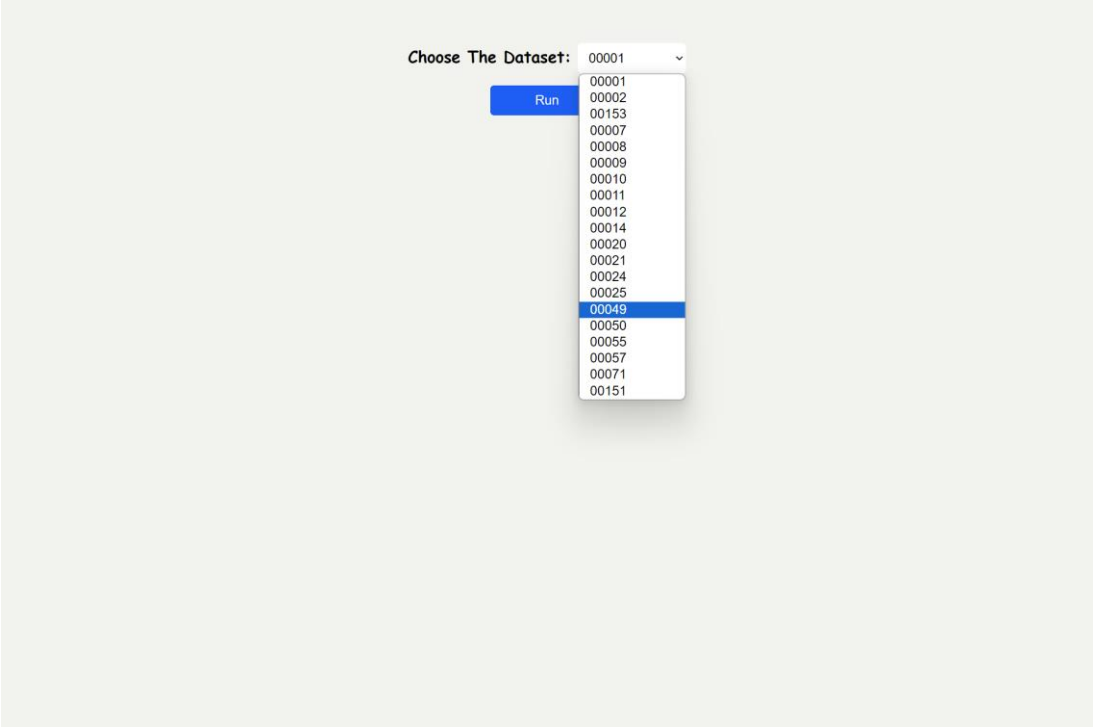
[6] "Visual Studio Code — Code Editing. Redefined". [Ηλεκτρονικό]. Διαθέσιμο από: https://code.visualstudio.com/

Παράρτημα Α

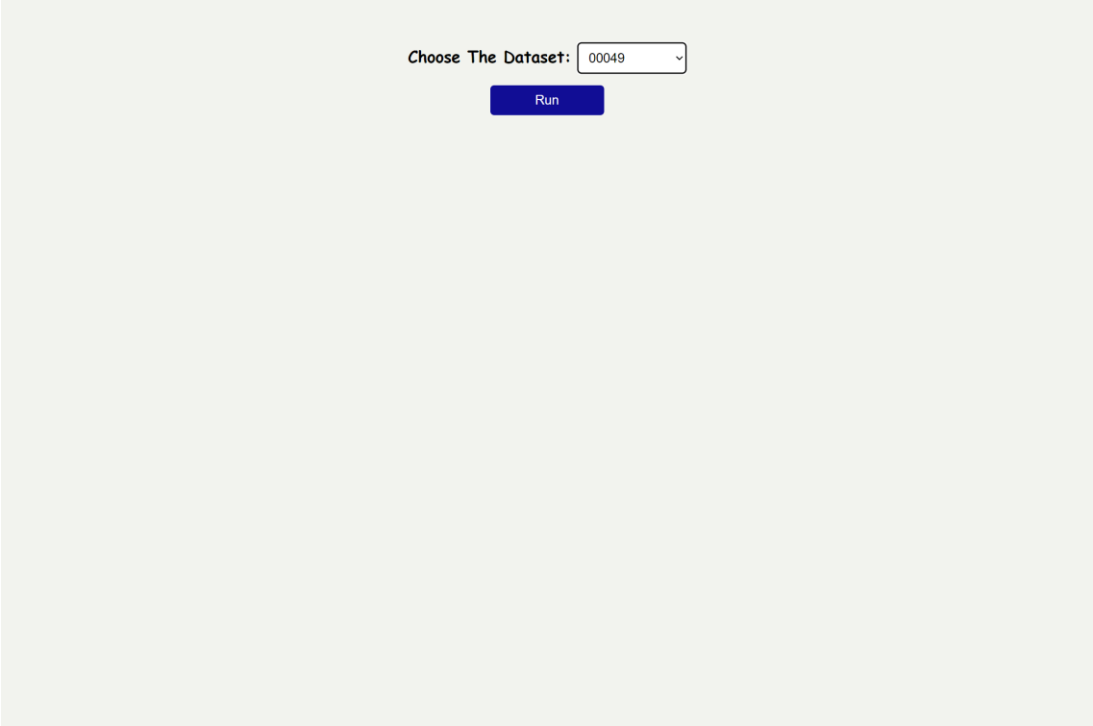
Στο παράρτημα αυτό ακολουθείται παράδειγμα εκτέλεσης του αλγορίθμου.

Επιλέγουμε ποιο dataset θέλουμε να τρέξουμε.

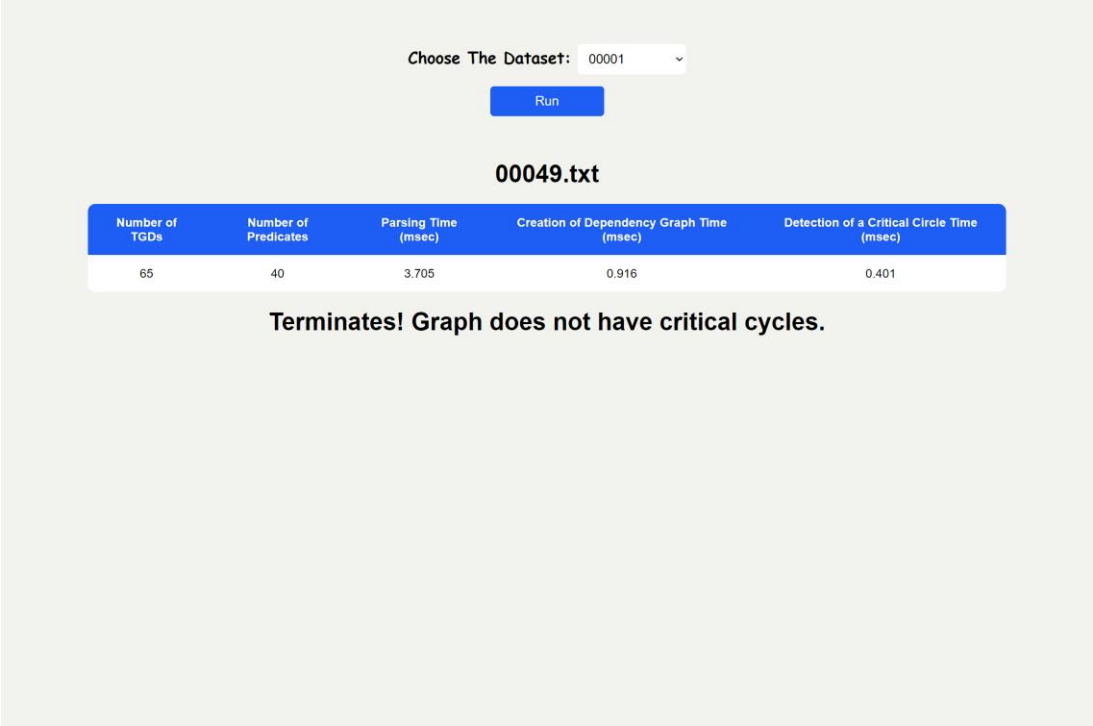




Πατάμε «Run», και ο αλγόριθμος τρέχει με βάση αυτό το dataset.



Τυπώνεται ο πίνακας με τα δεδομένα του dataset και τους χρόνους τις κάθε διαδικασίας σε msec. Επίσης, τυπώνεται και το μήνυμα αν θα τερματίζει ή όχι, δηλαδή αν ο εξαρτημένος γράφος έχει κρίσιμο κύκλο ή όχι.



Εδώ βλέπουμε ένα παράδειγμα με ένα dataset όπου υπάρχει κρίσιμος κύκλος στον εξαρτημένο γράφο, άρα δεν τερματίζει.