ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

Το πρόβλημα της σύνθεσης προγραμμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΙΚΑΤΕΡΙΝΗ Α. ΤΣΙΒΡΑ

Επιβλέπων καθηγητής: Νικόλαος Παπασπύρου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Το πρόβλημα της σύνθεσης προγραμμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΙΚΑΤΕΡΙΝΗ Α. ΤΣΙΒΡΑ

Επιβλέπων καθηγητής: Νικόλαος Παπασπύρου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την …….

……	……	……
Νικόλαος Παπασπύρου	Γεώργιος Στάμου	Γεώργιος Γκούμας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2023

……...
Αικατερίνη Α. Τσιβρά
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Νικόλαο Παπασπύρου για τις συμβουλές και την υποστήριξή του καθόλη τη διάρκεια εκπόνησης της διπλωματικής εργασίας, καθώς επίσης και για την έμπνευση και την αγάπη που μου εμφύσησε για τις γλώσσες προγραμματισμού από το πρώτο κιόλας εξάμηνο των σπουδών μου. Θα ήθελα επίσης να ευχαριστήσω τον Δρ. Δημήτριο Βυτινιώτη για την επιστημονική καθοδήγηση και τις πληροφορίες που μου παρείχε για την υλοποίηση της εργασίας. Τέλος ευχαριστώ τους φίλους που απέκτησα μέσω αυτής της διαδρομής στο πανεπιστήμιο και κυρίως τον σύντροφό μου Στέργιο για την συνεχή και αδιάλειπτη υποστήριξή του, χωρίς την οποία δε θα είχε ολοκληρωθεί αυτή η εργασία.

Περίληψη

Η αυτόματη παραγωγή κώδικα αποτελούσε διαχρονικό όνειρο των ερευνητών στον χώρο της πληροφορικής. Αυτές οι προσπάθειες συγκεντρώνονται στο ερευνητικό αντικείμενο της σύνθεσης προγραμμάτων (program synthesis). Η αρχική προσέγγιση της σύνθεσης προγραμμάτων, παράλληλη με την εξέλιξη της επιστήμης της πληροφορικής, ήταν φορμαλιστική και στηριγμένη στην λογική. Αυτή η μέθοδος ήταν δύσχρηστη και δύσκολα επεκτεινόταν σε άλλα προβλήματα. Τα ισχυρά νευρωνικά δίκτυα επέτρεψαν στους ερευνητές να περάσουν από αυστηρούς φορμαλισμούς σε πιο αφηρημένες προδιαγραφές το οποίο μετέτρεψε τη σύνθεση προγραμμάτων σε πιο ισχυρό εργαλείο αλλά με σημαντικά δυσκολότερη ερμηνεία. Οι λύσεις που σχεδιάζονται κι ερευνώνται σήμερα συνδυάζουν τα νευρωνικά δίκτυα με τυπικές μεθόδους. Αυτή η προσέγγιση έχει ήδη δώσει καλύτερα αποτελέσματα τα οποία μάλιστα ερμηνεύονται ευκολότερα από τον ανθρώπινο παράγοντα.

Λέξεις κλειδιά

Σύνθεση προγραμμάτων, νευρωνικά δίκτυα, τυπικές μέθοδοι, τεχνητή νοημοσύνη, γλώσσες προγραμματισμού, λογική, νευροσυμβολική σύνθεση

Abstract

Automatic code generation has been a perennial dream of computer science researchers. These efforts are concentrated in the research area of program synthesis. The original approach to program synthesis, parallel to the evolution of computer science, was formalistic and based on logic. This method was unwieldy and difficult to extend to other problems. Powerful neural networks have allowed researchers to move from strict formalisms to more abstract requirements, which has turned program synthesis into a more powerful tool but significantly more difficult to interpret. Solutions being designed and researched today combine neural networks with standard methods. This approach has already given better results which are even easier to interpret by the human factor.

Keywords

Program synthesis, neural networks, formal methods, artificial intelligence, programming language, logic, neurosymbolic synthesis

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Την τελευταία δεκαετία, το ενδιαφέρον για την πρόκληση της σύνθεσης προγραμμάτων (program synthesis) ολοένα και αυξάνεται. Μεγάλοι επιστήμονες όπως οι Church, Büchi, Landweber, Manna και Waldinger θεμελίωσαν την προσπάθεια αυτή και μέσω της προσφοράς του καθενός πήρε τη σύγχρονη μορφή της. Σήμερα η έρευνα των Armando Solar-Lezama, Gulwani, Chaudhuri, Dillig κ.α έχει ανοίξει νέες προοπτικές και δυνατότητες. Πλέον οι εφαρμογές της σύνθεσης προγραμμάτων έχουν εξαπλωθεί σε δεκάδες πεδία, ορισμένες έχουν σημειώσει τεράστια εμπορική επιτυχία και μπορούν να χρησιμοποιηθούν τόσο από προγραμματιστές λογισμικού όσο και απλούς χρήστες.

1.1 Σκοπός

Σκοπός της εργασίας είναι η μελέτη του προβλήματος της σύνθεσης προγραμμάτων, της εξέλιξής του μέσα στα χρόνια και των μεθόδων που χρησιμοποιούνται για την κατασκευή ενός synthesizer.

Από την αρχή της ανάπτυξης της τεχνητής νοημοσύνης η σύνθεση προγραμμάτων θεωρείται το άγιο δισκοπότηρο της επιστήμης των υπολογιστών. Ορισμένοι επιστήμονες όπως ο Amir Pnueli πιστεύουν ότι πρόκειται για ένα από τα κεντρικότερα προβλήματα της θεωρίας προγραμματισμού[1] καθώς θέτει ένα βασικό ερώτημα: Μπορεί ένας υπολογιστής να προγραμματίσει τον εαυτό του; Ο James Bornholt δίνει μια πολύ απλή περιγραφή της σύνθεσης προγραμμάτων ως εξής:

“Δε θα ήταν απλούστερο για μας να λέμε στον υπολογιστή *τι* θέλουμε να κάνει το πρόγραμμα και να αφήνουμε τον υπολογιστή να βρει το *πώς* θα το πετύχει αυτό; Είναι η απόλυτη αφαιρετική διαδικασία: Ο προγραμματιστής λέει στον υπολογιστή τι θέλει και όχι πώς θα γίνει αυτό που θέλει, έτσι απαλλάσσεται από τις λεπτομέρειες της υλοποίησης.”[2] Η σύνθεση προγραμμάτων συνίσταται λοιπόν στην αυτόματη παραγωγή προγραμμάτων τέτοιων ώστε να ικανοποιούνται κάποιες δοθείσες προδιαγραφές.

Στο συγκεκριμένο πόνημα γίνεται μια ολιστική παρουσίαση της σύνθεσης προγραμμάτων, οι εφαρμογές, οι προκλήσεις, η ανάπτυξη των τεχνικών που χρησιμοποιήθηκαν από την απαρχή του μέχρι σήμερα καθώς και οι πιθανές κατευθύνσεις εξέλιξης στο μέλλον.

1.2 Ιστορική αναδρομή

Ο Alan Turing είχε πρώτος το όραμα της αυτόματης παραγωγής κώδικα μόλις από το 1946. Ειδικότερα, στην αναφορά του προς το Εθνικό Εργαστήριο Φυσικής του Λονδίνου έγραφε ότι η

κατασκευή των πινάκων εντολών θα είναι μια διαδικασία ιδιαιτέρως ενδιαφέρουσα, δίχως τον κίνδυνο να γίνει κουραστική διότι τα τεχνικότερα μέρη της διαδικασίας μπορεί να τα αναλάβει το ίδιο το μηχάνημα[3].

Μετά από 10 χρόνια οι Newell, Simon και Shaw κατασκεύασαν το logic theorist, ένα πρόγραμμα στον υπολογιστή που μπορούσε να παράξει αυτόματα αποδείξεις θεωρημάτων. Βρήκε αποδείξεις για 38 από τα πρώτα 52 θεωρήματα του Principia Mathematica, ορισμένες μάλιστα κομψότερες από τις προϋπάρχουσες[4].

Τον επόμενο χρόνο, το 1957 ο Alonzo Church επιχείρησε να κατασκευάσει κυκλώματα –όχι ακόμη προγράμματα – τα οποία έπρεπε να ικανοποιούν συγκεκριμένες μαθηματικές προδιαγραφές. Θεωρείται από πολλούς ερευνητές η πρώτη φορά που θεμελιώθηκε το πρόβλημα της σύνθεσης προγραμμάτων και αναφέρονται σε αυτό ως "Το πρόβλημα του Church"[5].

Εικόνα 1: Alonzo Church

Την ίδια χρονιά κι ενώ βρισκόμαστε στην περίοδο της γέννησης της FORTRAN, στο paper "The FORTRAN automatic coding system" αναφέρεται ότι ο στόχος για τον καινούριο υπολογιστή της IBM είναι "να γράφει τον κώδικα για προβλήματα από μόνος του και να παράγει εξίσου καλά προγράμματα όσο και αυτά των προγραμματιστών (αλλά χωρίς τα λάθη).[6]

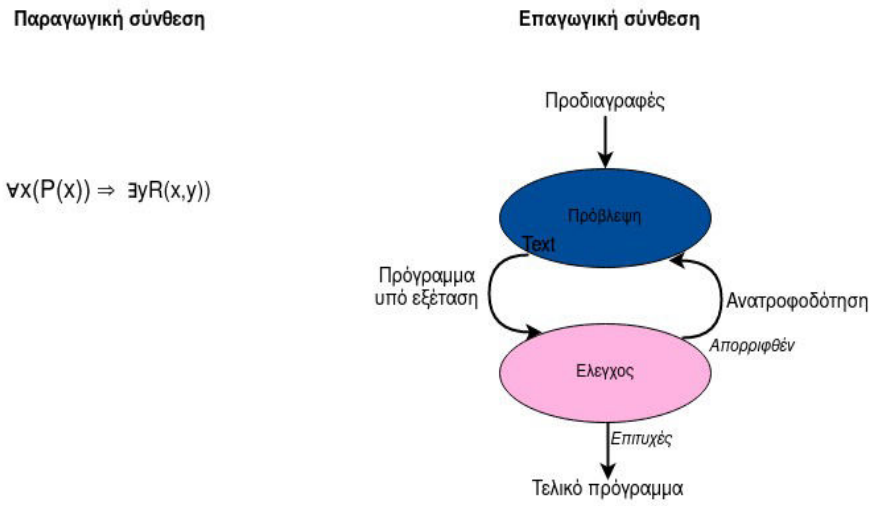
Λίγο αργότερα, τη δεκαετία του '60 το ενδιαφέρον γύρω από τη σύνθεση προγραμμάτων μεγάλωσε, ειδικότερα σε συνδυασμό με τη βοήθεια των πρώτων προγραμμάτων αυτόματης απόδειξης θεωρημάτων. Αναπτύχθηκε έτσι η ιδέα ότι αυτά τα προγράμματα μπορούν να βοηθήσουν στην κατασκευή αποδείξεων για τις δοθείσες προδιαγραφές, οι οποίες δίνονταν με τη μορφή λογικών προτάσεων. Από αυτές τις αποδείξεις ύστερα εκμαιευόταν το αντίστοιχο πρόγραμμα. Αξιοσημείωτη δουλειά σε αυτή την κατεύθυνση προσέφεραν οι Green, Manna και Waldinger ενώ την ίδια περίοδο οι Büchi και Landweber προσέγγισαν το πρόβλημα με τη χρήση αυτομάτων (automata-theoritic approach).

Μια ακόμη τεχνική που έγινε πολύ δημοφιλής αμέσως μετά ήταν η σύνθεση με βάση την μετατροπή (transformation-based synthesis).[7] Συγκεκριμένα η κεντρική ιδέα ήταν η εξής: Λαμβάνουμε τις προδιαγραφές που πρέπει να πληροί το πρόγραμμα ως μια λογική πρόταση υψηλού επιπέδου. Την υποβάλλουμε σε διαδοχικές μετατροπές χρησιμοποιώντας λογικούς κανόνες συμπερασμού μέχρι να καταλήξουμε στο ζητούμενο πρόγραμμα σε χαμηλού επιπέδου μορφή[1].

Μέχρι αυτό το σημείο η οπτική γωνία των ερευνητών ήταν καθαρά μαθηματική και ο τρόπος αντιμετώπισης του προβλήματος η παραγωγική συλλογιστική, σύμφωνα με την οποία ξεκινώντας από μια γενικότερη πρόταση καταλήγουμε σε μια ειδικότερη. Σημείο αφετηρίας λοιπόν ήταν η γενικότερη πρόταση (η οποία προέκυπτε από τις προδιαγραφές) η οποία θεωρούνταν αξιωματικά αληθής και προσπαθούσαν να φτάσουν σε μια ειδικότερη, στην απόδειξή της (η οποία οδηγούσε στο ζητούμενο πρόγραμμα) εφαρμόζοντας κανόνες.

Η παραγωγική λογική είχε ένα βασικό πλεονέκτημα και αυτό ήταν η απόδοσή της. Δεδομένου ότι η τεχνική αυτή εφαρμοζόταν ως επί το πλείστον πάνω σε μερικώς ανεπτυγμένα προγράμματα τα οποία ικανοποιούσαν τις προδιαγραφές σπάνια χρειαζόταν να απορριφθεί κάποιο πρόγραμμα ως μη έγκυρο. Έτσι δεν υπήρχε και η ανάγκη για οπισθοδρόμηση. Είχε όμως και αρκετά μειονεκτήματα.

Αρχικά, οι προδιαγραφές έπρεπε να εκφράζονται σε κάποια τυπική μορφή, μια μαθηματική συνάρτηση είτε μια πρόταση κατηγορικής λογικής. Δεν ήταν πάντα εύκολη η διαδικασία μετάφρασής τους σε μια τέτοια μορφή. Αντίστοιχα απαιτούνταν η πλήρης αξιωματοποίηση της γλώσσας στην οποία θα ήταν γραμμένο το τελικό πρόγραμμα. Επιπλέον, η εγκυρότητα των προδιαγραφών ήταν μία ακόμη αδυναμία αυτής της μεθόδου. Είναι πολλές φορές δύσκολο για τον χρήστη να εκφράσει ακριβώς τι θέλει να κάνει το πρόγραμμα με αποτέλεσμα να υπάρξουν περισσότερα από ένα προγράμματα που καλύπτουν τις προδιαγραφές. Δεν ήταν σπάνιο φαινόμενο η διαχείριση των προδιαγραφών να είναι τόσο περίπλοκη όσο και η παραγωγή του ίδιου του τελικού προγράμματος.



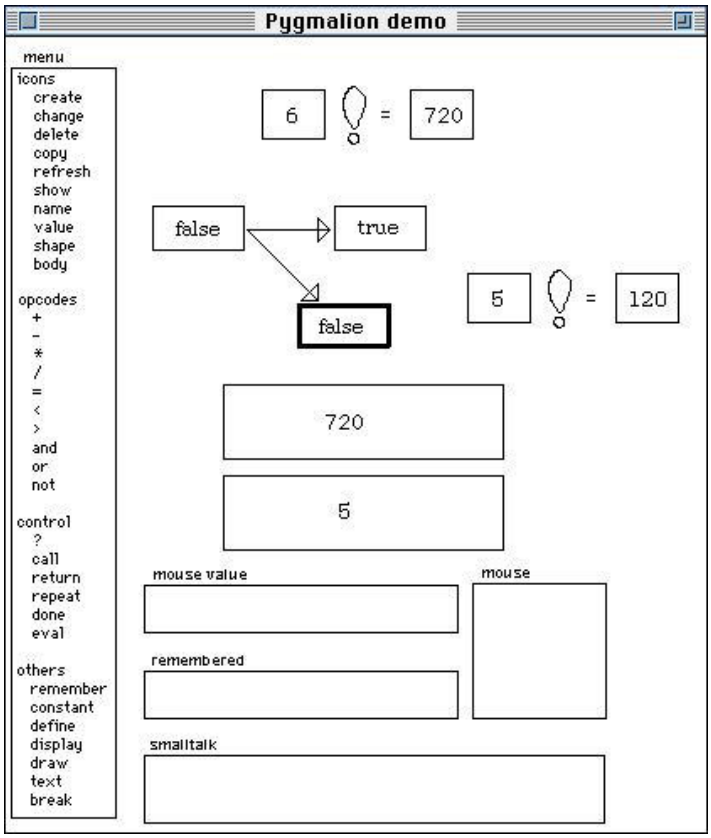
Εικόνα 2: Οι τρόποι λειτουργίας της παραγωγικής και της επαγωγικής σύνθεσης

Έτσι στα μέσα της δεκαετίας του 70 ξεκίνησε να μετατοπίζεται το ενδιαφέρον από την παραγωγική στην επαγωγική συλλογιστική, όσον αφορά την διαχείριση των προδιαγραφών. Σύμφωνα με την επαγωγική λογική εξάγεται ένα γενικότερο συμπέρασμα βασιζόμενο σε ειδικότερες και συγκεκριμένες παρατηρήσεις. Στην περίπτωση της σύνθεσης προγραμμάτων άρχισαν να χρησιμοποιούνται ως προδιαγραφές ζευγάρια εισόδου/εξόδου (programming by example ή PBE), περιγραφές προγράμματος (programming by demonstration ή PBD) ή σπανιότερα ακόμη και κάποιο πρόγραμμα ως σημείο αναφοράς.

Όσον αφορά τα ζευγάρια εισόδου/εξόδου, η ιδέα είναι: Δεδομένου ενός ζευγαριού τιμών ζητείται ένα πρόγραμμα που δεχόμενο ως είσοδο την πρώτη τιμή, δίνει ως έξοδο την δεύτερη, πχ factorial(6)=720. Η περίπτωση της περιγραφής προγράμματος μοιάζει με εκείνη των ζευγαριών εισόδου/εξόδου, αλλά με την διαφορά ότι δίνεται μια επιπλέον επίδειξη του τρόπου που θέλουμε να λειτουργεί το πρόγραμμα, πχ factorial(6) = 6 * (5 * (4 * (3 * (2 *1)))) = 720.

Ήδη από το 1970 ο Patrick Winston είχε δημοσιεύσει μια μελέτη σχετικά με την δυνατότητα γενίκευσης με βάση ένα συγκεκριμένο σετ δεδομένων, παρόλο που αυτή η εργασία δεν είχε σχέση με τον αυτόματο προγραμματισμό. Ύστερα, το 1975 κατασκευάστηκε το πρώτο PBD σύστημα, το Pygmalion. Επρόκειτο για ένα προγραμματιστικό περιβάλλον που χρησιμοποιούσε εικονίδια. Η αρχική εικόνα θα μετατρεπόταν διαδοχικά μέχρις ότου στην τελευταία θα υπήρχε η ζητούμενη

πληροφορία. Αυτό το πετύχαινε με τη βοήθεια ενός διαδραστικού επεξεργαστή με “μνήμη”. Δηλαδή αν πραγματοποιούνταν μια φορά μια επεξεργασία της εικόνας από το χρήστη, έπειτα μπορούσε να την επαναλάβει ο επεξεργαστής ακόμη και σε άλλο περιεχόμενο.



Εικόνα 3: Στιγμιότυπο λειτουργίας του συστήματος Pygmalion

Τον ίδιο χρόνο ο Shaw ολοκλήρωσε ένα σύστημα που μπορούσε να συνθέσει προγράμματα σε LISP βασισμένο μόνο σε ένα ζευγάρι εισόδου/εξόδου. Αργότερα, ο Summers το 1977 και ο Biermann το 1978 ανέπτυξαν επιπλέον μεθόδους για την παραγωγή προγραμμάτων σε LISP με τη χρήση ζευγαριών εισόδου/εξόδου. Παρόλο που τα συστήματα αυτά ήταν δυσπροσάρμοστα και τα προγράμματα που παρήγαγαν ήταν συγκεκριμένα και περιορισμένα, ήταν ωστόσο πολύ σημαντικά για την ανάπτυξη του πεδίου.

Τις επόμενες δεκαετίες το ενδιαφέρον για τη σύνθεση προγραμμάτων έφθινε και τη θέση του πήρε η ενασχόληση με το machine learning. Μέχρι τη δεκαετία του 90 κατά την οποία υπήρξαν ξανά κάποιες μελέτες πάνω στο αντικείμενο. Πιο συγκεκριμένα το 1994 ο John Koza ανέλυσε μια μέθοδο αυτόματης εξέλιξης κώδικα εμπνευσμένη από την θεωρία του Δαρβίνου για την εξέλιξη των ειδών. Η διαδικασία που ακολουθούνταν ήταν η ανάπτυξη ενός τυχαίου “πληθυσμού” προγραμμάτων και η διαρκής εξέλιξή τους σε νέες “γενεές” μέχρι την δημιουργία του ζητούμενου προγράμματος. Η εξέλιξη αυτή καθοριζόταν από συγκεκριμένες προδιαγραφές. Την ίδια περίοδο η Tessa Lau προσπάθησε να φέρει τεχνικές από το machine learning στη σύνθεση προγραμμάτων. Στόχος της ήταν να απομακρυνθούμε από τα περιοριστικά συστήματα που υπήρχαν μέχρι τότε και να προχωρήσουμε σε γενικότερες τεχνικές, ικανές να λειτουργήσουν σε πολλά προβλήματα PBD.

Γρήγορα όμως το ενδιαφέρον χάθηκε ξανά, καθώς τα αποτελέσματα δεν ήταν τα αναμενόμενα. Οι τεχνικές που υπήρχαν δεν ήταν αρκετά ευέλικτες ώστε να έχουν ουσιαστική πρακτική εφαρμογή. Χαρακτηριστικό είναι το γεγονός ότι η Tessa Lau δημοσίευσε το άρθρο “Γιατί τα PBD συστήματα αποτυγχάνουν: Μαθήματα για την εφαρμοζόμενη τεχνητή νοημοσύνη” (Why Programming-By-Demonstration Systems Fail: Lessons Learned for Usable AI)[32] το 2009, λίγο πριν ξεκινήσει το νέο κύμα ενθουσιασμού πάνω στο αντικείμενο[3].

Εικόνα 4: Armando Solar-Lezama

Σύμφωνα με αυτή την σύγχρονη προσέγγιση που ξεκίνησε λίγο πριν το 2010, η είσοδος που δίνεται στα συστήματα σύνθεσης είναι όχι μόνο οι προδιαγραφές αλλά και ένα σετ συντακτικών κανόνων κάποιας γραμματικής (syntax-guided synthesis ή syGuS).[8] Έτσι το ζητούμενο πρόγραμμα πρέπει όπως και στις προηγούμενες μεθόδους να πληροί κάποιες προδιαγραφές, αλλά τώρα υπάρχει και μια συντακτική κατεύθυνση που πρέπει να ακολουθήσει η δημιουργία του. Αυτό είναι πολύ ωφέλιμο γιατί καταρχάς μειώνει τον χώρο αναζήτησης και δεύτερον τα παραγόμενα προγράμματα είναι πιο ευνόητα γιατί προέρχονται από μία γραμματική. Πρωτοπόρος αυτής της τεχνικής ήταν ο Armando Solar-Lezama το 2008 με το Sketching.[9] Σύμφωνα με αυτό, ο προγραμματιστής εκφράζει την υψηλού επιπέδου οπτική του τι πρέπει να κάνει το ζητούμενο πρόγραμμα, δίνοντας μη ολοκληρωμένα κομμάτια κώδικα. Ο synthesizer ασχολείται με τις χαμηλού επιπέδου λεπτομέρειες, ολοκληρώνει δηλαδή το πρόγραμμα με βάση κάποιες προδιαγραφές που δίνει ο προγραμματιστής.

Ακόμα ένα παράδειγμα είναι το FlashFill (2011) του Sumit Gulwani.[10] Πρόκειται για μια τεχνική μετατροπής string η οποία αποτελεί πια μια λειτουργία του Microsoft Excell. Ο χρήστης συμπληρώνει στο πρώτο κελί την μετατροπή που θέλει να γίνει και τα υπόλοιπα κελιά συμπληρώνονται αυτόματα.

Τα τελευταία χρόνια υπάρχουν αρκετές δομές meta-synthesis που υποστηρίζουν εφαρμογές σύνθεσης προγραμμάτων. Ο χρήστης δηλαδή δίνει κάποιες πληροφορίες στο σύστημα, όπως μία γλώσσα ή μια γραμματική που πρέπει να υποστηρίζεται από το πρόγραμμα που θα παραχθεί και κάποιες προδιαγραφές (όπως παραδείγματα εισόδου/εξόδου) και το σύστημα αναλαμβάνει να δώσει έναν synthesizer κατάλληλο για την εκάστοτε εφαρμογή. Παραδείγματα αυτής της προσέγγισης είναι αρχικά το Sketch που αναφέρθηκε πρωτύτερα αλλά και άλλα συστήματα, όπως το Rosette του 2013 και το Prose του 2015.

1.3 Δομή της εργασίας

Η εργασία είναι οργανωμένη στα εξής κεφάλαια:

- 2ο Κεφάλαιο: Αναλύεται το απαραίτητο θεωρητικό υπόβαθρο ώστε να γίνουν κατανοητές οι έννοιες που αναφέρονται.
- 3ο Κεφάλαιο: Εξηγείται αναλυτικά η φιλοσοφία της σύνθεσης προγραμμάτων, οι εφαρμογές του και οι προκλήσεις που παρουσιάζονται.

- 4ο Κεφάλαιο: Παρατίθενται οι κλασικές προσεγγίσεις στο πρόβλημα της σύνθεσης προγραμμάτων.
- 5ο Κεφάλαιο: Γίνεται εκτενής παρουσίαση κάποιων σύγχρονων μεθόδων σύνθεσης προγραμμάτων.
- 6ο Κεφάλαιο: Επίλογος και σχετικά συμπεράσματα

Η εργασία ολοκληρώνεται με την παρουσίαση της βιβλιογραφίας που χρησιμοποιήθηκε.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Σε αυτή την ενότητα εισάγονται οι απαραίτητες μαθηματικές έννοιες και τα αναγκαία θεωρητικά στοιχεία για την κατανόηση των συστημάτων σύνθεσης προγραμμάτων που αναλύονται στη συνέχεια καθώς και των συμπερασμάτων που προκύπτουν από την εργασία.

2.1 Λογική

Η επιστήμη της λογικής μελετά τους τρόπους θεμελίωσης ενός συλλογισμού. Επιδιώκει να ορίσει τους κανόνες που διαχωρίζουν έναν βάσιμο συλλογισμό από έναν σαθρό, κανόνες τους οποίους στη συνέχεια απλοποιεί και συστηματοποιεί.[11] Η λογική αποτελεί σημαντικό εργαλείο πολλών πνευματικών δραστηριοτήτων, όπως της φιλοσοφίας, των μαθηματικών και της πληροφορικής.[12]

Η λογική αρχικά καθιερώθηκε ως κλάδος της φιλοσοφίας από τον Αριστοτέλη, ωστόσο μελετήθηκε και από άλλους αρχαίους πολιτισμούς εκτός του ελληνικού, όπως τον κινεζικό, τον ινδικό και τον περσικό. Κατά τα μεσαιωνικά χρόνια μάλιστα η λογική μαζί με τη ρητορική και τη γραμματική συνέθετε την trivium, το τρίπτυχο δηλαδή που αποτέλεσε το θεμέλιο της κλασικής εκπαίδευσης της εποχής.

Στη συνέχεια αναλύονται ορισμένες μορφές της λογικής που έχουν βοηθήσει την ανάπτυξη της επιστήμης των υπολογιστών.

Τυπική Λογική

Η τυπική λογική παρέχει έναν τρόπο για την αποσαφήνιση και την τυποποίηση της διαδικασίας της ανθρώπινης σκέψης.[12] Πιο συγκεκριμένα εξετάζει τους τρόπους δόμησης των επιχειρημάτων, των δηλώσεων, των προτάσεων και τις μορφές συμπερασματολογίας χρησιμοποιώντας τυπικές δομές και φόρμες. Αυτές οι δομές εγγυώνται έγκυρα συμπεράσματα ή παράγουν τις απαραίτητες προϋποθέσεις για την εξαγωγή ενός συμπεράσματος. Δηλαδή τα συμπεράσματα προκύπτουν ως ειδικές εφαρμογές γενικευμένων και αφηρημένων κανόνων. Η πρώτη γνωστή μελέτη της τυπικής λογικής παρατηρείται στα έργα του Αριστοτέλη.[13]

πχ

Όποτε βρέχει παίρνω ομπρέλα. *(Δήλωση)*

Σήμερα βρέχει *(Δήλωση)*

επομένως, σήμερα θα πάρω ομπρέλα. *(Συμπέρασμα)*

Συμβολική Λογική / Μαθηματική Λογική

Η μαθηματική λογική είναι ένα πεδίο μεταξύ των μαθηματικών και της επιστήμης των υπολογιστών ωστόσο είναι στενά συνδεδεμένο και με τη φιλοσοφική λογική.[14] Μελετά τις εφαρμογές της τυπικής λογικής σε διάφορες περιοχές των μαθηματικών και εξετάζει τους έγκυρους ισχυρισμούς και συλλογισμούς με τη βοήθεια μαθηματικών εργαλείων. Συχνά αποκαλείται και συμβολική λογική λόγω της χρήσης συμβολικών αναπαραστάσεων. Χρησιμοποιούνται δηλαδή σύμβολα αντί λέξεων της φυσικής γλώσσας προκειμένου να μελετηθούν οι ισχυρισμοί ανεξάρτητα από το περιεχόμενο των λογικών προτάσεων, εκφράζοντάς τους στη συμβολική τους μορφή. Θα μπορούσε να πει κανείς ότι η συμβολική λογική είναι μια στενογραφία της λογικής.

Προτασιακή Λογική

Η προτασιακή λογική (ή προτασιακός λογισμός) είναι η απλούστερη μορφή της μαθηματικής λογικής. [15] Στην προτασιακή λογική κάθε γεγονός του πραγματικού κόσμου αναπαρίσταται με μία λογική πρόταση η οποία χαρακτηρίζεται από μία λογική τιμή, μπορεί να είναι είτε αληθής είτε ψευδής.[16] Οι λογικές προτάσεις μπορούν να συνδυαστούν με τη χρήση λογικών συμβόλων και η προτασιακή λογική αναλύει τους συσχετισμούς της αληθοτιμής των αρχικών προτάσεων και των τελικών. Στον πίνακα 1 παρουσιάζονται οι σύνδεσμοι της προτασιακής λογικής και ο πίνακας 2 είναι ένας πίνακας αληθείας, δείχνει δηλαδή την τιμή αληθείας της τελικής πρότασης που σχηματίζεται από κάθε λογικό σύνδεσμο και απλές λογικές προτάσεις.[17]

Σύνδεσμοι	Ονομασία
¬	Άρνηση (Λογικό Οχι/Not)
∧	Σύζευξη (Λογικό Και/And)
∨	Διάζευξη (Λογικό Ή/Or)
→	Συνεπαγωγή (Εάν τότε/If then)
↔	Ισοδυναμία (Αν και μόνο αν/If and only if)

Πίνακας 1: Οι σύνδεσμοι του προτασιακού λογισμού

p	q	¬p	p∧q	p∨q	p→q	p↔q
Α	Α	Ψ	Α	Α	Α	Α
Α	Ψ	Ψ	Ψ	Α	Ψ	Ψ
Ψ	Α	Α	Ψ	Α	Α	Ψ
Ψ	Ψ	Α	Ψ	Ψ	Α	Α

Πίνακας 2: Πίνακας αληθείας

Λογική Πρώτου Βαθμού (ή Κατηγορηματική Λογική)

Όπως είδαμε παραπάνω η προτασιακή λογική μελετά τη δομή και τη διατύπωση επιχειρημάτων (όπως και κάθε μορφή τυπικής λογικής) αλλά μόνο όσον αφορά λογικά αντικείμενα, λογικές προτάσεις δηλαδή που φέρουν μία τιμή αληθείας, ψευδής ή αληθής.[18] Η λογική πρώτου βαθμού επεκτείνει την προτασιακή λογική δίνοντας μια μεγαλύτερη ελευθερία έκφρασης. Συγκεκριμένα χρησιμοποιεί και μη λογικά αντικείμενα (όπως αριθμούς και σύνολα) αλλά και σχέσεις μεταξύ των αντικειμένων. Επίσης, ένα σημαντικό εργαλείο της λογικής πρώτου βαθμού είναι οι ποσοδείκτες με τη βοήθεια των οποίων επιτυγχάνεται η αναφορά σε άπειρο πλήθος αντικειμένων, κάτι που δεν ισχύει για την προτασιακή λογική. Τέλος, μια σημαντική επέκταση είναι και η χρήση μεταβλητών, κάτι που επιτρέπει την αναπαράσταση “γενικής” γνώσης. Οι λογικοί σύνδεσμοι της πρωτοβάθμιας λογικής είναι ίδιοι με αυτούς της προτασιακής λογικής, φαίνονται δηλαδή στον πίνακα 1. Τους ποσοδείκτες τους βλέπουμε στον πίνακα 3. [19]

Ποσοδείκτες	Ονομασία
$\forall$	Καθολικός ποσοδείκτης (για κάθε)
$\exists$	Υπαρξιακός ποσοδείκτης (υπάρχει)

Πίνακας 3: Ποσοδείκτες λογικής πρώτου βαθμού

Ακολουθεί ένα παράδειγμα λογικής πρώτου βαθμού:

Έστω μία πρόταση που σε φυσική γλώσσα εκφράζεται ως: “Κάθε άνθρωπος έχει μητέρα”

Σε λογική πρώτου βαθμού αυτή η πρόταση γίνεται :
$\forall X$ άνθρωπος$(X) \rightarrow \exists Y$ μητέρα(Y,X) και διαβάζεται ως εξής:
Για κάθε μεταβλητή X που έχει το χαρακτηριστικό “άνθρωπος” (δηλαδή για κάθε άνθρωπο X) υπάρχει μια μεταβλητή Y που έχει το χαρακτηριστικό “μητέρα του X”.

2.2 Τυπικά συστήματα

Τυπικό σύστημα ονομάζεται μια αφηρημένη δομή που περιλαμβάνει μια τυπική γλώσσα και ένα σύνολο κανόνων συμπερασμού. [20]Η γλώσσα επιτρέπει στο σύστημα τον σχηματισμό εκφράσεων, κάποιες από τις οποίες ονομάζονται αξιώματα, γιατί δεχόμαστε εξαρχής ότι είναι αληθείς και κάποιες άλλες θεωρήματα γιατί προκύπτουν από τα αξιώματα και τους κανόνες συμπερασμού. Τα τυπικά συστήματα χρησιμοποιούνται είτε για την περιγραφή φαινομένων του περιβάλλοντος είτε μελετούνται για κάποιες εγγενείς τους ιδιότητες. Η προτασιακή και η κατηγορηματική λογική θα μπορούσαν να θεωρηθούν δύο απλά τυπικά συστήματα.

Τυπικές γλώσσες

Κάθε γλώσσα ορίζεται ως ένα σύνολο συμβολοσειρών που προέρχονται από ένα πεπερασμένο αλφάβητο. Οι συμβολοσειρές αυτές καθορίζονται από ένα συγκεκριμένο σύνολο κανόνων. Το χαρακτηριστικό που διαφοροποιεί τις τυπικές γλώσσες είναι ότι διαμορφώνονται με βάση

μαθηματικούς κανόνες και τύπους ή τύπους που μπορούν να ερμηνευθούν από μια μηχανή. Παραδείγματα τυπικών γλωσσών είναι οι γλώσσες προγραμματισμού. Κάθε γλώσσα (φυσική ή τυπική) προσδιορίζεται από το αλφάβητο, το λεξιλόγιο, τη γραμματική και τη σημασιολογία της.

Αλφάβητο ονομάζεται το μη κενό σύνολο των στοιχείων που χρησιμοποιούνται από μια γλώσσα. Περιλαμβάνει σύμβολα, γράμματα και ψηφία. Ένα υποσύνολο των ακολουθιών που σχηματίζουν τα στοιχεία του αλφαβήτου αποτελεί το λεξιλόγιο της γλώσσας. Οι ακολουθίες αυτές ονομάζονται λέξεις στην περίπτωση των φυσικών γλωσσών και tokens στην περίπτωση των τυπικών γλωσσών.

2.2.1 Γραμματική

Η γραμματική μιας τυπικής γλώσσας είναι το σύνολο των κανόνων που καθορίζουν την ορθότητα της διάταξης και της σύνδεσης των tokens σύμφωνα με το συντακτικό της γλώσσας. Έχει να κάνει μόνο με τη μορφή της γλώσσας και όχι με το νόημα ή τη χρήση των συμβολοσειρών.

Μία τυπική γραμματική G(V, T, P, S) συντίθεται από τα εξής στοιχεία:

- Ένα πεπερασμένο σύνολο V από μη τερματικά σύμβολα
- Ένα πεπερασμένο σύνολο T από τερματικά σύμβολα
- Ένα πεπερασμένο σύνολο P από κανόνες παραγωγής
- Ένα αρχικό σύμβολο S

όπου

- $V \cap T = \emptyset$
- $S \in V$

Τα τερματικά σύμβολα T της γραμματικής G ταυτίζονται με το αλφάβητο της γλώσσας L που περιγράφει η γραμματική. Με άλλα λόγια αν έχουμε μια γλώσσα της οποίας το αλφάβητο είναι το {a,b} τότε τα τερματικά σύμβολα της γραμματικής της θα είναι τα a και b. [21]

Τα μη τερματικά σύμβολα V μιας τυπικής γραμματικής G είναι σύμβολα τα οποία μέσω των κανόνων παραγωγής αντικαθίστανται από άλλα, γι’ αυτό και ονομάζονται και συντακτικές μεταβλητές. Το αρχικό σύμβολο S είναι ένα από τα μη τερματικά σύμβολα.

Τέλος οι κανόνες παραγωγής καθορίζουν ποιο σύμβολο παράγεται από ποιο.[22] Κάθε κανόνας παραγωγής περιλαμβάνει μια κεφαλή ή αριστερό μέρος (τη συμβολοσειρά που θα αντικατασταθεί) και ένα σώμα ή δεξί μέρος (τη συμβολοσειρά που θα την αντικαταστήσει). Οι κανόνες παραγωγής ανάλογα με την αυστηρότητα ή την ευελιξία που έχουν επιτρέπουν την έκφραση λιγότερων ή περισσότερων αντίστοιχα τυπικών γλωσσών, έχουν δηλαδή μικρότερη ή μεγαλύτερη παραγωγική ισχύ. Ο Νόαμ Τσόμσκι το 1956 ταξινόμησε τις τυπικές γραμματικές ανάλογα με την παραγωγική τους ισχύ σε μία ιεραρχία που ονομάστηκε ιεραρχία Τσόμσκι. Πρόκειται για μια κατηγοριοποίηση των γραμματικών που φάνηκε εξαιρετικά χρήσιμη στην επιστήμη των υπολογιστών:

Γενικές Γραμματικές

Οι γενικές γραμματικές ανήκουν στην ευρύτερη κατηγορία γραμματικών, τις γραμματικές τύπου 0. Οι παραγωγικοί κανόνες των γενικών γραμματικών είναι της μορφής:

$\alpha \rightarrow \beta$, $\alpha \neq \epsilon$, όπου $\alpha, \beta \in (V \cup T)^*$

Αυτό σημαίνει ότι τόσο στο δεξί όσο και στο αριστερό μέρος μπορεί να βρίσκεται οποιαδήποτε συμβολοσειρά τερματικών συμβόλων. Δεν υπάρχει κανένας περιορισμός όσον αφορά τις συμβολοσειρές των κανόνων παραγωγής, με εξαίρεση το ότι δεν παράγεται καμία συμβολοσειρά από την κενή.

Γραμματικές Με Συμφραζόμενα

Στις γραμματικές τύπου 1 βρίσκονται οι γραμματικές με συμφραζόμενα. Οι κανόνες παραγωγής είναι της μορφής:

$\alpha A \beta \rightarrow \alpha \gamma \beta$, όπου $\alpha, \beta, \gamma \in (V \cup T)^*$, $A \in V$, $\gamma \neq \epsilon$ και επιτρέπεται $S \rightarrow \epsilon$

Σύμφωνα με αυτό τον κανόνα, οι συμβολοσειρές α, β και γ μπορούν να περιλαμβάνουν οποιαδήποτε σύμβολα της αλφαβήτου της γραμματικής. Το Α είναι μη τερματικό σύμβολο (συνήθως με κεφαλαία συμβολίζουμε τα μη τερματικά σύμβολα) και τέλος η συμβολοσειρά γ δεν μπορεί να είναι κενή.[23]

Διαισθητικά αυτός ο κανόνας σημαίνει ότι αν το σύμβολο Α πλαισιώνεται από τα συμφραζόμενα α και β τότε μπορεί να αντικατασταθεί από τη συμβολοσειρά τερματικών και μη τερματικών συμβόλων γ.

Γραμματικές Χωρίς Συμφραζόμενα

Οι γραμματικές χωρίς συμφραζόμενα ανήκουν στις γραμματικές τύπου 2. Η μορφή των κανόνων παραγωγής τους είναι:

$A \rightarrow \alpha$, όπου $A \in V$ και $\alpha \in (V \cup T)^*$

Το Α είναι και πάλι μη τερματικό σύμβολο ενώ το α είναι συμβολοσειρά αποτελούμενη από οποιαδήποτε τερματικά σύμβολα (μπορεί να είναι και κενή).

Η διαφορά των γραμματικών χωρίς συμφραζόμενα από αυτές με συμφραζόμενα είναι ότι σε αυτή την περίπτωση η αντικατάσταση μπορεί να γίνει ανεξαρτήτως των συμφραζομένων που μπορεί να συνοδεύουν το Α, γι’ αυτό και τα συμφραζόμενα εδώ είναι κενά.

Παρότι οι γραμματικές χωρίς συμφραζόμενα δεν έχουν την εκφραστική ισχύ των προαναφερθέντων, οι γλώσσες που αναπαριστούν έχουν αρκετή ισχύ για να περιγράψουν τις περισσότερες γλώσσες προγραμματισμού, όπως επίσης αριθμητικές εκφράσεις και υποσύνολα φυσικών γλωσσών. Είναι πολύ σημαντικές για την επιστήμη των υπολογιστών γιατί δίνουν μια ακριβή και ταυτόχρονα εύκολα κατανοητή περιγραφή του συντακτικού μιας γλώσσας προγραμματισμού. [24]Συνήθως οι γλώσσες χωρίς συμφραζόμενα αναπαρίστανται σε μορφή Backus-Naur. [25]Ενα παράδειγμα ενός κανόνα παραγωγής σε Backus-Naur είναι:

<digit> ::= 0|1|2|3|4|5|6|7|8|9

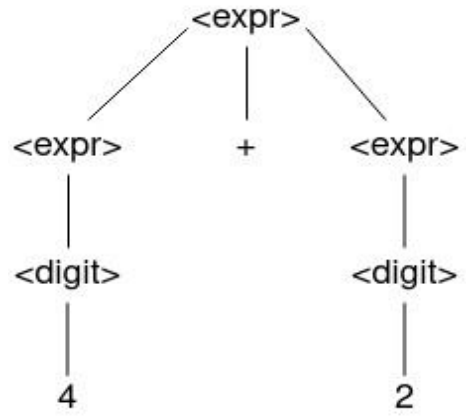
Αυτός ο κανόνας μας λέει ότι ένα ψηφίο μπορεί να οριστεί ως ένα από τα: 0,1,2,3,4,5,6,7,8,9. Τα σύμβολα “<”,”>” χρησιμοποιούνται για να περιγράψουν ένα μη τερματικό σύμβολο, στο συγκεκριμένο παράδειγμα το digit είναι μη τερματικό και τα 0,1,2,3,4,5,6,7,8,9 είναι τα τερματικά.

Ένας ακόμη τρόπος περιγραφής μιας γραμματικής χωρίς συμφραζόμενα είναι μέσω ενός συντακτικού δέντρου. Πρόκειται για μια δενδρική δομή που περιγράφει την συντακτική δομή μιας γραμματικής και δείχνει γραφικά πώς ένα αρχικό σύμβολο μιας γραμματικής οδηγεί σε μια συμβολοσειρά της γλώσσας.

πχ έστω μια απλή γραμματική που κάνει την πρόσθεση δύο αριθμών:

<exp> ::= <exp> + <exp>|<digit>
<digit> ::= 0|1|2|3|4|5|6|7|8|9

και μια έκφραση της γραμματικής “4+2”. Το συντακτικό δέντρο που περιγράφει αυτή την έκφραση είναι το εξής:



Εικόνα 5: Συντακτικό δέντρο

Κανονικές Γραμματικές

Τέλος, στις γραμματικές τύπου 3 ανήκουν οι κανονικές γραμματικές.[26] Πρόκειται για την πιο αδύναμη μορφή τυπικών γλωσσών σε παραγωγική ισχύ και είναι ευρέως χρησιμοποιούμενες. Οι κανόνες παραγωγής τους είναι δεξιογραμμικές (right-linear) ή αριστερογραμμικές (left-linear). Στην πρώτη περίπτωση οι κανόνες παραγωγής είναι:

$A \rightarrow u$, $A \rightarrow uB$, $A \rightarrow \varepsilon$

ενώ στη δεύτερη:

$A \rightarrow u$, $A \rightarrow Bu$, $A \rightarrow \varepsilon$ όπου $A,B \in V$ και $u \in T^*$

Βλέπουμε δηλαδή ότι το αριστερό μέρος περιλαμβάνει μόνο ένα μη τερματικό σύμβολο και το δεξί περιλαμβάνει μια ακολουθία τερματικών συμβόλων πλαισιωμένη από ένα μη τερματικό σύμβολο είτε στα αριστερά (left-linear) είτε στα δεξιά(right-linear).

Μια τυπική γραμματική ονομάζεται κανονική όταν μπορεί να αποφασιστεί εάν μια συμβολοσειρά ανήκει ή όχι στη γλώσσα που αναπαριστά με τη χρήση ενός αλγορίθμου ή ισοδύναμα μιας μηχανής πεπερασμένης μνήμης εξετάζοντας διαδοχικά όλα τα σύμβολα.

Ένας άλλος τρόπος ορισμού των κανονικών γραμματικών είναι ότι οι γλώσσες που αναπαριστούν μπορούν να περιγραφούν από μία κανονική έκφραση. Μια κανονική έκφραση (regular expression ή regex) είναι μια ακολουθία χαρακτήρων και συμβόλων που καθορίζει ένα συγκεκριμένο μοτίβο. Κάποια από τα σημαντικότερα σύμβολά τους είναι τα εξής:

Σύμβολο	Σημασία
a?	Το σύμβολο a καμία ή μία φορά
a*	Το σύμβολο a μηδέν ή περισσότερες φορές
a+	Το σύμβολο a μία ή περισσότερες φορές
a|b	Είτε το σύμβολο a είτε το b

Πίνακας 4: Σύμβολα κανονικών εκφράσεων

πχ

η κανονική έκφραση (0*10*) περιγράφει μια γλώσσα L η οποία περιλαμβάνει ακριβώς έναν άσσο και οσαδήποτε μηδενικά πριν και μετά από αυτόν, δηλαδή L= {1, 01, 1000000, 0010, … }

Εικόνα 6: Οι κατηγορίες γραμματικών ανάλογα με την παραγωγική τους ισχύ

2.2.3 Σημασιολογία

Η σημασιολογία (semantics) είναι το σύνολο των κανόνων που καθορίζει το νόημα των συμβολοσειρών και κατ' επέκταση των εκφράσεων μιας γλώσσας. Επιτυγχάνεται με τη βοήθεια μιας μαθηματικής μελέτης των λειτουργιών που πραγματοποιεί μια τυπική γλώσσα. Στη συνέχεια αυτοί οι μαθηματικοί υπολογισμοί αποδίδονται στη συντακτική ανάλυση της γλώσσας.

Η σημασιολογία δηλαδή περιγράφει τις διαδικασίες που πραγματοποιούνται από έναν υπολογιστή όταν εκτελεί ένα πρόγραμμα στην υπό μελέτη γλώσσα. Αυτό γίνεται είτε δίνοντας μία σχέση εισόδου-εξόδου ενός προγράμματος είτε εξηγώντας τον τρόπο εκτέλεσης του προγράμματος.

2.3 Τεχνητή νοημοσύνη

Η αναπαράσταση της γνώσης αποτελεί πεδίο έρευνας τις τελευταίες δεκαετίες. Οι αριθμητικοί υπολογισμοί είναι ένα πεδίο στο οποίο οι ανθρώπινες νοητικές λειτουργίες έχουν υποκατασταθεί με επιτυχία από τον Η/Υ κάτι που αναδεικνύει τη δυνατότητα υλοποίησης περαιτέρω ανεπτυγμένων συστημάτων τα οποία και θα επιτυγχάνουν μεγάλες ταχύτητες λογισμού σε ακόμα πιο εξεζητημένες εφαρμογές.

Στα πλαίσια αυτής της λογικής ξεκίνησε η πρόοδος της τεχνητής νοημοσύνης. Η τεχνητή νοημοσύνη ορίζεται λοιπόν ως ο κλάδος της επιστήμης των υπολογιστών που ασχολείται με τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς τα οποία υπονοούν έστω και στοιχειώδη ευφυΐα: μάθηση, προσαρμοστικότητα, εξαγωγή συμπερασμάτων, κατανόηση από συμφραζόμενα, επίλυση προβλημάτων κλπ. Ο John Mccarthy όρισε τον τομέα αυτόν ως «*επιστήμη και μεθοδολογία της δημιουργίας νοημόνων μηχανών*». [27]

Ένας από τους κλάδους της τεχνητής νοημοσύνης είναι και η αυτοματοποίηση λογισμικού στην οποία ανήκει και η σύνθεση προγραμμάτων μεταξύ άλλων. Θα ήταν ωφέλιμη μια αναφορά σε έννοιες σχετικές με τη σύνθεση προγραμμάτων ώστε να γίνουν περισσότερο κατανοητές οι διαφορές τους.

2.3.1 Μεταγλωττιστές

Μια από τις πρώτες μορφές αυτοματοποίησης λογισμικού είναι οι μεταγλωττιστές. Ο μεταγλωττιστής είναι ένα ειδικό πρόγραμμα που μεταφράζει ένα πρόγραμμα από μία γλώσσα σε μία άλλη. Συνήθως λαμβάνει το πρόγραμμα σε μια γλώσσα υψηλού επιπέδου, κατανοητή από τον άνθρωπο και τη μεταφράζει σε μία γλώσσα χαμηλού επιπέδου, όπως γλώσσα μηχανής, με σκοπό να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.

Παρατηρείται λοιπόν, ότι υπάρχουν αρκετές ομοιότητες μεταξύ των μεταγλωττιστών και της σύνθεσης προγραμμάτων. Και στις δύο περιπτώσεις παράγεται κώδικας με βάση κάποια υψηλού επιπέδου περιγραφή.

Ωστόσο, από τη σύνθεση προγραμμάτων αναμένεται κάτι περισσότερο από μια απλή μετάφραση ενός προγράμματος από μια μορφή σε μία άλλη, αναμένεται να βρει με ποιον τρόπο θα το επιτύχει

αυτό. Μια ακόμη σημαντική διαφορά είναι ότι η σύνθεση προγραμμάτων περιλαμβάνει αναζήτηση στη διαδικασία που ακολουθεί, ώστε να βρεθεί το κατάλληλο πρόγραμμα για τις δοθείσες προδιαγραφές, κάτι που δεν ισχύει για τους μεταγλωττιστές. Παρόλα αυτά ο διαχωρισμός δεν είναι ξεκάθαρος και απόλυτος μιας και οι μεταγλωττιστές νέας εποχής είναι ικανοί να εφαρμόσουν κάποιες από αυτές τις λειτουργίες.

2.3.2 Λογικός προγραμματισμός

Ένας άλλος κλάδος της τεχνητής νοημοσύνης που είναι κοντινός με τη σύνθεση προγραμμάτων είναι ο λογικός προγραμματισμός. Ο λογικός προγραμματισμός βασίζεται σε μεγάλο βαθμό στη Μαθηματική Λογική, αλλά την προεκτείνει εισάγοντας νέες μεθόδους αυτοματοποιημένης συμπερασματολογίας. Πιο συγκεκριμένα, ο λογικός προγραμματισμός είναι μια υπολογιστική κωδικοποίηση που συνδυάζει δύο κύριες αρχές:

• χρησιμοποιεί λογική για να αναπαραστήσει τις προδιαγραφές
• χρησιμοποιεί το μηχανισμό των αποδείξεων και των συμπερασμάτων για να τις επεξεργαστεί

Η διαφορά του από τη σύνθεση προγραμμάτων είναι αρχικά η γενικευμένη αντιμετώπιση που έχει. Η προσπάθεια δηλαδή επικεντρώνεται στην εύρεση ενός γενικού αλγορίθμου που θα λύνει κάθε πρόβλημα. Επιπλέον, το αποτέλεσμα δίνεται στον χρόνο εκτέλεσης.

2.3.3 Μηχανική μάθηση

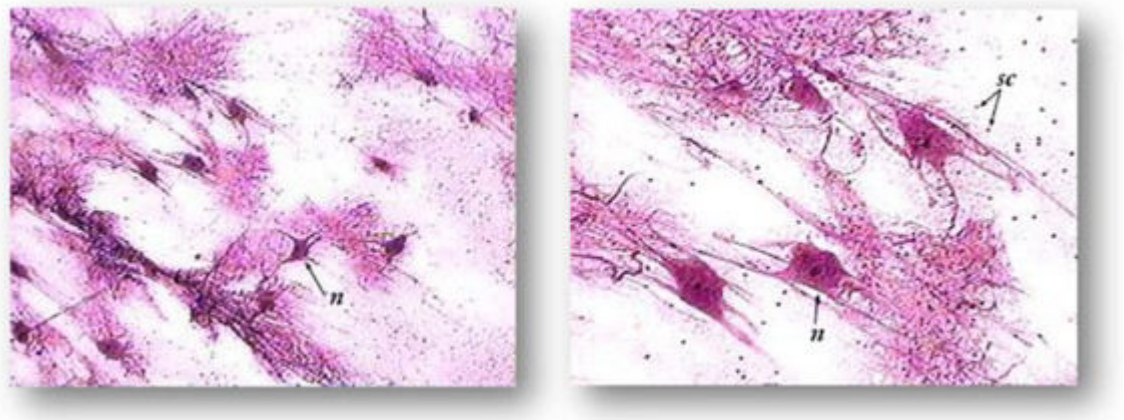
Η μηχανική μάθηση είναι ένας τρίτος τομέας αυτοματοποίησης που σχετίζεται με τη σύνθεση προγραμμάτων. Η μηχανική μάθηση διερευνά τους τρόπους εντοπισμού κατάλληλων συναρτήσεων, τέτοιων ώστε να ικανοποιούν δοθέντα σύνολα δεδομένων. Η μηχανική μάθηση λοιπόν, θα μπορούσε να θεωρηθεί μια υποκατηγορία της σύνθεση προγραμμάτων όπου οι προδιαγραφές δίνονται με τη μορφή δεδομένων.

Ωστόσο, όσον αφορά τη μηχανική μάθηση ο χώρος των συναρτήσεων που εξετάζει ο αλγόριθμος είναι πολύ αυστηρά καθορισμένος, ενώ στην προσέγγιση της σύνθεσης προγραμμάτων το ενδιαφέρον επικεντρώνεται σε γενικότερους αλγορίθμους που μπορούν να χρησιμοποιηθούν σε γενικές κλάσεις προβλημάτων. Ένα δεύτερο διαχωριστικό σημείο είναι ότι η μηχανική μάθηση μπορεί να ανταπεξέλθει σε περιπτώσεις όπου τα δεδομένα που δίνονται είναι “θορυβώδη” (noisy data), δηλαδή ανάμεσά τους υπάρχουν δεδομένα που δε βγάζουν νόημα, έχουν με κάποιον τρόπο διαφθαρεί ή γενικότερα δεν μπορούν να γίνουν αντιληπτά και χρησιμοποιήσιμα από μια μηχανή. Αντιθέτως, για να πραγματοποιηθεί επιτυχώς σύνθεση προγραμμάτων οι προδιαγραφές θα έπρεπε να ήταν απολύτως ακριβείς. Αυτή η διαφορά όμως σήμερα δεν ισχύει και τόσο έντονα μιας και υπάρχει μεγάλη εξέλιξη σε αλγορίθμους που επιτρέπουν στη σύνθεση προγραμμάτων να λειτουργεί επιτυχώς ακόμα και χωρίς οι προδιαγραφές να είναι απολύτως σωστές ή ακόμα και να έχουν ορισμένες ελλείψεις. [3]

2.4 Νευρωνικά δίκτυα

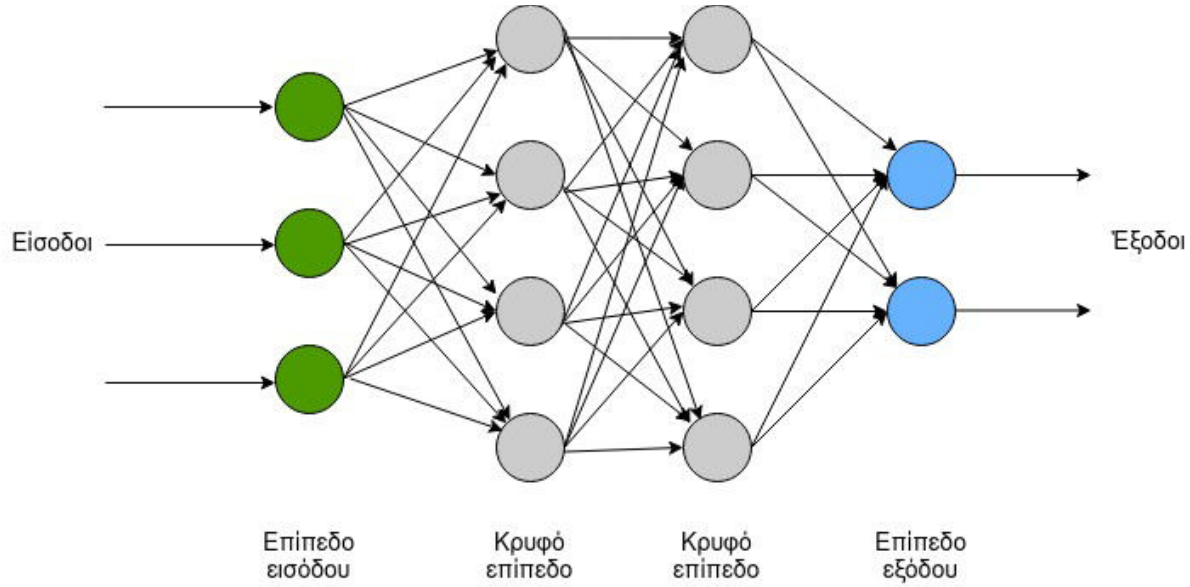
Νευρωνικό δίκτυο ή τεχνητό νευρωνικό δίκτυο ονομάζεται ένα δίκτυο από απλούς υπολογιστικούς κόμβους (νευρώνες ή νευρώνια), διασυνδεδεμένους μεταξύ τους.[28] Πρόκειται για ένα αφηρημένο αλγοριθμικό κατασκεύασμα που εμπίπτει στον τομέα της υπολογιστικής

νοημοσύνης και αποτελεί υποκατηγορία της μηχανικής μάθησης. Το όνομα και η δομή τους προέρχεται από το Κεντρικό Νευρικό Σύστημα (ΚΝΣ) το οποίο και προσπαθεί να προσομοιώσει.



Εικόνα 7: Νευρωνικό δίκτυο σε μικροσκόπιο όπως βρίσκεται στον ανθρώπινο εγκέφαλο

Πηγή: Deep Learning και Νευρωνικά Δίκτυα με απλά λόγια, Ανάρτηση στο Linkedin, George Koutsoudakis, 1η Ιανουαρίου 2021



Εικόνα 8: Παράδειγμα τεχνητού νευρωνικού δικτύου

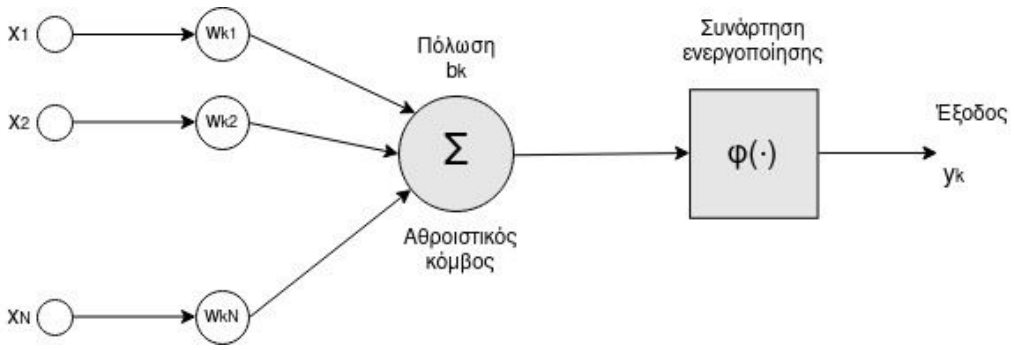
Τα δομικά στοιχεία του δικτύου είναι οι νευρώνες. Όπως φαίνεται και στην εικόνα 8 κάθε νευρώνας δέχεται μία ή περισσότερες εισόδους (είτε από το περιβάλλον είτε από άλλους νευρώνες προηγούμενου επιπέδου) και έχει μία ή περισσότερες εξόδους (είτε σε άλλους νευρώνες είτε στο περιβάλλον). Υπάρχουν τρεις τύποι νευρώνων: οι νευρώνες εισόδου, οι νευρώνες εξόδου και οι υπολογιστικοί νευρώνες ή κρυμμένοι νευρώνες. Από αυτούς καθορίζονται και τα 3 επίπεδα που φαίνονται στην εικόνα 8: το επίπεδο εισόδου, τα κρυφά επίπεδα και το επίπεδο εξόδου (ένα δίκτυο της απλούστερης μορφής θα μπορούσε να μην περιλαμβάνει κρυφά επίπεδα, θα ήταν ισοδύναμο δηλαδή με μια γραμμική παλινδρόμηση).

Οι νευρώνες εισόδου δέχονται τα δεδομένα από το περιβάλλον και τα προωθούν στους υπολογιστικούς νευρώνες χωρίς να πραγματοποιήσουν κάποια επεξεργασία. Αντίστοιχα οι νευρώνες εξόδου δέχονται τα τελικά αριθμητικά αποτελέσματα και τα διοχετεύουν στο περιβάλλον.

Στα κρυφά επίπεδα είναι που γίνεται η επεξεργασία των δεδομένων. Σε κάθε μία από τις εισερχόμενες συνδέσεις του, ο κάθε κόμβος εκχωρεί έναν αριθμό που λέγεται συναπτικό βάρος. Όταν το δίκτυο είναι ενεργό, ο κόμβος λαμβάνει έναν διαφορετικό αριθμό σε κάθε μία από τις συνδέσεις του και το πολλαπλασιάζει με το βάρος. Ύστερα, προσθέτει τα γινόμενα που προέκυψαν και καταλήγει με έναν αριθμό. Εάν ο αριθμός αυτός είναι μεγαλύτερος από μία συγκεκριμένη τιμή κατωφλίου, τότε ο νευρώνας ενεργοποιείται. Εάν είναι μικρότερος, τότε ο νευρώνας παραμένει ανενεργός. Αυτή την απόφαση της ενεργοποίησης ή μη του νευρώνα την παίρνει μια συνάρτηση που εκτελείται εσωτερικά σε κάθε νευρώνα και λόγω της λειτουργίας της ονομάζεται συνάρτηση ενεργοποίησης. Τέλος, συχνά χρησιμοποιείται μία σταθερά τιμή με την ονομασία πόλωση (bias) που προστίθεται στο γινόμενο των βαρών και των σημάτων εισόδου. Ο ρόλος της πόλωσης είναι να αντισταθμίζει το αποτέλεσμα. Βοηθά τα μοντέλα να μετατοπίσουν τη λειτουργία ενεργοποίησης προς τη θετική ή την αρνητική πλευρά.

Έστω ότι θεωρούμε x_i την i-οστή εισερχόμενη σύνδεση του κόμβου k, w_{ki} το συναπτικό βάρος που της εκχωρείται και $\varphi(\cdot)$ τη συνάρτηση ενεργοποίησης. Επίσης η τιμή πόλωσης συμβολίζεται με b_k, η τιμή κατωφλίου με θ_k, η έξοδος με y_k ενώ το σύνολο των νευρώνων είναι N. Τότε η εξίσωση που περιγράφει τη λειτουργία ενός νευρώνα k είναι η εξής:

$$y_k = \varphi\left(\sum_{i=0}^{N} x_i w_{ki} + b_k \right)$$



Εικόνα 9: Η λειτουργία ενός τεχνητού νευρώνα

Ένα από τα βασικότερα χαρακτηριστικά των νευρωνικών δικτύων είναι ότι έχουν τη δυνατότητα να εκπαιδεύονται. Η εκπαίδευση ενός τεχνητού νευρωνικού δικτύου έγκειται στην σταδιακή τροποποίηση των βαρών και της πόλωσής του και αποσκοπεί στη βαθμιαία σύλληψη της πληροφορίας η οποία στη συνέχεια θα είναι διαθέσιμη προς ανάκτηση.

Τα βάρη w_k και η πόλωση b_k αρχικοποιούνται σε μία τυχαία τιμή.[29] Τα δεδομένα εκπαίδευσης δίνονται στο επίπεδο εισόδου του νευρωνικού δικτύου. Προχωρούν ύστερα στα κρυφά επίπεδα του δικτύου και μέσα από τις μαθηματικές διαδικασίες που είδαμε καταλήγουν εξελιγμένες και τροποποιημένες στο επίπεδο εξόδου.

Συνοπτικά, τα βήματα που ακολουθεί η διαδικασία εκπαίδευσης είναι τα εξής:

1) Το νευρωνικό δίκτυο παίρνει δεδομένα από το περιβάλλον
2) Τα δεδομένα επιφέρουν αλλαγές στις ελεύθερες παραμέτρους του δικτύου, δηλαδή αλλάζουν την εσωτερική του δομή
3) Το δίκτυο αποκρίνεται με διαφορετικό τρόπο στο περιβάλλον εξαιτίας των μεταβολών που συνέβησαν

Υπάρχουν πολλές μέθοδοι εκπαίδευσης νευρωνικών δικτύων. Ανάλογα με τη φύση του εκάστοτε προβλήματος και τη δομή του δικτύου που βρίσκεται υπό μελέτη γίνεται και η επιλογή της κατάλληλης μεθόδου. Τρεις είναι οι βασικές μέθοδοι εκπαίδευσης ενός τεχνητού νευρωνικού δικτύου: μάθηση με επίβλεψη (supervised learning), μάθηση χωρίς επίβλεψη (unsupervised learinig) και ενισχυτική μάθηση (reinforcement learning).

Εικόνα 10: Οι βασικές μέθοδοι εκπαίδευσης τεχνητών νευρωνικών δικτύων

<u>Μάθηση με επίβλεψη</u>

Η μάθηση με επίβλεψη είναι η συνηθέστερα χρησιμοποιούμενη μέθοδος μάθησης νευρωνικών δικτύων.[30] Πρόκειται για μάθηση βάσει παραδειγμάτων τα οποία εξασφαλίζονται είτε από κάποιον αλγόριθμο είτε από κάποιο πρόσωπο. Τα παραδείγματα είναι ζευγάρια εισόδου και επιθυμητής εξόδου τα οποία διαθέτουν μία ετικέτα. Το νευρωνικό δίκτυο δεχόμενο τις εισόδους των ζευγαριών δίνει ένα αρχικό αποτέλεσμα που συνήθως διαφέρει από το επιθυμητό. Η διαφορά των επιθυμητών

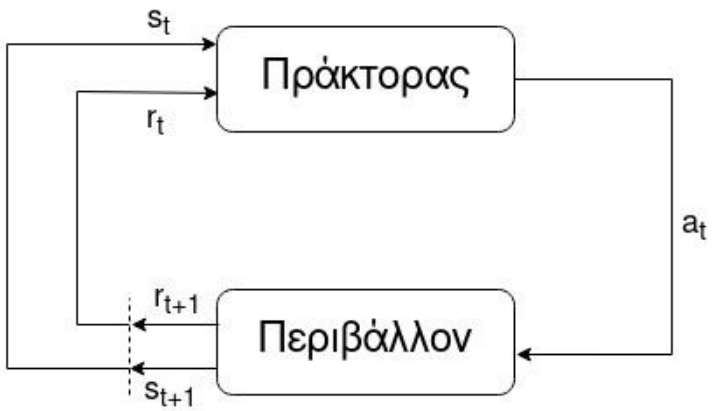
από τις πραγματικές εξόδους ονομάζεται σφάλμα και με βάση αυτό αρχίζει η τροποποίηση των συναπτικών βαρών και των πολώσεων του δικτύου. Μέσω διαδοχικών επαναλήψεων το νευρωνικό δίκτυο προσεγγίζει ικανοποιητικά την επιθυμητή απόκριση και τότε είναι που τερματίζεται η διαδικασία εκπαίδευσης. Η γνώση του περιβάλλοντος δηλαδή έχει περάσει στο δίκτυο και τα συναπτικά βάρη που προέκυψαν εκφράζουν την μακροπρόθεσμη μνήμη του δικτύου. Τότε το μοντέλο είναι έτοιμο να κάνει προβλέψεις σε διαφορετικά παραδείγματα από αυτά στα οποία εκπαιδεύτηκε.

Μάθηση χωρίς επίβλεψη

Στην περίπτωση της μάθησης χωρίς επίβλεψη χρησιμοποιούνται μόνο δεδομένα εισόδου χωρίς επιθυμητές εξόδους ή ετικέτες. Το νευρωνικό δίκτυο καλείται να προσομοιώσει τη δυνατότητα του ανθρώπου για παρατήρηση, δηλαδή να βρει κρυμμένα μοτίβα και συσχετίσεις βασιζόμενο μόνο στις ιδιότητες των δεδομένων που του δόθηκαν. Το αποτέλεσμα είναι η ομαδοποίηση των δεδομένων σε έναν πεπερασμένο αριθμό ομάδων.

Ενισχυτική μάθηση

Η ενισχυτική μάθηση είναι μια τεχνική που μοιάζει με την μάθηση με επίβλεψη με τη διαφορά ότι αντί να δίνεται στο σύστημα η σωστή έξοδος για κάθε παράδειγμα εισόδου, δίνεται ένα σύνολο επιτρεπόμενων δράσεων, κανόνων και πιθανών τελικών καταστάσεων. Η οντότητα που εκπαιδεύεται ονομάζεται πράκτορας (agent). Ο πράκτορας αλληλεπιδρά με το περιβάλλον επιλέγοντας ενέργειες (a_t) και εκείνο αποκρίνεται δίνοντάς του καινούριες καταστάσεις (s_t). Είναι μια μέθοδος εκμάθησης μέσω της ανταμοιβής (r_t), ο πράκτορας δηλαδή κερδίζει ανταμοιβές (rewards) από το περιβάλλον οι οποίες μεταφράζονται σε ειδικές αριθμητικές τιμές. Στόχος του συστήματος είναι να μεγιστοποιήσει τις τιμές αυτές. Αυτός ο τρόπος εκπαίδευσης είναι λιγότερο διαδεδομένος σε σχέση με τους προαναφερθέντες. Θεωρείται κατάλληλος για τον έλεγχο εφαρμογών συστημάτων.



Εικόνα 11: Ο τρόπος λειτουργίας της ενισχυτικής μάθησης

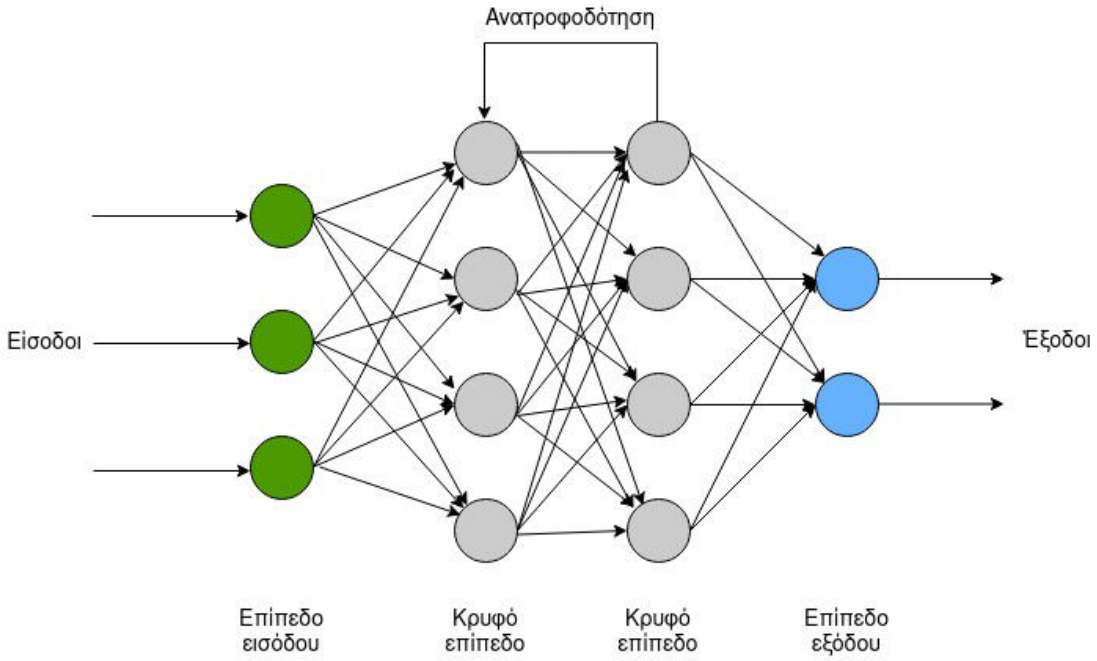
Στη συνέχεια ορίζονται κάποιοι τύποι νευρωνικών δικτύων που θα αναφερθούν και στο κεφάλαιο 5.

2.4.1 Νευρωνικά δίκτυα πρόσθιας τροφοδότησης

Στα τεχνητά νευρωνικά δίκτυα πρόσθιας τροφοδότησης (feed-forward) το σήμα διαδίδεται μόνο προς μία κατεύθυνση. Η έξοδος κάθε επιπέδου επηρεάζει μόνο τα επόμενα επίπεδα. Αυτό σημαίνει ότι σε ένα δίκτυο πρόσθιας τροφοδότησης δεν υπάρχουν κύκλοι. Πρόσθιας τροφοδότησης είναι το δίκτυο της εικόνας 8.

2.4.2 Νευρωνικά δίκτυα με ανατροφοδότηση

Τα νευρωνικά δίκτυα με ανατροφοδότηση είναι δίκτυα στα οποία ξεκινώντας από έναν νευρώνα και ακολουθώντας τη φορά ενεργοποίησης του δικτύου είναι δυνατόν να καταλήξουμε και πάλι στον ίδιο νευρώνα, με άλλα λόγια είναι δίκτυα στα οποία υπάρχουν κύκλοι.[31] Αυτού του τύπου η αρχιτεκτονική είναι ιδιαίτερα χρήσιμη σε περιπτώσεις όπου η χρονική διαδοχή των δεδομένων είναι σημαντική, όπως πχ για αναγνώριση ομιλίας και αυτό ακριβώς είναι που τα διαφοροποιεί από άλλους τύπους νευρωνικών δικτύων: η “προσωρινή μνήμη” τους που διαμορφώνεται εφόσον προγενέστερες είσοδοι μπορούν να επηρεάσουν επόμενες εισόδους, οπότε και εξόδους. Πολλές δημοφιλείς εφαρμογές χρησιμοποιούν νευρωνικά δίκτυα με ανατροφοδότηση , όπως η εικονική βοηθός Siri και η διαδικτυακή μεταφραστική υπηρεσία Google Translate.



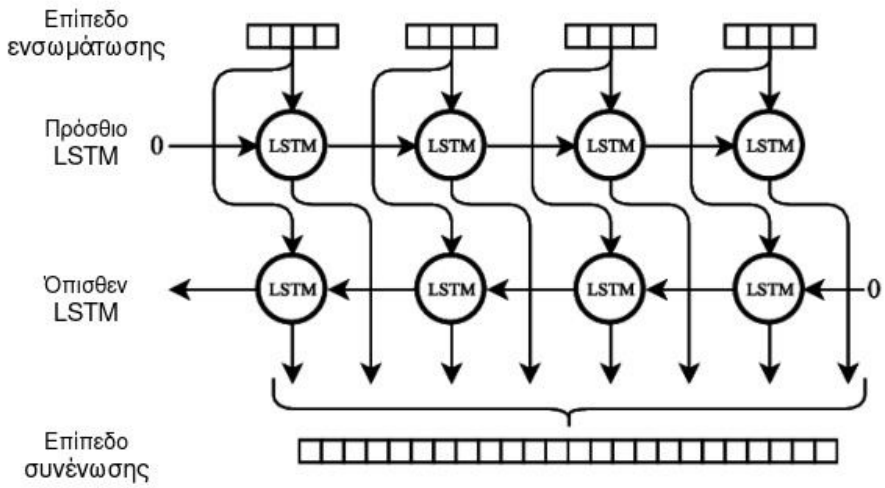
Εικόνα 12: Τεχνητό νευρωνικό δίκτυο με ανατροφοδότηση

2.4.3 LSTM (Long Short-Term Memory) νευρωνικά δίκτυα

Τα LSTM είναι υποκατηγορία των νευρωνικών δικτύων με ανατροφοδότηση. Τα απλά νευρωνικά δίκτυα με ανατροφοδότηση, ενώ είναι πολύ αποδοτικά σε περιπτώσεις άμεσης χρονικής διαδοχής των δεδομένων, δεν έχουν αντίστοιχη επιτυχία όταν υπάρχει υπολογίσιμο χρονικό διάστημα μεταξύ της εισόδου και της αναμενόμενης εξόδου. Για να αντιμετωπιστεί αυτή η αδυναμία σχεδιάστηκαν τα LSTM που έχουν τη δυνατότητα να ανακαλέσουν δεδομένα σημαντικά για την τρέχουσα κατάσταση, ακόμη κι αν αυτά δεν προέκυψαν στο πρόσφατο παρελθόν. Αυτό επιτυγχάνεται με μία τροποποιημένη αρχιτεκτονική, στην οποία τα κρυφά επίπεδα του νευρωνικού δικτύου διαθέτουν τρεις πύλες: την πύλη εισόδου (input gate), την πύλη εξόδου (output gate) και την πύλη λήθης (forget gate). Αυτές οι πύλες ελέγχουν τη ροή πληροφοριών που είναι απαραίτητες για τον υπολογισμό της εξόδου του δικτύου.

2.4.4 Αμφίδρομα (Bidirectional) LSTM νευρωνικά δίκτυα

Τα αμφίδρομα LSTM είναι ειδική περίπτωση LSTM νευρωνικών δικτύων. Σε αυτό το μοντέλο τα δεδομένα εισόδου χωρίζονται σε δύο ομάδες και εκπαιδεύονται παράλληλα δύο νευρωνικά δίκτυα με αντίθετη κατεύθυνση. Έτσι τα αμφίδρομα LSTM δε χρησιμοποιούν μόνο μνήμη από πρότερες καταστάσεις, έχουν και τη δυνατότητα να χρησιμοποιήσουν “μελλοντικές” καταστάσεις για να βελτιώσουν την απόδοσή τους.



Εικόνα 13: Αμφίδρομο LSTM νευρωνικό δίκτυο

Κεφάλαιο 3

Το πρόβλημα της σύνθεσης προγραμμάτων

3.1 Εισαγωγή στη σύνθεση προγραμμάτων

Από την εποχή που η επιστήμη των υπολογιστών εξερεύνησε για πρώτη φορά τις δυνατότητες της αυτόματης παραγωγής κώδικα μέχρι σήμερα, πολλές φορές ο ενθουσιασμός γύρω από τη σύνθεση προγραμμάτων φούντωνε και αντίστοιχα υποχωρούσε όταν οι ερευνητές αδυνατούσαν να υπερπηδήσουν τους περιορισμούς που συναντούσαν. Η τελευταία ύφεση που παρατηρήθηκε ήταν από την δεκαετία του 1990 μέχρι λίγο πριν το 2010 διότι κυριαρχούσε η πεποίθηση ότι παρά τα σημαντικά επιτεύγματα που είχαν σημειωθεί, η σύνθεση προγραμμάτων δε θα μπορούσε να χρησιμοποιηθεί ευρέως. Σε αντίθεση με την μηχανική μάθηση της οποίας τα μοντέλα δέχονταν εκατοντάδες ή ακόμη και χιλιάδες παραδείγματα στη διαδικασία της εκπαίδευσης στην περίπτωση της σύνθεσης προγραμμάτων ο χρήστης έδινε στο σύστημα πέντε με δέκα παραδείγματα. Επίσης δεν υπήρχαν μοντέλα εύχρηστα ώστε να μπορούν να χρησιμοποιηθούν από απλούς χρήστες αλλά ούτε και προσαρμοστικά ώστε να συνδυάζονται με άλλα λογισμικά.[32]

Αυτό άλλαξε όταν εισήχθη η μέθοδος του syntax-guided synthesis. Η αρχή έγινε από τον Armando Solar-Lezama όταν στα πλαίσια της πτυχιακής του εργασίας για το διδακτορικό του το 2008 παρουσίασε μία νέα προσέγγιση, την CEGIS (counter-example guided inductive synthesis) μέσω της τεχνικής του Sketching. Επρόκειτο για ένα σύστημα που διέθετε μία γεννήτρια προγραμμάτων και έναν ελεγκτή και λειτουργούσε με επαναλήψεις. Σε κάθε επανάληψη η γεννήτρια πρότεινε ένα καινούριο πρόγραμμα αλλά το στοιχείο κλειδί είναι ότι ο ελεγκτής σε περίπτωση απόρριψης του προγράμματος παρήγαγε ένα αντιπαράδειγμα δεδομένο εισόδου. Με αυτό το αντιπαράδειγμα φρόντιζε να αποκλείσει όχι μόνο το συγκεκριμένο πρόγραμμα αλλά και κάθε άλλο πρόγραμμα που θα αποτύγχανε για τη συγκεκριμένη είσοδο. Έτσι τα προγράμματα που πρότεινε η γεννήτρια στον ελεγκτή ήταν αποτέλεσμα επαγωγικής διαδικασίας και όχι τυχαίες προτάσεις.

Αυτή η νέα τάση που βοηθούσε την αναζήτηση μειώνοντας τον αριθμό των πιθανών προγραμμάτων ξεκίνησε την άνθηση στην έρευνα και εξέλιξη της σύνθεσης προγραμμάτων που συνεχίζεται μέχρι σήμερα.

Για να γίνουν όμως αυτές οι μέθοδοι περισσότερο κατανοητές πρέπει να ορίσουμε με σαφήνεια τη σύνθεση προγραμμάτων.

3.1.1 Ορισμός

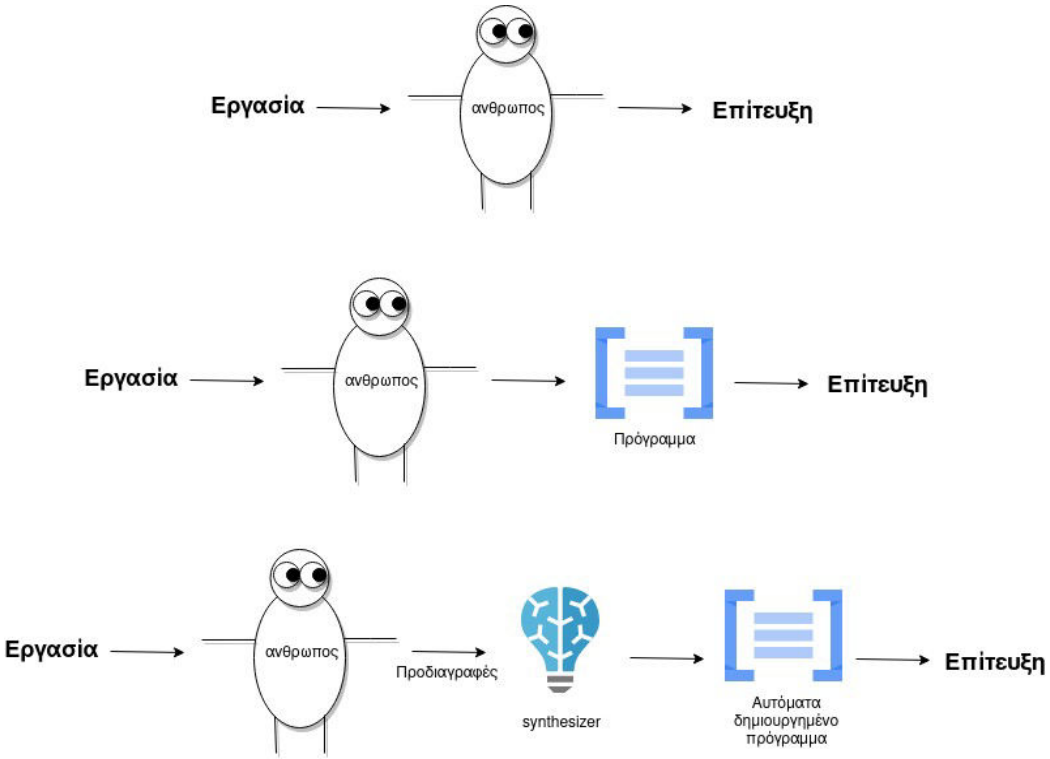
Η σύνθεση προγραμμάτων είναι ένα διεπιστημονικό πεδίο μεταξύ γλωσσών προγραμματισμού, τυπικών μεθόδων και τεχνητής νοημοσύνης. Αυτό που επιτυγχάνει είναι η αυτόματη παραγωγή κώδικα , ο ορισμός του όμως είναι δύσκολος καθώς τα όρια που το διαχωρίζουν από άλλες μορφές αυτοματοποίησης λογισμικού είναι συγκεχυμένα. Έννοιες όπως μεταγλωττιστές, λογικός προγραμματισμός και μηχανική μάθηση είναι συγγενικές με τη σύνθεση προγραμμάτων.

Ωστόσο το ιδιαίτερο χαρακτηριστικό του είναι ότι στην περίπτωση της σύνθεσης προγραμμάτων ο χρήστης περιγράφει στον synthesizer τι πρέπει να κάνει το τελικό πρόγραμμα που θα παραχθεί αλλά το πώς θα γίνει αυτό εναποτίθεται στον ίδιο τον synthesizer. Η περιγραφή του ζητούμενου προγράμματος γίνεται με τη μορφή συντακτικών ή/και σημασιολογικών προδιαγραφών. Οπότε η σύνθεση προγραμμάτων είναι το σύνολο των τεχνικών που δίνουν τη δυνατότητα αυτόματης δημιουργίας προγραμμάτων τέτοιων ώστε να ικανοποιούνται συγκεκριμένες προδιαγραφές.

3.1.2 Σημασία της σύνθεσης προγραμμάτων

Η σύνθεση προγραμμάτων είναι ιδιαίτερα σημαντική τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο. Η προσφορά της είναι μεγάλη στην επιστημονική κοινότητα και τη βιομηχανία.

Καταρχάς η επιστήμη εκ φύσεως έχει ως στόχο τη διεύρυνση των ορίων των ανθρώπινων δυνατοτήτων με όπλο τη γνώση. Από επιστημονικής άποψης λοιπόν, η επίτευξη της δημιουργίας ενός προγράμματος που δημιουργεί ένα άλλο πρόγραμμα προχωράει ακόμη ένα σκαλοπάτι τα όρια των δυνατοτήτων μας και ανοίγει ένα παράθυρο νέων προοπτικών. Επίσης, από φιλοσοφική σκοπιά θα μπορούσε να πει κανείς ότι κλείνει έναν κύκλο ανθρώπινης δημιουργίας. Πετύχαμε αρχικά την αυτοματοποίηση κατασκευάζοντας ηλεκτρονικούς υπολογιστές. Η σύνθεση προγραμμάτων μας πηγαίνει στην αυτοματοποίηση της αυτοματοποίησης. Με άλλα λόγια, ο άνθρωπος περνάει από το να βρίσκει λύσεις στα προβλήματα που του παρουσιάζονται στο να δημιουργεί λύτες. Έτσι, αυτό που έχει να κάνει πια είναι να περιγράψει και να οριοθετήσει το πρόβλημα, χωρίς να αναλαμβάνει να το λύσει.



Εικόνα 14: Η εξέλιξη του τρόπου επίλυσης προβλημάτων από τον άνθρωπο

Επιπλέον η σύνθεση προγραμμάτων μπορεί να βοηθήσει την επιστήμη στην περιγραφή του κόσμου με τρόπους που άλλες μέθοδοι, όπως η μηχανική μάθηση έχουν αποτύχει.

Η επιστήμη εγγενώς έχει να κάνει με τη συλλογή δεδομένων. Η παρατήρηση των δεδομένων οδηγεί τους επιστήμονες σε υποθέσεις. Ύστερα, οι υποθέσεις ελέγχονται με τη βοήθεια πειραμάτων και ανάλογα με τα αποτελέσματα οι υποθέσεις επικυρώνονται, απορρίπτονται ή μετατρέπονται.

Τα τελευταία χρόνια η εξέλιξη της μηχανικής μάθησης έχει δώσει τη δυνατότητα αυτοματοποίησης αυτής της διαδικασίας μέσω αλγορίθμων που βασίζονται σε δεδομένα που μπορούν να δημιουργήσουν νέες επιστημονικές υποθέσεις και να καθοδηγήσουν το σχεδιασμό των πειραμάτων. Τα αποτελέσματα είναι εντυπωσιακά και αξιοσημείωτα, ωστόσο τα μοντέλα μηχανικής μάθησης δεν έχουν καταφέρει ακόμη να εξηγήσουν φυσικά φαινόμενα ούτε και να μοντελοποιήσουν την επιστημονική αλληλουχία αιτίου και αποτελέσματος. Απαιτούν πολύ μεγάλη ποσότητα δεδομένων και τα αποτελέσματά τους παρουσιάζουν μεγάλη διακύμανση σε διαφορετικές δοκιμές. Αυτή η συμπεριφορά αντιτίθεται με τη σταθερή διαδικασία της επιστημονικής εξέλιξης.

Αντιθέτως, ο συνδυασμός τυπικών μεθόδων, γλωσσών προγραμματισμού και σύνθεσης προγραμμάτων μπορεί να έχει καλύτερα αποτελέσματα. Ο σχεδιασμός μιας γλώσσας προγραμματισμού υψηλού επιπέδου μπορεί να περιγράφει μοντέλα επιστημονικών διαδικασιών και φυσικές ιεραρχικές δομές και περιορισμούς του περιβάλλοντος. Έπειτα, η εξαγωγή αποτελεσμάτων με βάση τα δεδομένα μπορεί να μοντελοποιηθεί με τη βοήθεια της σύνθεσης προγραμμάτων: η δημιουργία δηλαδή ενός προγράμματος που ταιριάζει βέλτιστα στα πειραματικά δεδομένα.[33]

Τέλος η βιομηχανία έχει ωφεληθεί ιδιαίτερα από την αυτοματοποίηση παραγωγής κώδικα. Οι προγραμματιστές διευκολύνονται στο έργο τους και άνθρωποι χωρίς εξειδικευμένες γνώσεις

μπορούν να παράγουν βοηθητικό λογισμικό. Η σύνθεση προγραμμάτων έχει επεκταθεί ακόμη και σε τομείς όπως η εκπαίδευση και η βιολογία.

3.2 Εφαρμογές

Οι εφαρμογές της σύνθεσης προγραμμάτων ποικίλουν και εξαπλώνονται σε πολλά και διαφορετικά πεδία.

Πρώτον, η σύνθεση προγραμμάτων είναι ιδιαίτερα ωφέλιμη στη δουλειά των προγραμματιστών λογισμικού. Είναι πολύ σύνηθες να κάνουν αλλαγές ή βελτιώσεις στους κώδικες και κάτω από την πίεση των προθεσμιών αυτό γίνεται βιαστικά και όχι με τον βέλτιστο τρόπο. Το αποτέλεσμα είναι ένας κώδικας που ναι μεν λειτουργεί αλλά είναι κακογραμμένος, δύσκολος στην κατανόηση και με πολλά τμήματα να επαναλαμβάνονται. Τότε είναι απαραίτητο να γίνει μια ανακατασκευή του κώδικα (refactoring), μια αλλαγή δηλαδή που διατηρεί την λειτουργία του – οπότε τη σημασιολογία του - αλλά βελτιώνει τις δυνατότητες δοκιμής, επέκτασης, συντήρησης και κατανόησής του. Η σύνθεση προγραμμάτων είναι ιδανική για την ανακατασκευή προγραμμάτων έχοντας ως δεδομένο τη σημασιολογία τους, το τι δηλαδή πρέπει να κάνουν.

Επιπλέον, οι προγραμματιστές χρησιμοποιούν τη σύνθεση προγραμμάτων για επαλήθευση της ορθότητας των προγραμμάτων τους, για έλεγχο τερματισμού ή μη τερματισμού ενός προγράμματος ή και για να εντοπίσουν λάθη.

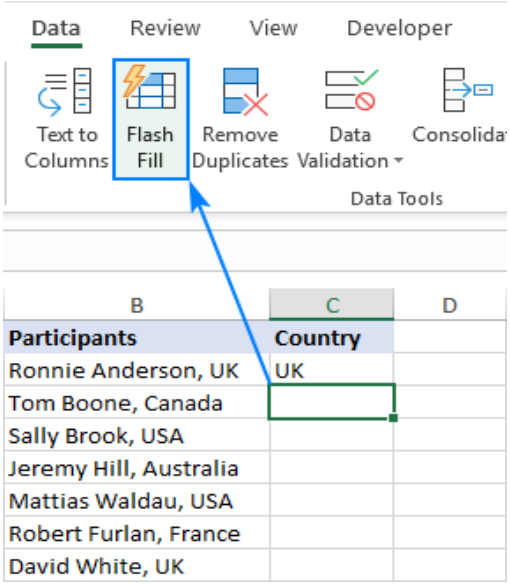
Ένας ακόμη τομέας όπου χρησιμοποιείται η σύνθεση προγραμμάτων είναι αυτός των ενσωματωμένων συστημάτων και πιο συγκεκριμένα η δημιουργία ψηφιακών ελεγκτών. Κάθε σύγχρονη συσκευή σήμερα, από τηλεοράσεις μέχρι και φρυγανιέρες περιλαμβάνει έναν επεξεργαστή ο οποίος ελέγχεται από ένα λογισμικό. Αυτοί οι ελεγκτές μπορούν να παραχθούν αυτόματα με τη χρήση σύνθεσης προγραμμάτων.

Παρουσιάζει επίσης μεγάλο ενδιαφέρον η χρήση της σύνθεσης προγραμμάτων στην εκπαίδευση. Οι δάσκαλοι χρειάζεται να σκέφτονται διαρκώς νέα προβλήματα για τους μαθητές τους τα οποία πρέπει να έχουν συγκεκριμένα χαρακτηριστικά (πχ επίπεδο δυσκολίας, γενική ιδέα). Αυτή η μονότονη δουλειά μπορεί να γίνεται με αυτόματη παραγωγή παρόμοιων προβλημάτων, έτσι στον κάθε μαθητή δίνεται ένα ξεχωριστό πρόβλημα, όλα όμως έχουν τα ίδια χαρακτηριστικά και τον ίδιο τρόπο λύσης. Η σύνθεση προγραμμάτων θα μπορούσε να βοηθήσει και στην διαδικασία διόρθωσης των εργασιών, γλιτώνοντας χρόνο για τους εκπαιδευτικούς και βοηθώντας τη διαδικασία βαθμολόγησης. [34]

Πρέπει επιπροσθέτως να γίνει αναφορά στην πιο σημαντική εφαρμογή της σύνθεσης προγραμμάτων στις φυσικές επιστήμες και πιο συγκεκριμένα στη βιολογία. Οι επιστήμονες έχουν καταφέρει να μοντελοποιήσουν την κυτταρική συμπεριφορά χρησιμοποιώντας προγράμματα που αντιπροσωπεύουν συστατικά όπως πχ πρωτεΐνες. Κάθε συστατικό αλληλεπιδρά με τα γειτονικά και κάτω από ορισμένες συνθήκες αλλάζει κατάσταση. Πρόκειται για μία ιδιαίτερα πολύπλοκη διαδικασία ακόμη κι αν ο αριθμός των συστατικών δεν είναι μεγάλος. Η σύνθεση προγραμμάτων

μπορεί να κάνει αυτή τη διαδικασία μοντελοποίησης αυτόματα, χρησιμοποιώντας πρότερη γνώση και δεδομένα. [33]

Τέλος υπάρχουν εφαρμογές της σύνθεσης προγραμμάτων που βοηθούν εκατομμύρια ανθρώπους που δεν έχουν καμία σχέση με τον προγραμματισμό. Το πιο χαρακτηριστικό παράδειγμα είναι το FlashFill, μια εφαρμογή του Microsoft Excell που γλιτώνει χρόνο και κόπο κάθε χρήστη που εκτελεί αποθήκευση ή επεξεργασία δεδομένων.[35]



Εικόνα 15: Παράδειγμα λειτουργίας του FlashFill

3.3 Προκλήσεις

Τρεις είναι οι βασικές προκλήσεις που παρουσιάζει το πρόβλημα της σύνθεσης προγραμμάτων: η αβεβαιότητα της πρόθεσης του χρήστη, η δημιουργία νέων αλγορίθμων και δομών και τέλος η προσαρμογή σε ένα διαρκώς μεταβαλλόμενο περιβάλλον λογισμικού. Στη βιβλιογραφία αυτά τα τρία θέματα, η πρόθεση, η εφεύρεση και η προσαρμοστικότητα αναφέρονται ως οι τρεις πυλώνες του μηχανικού προγραμματισμού. [36]

Εικόνα 16: Διάγραμμα Venn για τις βασικές προκλήσεις της σύνθεσης προγραμμάτων

Πρόθεση

Η αβεβαιότητα της πρόθεσης του χρήστη συνοψίζεται στην ερώτηση “πώς μπορεί να εξηγήσει ο χρήστης τον στόχο του;” και στους τρόπους που μπορεί να απαντηθεί αυτό το ερώτημα. Η πρόθεση του χρήστη μεταφράζεται ουσιαστικά στις προδιαγραφές, στα ζητούμενα κριτήρια δηλαδή που πρέπει να πληροί το πρόγραμμα που θα παραχθεί. Η μορφή όμως με την οποία αυτές θα αποδοθούν έχει τεράστια σημασία.

Υπάρχουν πολλές μορφές αναπαράστασης των προδιαγραφών με κυριότερες μία τυπική πρόταση κατηγορικής λογικής, μια μη τυπική περιγραφή του ζητούμενου προγράμματος σε φυσική γλώσσα, ζευγάρια εισόδου/εξόδου όπως και ένα άλλο πρόγραμμα ως σημείο αναφοράς.

Την τελευταία δεκαετία η μελέτη των επιστημόνων πάνω στο θέμα των προδιαγραφών έχει αποδώσει 2 σημαντικά συμπεράσματα:

1. Οι μη τυπικές μέθοδοι αναπαράστασης των προδιαγραφών παρά την ευκολία στη χρήση περιέχουν εκ φύσεως τον κίνδυνο της ασάφειας. Παρατηρήθηκε όμως, ότι αν κανείς θέσει αυστηρούς περιορισμούς στα συστατικά που “επιτρέπονται” για τη σύνθεση του προγράμματος και στον τρόπο που αυτά μπορούν να συνδυαστούν, τότε μπορεί να αποτραπεί σε μεγάλο βαθμό το χαρακτηριστικό της ασάφειας. Παράδειγμα σε αυτή την κατεύθυνση αποτελεί το FlashFill το οποίο ψαλιδίζει το χώρο αναζήτησης με τη χρήση μιας γλώσσας ειδικού σκοπού (domain specific language).[37] Μια γλώσσα ειδικού σκοπού είναι σχεδιασμένη για μια συγκεκριμένη εφαρμογή, αυτό την κάνει να έχει λιγότερα χαρακτηριστικά από μια γλώσσα γενικού σκοπού, οπότε και πολύ πιο περιορισμένο συντακτικό. Αυτό σημαίνει ότι όποιο πρόγραμμα δεν έχει αυτής της μορφής το συντακτικό απορρίπτεται αμέσως και ο χώρος αναζήτησης μειώνεται σημαντικά.
2. Έχει παρατηρηθεί ότι σε πολλές περιπτώσεις ο συνδυασμός διαφορετικών ειδών προδιαγραφών δίνει την καλύτερη λύση και περιγράφει ακριβέστερα σύνθετες λειτουργίες. Μία από τις πρώτες φορές που προτάθηκε αυτή η προσέγγιση ήταν το 2012 με το Εργαλείο Προγραμματισμού Storyboard (Storyboard Programming Tool)[38] που συνδύαζε αφηρημένα διαγράμματα, συγκεκριμένα παραδείγματα και έναν σκελετό του κώδικα.

Μια ακόμη περίπτωση του προβλήματος της αβεβαιότητας της πρόθεσης είναι η χρήση ανεπαρκών προδιαγραφών. Σε αυτή την περίπτωση τα προγράμματα που θα ικανοποιούν τις δοθείσες προδιαγραφές θα είναι περισσότερα του ενός, οπότε θα είναι δύσκολο να γίνει η επιλογή του σωστού. Από τη μία πλευρά αυτό θα μπορούσε να θεωρηθεί σφάλμα του χρήστη και να δοθεί τυχαία ένα από τα προγράμματα που πληρούν τα κριτήρια, από την άλλη όμως στην πράξη μια σωστή επιλογή μπορεί να κάνει τη διαφορά ενός συστήματος που είναι χρήσιμο από ένα που δεν είναι.

Εφεύρεση

Από τη στιγμή που οριστικοποιείται το τι ζητά ο χρήστης έρχεται η ώρα να παραχθεί όντως ο ζητούμενος κώδικας. Αυτή είναι και η κεντρική πρόκληση της σύνθεσης προγραμμάτων καθώς θα μπορούσε να περιλαμβάνει την επινόηση νέων αλγοριθμικών λύσεων για ένα πρόβλημα.

Το δυσκολότερο μέρος της κατασκευής του προγράμματος είναι συνήθως η διαχείριση του χώρου αναζήτησης. Στη γενική της μορφή η σύνθεση προγραμμάτων εκτελεί κάποιου είδους αναζήτηση στο πεδίο των πιθανών προγραμμάτων με σκοπό να βρεθεί ένα που να έχει τη ζητούμενη συμπεριφορά. Τα ερωτήματα στα οποία επικεντρώνεται η έρευνα είναι τα εξής: "Ποιος είναι ο βέλτιστος τρόπος να εκφραστεί ο χώρος αναζήτησης και πώς μπορεί να εξερευνηθεί αποδοτικά;"

Μία προσέγγιση στην οποία έχουν καταλήξει οι ερευνητές είναι η χρήση συντακτικών δέντρων. Συγκεκριμένα, τα προγράμματα του χώρου αναζήτησης αναπαριστάνονται με συντακτικά δέντρα – συνήθως αφηρημένα συντακτικά δέντρα(ASTs). Με αυτό τον τρόπο μειώνεται ο χώρος αναζήτησης χωρίς ενδελεχή έλεγχο σε κάθε πρόγραμμα. Αυτό συμβαίνει γιατί είναι εύκολο να αποκλείσει κανείς μια ολόκληρη ομάδα προγραμμάτων των οποίων το συντακτικό δέντρο περιλαμβάνει μια δομή που δε θα μπορούσε να ανήκει στη σωστή λύση.

Μια δεύτερη προσέγγιση είναι η αναπαράσταση όλου του χώρου αναζήτησης με συμβολικό τρόπο. Αυτό μπορεί να γίνει είτε χρησιμοποιώντας μια αναπαράσταση για κάποιον ειδικό σκοπό είτε στην περίπτωση της σύνθεσης βάσει περιορισμών (constraint-based synthesis) συρρικνώνοντας το χώρο αναζήτησης σε ένα σύνολο περιορισμών που μπορούν να συνδεθούν σε ένα πρόγραμμα.

Πολλές από αυτές τις τεχνικές, ιδίως αυτές που σχεδιάστηκαν για περιπτώσεις με μεγάλο πεδίο προδιαγραφών (και όχι απλά ζευγάρια εισόδου/εξόδου) είναι ικανές να αποφασίσουν αυτόματα αν ένα υποψήφιο πρόγραμμα είναι σωστό ή όχι.

Παρά τον μακρύ δρόμο που έχει διανυθεί, τα σημερινά συστήματα σύνθεσης προγραμμάτων σπανίως βρίσκουν εφαρμογή σε μεγέθη της βιομηχανίας προγραμματισμού. Για παράδειγμα η νέα τεχνική υπερβελτιστοποίησης[i] των Phothilimthana et al[39] μπορεί να εξερευνήσει ένα πεδίο μεγέθους 10^{79}. Ωστόσο η υλοποίηση της συνάρτησης κατακερματισμού MD5 θα απαιτούσε την εξερεύνηση ενός χώρου μεγέθους 10^{5943} [ii].

Αυτό συμβαίνει γιατί υπάρχουν ορισμένα θεμελιώδη εμπόδια στις σύγχρονες προσεγγίσεις των ερευνητών. Πιο συγκεκριμένα, ο χώρος αναζήτησης αυξάνεται εκθετικά σε σχέση με το μέγεθος του ζητούμενου προγράμματος. Αυτό κάνει τη δημιουργία ενός προγράμματος μεγαλύτερου από 12 γραμμές περίπου πολύ δύσκολη.

Κάποια συστήματα έχουν καταφέρει να αποδώσουν και με πιο πολύπλοκα προγράμματα είτε κατασκευάζοντάς τα σταδιακά[40], είτε σπάζοντας το πρόβλημα σε επιμέρους μικρότερα προβλήματα, είτε παρέχοντας στον synthesizer διεπαφές επιμέρους συστατικών προς χρήση[41], είτε

i Υπερβελτιστοποίηση είναι το έργο της σύνθεσης μιας βέλτιστης ακολουθίας εντολών που είναι λειτουργικά ισοδύναμη με ένα δεδομένο κομμάτι κώδικα.

ii MD5 (Message Digest Method 5) είναι ένας κρυπτογραφικός αλγόριθμος κατακερματισμού που χρησιμοποιείται για να συμπιέσει ένα string οποιουδήποτε μήκους σε έναν δυαδικό αριθμό 128 bit.

αξιοποιώντας μία τυχόν ειδικού πεδίου δομή του προβλήματος για την αποσύνθεσή του σε έναν μεγάλο αριθμό ανεξάρτητων υποπροβλημάτων[42].

Η διαχείριση του χώρου αναζήτησης παραμένει ένα ανοιχτό πεδίο έρευνας με βαρύτητα σε νέες αλγοριθμικές μεθόδους και εκμετάλλευση των δυνατοτήτων των γλωσσών ειδικού πεδίου (domain-specific language).

Προσαρμογή

Θα μπορούσε να υποτεθεί ότι η σύνθεση προγραμμάτων περιλαμβάνει τη σύλληψη ενός αλγορίθμου και ύστερα την προσπάθεια αυτόματης υλοποίησης αυτού του αλγορίθμου. Η πραγματικότητα όμως της παραγωγής λογισμικού περιλαμβάνει ήδη υπάρχοντα κομμάτια κώδικα, επιδιόρθωση λαθών, βελτίωση κώδικα και κάθε είδους διαδικασίες συντήρησης. Το ερώτημα λοιπόν της προσαρμογής είναι αν είναι δυνατόν η σύνθεση προγραμμάτων να έχει εφαρμογές όχι μόνο σε μεμονωμένες λειτουργίες αλλά και σε ένα ευρύτερο περιβάλλον και να προσφέρει σε πραγματικές συνθήκες προγραμματισμού. Ορισμένα συστήματα σύνθεσης προγραμμάτων σημειώνουν πρόοδο και εξερευνούν δυνατότητες σε αυτό τον τομέα.

Αξίζει να αναφερθεί επίσης η σημασία των δεδομένων όσον αφορά τη σύνθεση προγραμμάτων. Τα δεδομένα έρχονται σε διάφορες μορφές, όπως ζευγάρια εισόδου/εξόδου, γλώσσες ειδικού πεδίου κ.ά. αλλά είναι πάντα απαραίτητα για την αξιοποίηση των εκάστοτε αλγορίθμων. Η εξάρτηση της σύνθεσης προγραμμάτων αλλά και γενικότερα της τεχνητής νοημοσύνης από τα δεδομένα είναι που καθιστά τα ανοιχτά προβλήματα για τη διαχείριση των δεδομένων τόσο σημαντικά.

Τέλος, πρέπει να σημειωθεί ότι το βασικότερο χαρακτηριστικό των προκλήσεων στο πρόβλημα της σύνθεσης προγραμμάτων είναι ότι είναι αλληλένδετα και το ένα επηρεάζει το άλλο. Κάθε προσπάθεια που γίνεται σε μία εκ των προκλήσεων πρέπει πάντα να λαμβάνει υπόψη και τις άλλες και τι επίδραση μπορεί να έχει σε αυτές. Θα μπορούσε για παράδειγμα ένα σύστημα να εξυπηρετεί την ανάγκη για προσαρμοστικότητα, να μεταφράζει δηλαδή το πρόγραμμα που παρήχθη σε άλλη, απλούστερη μορφή ή σε άλλη γλώσσα με σκοπό την απορρόφησή του σε ένα ευρύτερο λογισμικό. Αν όμως δε φρόντιζε να κρατήσει τα ονόματα κάποιων σημαντικών μεταβλητών θα μπορούσε να εμποδίσει τη λειτουργία κάποιου άλλου συστήματος που επικεντρώνεται στην πρόθεση του χρήστη και με βάση τα ονόματα εκτιμά αν ένα κομμάτι κώδικα είναι σχετικό με μια συγκεκριμένη λειτουργία.

Κεφάλαιο 4

Παραδοσιακές μέθοδοι σύνθεσης προγραμμάτων

4.1 Το πλαίσιο των Manna και Waldinger

Το πλαίσιο των Manna και Waldinger εισήχθη το 1980. Πρόκειται για ένα παράδειγμα παραγωγικής λογικής όπου το ζητούμενο πρόγραμμα προκύπτει μέσα από τη διαδικασία απόδειξης των προδιαγραφών. [43]

4.1.1 Η περιγραφή της μεθόδου

Είναι μια προσέγγιση στα πλαίσια της οποίας οι δοθείσες προδιαγραφές εκφράζονται με τη μορφή μιας πρότασης κατηγορικής λογικής, έστω R(a,z) όπου το a είναι μια σταθερά και το z μια μεταβλητή. Η πρόταση αυτή που περιγράφει τις προδιαγραφές ονομάζεται στόχος. Διατίθενται ακόμη τα δεδομένα, δηλαδή κάποιοι ισχυρισμοί οι οποίοι θεωρούνται αληθείς αξιωματικά, έστω P(a). Προσπαθούμε ύστερα να αποδείξουμε ότι για κάποια τιμή του z ο στόχος R(a,z) είναι αληθής, λαμβάνοντας υπόψη ότι οι ισχυρισμοί P(a) είναι επίσης αληθείς. Για να επιτευχθεί αυτό κατασκευάζουμε την παρακάτω ακολουθία:

Δεδομένα	Στόχος	Πρόγραμμα
P(a)	R(a,z)	z

Πίνακας 5: Η βασική ακολουθία του πλαισίου Manna και Waldinger

Στη συνέχεια ακολουθούμε διαδοχικά βήματα ανάλυσης των προτάσεων P(a) και R(a,z) και ξέρουμε ότι φτάσαμε στο επιθυμητό αποτέλεσμα αν καταλήξουμε σε μια ακολουθία της μορφής του πίνακα 6 ή του πίνακα 7.

	αληθής	t

Πίνακας 6: Ακολουθία απόδειξης αληθείας στόχων

ψευδής		t

Πίνακας 7: Ακολουθία απόδειξης ψευδών αξιωμάτων

Τότε το η έκφραση t είναι το αποτέλεσμα. Εφόσον η παραγωγή του προγράμματος γίνεται με χρήση μαθηματικών φορμαλισμών και ξεκινάει παίρνοντας ως δεδομένες τις προδιαγραφές, το τελικό αποτέλεσμα θα μπορούσε να θεωρηθεί σωστό εκ κατασκευής.

[5]Υπάρχουν κάποιοι περιορισμοί στο τι είδους προγράμματα μπορούν να συνθεθούν με αυτή την τεχνική, ωστόσο παρόλο που πρόκειται κυρίως για λιτά προγράμματα οι μελέτες έχουν δείξει ότι μπορούν να παραχθούν αλγόριθμοι ενοποίησης, υπολογισμού τετραγωνικής ρίζας, υπολογισμού υπολοίπου, ταξινόμησης κα.

Ακολουθεί ένα παράδειγμα όπου φαίνεται καθαρότερα η διαδικασία που ακολουθείται. Έχουν προστεθεί δύο ακόμη στήλες για λόγους χρηστικότητας, η πρώτη στήλη η οποία δίνει την αρίθμηση των γραμμών (ώστε να υπάρχει η δυνατότητα παραπομπής σε συγκεκριμένη γραμμή) και η πέμπτη στήλη η οποία προσφέρει μια εξήγηση για το από πού προκύπτουν τα στοιχεία κάθε γραμμής.

Οι γραμμές 1, 2 και 3 περιλαμβάνουν τα αξιώματα, τις προτάσεις δηλαδή που δεχόμαστε ως αληθείς. Στην επόμενη γραμμή, την 10 έχουμε την προδιαγραφή σε τυπική γλώσσα. Σε φυσική γλώσσα θα ήταν “Το μέγιστο είναι μεγαλύτερο ή ίσο κάθε αριθμού που δόθηκε και είναι ένας από τους αριθμούς που δόθηκαν”. Στον πίνακα είναι στη μορφή: $x \leq M \wedge y \leq M (x = M \vee y = M)$ όπου x,y είναι μεταβλητές και M είναι σταθερά.

Στη συνέχεια, στην γραμμή 11 εφαρμόζεται η επιμεριστική ιδιότητα ενώ στις γραμμές 12 και 13 γίνεται ο διαχωρισμός της συνολικής πρότασης σε δύο επιμέρους (εφόσον συνδέονταν μέσω της διάζευξης). Στη γραμμή 14 γίνεται αντικατάσταση του M με το x (γιατί υπήρχε ο όρος x=M) και διαγράφοντας έναν όρο που κατέληξε σε αξίωμα(το αξίωμα στη γραμμή 2) φτάνουμε στην γραμμή 15. Έπειτα με τη βοήθεια του αξιώματος στη γραμμή 3 αναλύεται η πρόταση περαιτέρω και καταλήγει στη μορφή της γραμμής 16. Γίνεται ακριβώς η ίδια ανάλυση και του δεύτερου μέρους που προέκυψε μετά τον διαχωρισμό στις γραμμές 17 και 18.

Τέλος, στην γραμμή 19 ενώνονται τα αποτελέσματα των γραμμών 16 και 18. Η γραμμή 15 μας λέει ότι στην περίπτωση που $y \leq x$ τότε έχουμε το έγκυρο αποτέλεσμα ότι μέγιστο είναι το x. Η πρόταση υπέστη μια μετατροπή και καταλήγει στη μορφή της γραμμής 16 μόνο και μόνο για να φανεί ότι η πρόταση είναι συμπληρωματική με εκείνη της γραμμής 18. Η γραμμή 18 εκφράζει ότι στην περίπτωση που $x \leq y$ τότε έχουμε το έγκυρο αποτέλεσμα ότι το y είναι το μέγιστο. Εφόσον λοιπόν οι γραμμές 16 και 18 δίνουν από ένα αποτέλεσμα στη στήλη του προγράμματος η γραμμή 19 χρησιμοποιεί έναν όρο υπόθεσης. Και αφού προέκυψε το “αληθές” στη στήλη των στόχων, αυτό σημαίνει ότι έχουμε την περίπτωση του πίνακα 6, δηλαδή ό,τι βρίσκεται στη στήλη του προγράμματος είναι το τελικό μας πρόγραμμα.

No	Δεδομένα	Στόχοι	Πρόγραμμα	Προέλευση
1	$A = A$			Αξίωμα
2	$A \leq A$			Αξίωμα
3	$A \leq B \vee B \leq A$			Αξίωμα
10		$x \leq M \wedge y \leq M (x = M \vee y = M)$	M	Προδιαγραφές
11		$(x \leq M \wedge y \leq M \wedge x = M) \vee (x \leq M \wedge y \leq M \wedge y = M)$	M	Επιμεριστική ιδιότητα(10)
12		$x \leq M \wedge y \leq M \wedge x = M$	M	Διαχωρισμός(11)
13		$x \leq M \wedge y \leq M \wedge y = M$	M	Διαχωρισμός(11)
14		$x \leq x \wedge y \leq x$	x	Ανάλυση(12,1)
15		$y \leq x$	x	Ανάλυση(14,2)
16		$\neg(x \leq y)$	x	Ανάλυση(15,3)
17		$x \leq y \wedge y \leq y$	y	Ανάλυση(13,1)
18		$x \leq y$	y	Ανάλυση(17,2)
19		αληθές	x<y ? y : x	Ανάλυση(18,16)

Πίνακας 8: Παράδειγμα σύνθεσης προγράμματος που υπολογίζει τον μέγιστο μεταξύ δύο αριθμών

4.1.2 Συμπεράσματα

Το πλαισίο των Manna και Waldinger ήταν πολύ σημαντικό για το ερευνητικό πεδίο της σύνθεσης προγραμμάτων. Αρχικά παρείχε ένα επίσημο πλαίσιο για τη σύνθεση προγραμμάτων, το οποίο επέτρεψε συστηματικούς και αυστηρούς μετασχηματισμούς από προδιαγραφές σε προγράμματα. Ακόμη, οι μετασχηματισμοί ορθότητας που ανέπτυξαν διασφαλίζουν ότι το πρόγραμμα που προκύπτει πληροί τις αρχικές προδιαγραφές.

Τα παραπάνω χαρακτηριστικά για πρώτη φορά μπορούν να εκτελεστούν με συστηματικό και αυτοματοποιημένο τρόπο μειώνοντας την ανάγκη για χειροκίνητη παρέμβαση και επιτρέποντας την αποτελεσματική παραγωγή σωστών προγραμμάτων. Το έργο των Manna και Waldinger έθεσε τα θεμέλια για μετέπειτα έρευνα στον τομέα της σύνθεσης προγραμμάτων, εμπνέοντας πολλούς ερευνητές στην αυτοματοποιημένη παραγωγή προγραμμάτων και καθόρισε την προσέγγιση των τυπικών μεθόδων.

Κεφάλαιο 5

Σύγχρονες μέθοδοι σύνθεσης προγραμμάτων

5.1 Neuro-Symbolic Program Synthesis

Μία ενδιαφέρουσα προσέγγιση πάνω στο πρόβλημα της σύνθεσης προγραμμάτων με χρήση νευρωνικών δικτύων προτάθηκε το 2017 από τους Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou και Pushmeet Kohli.[44]

Τα νευρωνικά δίκτυα είναι από τα βασικότερα εργαλεία που χρησιμοποιούνται στα πλαίσια της σύνθεσης προγραμμάτων, τα μοντέλα που παράγουν ωστόσο έχουν κάποιους περιορισμούς. Πιο συγκεκριμένα, είναι υπολογιστικά ακριβά, το νευρωνικό δίκτυο πρέπει να εκπαιδεύεται ξεχωριστά για κάθε ένα πρόγραμμα και τέλος είναι δύσκολο να επαληθεύσει κανείς την ορθότητα του αποτελέσματος.

Η συγκεκριμένη προσέγγιση όμως, όχι μόνο παράγει προγράμματα με βάση νέα δεδομένα εισόδου/εξόδου αλλά και προγράμματα για τα οποία δεν είχε εκπαιδευτεί στη φάση του training.

Ο τρόπος που το πετυχαίνει αυτό είναι καταρχάς ακολουθώντας μια διαδικασία που περιλαμβάνει δύο φάσεις:

- Το cross correlation δίκτυο εισόδου/εξόδου παίρνει ένα σύνολο δεδομένων εισόδου/εξόδου και δίνει μια συνεχή αναπαράσταση των δεδομένων αυτών.
- Το R3NN (το αναδρομικό-αντίστροφο-αναδρομικό δίκτυο) παίρνει την αναπαράσταση των δεδομένων που προκύπτει από την προηγούμενη φάση και δίνει ένα πρόγραμμα επεκτείνοντας σταδιακά τμήματα προγραμμάτων.

Ένα ακόμη σημαντικό ζήτημα είναι η επιλογή της γλώσσας στην οποία θα είναι γραμμένο το τελικό πρόγραμμα (αυτή η γλώσσα ονομάζεται target language). Τα τελευταία χρόνια γίνεται όλο και πιο συχνά η επιλογή μιας DSL (domain specific language). DSL είναι μια περιορισμένη γλώσσα προγραμματισμού η οποία χρησιμοποιείται μόνο για μια πολύ εξειδικευμένη λειτουργία. Διαθέτει λοιπόν μόνο τα απαραίτητα χαρακτηριστικά για τη λειτουργία αυτή. Όλα αυτά την καθιστούν απλούστερη, οπότε και ευκολότερη στη διαχείριση.

Η target language αυτού του μοντέλου είναι μια DSL βασισμένη σε αυτή που χρησιμοποιήθηκε για το FlashFill. Ειδικότερα, είναι μια γλώσσα που με τη βοήθεια regular expressions πραγματοποιεί μετρατροπή strings.

Ένα επιπλέον ζήτημα που αποτελεί ένα από το δυσκολότερα προβλήματα στη σύνθεση προγραμμάτων είναι ο χειρισμός του προβλήματος του χώρου αναζήτησης, δηλαδή του υποθετικού συνόλου όλων των πιθανών προγραμμάτων. Έχουν εξερευνηθεί πολλοί τρόποι αντιμετώπισης αυτού του προβλήματος: enumerative, stochastic, constraint-based, version-space algebra based

algorithms. Στην περίπτωση αυτής της μελέτης δε γίνεται εκτενής αναζήτηση, αντιθέτως το R3NN επεκτείνει σταδιακά το πρόγραμμα (το οποίο βρίσκεται στη μορφή δέντρου) διαλέγοντας ποιο μη τερματικό σύμβολο να αναλύσει χρησιμοποιώντας κανόνες της context-free γραμματικής της γλώσσας μας.

5.1.1 Ο ορισμός του προβλήματος

Ο τυπικός ορισμός του προβλήματος είναι ο εξής:

Δοθείσας μιας DSL γλώσσας L στόχος είναι η αυτόματη παραγωγή ενός αλγορίθμου σύνθεσης προγραμμάτων A, τέτοιου ώστε δίνοντάς του ένα σετ παραδειγμάτων εισόδου/εξόδου {(i_1, o_1),…, (i_n,o_n)} ο A να επιστρέφει ένα πρόγραμμα P τέτοιο ώστε να ικανοποιεί τα παραδείγματα εισόδου/εξόδου, δηλαδή:

$$\forall \; j : 1 \le j \le n \; P(i_j) = o_j$$

	Είσοδος υ	Έξοδος
1	William Henry Charles	Charles, W.
2	Michael Johnson	Johnson, M.
3	Barack Rogers	Rogers, B.
4	Martha D. Saunders	Saunders, M.
5	Peter T Gates	Gates, P.

Πίνακας 9: Ένα παράδειγμα λειτουργίας του FlashFill με μετατροπή του ονόματος σε επώνυμο και αρχικό του μικρού ονόματος

String e	:=	Concat(f_1,…,f_n)
Substring f	:=	ConstStr(s)
	|	SubStr(υ, p_l , p_r)
Posision p	:=	(r, k, Dir)
	|	ConstPos(k)
Direction Dir	:=	Start | End
Regex	:=	s | T_1 … | T_n

Πλαίσιο 1: Οι συντακτικοί κανόνες της DSL για μετατροπή strings

$[Concat(f_1, \dots, f_n]_\upsilon = Concat([f_1]_\upsilon, \dots, [f_n]_\upsilon)$

$[ConstStr(s)]_\upsilon = s$

$[SubStr(\upsilon, p_l, p_r)]_\upsilon = \upsilon[[p_l]_\upsilon..[p_r]_\upsilon]$

$[ConstPos(k)]_\upsilon = k>0?\ k: len(s) + k$

$[(r, k, Start)]_\upsilon$ = Η αρχή της $k^{ης}$ αντιστοίχισης του r μέσα στο υ μετρώντας από την αρχή του (από το τέλος του αν k<0)

$[(r, k, End)]_\upsilon$ = Το τέλος της $k^{ης}$ αντιστοίχισης του r μέσα στο υ μετρώντας από την αρχή του (από το τέλος του αν k<0)

Πλαίσιο 2: Οι σημασιολογικοί κανόνες της DSL για μετατροπή strings

Ένα πρόγραμμα της DSL που περιγράφεται στα πλαίσια 1 και 2 και ικανοποιεί τα παραδείγματα στον πίνακα 9 είναι:

$Concat(f_1, ConstStr(“, ”), f_2, ConstStr(“.”))$, όπου

$f_1 \equiv SubStr(\upsilon, (“ ”, -1, End), ConstPos(-1))$ και $f_2 \equiv SubStr(\upsilon, ConstPos(0), ConstPos(1))$.

Αυτό το πρόγραμμα συνενώνει 4 strings: Ένα substring από το τέλος του τελευταίου whitespace μέχρι το τέλος του string, την σταθερά “,”, τον πρώτο χαρακτήρα του string και την σταθερά “.”

5.1.2 Επισκόπηση της προσέγγισης

Το ζητούμενο είναι η κατασκευή ενός μοντέλου που είναι ικανό να παράγει προγράμματα γραμμένα στην DSL που επελέγη, τέτοια ώστε να ικανοποιούν συγκεκριμένα ζευγάρια δεδομένων εισόδου/εξόδου.

Προς αυτό τον στόχο, είναι απαραίτητο αρχικά ένα ικανό δείγμα προγραμμάτων στην DSL καθώς και ένα σύνολο ζευγαριών εισόδου/εξόδου που χρησιμοποιούνται στη φάση της εκπαίδευσης. Συγκεντρώνεται έτσι ένας μεγάλος αριθμός προγραμμάτων στην DSL και ύστερα δίνονται κατάλληλα δεδομένα εισόδου. Συλλέγονται στη συνέχεια τα δεδομένα εξόδου και έτσι εξασφαλίζεται το σύνολο δεδομένων εισόδου/εξόδου.

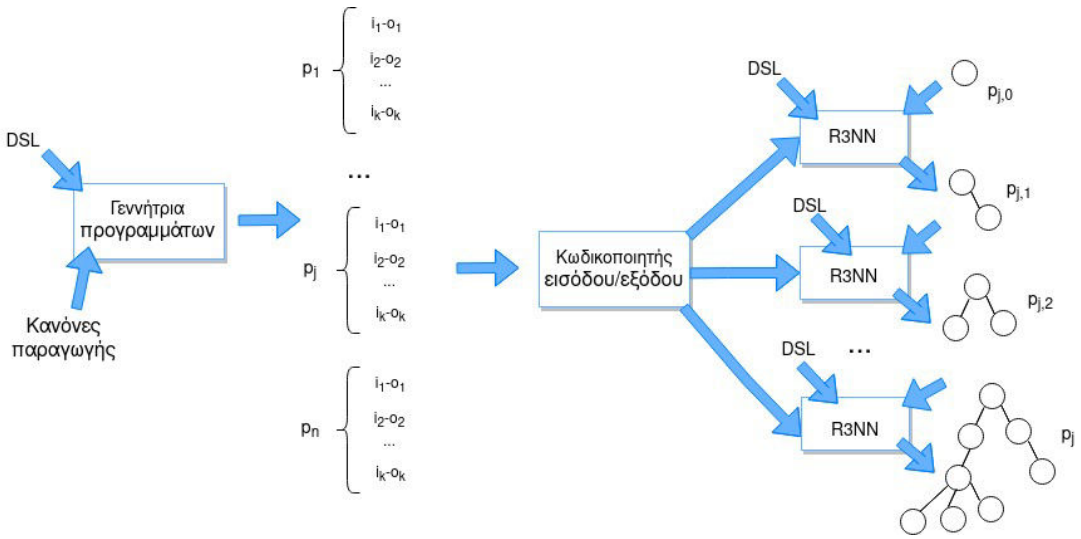
Τα δεδομένα αυτά δίνονται στον κωδικοποιητή. Δοκιμάστηκαν αρκετοί διαφορετικοί κωδικοποιητές, όπως ένας LSTM encoder και ένας cross correlation encoder. Τα αποτελέσματα αυτών δίνονται ως είσοδος στο R3NN.

Το R3NN σύστημα νευρωνικών δικτύων είναι αυτό που θα κατασκευάσει το τελικό πρόγραμμα με διαδοχικές δοκιμές επέκτασης του μερικώς ανεπτυγμένου προγράμματος.

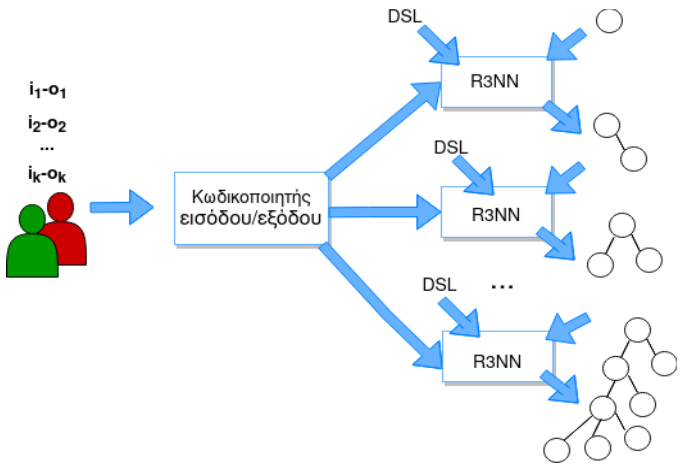
Κάθε γραμματική χωρίς συμφραζόμενα, έτσι και αυτή που περιγράφει την DSL στη συγκεκριμένη μελέτη μπορεί να προσεγγιστεί σαν ένα συντακτικό δέντρο με αρχικό σύμβολο S τη ρίζα του δέντρου, μη τερματικά σύμβολα τους εσωτερικούς κόμβους του δέντρου που

συνοδεύονται από κανόνες παραγωγής και τέλος τα τερματικά σύμβολα της γραμματικής που είναι τα φύλλα του δέντρου. Κάθε μερικώς ανεπτυγμένο πρόγραμμα λοιπόν, μπορεί να ερμηνευθεί ως ένα μερικώς ανεπτυγμένο δέντρο.

Μία απλή προσπάθεια ανάπτυξης του μη ολοκληρωμένου δέντρου θα ήταν η τυχαία επιλογή κόμβων (κανόνων παραγωγής δηλαδή) έως ότου το δέντρο φτάσει σε φύλλα (σε τερματικά σύμβολα). Αυτό το μοντέλο όμως αποδίδει σε κάθε εσωτερικό κόμβο μία πιθανότητα (που εκφράζει το πόσο πιθανό είναι αν επιλεγεί ο συγκεκριμένος κόμβος για ανάπτυξη να οδηγήσει σε ολοκληρωμένο πρόγραμμα) και με βάση αυτές τις πιθανότητες επιλέγονται ανάλογοι κανόνες παραγωγής.



Εικόνα 17: Το μοντέλο στη φάση της εκπαίδευσης



Εικόνα 18: Το μοντέλο στη φάση της δοκιμής

5.1.3 Ο πρώτος κωδικοποιητής I/O

Αρχικά τρέχουν δύο διαφορετικά βαθιά αμφίδρομα LSTM νευρωνικά δίκτυα που επεξεργάζονται τα δεδομένα εισόδου και εξόδου σε κάθε ζευγάρι δεδομένων. Παράγεται έτσι για κάθε ζευγάρι ένα διάνυσμα χαρακτηριστικών (feature) 4HT διαστάσεων, όπου T είναι το μέγιστο μήκος string από όλα τα strings εισόδου και εξόδου και H είναι το κορυφαίο κρυφό επίπεδο του LSTM.

Έπειτα συνενώνονται τα διανύσματα όλων των ζευγαριών και προκύπτει ένα διάνυσμα αναπαράστασης όλου του συνόλου δεδομένων εισόδου/εξόδου.

Για να ανακαλυφθούν τα τμήματα των string εισόδου που επαναλαμβάνονται και στην έξοδο χρησιμοποιήθηκε ένας κωδικοποιητής δεδομένων εισόδου/εξόδου. Ως είσοδο παίρνει τα ζευγάρια αναπαραστάσεων που προκύπτουν από τον πρώτο κωδικοποιητή. Για κάθε ζεύγος παραδειγμάτων, πρώτα σύρεται το μπλοκ χαρακτηριστικών εξόδου πάνω από το μπλοκ χαρακτηριστικών εισόδου και υπολογίζεται το εσωτερικό γινόμενο μεταξύ των στοιχείων που βρίσκονται στις αντίστοιχες θέσεις. Έπειτα αθροίζονται όλες οι χρονικές επικαλύψεις και τα διανύσματα όλων των ζευγαριών συνενώνονται σε ένα διάνυσμα 2(T-1) διαστάσεων. Οπότε υπάρχουν συνολικά 2 (T-1) πιθανοί τρόποι στοίχισης των μπλοκ χαρακτηριστικών εισόδου και εξόδου.

Υπάρχουν επίσης οι ακόλουθες παραλλαγές του κωδικοποιητή:

Η πρώτη περίπτωση είναι ο κωδικοποιητής διάχυτης συσχέτισης. Είναι σχεδόν ίδιος με τον κωδικοποιητή συσχέτισης αλλά αντί του αθροίσματος των χρονικών επικαλύψεων γίνεται συνένωση των διανυσμάτων που αντιστοιχούν σε όλα τα χρονικά βήματα με αποτέλεσμα μια τελική αναπαράσταση που περιλαμβάνει 2(T-1)T χαρακτηριστικά για κάθε ζευγάρι εισόδου/εξόδου.

Μια ακόμη περίπτωση είναι ο LSTM-sum κωδικοποιητής συσχέτισης. Σε αυτή την παραλλαγή δεν υπολογίζεται το εσωτερικό γινόμενο αλλά τρέχει ένα αμφίδρομο LSTM πάνω στα μπλοκ χαρακτηριστικών κάθε συσχέτισης. Αντιπροσωπεύεται κάθε συσχέτιση από την κρυφή

αναπαράσταση του LSTM του τελικού χρονικού βήματος **(?)** με αποτέλεσμα 2*H*2*(T-1) χαρακτηριστικά για κάθε ζευγάρι.

Τέλος δοκιμάστηκε ο κωδικοποιητής επαυξημένης διάχυτης συσχέτισης. Σε αυτή την προσέγγιση συνενώνονται η έξοδος του κωδικοποιητή διάχυτης συσχέτισης σχετικά με τη θέση κάθε χαρακτήρα με τον χαρακτήρα που βρέθηκε τελικά σε αυτή τη θέση. Ύστερα ένα LSTM νευρωνικό δίκτυο επεξεργάζεται αυτά τα συνδυασμένα χαρακτηριστικά και δίνει διανύσματα 4*H διαστάσεων για τα string εισόδου και εξόδου. Έπειτα το αποτέλεσμα του LSTM νευρωνικού δικτύου συνενώνεται με το αποτέλεσμα του κωδικοποιητή διάχυτης συσχέτισης παράγοντας τελικά από ένα διάνυσμα χαρακτηριστικών (4*H + T (T-1))

Μετά την κωδικοποίηση των δεδομένων επιλέγεται το βέλτιστο σημείο όπου μπορούν να προστεθούν οι κωδικοποιήσεις στο R3NN μοντέλο ώστε αυτές να αξιοποιηθούν με τον βέλτιστο τρόπο. Εξετάστηκαν 3 σημεία όπου μπορούν να προστεθούν:

Η πρώτη περίπτωση είναι το pre-conditioning, σύμφωνα με την οποία οι κωδικοποιήσεις του σετ εισόδου/εξόδου συνενώνονται με τις κωδικοποιήσεις των φύλλων-κόμβων. Ύστερα περνούν από ένα δίκτυο προετοιμασίας που τις επεξεργάζεται πριν το bottom-up πέρασμα του δέντρου. Αυτό το δίκτυο προετοιμασίας μπορεί να είναι είτε ένα πολυεπίπεδο feed-forward δίκτυο είτε ένα αμφίδρομο LSTM δίκτυο. Το να τρέξεις ένα LSTM δίκτυο πάνω στα φύλλα του δέντρου, επιτρέπει στο μοντέλο να γνωρίζει περισσότερα σχετικά με τη θέση κάθε φύλλου στο δέντρο.

Δεύτερη εξετάστηκε η περίπτωση του post-conditioning. Εδώ οι κωδικοποιήσεις των παραδειγμάτων συνενώνονται με τις updated κωδικοποιήσεις των φύλλων των δέντρων μετά το reverse-recursive πέρασμα. Το αποτέλεσμα δίνεται σε ένα δίκτυο προετοιμασίας πριν τον υπολογισμό των σκορ των επεκτάσεων.

Τέλος υπάρχει η δυνατότητα του root-conditioning. Σε αυτή την περίπτωση οι κωδικοποιήσεις των παραδειγμάτων εισόδου/εξόδου συνενώνονται με την κωδικοποίηση της ρίζας μετά την πρώτη bottom-up διάσχιση του δέντρου (την recursive) και το αποτέλεσμα δίνεται σε ένα δίκτυο προετοιμασίας. Η updated αναπαράσταση της ρίζας χρησιμοποιείται για το top-down πέρασμα.

Μετά από δοκιμή και των τριών προσεγγίσεων, το pre-conditioning είχε τα καλύτερα αποτελέσματα. Η χρήση και των τριών δεν έδωσε σημαντική βελτίωση από τη χρήση μόνο του pre-conditioning. Έτσι όλα τα πειράματα της έρευνας έγιναν με χρήση της pre-conditioning μεθόδου.

5.1.4 Νευρωνικό δίκτυο R3NN

Έπειτα τα δεδομένα περνούν στα νευρωνικά δίκτυα. Το R3NN σε κάθε βήμα της διαδικασίας παίρνει ως δεδομένα τις κωδικοποιήσεις της προηγούμενης φάσης, την γλώσσα DSL στην οποία θα είναι γραμμένο το ζητούμενο πρόγραμμα και τέλος το μερικώς ανεπτυγμένο πρόγραμμα στη μορφή ενός δέντρου.

Σε κάθε μερικώς ανεπτυγμένο δέντρο τα φύλλα αναπαριστούν σύμβολα (τερματικά και μη) και οι εσωτερικοί κόμβοι κανόνες παραγωγής. Αντιστοιχείται σε κάθε εσωτερικό κόμβο (δηλαδή σε κάθε κανόνα παραγωγής) μια πιθανότητα, δηλαδή πόσο πιθανό είναι να είναι καλή επιλογή ο συγκεκριμένος κανόνας παραγωγής ώστε να καταλήξουμε σε ολοκληρωμένο πρόγραμμα. Η πιθανότητα αυτή κρίνεται από δύο παράγοντες: από το ποιος είναι ο κανόνας παραγωγής και από τη θέση του φύλλου-κόμβου που πρόκειται να επεκταθεί σε σχέση με τους υπόλοιπους κόμβους.

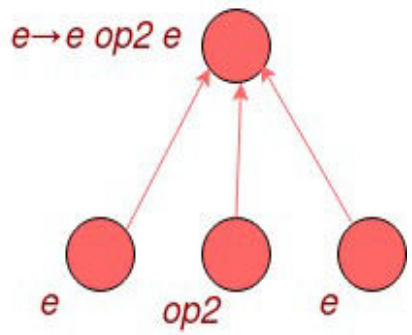
Όσον αφορά τον πρώτο παράγοντα, κρατείται μια ξεχωριστή αναπαράσταση όλων των κανόνων παραγωγής. Για τον δεύτερο παράγοντα χρησιμοποιείται μια αρχιτεκτονική που θυμίζει τη διάδοση πίστης (belief propagation) που επιτρέπει σε κάθε κόμβο να έχει μια ολιστική “αντίληψη” πάνω σε όλο το υπόλοιπο δέντρο.

Από τα αποτελέσματα αυτών των δύο υπολογίζεται η τελική πιθανότητα επιτυχίας κάθε κανόνα παραγωγής.

Η λειτουργία του R3NN αποτελείται από τα εξής βήματα:

1. Δίνεται για κάθε σύμβολο $s \in S$ μια αναπαράσταση M διαστάσεων φ(s)
2. Δίνεται για κάθε κανόνα παραγωγής $r \in R$ μια αναπαράσταση M διαστάσεων ω(r)
3. Για κάθε κανόνα παραγωγής r ένα βαθύ νευρωνικό δίκτυο f_r παίρνει σαν είσοδο την αναπαράσταση των συμβόλων της δεξιάς πλευράς του κανόνα και δίνει σαν έξοδο μια αναπαράσταση για τα σύμβολα της αριστερής πλευράς του κανόνα.

   Πχ για τον κανόνα: e → e op2 e είσοδος είναι τα e, op2 και e και έξοδος ο κανόνας παραγωγής, αυτό φαίνεται καλύτερα με μορφή δέντρου:

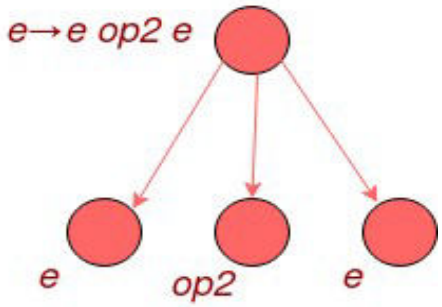


Εικόνα 19: Bottom-up διάσχιση δέντρου

4. Οπότε πρόκειται ουσιαστικά για μια bottom up διάσχιση του δέντρου. Μετά το τέλος της, ενώ έχουμε τις ζητούμενες αναπαραστάσεις για κάθε κόμβο, δεν έχουμε πληροφορίες για τη θέση κάθε κόμβου στο δέντρο.

5. Για κάθε κανόνα r εφαρμόζεται ακόμη ένα βαθύ νευρωνικό δίκτυο g_r που παίρνει σαν είσοδο μια αναπαράσταση των συμβόλων της αριστερής πλευράς κάθε κανόνα και παράγει ως αποτέλεσμα μια συνένωση των αναπαραστάσεων των συμβόλων της δεξιάς πλευράς του κανόνα. Αυτό αντιστοιχεί σε μία top down διάσχιση του δέντρου. Έτσι καταλήγουμε να

ξέρουμε για κάθε φύλλο την διαδρομή προς τη ρίζα. Ακόμη κι αν υπάρχουν 2 φύλλα-κόμβοι με τον ίδιο τύπο συμβόλου δεν μπορούν να θεωρηθούν ίδια γιατί τα διαχωρίζουν οι θέσεις τους στο δέντρο.



Εικόνα 20: Top-down διάσχιση δέντρου

Υπολογίζουμε έπειτα τα σκορ κάθε πιθανής επέκτασης. Τέλος η πιθανότητα κάθε πιθανής επέκτασης είναι το κανονικοποιημένο εκθετικό άθροισμα όλων των σκορ της.

Επίσης φάνηκε χρήσιμη η προσθήκη ενός ακόμη LSTM νευρωνικού δικτύου για την επεξεργασία των αναπαραστάσεων των φύλλων-κόμβων των δέντρων. Έτσι στον τελικό υπολογισμό των σκορ αντί των φύλλων-κόμβων χρησιμοποιήθηκαν τα κρυφά επίπεδα του νευρωνικού δικτύου, πράγμα που μείωσε το ελάχιστο μήκος διάδοσης της πληροφορίας μέσα στα δέντρα.

5.1.5 Σχετικά πειράματα

Αρχικά παράγεται ένα σύνολο προγραμμάτων της DSL. Για να επιτευχθεί αυτό απαριθμούνται όλα τα πιθανά προγράμματα της DSL που να έχουν έως 13 εντολές. Η επιλογή δεν είναι τυχαία, στη συγκεκριμένη γλώσσα κάθε πρόγραμμα μπορεί να παραχθεί ως ένωση μικρότερων προγραμμάτων που έχουν το πολύ 13 εντολές. Δηλαδή, η σημασιολογική ανάλυση των προγραμμάτων με 13 το πολύ εντολές είναι τα “άτομα” πάνω στα οποία χτίζεται η συγκεκριμένη γλώσσα. Τα προγράμματα αυτά έπειτα χωρίζονται στις ομάδες εκπαίδευσης, επικύρωσης και δοκιμής (test).

Στη φάση της δοκιμής υπάρχουν δύο είδη γενίκευσης στα οποία ελέγχεται η απόδοση του μοντέλου. Η πρώτη είναι η γενίκευση σε νέα δεδομένα, δίνεται δηλαδή στο μοντέλο ένα πρόγραμμα του οποίου το δέντρο είχε δει στη φάση της εκπαίδευσης και ένα σετ νέων δεδομένων

εισόδου/εξόδου που δεν έχει δει νωρίτερα. Η δεύτερη είναι η γενίκευση σε νέα προγράμματα, δίνεται δηλαδή ένα νέο πρόγραμμα καθώς και ένα σετ νέων δεδομένων εισόδου/εξόδου. Η ομάδα δεδομένων για εκπαίδευση αντιστοιχεί στο πρώτο είδος γενίκευσης, ενώ η ομάδα δοκιμής στο δεύτερο.

Ελέγχονται οι αποδόσεις όλων των παραλλαγών του μοντέλου, όλα τα είδη κωδικοποιητή καθώς και το R3NN σε σύγκριση με ένα απλό μοντέλο με επαναλήψεις. Σε κάθε περίπτωση δίνεται το μοντέλο με την καλύτερη απόδοση στα προγράμματα μήκους 13 και σε ένα σετ 238 λειτουργιών σημείων αναφοράς.

Στη φάση της εκπαίδευσης χρησιμοποιήθηκαν ομάδες των 8 προγραμμάτων κάθε φορά. Ορισμένα πειράματα που πραγματοποιήθηκαν υποδηλώνουν ότι ομάδες μεγαλύτερου μεγέθους θα βελτίωναν την απόδοση, όμως αυτό δεν επιλέχθηκε λόγω δυσκολιών εκτέλεσης και χρονικών περιορισμών. Κάθε πρόγραμμα παρήχθη με την χρήση ενός σετ 10 παραδειγμάτων εισόδου/εξόδου.

Στη φάση της δοκιμής, για κάθε πρόγραμμα που παρήχθη ελέγχεται αν είναι ισοδύναμο με ένα από 100 προγράμματα του μόντέλου. Αν αυτό ισχύει τότε το αποτέλεσμα κρίνεται επιτυχημένο. (δύο προγράμματα θεωρούνται ισοδύναμα αν παίρνοντας την ίδια είσοδο δίνουν την ίδια έξοδο).

Απόδοση των κωδικοποιητών:

Κωδικοποιητής εισόδου/εξόδου	**Εκπαίδευση**	**Δοκιμή**
LSTM	88%	88%
Συσχέτισης	67%	65%
Διάχυτης Συσχέτισης	89%	88%
LSTM-sum Συσχέτισης	90%	91%
Επαυξημένης Διάχυτης Συσχέτισης	91%	91%

Πίνακας 10: Η επίδραση διάφορων κωδικοποιητών στην ακρίβεια της απόδοσης

Στον πίνακα 10 βλέπουμε την απόδοση όλων των κωδικοποιητών που δοκιμάστηκαν στο μοντέλο με τη βοήθεια στη συνέχεια του R3NN. Παρατηρούμε ότι ο κωδικοποιητής συσχέτισης δεν είχε καλά αποτελέσματα, γεγονός που πιθανόν να οφείλεται στην άθροιση των χρονικών επικαλύψεων. Αυτό προκαλεί την απώλεια πληροφοριών σχετικά με τη κάθε κάθε κόμβου στο δέντρο. Την καλύτερη απόδοση είχαν οι κωδικοποιητές LSTM-sum συσχέτισης και Επαυξημένης Διάχυτης Συσχέτισης. Επίσης ο LSTM είχε απροσδόκητα καλή απόδοση.

Έγινε χρήση 100 προγραμμάτων για την αξιολόγηση των σετ εκπαίδευσης και δοκιμής. Τα αποτελέσματα της εκπαίδευσης είναι κάποιες φορές ελαφρώς μειωμένα διότι υπάρχουν περίπου 5 εκατομμύρια προγράμματα αλλά το μοντέλο στη διάρκεια της εκπαίδευσης βλέπει λιγότερα από 2 εκατομμύρια. Επιλέχθηκε ένα δείγμα 1000 προγραμμάτων – από το σύνολο των 5 εκατομμυρίων για την αναφορά των αποτελεσμάτων της εκπαίδευσης. Ένα δείγμα ίδιου μεγέθους ήταν και το σετ δοκιμής.

Έπειτα χρησιμοποιείται ο κωδικοποιητής LSTM-sum συσχέτισης αλλά αυτή η φορά όχι με το R3NN αλλά με ένα άλλο μοντέλο παραγωγής προγραμμάτων, το io2seq. Πρόκειται για ένα

νευρωνικό δίκτυο LSTM του οποίου τα αρχικά επίπεδα είναι συναρτήσεις των διανυσμάτων που παράγονται από τα δεδομένα εισόδου/εξόδου και παράγει ένα γραμμικοποιημένο συντακτικό δέντρο.

Μοντέλο	**Εκπαίδευση**	**Δοκιμή**
io2seq	44%	42%

Πίνακας 11: Η απόδοση του μοντέλου io2sec

Βλέπουμε ότι τα αποτελέσματά του, που φαίνονται στον πίνακα 11, είναι πολύ χειρότερα από τα αντίστοιχα του R3NN. Αυτό μπορεί να οφείλεται στο γεγονός ότι το io2seq έχει να πάρει περισσότερες αποφάσεις, όπως τα σύμβολα παρενθέσεων που υποδεικνύουν το επίπεδο του δέντρου στο οποίο βρίσκεται κάθε σύμβολο.

Όσον αφορά το καλύτερο R3NN μοντέλο που εκπαιδεύτηκε, αξιολογήθηκε επίσης την επίδραση του μεγέθους του δείγματος στην απόδοση. Βλέπουμε στον πίνακα 12 ότι όσο αυξάνει το δείγμα τόσο βελτιώνεται η απόδοση.

Μέγεθος δείγματος	**Εκπαίδευση**	**Δοκιμή**
1-βέλτιστο	60%	63%
1-πρόγραμμα	56%	57%
10-προγράμματα	81%	79%
50-προγράμματα	91%	89%
100-προγράμματα	94%	94%
300-προγράμματα	97%	97%

Πίνακας 12: Η επίδραση του μεγέθους του δείγματος στην ακρίβεια του μοντέλου. Το 1-βέλτιστό είναι ένα πρόγραμμα που παρήχθη επιλέγοντας σε κάθε βήμα την επέκταση με την μεγαλύτερη πιθανότητα

Δοκιμάζονται επίσης δύο μοντέλα σε 238 πραγματικές λειτουργίες του FlashFill που σχετίζονται με μετατροπή strings. Τα μοντέλα εκπαιδεύτηκαν το ένα με πέντε ζευγάρια εισόδου/εξόδου και το άλλο με δέκα.

Όσον αφορά το πρώτο, κατάφερε να παράξει προγράμματα για την ολοκλήρωση 91 διεργασιών, δηλαδή το 38,2%. Το μέγεθος των προγραμμάτων που απαιτούνταν για την διεκπεραίωση του συνόλου των διεργασιών ποικίλει από 4 έως και 63 γραμμές. Το συγκεκριμένο μοντέλο όμως έχει εκπαιδευτεί για την δημιουργία προγραμμάτων με μέγεθος έως τις 13 γραμμές. Αυτός είναι ο βασικός λόγος που το ποσοστό επιτυχίας δείχνει χαμηλό. Αν ληφθούν υπόψη μόνο οι διεργασίες που απαιτούν ένα πρόγραμμα όχι μεγαλύτερου μήκους από 13 τότε το ποσοστό επιτυχίας είναι 82,7%.

Βλέπουμε στον πίνακα 13 σταθερή αύξηση της απόδοσης όσο αυξάνεται το δείγμα μέχρι το μέγεθος 2000. Από κει κι έπειτα δεν παρατηρείται βελτίωση διότι ο αριθμός των προγραμμάτων στην DSL γλώσσα που χρησιμοποιείται με μήκος 11 είναι μεγαλύτερος των δύο εκατομμυρίων και οι επιλεγμένες τακτικές αναζήτησης δεν κλιμακώνονται καλά.

Μέγεθος δείγματος	Ποσοστό επιτυχίας μοντέλου
10	13%
50	21%
100	23%
200	29%
500	33%
1000	34%
2000	38%
5000	38%

Πίνακας 13: Η επίδραση του μεγέθους του δείγματος στο ποσοστό επιτυχίας

Το δεύτερο μοντέλο σημείωσε ποσοστό επιτυχίας 29%. Μια εκδοχή για αυτό το ποσοστό είναι ότι όσο περισσότερα ζευγάρια δίνονται ως είσοδος, τόσο περιορίζεται το εύρος των πιθανών προγραμμάτων. Ένας άλλος λόγος θα μπορούσε να είναι ότι τα περισσότερα ζευγάρια εισόδου/εξόδου μπερδεύουν τον κωδικοποιητή.

Τα μοντέλα είναι ικανά να παράξουν ένα πρόγραμμα με τη συνένωση το πολύ τριών substrings. Στους πίνακες 14, 15 και 16 υπάρχουν παραδείγματα λειτουργιών για τις οποίες το μοντέλο παρήγαγε επιτυχές πρόγραμμα.

Είσοδος υ	Έξοδος
[CPT-00350	[CPT-00350]
[CPT-00340]	[CPT-00340]
[CPT-114563]	[CPT-114563]
[CPT-1AB02	[CPT-1AB02]
[CPT-00360	[CPT-00360]

Πίνακας 14: Διόρθωση ιατρικών κωδικών προσθέτοντας αγκύλες που λείπουν

Είσοδος υ	Έξοδος
732606129	0x73
430257526	0x43
444004480	0x44
371255254	0x37
635272676	0x63

Πίνακας 15: Μετατροπή αριθμών σε δεκαεξαδικούς

Είσοδος υ	Έξοδος
John Doyle	John D.
Matt Walters	Matt W.
Jody Foster	Jody F.
Angela Lindsay	Angela L.
Maria Schulte	Maria S.

Πίνακας 16: Μετατροπή ονοματεπωνύμων σε όνομα και αρχικό του επωνύμου

Το μοντέλο δεν είναι ικανό να παράξει πρόγραμμα το οποίο πραγματοποιεί συνδυασμό περισσότερων από 4 substrings. Αυτό οφείλεται κυρίως σε προβλήματα επέκτασης σε προγράμματα μεγαλύτερου μεγέθους στη φάση της εκπαίδευσης. Στους πίνακες 17 και 18 βλέπουμε δύο παραδείγματα τέτοιων διεργασιών. Για την μετατροπή στον συνδυασμό των ονομάτων που φαίνεται στο (α) χρειάζονται 6 παράμετροι στην διαδικασία της συνένωσης ενώ για την μετατροπή του τηλεφωνικού αριθμού που φαίνεται στο (β) χρειάζονται 5 παράμετροι.

	Είσοδος υ	Έξοδος
1	John James Paul	John, James, and Paul.
2	Tom Mike Bill	Tom, Mike, and Bill.
3	Marie Nina John	Marie, Nina, and John.
4	Reggie Anna Adam	Reggie, Anna, and Adam.

Πίνακας 17: Συνδυασμός ονομάτων με διαφορετικό τρόπο

	Είσοδος υ	Έξοδος
1	(425) 221 6767	425-221-6767
2	206.225.1298	206-225-1298
3	617-224-9874	617-224-9874
4	425.118.9281	425-118-9281

Πίνακας 18: Μετατροπή τηλεφωνικού αριθμού σε άλλη μορφή

5.1.5 Συμπεράσματα

Συγκεντρωτικά, οι κυριότερες προσφορές αυτής της προσέγγισης είναι οι εξής:

- Μια νέα τεχνική για νευροσυμβολική σύνθεση προγραμμάτων όπου για να επιτευχθεί καλύτερη διαχείριση του χώρου αναζήτησης χρησιμοποιείται μια DSL
- Ένα μοντέλο R3NN που κωδικοποιεί και επεκτείνει τμήματα προγραμμάτων, όπου κάθε κόμβος του δέντρου διαθέτει μια πλήρη αναπαράσταση όλου του δέντρου
- Μια νέα αρχιτεκτονική, η cross-correlation I/O system η οποία δίνει μία συνεχή αναπαράσταση του συνόλου παραδειγμάτων εισόδου/εξόδου
- Ο έλεγχος του συνολικού μοντέλου δοκιμάζοντάς το στο πολύπλοκο πρόβλημα της μετατροπής strings.

5.2 Synthesizing Neurosymbolic Programs

Η πρόκληση της νευροσυμβολικής σύνθεσης είναι ο επιτυχημένος συνδυασμός παραδοσιακών και σύγχρονων μεθόδων στα πλαίσια της σύνθεσης προγραμμάτων.[45] Οι σύγχρονες μέθοδοι που έχουν κυριαρχήσει πια υστερούν σε σχέση με τις παραδοσιακές μεθόδους σε κάποια σημεία. Πιο συγκεκριμένα οι τυπικές μέθοδοι παράγουν προγράμματα high level που είναι εύκολα αντιληπτά από τον άνθρωπο, οπότε και πολύ πιο απλά στην ερμηνεία και τη διαχείριση. Αντίθετα οι σύγχρονες μέθοδοι που βασίζονται σε νευρωνικά δίκτυα πολλές φορές παράγουν ως τελικό αποτέλεσμα ένα πρόγραμμα το οποίο είναι ακατανόητο και εξαιρετικά πολύπλοκο.

Τα δεδομένα που έχουν να διαχειριστούν οι σύγχρονες μέθοδοι μπορούν εύκολα να φτάσουν σε πολύ υψηλά μεγέθη και ο χώρος αναζήτησης να γίνει τεράστιος. Από την άλλη πλευρά οι τυπικές μέθοδοι μπορούν να χρησιμοποιήσουν τυπική λογική για να σπάσουν το πρόβλημα που διαχειρίζονται σε μικρότερα όπως επίσης και την δομή της γλώσσας προγραμματισμού (στην οποία θα είναι γραμμένο το τελικό πρόγραμμα) για να μειώσουν τον χώρο αναζήτησης.

Όσον αφορά τα νευροσυμβολικά προγράμματα, δε διαφέρουν ουσιαστικά από ένα οποιοδήποτε πρόγραμμα σε μία γλώσσα υψηλού επιπέδου εκτός από το γεγονός ότι μπορούν να χρησιμοποιήσουν νευρωνικά δίκτυα σαν βιβλιοθήκες. Τα δεδομένα εισόδου και εξόδου τους μπορούν επίσης να παραχθούν από νευρωνικά δίκτυα. Τα παραγόμενα προγράμματα μπορούν να μελετηθούν και αναλυθούν χρησιμοποιώντας εργαλεία συμβολικών (τυπικών) μεθόδων.

Προκειμένου να γίνει η τεχνική του συνδυασμού των δύο μεθόδων αντιληπτή, παρακάτω παρατίθενται σχετικά παραδείγματα στην βιβλιογραφία όπου η σύνθεση προγραμμάτων προσεγγίζεται με μία μορφή συνδυασμού δύο μεθόδων.

5.2.1 Συμβολική σύνθεση νευρωνικών μονάδων

Έστω ότι έχουμε το πρόγραμμα $\lambda x.\ f(N_1(x), \dots, N_k(x))$, όπου f είναι μια έκφραση σε μία DSL γλώσσα και $N_1, \dots, N_k$ είναι νευρωνικά δίκτυα.

Ένα τέτοιο πρόγραμμα θα μπορούσε να έχει να κάνει με τη διαχείριση και την επεξεργασία οπτικών ή ηχητικών σημάτων. Αρχικά τα νευρωνικά δίκτυα θα έπαιρναν τα χαμηλού επιπέδου δεδομένα των σημάτων και θα έδιναν υψηλού επιπέδου χαρακτηριστικά τους. Έπειτα, η f θα λάμβανε αυτά τα χαρακτηριστικά και θα πραγματοποιούσε κάποιον υπολογισμό. Ενδεικτικά παρατίθεται βιβλιογραφία που ακολουθεί μεθόδους όμοιες με αυτή του παραδείγματος.

Στην εργασία των Valkov et al χρησιμοποιούνται higher order συναρτήσεις, δομές δηλαδή που χρησιμοποιούνται στον συναρτησιακό προγραμματισμό οι οποίες διαχειρίζονται νευρωνικά δίκτυα. Τα νευρωνικά δίκτυα επεξεργάζονται οπτικά σήματα και τα αποτελέσματά τους δίνονται στις συναρτήσεις για περαιτέρω υπολογισμούς.[46]

Μια ακόμη περίπτωση είναι η εργασία των Ellis et al όπου συντίθενται προγράμματα το οποία μεταφράζουν εικόνες σε latex κώδικά. Δίνονται δηλαδή σχέδια σαν είσοδος σε ένα νευρωνικό δίκτυο και εκείνο δίνει τις γενικότερες και πιο αφαιρετικές γραμμές του σχεδίου. Έπειτα αυτό δίνεται σε ένα συμβολικό πρόγραμμα το οποίο δίνει τμήματα κώδικα σε latex.[47]

5.2.2 Νευρωνική σύνθεση συμβολικών μονάδων

Έστω ένα πρόγραμμα $\lambda x.N(f_1(x), \dots, f_k(x))$ όπου N είναι ένα νευρωνικό δίκτυο και $f_1, \dots, f_k$ είναι εκφράσεις σε μια συμβολική γλώσσα DSL. Ένα τέτοιο πρόγραμμα θα μπορούσε να είναι πολύ χρήσιμο σε μια εφαρμογή πρόβλεψης βασισμένης στην επεξεργασία δεδομένων. Οι συμβολικές συναρτήσεις f θα εξήγαγαν σημαντικές πληροφορίες με βάση τις οποίες το N θα έδινε την βέλτιστη πρόβλεψη. Όπως και στο υποκεφάλαιο 5.2.1 έτσι κι εδώ οι συναρτήσεις N και f θα προέκυπταν από δεδομένα και προδιαγραφές.

Τέτοιου είδους μελέτη γίνεται στο paper των Cheung, Solar-Lezama, and Madden. Τα προγράμματα που συντίθενται παίρνουν ως είσοδο μια αλληλουχία περίπλοκων δεδομένων σχετικών με εκδηλώσεις από κάποιο μέσο κοινωνικής δικτύωσης και δίνουν ως αποτέλεσμα ποια εκδήλωση είναι πιθανότερο να ενδιαφέρει συγκεκριμένους χρήστες. Δίνοντας στο πρόγραμμα μια καινούρια εκδήλωση, οι συμβολικές συναρτήσεις (που είναι αυτόματα παραγόμενες) εξάγουν συγκεκριμένα χαρακτηριστικά το οποία ύστερα μπορούν να αποτελέσουν είσοδο σε ένα νευρωνικό

δίκτυο ή σε ένα στατιστικό μοντέλο. Το νευρωνικό δίκτυο έπειτα διανέμει βάρη σε αυτά τα χαρακτηριστικά με αποτέλεσμα να προκύψει η “ετικέτα” της εκδήλωσης.[48]

5.2.3 Αλγεβρική σύνθεση συμβολικών και νευρωνικών μονάδων

Έστω ένα πρόγραμμα της μορφής: λx. N(x)⊕f(x), όπου συνδυάζονται τα αποτελέσματα του νευρωνικού δικτύου N και της συμβολικής συνάρτησης f αν τους δοθεί η ίδια είσοδος. Μία τέτοια μέθοδος θα ήταν αποτελεσματική σε μια περίπτωση όπου ένα πρόγραμμα βασισμένο σε τυπικές μεθόδους και κατασκευασμένο με βάση προϋπάρχουσα γνώση δίνει κατά προσέγγιση τον βέλτιστο τρόπο να πραγματοποιηθεί μια λειτουργία. Εδώ η προσθήκη ενός νευρωνικού δικτύου θα αύξανε την απόδοση προβλέποντας και για πιο ιδιαίτερες συνθήκες λειτουργίας.

Μία τέτοια μέθοδος χρησιμοποιείται στην εργασία των Verma et al. Σε αυτή τη μελέτη επιλέγεται ένα συμβολικό πρόγραμμα για την υλοποίηση ενός μηχανισμού ελέγχου με την επιπλέον προσθήκη ενός νευρωνικού δικτύου το οποίο διαχειρίζεται περιπτώσεις άγνωστου περιβάλλοντος. Το συμβολικό πρόγραμμα είναι αξιόπιστο αλλά συντηρητικό στη διαχείριση. Η νευρωνική μονάδα κάνει το σύστημα ελέγχου πιο ευέλικτο και αυξάνει έτσι την απόδοσή του.[49]

5.2.4 Συμπεράσματα

Το πλεονέκτημα της μηχανικής μάθησης είναι ότι μπορεί να εξάγει προγράμματα από θορυβώδη (noisy) ζευγάρια εισόδου/εξόδου, ζευγάρια δηλαδή που περιλαμβάνουν ανούσια δεδομένα ή δεδομένα που δεν μπορούν να γίνουν αντιληπτά ή χρήσιμα στο σύστημα. Αντίθετα στις τυπικές μεθόδους οι προδιαγραφές οδηγούν στη δημιουργία μαθηματικών εκφράσεων που περιγράφουν τα συντακτικά ή/και σημασιολογικά χαρακτηριστικά του ζητούμενου προγράμματος. Στην νευροσυμβολική σύνθεση λοιπόν, μπορούμε να δούμε όλα αυτά τα εργαλεία και συνδυασμούς τους, όπως θορυβώδη ζευγάρια εισόδου/εξόδου, περιορισμούς στα χαρακτηριστικά των νευρωνικών δικτύων, συντακτικό σκελετό της γλώσσας του ζητούμενου προγράμματος, σημασιολογική περιγραφή του ζητούμενου προγράμματος με τη μορφή τυπικών προτάσεων κατηγορικής λογικής κ.α

Η νευροσυμβολική σύνθεση γεφυρώνει δύο πολύ διαφορετικές μεθόδους αυτόματου προγραμματισμού: Από τη μία οι παραδοσιακές μέθοδοι παράγουν ευανάγνωστο και εύκολα ερμηνεύσιμο για τον άνθρωπο κώδικα και από την άλλη η μηχανική μάθηση παράγει κώδικα που έχει τη μορφή μαύρου κουτιού, περιγράφει διαδικασίες που δεν μπορούν να περιγραφούν εύκολα με ανθρώπινη γλώσσα. Το πρόβλημα αυτό έχει πολλές προκλήσεις, πιο συγκεκριμένα:

- ο σχεδιασμός των DSL γλωσσών που βοηθούν στον περιορισμό τόσο των συμβολικών μονάδων του προγράμματος όσο και της αρχιτεκτονικής των νευρωνικών μονάδων
- η μοντελοποίηση της αβεβαιότητας
- ο αποτελεσματικός συνδυασμός της αναζήτησης των συμβολικών τμημάτων του προγράμματος με την κυρτή βελτιστοποίηση των παραμέτρων των νευρωνικών μονάδων
- η απόφαση του πότε πρέπει να επιλεγεί η ανάπτυξη ενός συγκεκριμένου αλγορίθμου σύνθεσης

- η διασφάλιση ότι το πρόγραμμα που παρήχθη ικανοποιεί το σενάριο χειρότερης περίπτωσης

Κεφάλαιο 6

6.1 Συμπεράσματα και μελλοντικές επεκτάσεις

Η αυτόματη παραγωγή κώδικα αποτελούσε διαχρονικό όνειρο των ερευνητών στον χώρο της πληροφορικής. Θεμελιωτές της σύγχρονης επιστήμης της πληροφορικής όπως ο Alan Turing είχαν από νωρίς εντοπίσει την δυνατότητα μιας αφηρημένης μηχανής να δημιουργήσει κώδικα. Αυτές οι προσπάθειες συγκεντρώνονται στο ερευνητικό αντικείμενο της σύνθεσης προγραμμάτων.

Η αρχική προσέγγιση της σύνθεσης προγραμμάτων, παράλληλη με την εξέλιξη της επιστήμης της πληροφορικής, ήταν βασισμένη στα μαθηματικά. Προσομοίωνε δηλαδή τον τρόπο απόδειξης ενός μαθηματικό θεωρήματος με την χρήση ενός συνόλου αξιωμάτων και κανόνων παραγωγής. Αν κι αυτή η μέθοδος είχε ενδιαφέροντα αποτελέσματα, αντιμετώπιζε ωστόσο σημαντικούς περιορισμούς στο εύρος των προβλημάτων που μπορούσε να λύσει και στην χειρονακτική εργασία που απαιτούσε από τους ερευνητές. Η σύνθεση προγραμμάτων έτσι έμεινε για καιρό σαν ένα παρακλάδι της επιστήμης της πληροφορικής με λίγες πρακτικές εφαρμογές.

Η σύνθεση προγραμμάτων ήρθε στην επιφάνεια ξανά όταν οι ενισχυμένες ικανότητες των υπολογιστών επέτρεψαν την εφαρμογή νέων εργαλείων όπως τα νευρωνικά δίκτυα. Τα αποτελέσματα ήταν ελπιδοφόρα και οδήγησαν και σε ορισμένες εμπορικές εφαρμογές (όπως το flashfill MS-Excel). Τα ισχυρά νευρωνικά δίκτυα επέτρεψαν στους ερευνητές να περάσουν από αυστηρούς φορμαλισμούς σε πιο αφηρημένες προδιαγραφές, όπου ακόμη και μερικά απλά παραδείγματα εισόδου/εξόδου ήταν αρκετή περιγραφή στα συγκεκριμένα μοντέλα χωρίς να χρειάζεται ανθρώπινη επίβλεψη. Ωστόσο τα αποτελέσματα του κώδικα ήταν δυσνόητα και δεν ήταν δυνατό να κατανοηθεί πλήρως η λειτουργία τους κι άρα να προσαρμοστούν σε παρεμφερή προβλήματα.

Προκειμένου να μεγιστοποιηθούν τα θετικά από τις προηγούμενες μεθόδους, οι λύσεις που σχεδιάζονται κι ερευνούνται σήμερα συνδυάζουν τα νευρωνικά δίκτυα με τυπικές μεθόδους. Συνήθως η λύση υλοποιείται σε διακριτά στάδια όπου πρώτα τα δεδομένα ομαδοποιούνται με την χρήση μια εκ των δύο μεθόδων και μετά η νέα είσοδος αποτελεί βάση για το δεύτερο στάδιο όπου και δημιουργείται ο ζητούμενος κώδικας. Αυτή η προσέγγιση έχει ήδη δώσει καλύτερα αποτελέσματα τα οποία μάλιστα ερμηνεύονται ευκολότερα από τον ανθρώπινο παράγοντα.

6.2 Μελλοντικές επεκτάσεις

Παρά τα καθαρά οφέλη της μεθόδου του συνδυασμού νευρωνικών δικτύων και τυπικών μεθόδων υπάρχουν ακόμη σημαντικά ζητήματα που πρέπει να ερευνηθούν περαιτέρω. Η έρευνα αυτή θα βοηθήσει σημαντικά την πρόοδο της σύνθεσης προγραμμάτων θέτοντας τον άνθρωπο ένα βήμα πιο κοντά στην δημιουργία αυτόματου “λύτη προβλημάτων”. Συγκεκριμένα, ορισμένες σημαντικές ερευνητικές προκλήσεις παρατίθενται παρακάτω.

Μια σημαντική πρόκληση στο μέλλον θα είναι αυτή της επαλήθευσης ορθότητας. Για τον χρήστη είναι ιδιαίτερα σημαντικό να γνωρίζει ότι το πρόγραμμα του παρήχθη είναι το σωστό,

ειδικά όταν πρέπει να εκτελεστεί περισσότερες από μία φορές, σε ευαίσθητα δεδομένα ή σε μεγάλους όγκους δεδομένων όπου τα αποτελέσματα δεν είναι εύκολο να επαληθευτούν με μη αυτόματο τρόπο.

Μία ακόμη ενδιαφέρουσα κατεύθυνση θα ήταν η διευκόλυνση της χρήσης των μοντέλων σύνθεσης προγραμμάτων από απλούς χρήστες. Για να επιτευχθεί αυτό πρέπει οι τρόποι εισαγωγής των δεδομένων να μπορούν να προσαρμοστούν στον επιθυμητό τρόπο έκφρασης του χρήστη, όπως πχ φυσική γλώσσα, περιγραφές και λέξεις κλειδιά. Αυτή η προσέγγιση έχει αρχίσει να χρησιμοποιείται ευρέως σε εμπορικές πλατφόρμες, όπως το chatGTP[50] και το Bard.

Ένα ακόμη χαρακτηριστικό με περιθώρια εξέλιξης είναι η προσαρμοστικότητα. Η δυνατότητα δηλαδή αναμόχλευσης δεδομένων είτε από προηγούμενες εκτελέσεις του μοντέλου από τον ίδιο χρήστη ή ακόμη και από άλλους χρήστες.

Ιδιαίτερη σημασία επίσης δίνεται στην επέκταση του εύρους των παραγόμενων προγραμμάτων. Οι τρέχουσες τεχνικές σύνθεσης προγραμμάτων είναι ικανές να συνθέσουν μόνο μικρά προγράμματα με μερικές γραμμές εντολών. Η κλιμάκωση του μεγέθους των προγραμμάτων είναι άρα ενεργός τομέας έρευνας κι εξέλιξης. Σε αυτήν την κατεύθυνση θα μπορούσαν να ερευνηθούν ισχυρές στατιστικές τεχνικές βαθειάς μάθησης για να επεξεργαστούν τα δεδομένα εισόδου.

Τέλος, η βιομηχανοποίηση της σύνθεσης προγραμμάτων μπορεί να το καταστήσει προσιτό για το γενικό κοινό των μηχανικών, και να βοηθήσει ουσιαστικά την διαδικασία τεχνολογίας λογισμικού. Η σύνθεση προγραμμάτων είναι ένα σχετικά νέο πεδίο, και επομένως δεν είναι ακόμη τόσο διαδεδομένο. Περαιτέρω ανάπτυξη πλαισίων σύνθεσης και ενιαίων πλαισίων ανοικτού λογισμικού θα βοηθούσε σε αυτή την κατεύθυνση.

Βιβλιογραφία

1 Gulwani, Sumit, Oleksandr Polozov, and Rishabh Singh. "Program synthesis." *Foundations and Trends® in Programming Languages 4.1-2 (2017): 1-119.*

2 Post στη σελίδα της επιστήμης των υπολογιστών του πανεπιστημίου του Τέξας στο Όστιν, Program Synthesis Explained, James Bornholt, 6 January 2015

3 Διαλέξεις του μαθήματος "Introduction to Program Synthesis". Armando Solar-Lezama, 2018

4 Crevier, Daniel. *AI: the tumultuous history of the search for artificial intelligence*. Basic Books, Inc., 1993.

5 Wikipedia, Λήμμα "Program Synthesis", Νοέμβριος 2022

6 Backus, John W., et al. "The FORTRAN automatic coding system." *Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability*. 1957.

7 Manna, Zohar, and Richard Waldinger. "Knowledge and reasoning in program synthesis." *Artificial intelligence* 6.2 (1975): 175-208.

8 Alur, Rajeev, et al. *Syntax-guided synthesis*. IEEE, 2013.

9 Solar-Lezama, Armando. *Program synthesis by sketching*. University of California, Berkeley, 2008.

10 Gulwani, Sumit. "Automating string processing in spreadsheets using input-output examples." *ACM Sigplan Notices* 46.1 (2011): 317-330.

11 New World Encyclopedia, Λήμμα "Formal Logic", Μάϊος 2023

12 Wikipedia, Λήμμα "Λογική", Απρίλιος 2023

13 New World Encyclopedia, Λήμμα "Formal Logic", Μάϊος 2023

14 Detlovs, Vilnis, and Karlis Podnieks. "Introduction to mathematical logic." *University of Latvia* (2011).

15 Meyling, Michael. "Formal Predicate Calculus." (2013).

16 Σακελλαρίου, Ηλίας, et al. "Τεχνικές Λογικού Προγραμματισμού για Επίλυση Προβλημάτων." (2016).

17 Διαλέξεις του μαθήματος "Διακριτά Μαθηματικά", Εθνικό Μετσόβιο Πολυτεχνείο, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Διάλεξη "Στοιχεία Προτασιακής Λογικής", 2021

18 Wikipedia, Λήμμα "First-order logic", Μάιος 2023

19 Διαλέξεις του μαθήματος "Διακριτά Μαθηματικά", Εθνικό Μετσόβιο Πολυτεχνείο, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Διάλεξη "Στοιχεία Κατηγορηματικής Λογικής", 2021

20 Smullyan, R. M. "Theory of Formal Systems Annals of Mathematics Studies no. 47." (1961).

21 Chomsky, Noam. "Syntactic structures." *Syntactic Structures*. De Gruyter Mouton, 2009.

22 Chomsky, Noam, and Marcel P. Schützenberger. "The algebraic theory of context-free languages." *Studies in Logic and the Foundations of Mathematics*. Vol. 26. Elsevier, 1959. 118-161.

23 Διαλέξεις του μαθήματος "Γλωσσική Τεχνολογία", Πανεπιστήμιο Πατρών, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Διάλεξη "Συντακτική Ανάλυση", 2012-2013

24 Lars Marius Garshol, 'Αρθρο "BNF and EBNF: What are they and how do they work?"

25 Πλατφόρμα Isaac Computer Science, Recursion in BNF

26 Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. "Introduction to automata theory, languages, and computation." *Acm Sigact News* 32.1 (2001): 60-65.

27 Kaplan, Andreas. *Artificial Intelligence, Business and Civilization: Our Fate Made in Machines*. Routledge, 2022.

28 Haykin, Simon. "Neural networks: a comprehensive foundation prentice-hall upper saddle river." *NJ MATH Google Scholar* (1999).

29 Hardesty, Larry. "Explained: neural networks." *MIT News* 14 (2017).

30 Renu Khandelwal, Άρθρο Supervised, Unsupervised, and Reinforcement Learning, 20 Ιουλίου 2022

31 Mandic Danilo, P., and Chambers Jonathon. "Recurrent neural networks for prediction: learning algorithms, architectures and stability." (2001).

32 Lau, Tessa. "Why programming-by-demonstration systems fail: Lessons learned for usable ai." *AI Magazine* 30.4 (2009): 65-65.

33 Swarat Chaudhuri, Blogpost Sigplan, άρθρο Understanding the World with Program Synthesis, 4 Νοεμβρίου 2019

34 David, Cristina, and Daniel Kroening. "Program synthesis: challenges and opportunities." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 375.2104 (2017): 20150403.

35 Gulwani, Sumit, William R. Harris, and Rishabh Singh. "Spreadsheet data manipulation using examples." *Communications of the ACM* 55.8 (2012): 97-105.

36 Gottschlich, Justin, et al. "The three pillars of machine programming." *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. 2018.

37 Fowler, Martin, Άρθρο "Domain-Specific Languages Guide", (2019)

38 Madhusudan, P., and Sanjit A. Seshia. "Computer Aided Verification." CAV, 2012.

39 Phothilimthana, Phitchaya Mangpo, et al. "Scaling up superoptimization." *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*. 2016.

40 Perelman, Daniel, et al. "Test-driven synthesis." *ACM Sigplan Notices* 49.6 (2014): 408-418.

41 Polikarpova, Nadia, Ivan Kuraj, and Armando Solar-Lezama. "Program synthesis from polymorphic refinement types." *ACM SIGPLAN Notices* 51.6 (2016): 522-538.

42 Inala, Jeevana Priya, Rohit Singh, and Armando Solar-Lezama. "Synthesis of domain specific CNF encoders for bit-vector solvers." *Theory and Applications of Satisfiability Testing–SAT 2016: 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings 19*. Springer International Publishing, 2016.

43 Manna, Zohar, and Richard Waldinger. "A deductive approach to program synthesis." *ACM Transactions on Programming Languages and Systems (TOPLAS)* 2.1 (1980): 90-121.

44 Parisotto, Emilio, et al. "Neuro-symbolic program synthesis." *arXiv preprint arXiv:1611.01855* (2016).

45 Chaudhuri, Swarat. "Synthesizing Neurosymbolic Programs" *ACM Sigplan Notices* (2020)

46 Valkov, Lazar, et al. "Houdini: Lifelong learning as program synthesis." *Advances in neural information processing systems* 31 (2018).

47 Ellis, Kevin, et al. "Learning to infer graphics programs from hand-drawn images." *Advances in neural information processing systems* 31 (2018).

48 Cheung, Alvin, Armando Solar-Lezama, and Samuel Madden. "Using program synthesis for social recommendations." *Proceedings of the 21st ACM international conference on Information and knowledge management*. 2012.

49 Verma, Abhinav, et al. "Imitation-projected programmatic reinforcement learning." *Advances in Neural Information Processing Systems* 32 (2019).

50 Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.