==========
GREETING
==========
Γεια σας, είμαι ο Ορέστης Μελκονιάν και κάνω την πρακτική μου εδώ στο Δημόκριτο με γενικότερο αντικείμενο τις γλώσσες προγραμματισμού και ειδικά τη σχεδιάση μιας γλώσσας ειδικού-σκοπού, σε αντίθεση με τις γλώσσες γενικού σκοπού (C,java,php,...), ενσωματωμένη στο ΡΟΣ, η οποία προσφέρει μια διαφορετική και αφηρημένη οπτική στην μοντελοποίηση των προβλημάτων που εμφανίζονται στη ρομποτική.
==========
OVERVIEW
==========
Θα σας κάνω μια μικρή εισαγωγή για να δείτε τα κύρια κίνητρα για τη δουλειά που θα κάνω. Μετά θα δούμε 2 παραδείγματα στα οποία θα φανούν κάποια από τα προβλήματα των τωρινών τεχνολογιών, τα οποία θα δούμε και αναλυτικά μαζί με τις λύσεις που θα προσφέρει η γλώσσα αυτή. Τελος θα γίνει μια επισκόπηση σχετικής έρευνας που έχει γίνει σε διαφορετικούς τομείς.
==========
INTRODUCTION
==========
Η ρομποτική είναι ένα άκρως διεπιστημονικό πεδίο, οποτε τα συστηματα που υλοποιουνται ειναι απο τη φυση τους πολυπλοκα, μεγαλης κλιμακας και τιθεται κινδυνος μη-συντηρισημοτητας, δυσνοητου κωδικα και ελλειψης καταλληλων εργαλειων που αρμοζουν σε τετοιου ειδους project
FUTURE
Επισης πρεπει να σκεφτουμε και την εκρηκτικη αναπτυξη που θα εχει ως πεδιο τα επομενα χρονια, στα οποια τα ρομποτικα συστηματα θα ειναι πανταχου παρων και θα υπαρχουν απειρες δυνατοτητες με την αξιοποιηση του Διαδικτυου. Θα ειναι αναγκαια η γρηγορη σχεδιαση και υλοποιηση πολυποκων μοντελων με το ελαχιστο κοστος. Το ΡΟΣ εχει κανει ενα πολυ καλο βημα σε αυτή τη προσπαθεια, με το να προσφερει αφαιρεση ως προς το hardware kai mia open-source κοινοτητα με πληθωρα απο packages για συγκεκριμενες λειτουργιες, το οποιο ευνοει την επαναχρησιμοποιηση και το modularity ολου του συστηματος.
==========
MOTIVATION
==========
Γιατι ομως να κατσω να φτιαξω μια καινουρια γλωσσα? Δεν αρκουν οι υπαρχουσες? Για να απαντηθει αυτο πρεπει να ερωτηθουμε

- Εχουμε την καταλληλη αφαιρετικοτητα ετσι ωστε να μπορουμε με σχετικη ευκολια να υλοποιησουμε την μοντελοποιηση ενος προβληματος ρομποτικης που εχουμε στο μυαλο μας?
- Τα ρομποτικα συστηματα ειναι πραγματικου χρονου, οποτε εχουν εγγενη την εννοια του χρονου. Μας προσφερουν οι τωρινες γλωσσες εναν φυσικό τροπο να εκφρασουμε τον χρονο?
- Σε αυτα τα συστηματα η πληροφορια ρεει αναμεσα στα υποσυστηματα που τα απαρτουν για ακαθοριστο χρονο. Μας προσφερεται ευκολια χειρισμου τετοιων streams η πρεπει να καθομαστε να φτιαχνουμε ad-hoc λυσεις με buffers κλπ?
- Μηπως χανουμε χρονο σε επουσιωδη τεχνικα θεματα, τα οποια θα μπορουσε καλλιστα να τα κανει αυτοματα ενας εξυπνος compiler?
- Επισης, τα περισσοτερα προβληματα στο πεδιο αυτο δεν θα μπορουσαν να μοντελοποιηθουν κομψα και αποτελεσματικα από το υπολογιστικο μοντελου

του dataflow programming (που βλεπει το συστημα ως ενα κατευθυνομενο γραφο με κομβους να ειναι υπολογισμοι πανω σε δεδομενα και οι ακμες η πληροφορια που ρεει μεταξυ των κομβων)? Αν ναι, σιγουρα θα βοηθουσε μια dataflow γλωσσα.

==========
EXAMPLE
==========

Ας δουμε ενα παραδειγμα.
Εχουμε ενα drone που θελουμε να πεταξει αυτονομα απο την αρχη ενος χολ στο απεναντι τελος, εχοντας στη διαθεση του 4 sonar σενσορες(πανω,κατω,αριστερα,δεξια) που δινουν τιμες για την αποσταση απο τον αντιστοιχο τοιχο. Υποθετουμε οτι το drone ισορροπει παντα, δεν υπαρχει οπισθοδρομηση στο μονοπατι, και ειναι δυνατο να φτασει στο τελος με τις εντολες που διαθετουμε. Καθε χρονικη στιγμη του δινουμε εντολη για να κινηθει προσ μια απο αυτες τισ κατευθυνσεις και εαν δεν δωσουμε καποια εντολη προχωραει λιγο ευθεια. Καποιες φορες οι σενσορες δυσλειτουργουν και επιστρεφουν ακυρες(αρνητικες) τιμες.

ROS CODE

Εδω βλεπουμε τον κωδικα σε roscpp. Εχουμε μια callback funtion για καθε αισθητηρα που αποθηκευουν την τιμη που μας δινει στην καταλληλη θεση ενος πινακα. Υποθετουμε παλι οτι υπαρχει συγχονισμος των αισθητηρων και παντα θα ερθει η τελευταια τιμη απο το δεξι αισθητηρα. Ανιχνευουμε αρχικα το προβλημα του ελεγχου για εγκυρες τιμες στη αρχη καθε callback συναρτησης. Δεν θα ηταν πιο λογικο αυτο το φιλτραρισμα να γινεται καπου εξωτερικα? Επισης, σιγουρα το να ακυρωσουμε την τελευταια μας υποθεση θα δημιουργεισει πολλα προβληματα αφου θα χρειαστει να κραταμε και να διαχειριζομαστε buffers και γενικα να συχρονισουμε το συστημα manually.

FLOW CODE

Ας δουμε τωρα τον κωδικα μιας πιο δηλωτικης, dataflowish γλωσσας. Η μοντελοποιηση ως διαγραμμα dataflow ειναι απλη και διαισθητικα - Ενωνουμε τα τεσσερα streams σε ενα, φιλτραρουμε τις αρνητικες τιμες, και περναμε το προκυπτον stream σε μια συναρτηση που αποφασιζει την επομενη κινηση του drone. Ο ολικος συγχρονισμος του συστηματος γινεται αυτοματα απο το υποκειμενο συστημα που υλοποιει τη συναρτηση zip και θα μπορει να παιρνει παραμετρους σχετικα με το πως θα το κανει αυτο (δλδ να περιμενει μια καινουρια τιμη απο καθε stream, να δινει τιμη για καθε καινουρια τιμη συνδυαζοντας τις προηγουμενες, κλπ).
Ενα αλλο κρυφο πλεονεκτημα αυτης της προσεγγισης ειναι η δυνατοτητα του compiler να κανει σημασιολογικα-ορθες ανακαταταξεις στις πραξεις πανω στα streams, για λογους αποδοσης. Οποτε το συστημα μπορει να περιλαμβανει εναν σοβαρο optimizer με καλες προοπτικες.

==========
ANOTHER EXAMPLE
==========

Στον dataflow κωδικα που μολις ειδαμε, δεν υπηρχε το abstraction που ισως θα περιμενατε αφου παλι με ορους node,publish,subscribe ειχαμε να κανουμε. Αυτο ειναι απαραιτητο απο πλευρας compatability με το τεραστιο οικοσυστημα του ROΣ. Πρεπει σιγουρα να υπαρχει η δυνατοτητα εκφρασης των βασικων συνδετικων στοιχειων της αρχιτεκτονικης αυτης.
Ας δουμε το πραγματικα αφαιρετικο προσωπο της γλωσσας αυτης. Εδω βλεπουμε την υλοποιηση ενος PID controller που ειναι ενα αρκετα συνηθες feedback control loop. Σκοπος του ειναι να ελαχιστοποιησει ενα κοστος, που ειναι η διαφορα μιας τιμης ελεγχου απο μια επιθυμητη τιμη. Λειτουργει επαναληπτικα, με βηματικη μειωση του κοστους. Εξ ορισμου εχουμε το

dataflow διαγραμα και ειναι πανευκολο να το γραψουμε σε μια καταλληλα-κατασκευασμενη γλωσσα. Παρατηρουμε οτι ο κωδικας ειναι τελειως δηλωτικος και αφαιρετικος, αφου γραφουμε το ΤΙ θα γινει παραμελωντας το ΠΩΣ θα γινει. Επισης, παραμελουμε τις τεχνικες ιδιοτροπιες του ΡΟΣ, δηλαδη εννοεις οπως nodes,pusblish/subscribe,το setup των rosmsg, κλπ...

==========

PROBLEMS

==========

Ας δουμε πιο γενικα τα προβληματα που υπαρχουν

==========

SCALABILITY

==========

Στο θεμα της κλιμακωσης, ειδαμε οτι για πολυπλοκα schemes που συνδυαζουν πολλα streams υπαρχει αναγκη για "εσωτερικα υδραυλικα" μπλεγμενα με τον υπολογισμο που θα γινει πανω σε αυτα. Αυτο δεν κλιμακωνει και κανει δυσκολο την συντηρηση των προγραμματων. Ενα σχεδιαστικο προβλημα ειναι οτι δεν διαφοροποιουμε την πληροφορια που ρεει στο συστημα με τους υπολογισμους.

==========

UNTYPED TOPICS

==========

Ενα αλλο προβλημα ειναι οτι τα topics δεν υποκεινται στον αυστηρο ελεγχο του συστηματος τυπων. Οποτε τα streams δεν ειναι type-safe και δεν μπορουμε να δηλωσουμε περιορισμους πανω σε αυτα, με σκοπο να πιανουμε προβληματα στη φαση της μεταγλωττισης και οχι στο runtime. Πχ τρεχουμε ενα αλγοριθμο για visual detection πανω σε ενα audio stream. Γιατι να μην μας πει κατευθειαν ο compiler οτι δεν ειναι σημασιολογικα σωστο αυτο που παμε να κανουμε?

==========

TIME MODELLING

==========

Η θεωρια ελεγχου μοντελοποιει τη ρομποτικη συμπεριφορα κυριως με διαφορικες εξισωσεις στο πεδιο του χρονου. Οι τωρινες γλωσσες δεν μας προσφερουν εναν φυσικο τροπο να το εκφρασουμε αυτο ευκολα, παρολο που ο χρονος ειναι κατι εμφυτο στην καθημερινη αντιληψη.

==========

COMPILER RESTRICTION

==========

Επισης, ειμαστε πολυ συγκεκριμενοι στην τοπολογια του runtime, λεγοντας που να τρεξει το καθε rosnode ενω θα μπορουσε να αφησουμε εναν optimizer να παρει αυτες τις αποφασεις, πχ για να εχουμε τον ελαχιστο χρονο εκτελεσης ή να εχουμε την λιγοτερη καταναλωση μπαταριας.

==========

VERIFIABILITY

==========

Στο μελλον τα ρομποτικα συστηματα θα πρεπει να εχουν μεγαλυτερη ευθυνη, απολυτη αυτονομια και θα υπεισελθουν σε πολλους τομεις της κοινωνιας. Οποτε θα πρεπει να εχουμε τυπικες μεθοδους αποδειξης ορθοτητας των προγραμματων. Πχ babysitter χτυπαει το μωρο με το πιατο

Αυτο ειναι πολυ δυσκολο, εως ακατορθωτο σε μια γλωσσα γεματη side-effects οπως η C, ομως εχουν γινει αρκετα βηματα στον συναρτησιακο προγραμματισμο(που αρκετες αρχες αυτου υπαρχουν και στον dataflow προγραμματισμο).

==========

SOLUTIONS
==========
Τωρα, πως θα αντιμετωπιστουν αυτα τα προβληματα μεσω της σχεδιασης μια γλωσσας?

==========
A DSL DATAFLOW LANGUAGE
==========

Αρχικα, θα ειναι μια dataflow συναρτησιακη γλωσσα. Ο χρηστης-coder θα δινει στο συστημα μια high-level περιγραφη του διαγραμματος και ο compiler θα κανει ολα τα υπολοιπα, παραγωντας τελικα εγκυρο ros κωδικα(rosjava?roshask?roslisp?) που εκτελειται κανονικα στο ΡΟΣ

==========
ROS DEFINES DATAFLOW
==========

Γιατι dataflow? Επειδή η αρχιτεκτονικη του ΡΟΣ ουσιαστικα οριζει ενα διαγραμμα dataflow με κομβους τα rosnodes και ακμες τα topics.

==========
OPERATORS
==========

Η γλωσσα Θα μας προσφερει χρησιμους τελεστες πανω σε streams με τον συνδυασμο των οποιων θα μπορουμε να περιγραψουμε ενα αυθαιρετο διαγραμμα.

==========
TIME MODEL
==========

Η εννοια του χρονου υπαρχει μεσα στη γλωσσα, απο κατασκευης.

ετσι μια διαφορικη εξισωση που περιγραφει καποια ιδιοτητα της συμπεριφορας του robot μπορει να μετατραπει σε κωδικα με μηδαμινη προσπαθεια. Πχ η μετατοπιση ενος differential-drive robot

==========
DATA/CONTROL SEPARATION
==========

Υπάρχει διαχωρισμος των δεδομενων που ανταλλασονται απο τους υπολογισμους αυτους καθ'αυτους. Οπως ειδαμε και στο 1ο παραδειγμα το φιλτραρισμα ενος streams δεν εχει καμια δουλεια μεσα σε μια callback συναρτηση.

==========
TOPICS AS STREAMS
==========

Τα topics ειναι first-class citizens, δηλαδη εχουν τυπους, περνιουνται απο και σε συναρτησεις, οποτε εχουμε ενα τροπο αναλυτικης τους περιγραφης ως ξεχωριστες οντοτητες. Επισης, δεν θα χρειαζεται να φτιαχνουμε manually τα msg...

==========
PURE FUNCTIONS TO THE RESCUE
==========

Οσον αφορα το θεμα του verifiability η γλωσσα θα ενθαρυνει τη χρηση οσο το δυνατον περισσοτερων pure functions, ετσι ωστε να υπαρχει αυτο που λεμε referential transparency, δηλαδη μια διαδικασια που δεν εχει side-effects θα επιστρεφει παντα την ιδια τιμη με ιδια εισοδο, σε οποιο context και αν εκτελεστει. Αυτο μας επιτρεπει να εχουμε μεθοδους αυτοματης αποδειξης ορθοτητας.

==========
DYNAMIC RECONFIGURATION

==========

Η high-level περιγραφη που δινει ο χρηστης δεν οριζει το που θα εκτελεστουν τα nodes, ετσι ωστε ο compiler να αναθετει nodes δυναμικα για να μεγιστοποιησει μια μετρικη(καταναλωση ενεργειας, χρονος εκτελεσης,...)

==========

HARNESS THE CLOUD!

==========

Το γεγονος οτι το συστημα εχει την δυνατοτητα χειρισμου streams, σιγουρα θα προσφερει σημαντικη βοηθεια στην αξιοποιηση των web τεχνολογιων.

==========

RELATED WORK

==========

Υπαρχει πολυ υλικο ερευνας σχετικης με τα προβληματα που θιξαμε ως τωρα

==========

FRP

==========

Γενικοτερα, τα ρομποτικα συστηματα ειναι τελειο παραδειγμα reactive συστηματων. Εχει αναπτυχθει μια αρκετα θεμελιωμενη θεωρια για αυτα τα συστηματα, που αντλει ιδέες κυριως απο τον συναρτησιακό προγραμματισμό.

Επισης, επικεντρωνεται σε υβριδικα συστηματα, δηλαδη συστηματα που περιεχουν διακριτα αλλα και συνεχη δεδομενα. Μοντελοποιει κομψα τον χρονο, και βαζει σε προτεραιοτητα το event και error handling. Εχει εφαρμογες στην αναπτυξη γραφικων περιβαλλοντων, στην ρομποτικη και στη μουσικη.

==========

ROBOTICS

==========

Συγκεκριμενα τωρα στο πεδιο της ρομποτικης

=======

ROSHASK

=======

Εχουμε το roshask που ειναι το πρωτο βημα για την αξιοποιηση των streams στο ROS. Ειναι ενα client-library που προσφερει το απαραιτητο API που χρειαζεται το ΡΟΣ.

====

FROB

====

Μια ομαδα απο το yale εχει αναπτυξει μια DSL για ρομποτικη που ακολουθει τις αρχες του FRP, η οποια εχει περασει απο πολλες διορθωσεις και εχει θιξει τα σημαντικοτερα θεματα που προκυπτουν.

==========

BIG DATA

==========

Ιδεες απο τη θεωρια του dataflow programming εχουν παρει επισης οι ερευνητες στο πεδιο του big data

====

AKKA

====

Το ακκα ειναι ενα framework για αναπτυξη εφαρμογων μαγαλης κλιμακας μεσω ανταλλαγης μηνυματων. Υλοποιει το actor model οπως το ΡΟΣ. Οι developers με τον καιρο μπουχτισαν με την manual διαχειριση των streams οποτε οδηγηθηκαν, οπως και γω, στην υλοποιηση ενος ΑΠΙ για streams. Και αυτη η τεχνολογια ειναι βαθια επηρεασμενη απο FRP.

=====
NAIAD
=====

Ενα πολυ προσφατο συστημα που ενοποιει σχεδον ολες τις σχετικες ιδεες περι dataflow,frp,etc ειναι το NAIAΔ που αντιμετωπιζει επιτυχως πολλα performance issues που ειχαν μεχρι τοτε τετοιου ειδους συστηματα

=================
ZIRIA
=================

Ακομα και σε μια τελειως διαφορετικη επιστημονικη περιοχη, αυτη των ασυρματων τηλεπικοινωνιων, φαινεται να παρουσιαζονται πολυ παρομοια προβληματα.

Η ZIRIA ειναι μια DSL για προγραμματισμο συστηματων ασυρματων τηλεπικοινωνιων που επισης εχει εμπνευστει απο το συναρτησιακο κοσμο και το FRP μοντελο και προσφερει ενα πιο αφαιρετικο τροπο περιγραφης τετοιων συστηματων. Ολοκληρωθηκε επιτυχως, εχοντας ως αποτελεσμα πολυ πιο ωραιο, συντομο κωδικα για την υλοποιηση πψ ενος wifi receiver (23k -> 3k), κρατωντας βεβαια σχεδον τις ιδιες επιδοσεις με τους low-level ανταγωνιστες του.

Στοχος της δουλειας μου ειναι αυτο να επαναληφθει στην περιοχη της ρομποτικης.

==========
CONCLUSION
==========

Κλεινοντας, θα ηθελα να επισημανω οτι σιγουρα θα υπαρξει ενα learning curve ειδικα για coders που δεν εχουν εξοικιωθει με το συναρτησιακο προγραμματισμο, ομως θα υπαρξει ανταμοιβη σε σχεση με την παραγωγικοτητα τους και την καθαροτητα και τη συντηρισιμοτητα του κωδικα που παραγουν που εχει ως αποτελεσμα πιο γρηγορη αναπτυξη της ερευνητικης διαδικασιας.

Επισης, οπως σε καθε προσπαθεια σχεδιασης μιας γλωσσας ειδικου σκοπου, κυριος παραγοντ
ας ειναι το feedback απο ειδικους στο εκαστοτε πεδιο, αφου για αυτους φτιαχνεται η γλωσσα στην τελικη. Αρα, η ανταποκριση που θα εχω απο σας θα παιξει μεγαλο ρολο στο εργο μου.

Τελος, συνειδητοποιυμε ενα γενικοτερο μοτιβο στο κοσμο της πληροφορικης, που υπαρχει πολυ πρην την ρομποτικη. Οσο πιο πολυπλοκα γινονται τα συστηματα, τοσο πιο υψηλο πρεπει να ειναι το επιπεδο αφαιρεσης των εργαλείων που χρησιμοποιουμε.
Γιατι αλλωστε δεν προγραμματιζουμε πια σε γλωσσα μηχανης?

==========
THE END
==========