Язык программирования роботов в терминах потоков данных

Г.А. Зимин
Санкт-Петербургский государственный университет
Кафедра системного программирования
Email: zimin.grigory@gmail.com

Д.А. Мордвинов
Санкт-Петербургский государственный университет
Кафедра системного программирования
Email: mordvinov.dmitry@gmail.com

***Аннотация*—В статье описывается язык программирования роботов в терминах потоков данных на основе модельно-ориентированного подхода. Дается краткий обзор языков программирования роботов. Рассматриваются подходы к реализации систем управления роботами. В заключение представлено решение типовой задачи создания системы управления на описанном языке с применением архитектуры Брукса.**

I. Введение

Взаимодействие с роботами и языки программирования для них — это актуальная тема исследований: статьи, освещающие связанные с ними вопросы постоянно обозреваются на крупных конференциях, к примеру ICRA[1], IROS[2]. Последние три десятилетия продолжается активное изучение и исследование возможностей применения визуальных языков программирования — каждый год проводится крупные конференции, такие как VL/HCC[3]. В робототехнике визуальные языки программирования также нашли свое применение[1], [2], [3], [4], [5]. Они позволяют быстрее создавать и нагляднее отображать системы управления роботами, обучать робототехнике школьников — примерами этого могут служить различные среды программирования роботов для начинающих, такие как NXT-G[4], TRIK Studio[5], ROBOLAB[6].

Программы управления роботом по своей природе реактивны: они обрабатывают данные, непрерывно приходящие с множества датчиков и посылают команды на приводы. В силу этого для программирования роботов хорошо подходят реактивные языки, или языки программирования потоков данных (data flow languages). Эти языки также активно развивались от текстовых к широко распространенным сейчас визуальным языкам потоков данных[6]. В случае программирования потоков данных наглядность визуальных языков особо превосходит текстовые, так как сами потоки данных явно отображены на диаграмме. Существуют большие и сложные среды программирования, такие как LabVIEW[7] и Simulink[8], которые предоставляют большой и даже громоздкий набор библиотек для программирования роботов. Подробнее про языки программирования роботов будет сказано в секции II.

Для обучения основам робототехники и кибернетики существует большое количество конструкторов роботов, таких как линейка конструкторов LEGO MINDSTORMS[9], контструктор TRIK[10]. Подавляющее большинство современных языков программирования, используемых для обучения программированию на них, основаны на модели потока управления, а не потока данных. При этом во время их освоения зачастую появляется ощущение неудобства решения на них типовых задач управления роботом. Дополнительная «ступень» в виде простого в освоении потокового языка была бы весьма полезной при переходе от учебных языков программирования к тем, что используются в индустрии. Целью данной работы является создание такого языка программирония конструкторов роботов TRIK.

Остаток работы устроен следующим образом: в секции II дан краткий обзор существующих визуальных языков программирования для роботов и их применени, в секции III обозреваются архитектуры, на которых может быть построена система управлениями роботами, в секциях IV и V дается описание языка и его применение к решению задачи управлнения роботом при помощи оператора с автоматическим избеганием столкновений, наконец, в секции VI коротко будут сформулированы итоги работы.

[1]http://www.icra2016.org/ [Дата обращения: 10 марта 2016]
[2]http://www.iros2016.org/ [Дата обращения: 10 марта 2016]
[3]https://sites.google.com/site/vlhcc2016/ [Дата обращения: 10 марта 2016]
[4]http://www.legoengineering.com/program/nxt-g/ [Дата обращения: 10 марта 2016]
[5]http://www.trikset.com/ [Дата обращения: 10 марта 2016]
[6]http://www.legoengineering.com/program/robolab/ [Дата обращения: 10 марта 2016]
[7]http://www.ni.com/labview/ [Дата обращения: 10 марта 2016]
[8]http://www.mathworks.com/products/simulink/ [Дата обращения: 10 марта 2016]
[9]http://www.lego.com/en-us/mindstorms/products [Дата обращения: 10 марта 2016]
[10]http://blog.trikset.com/p/blog-page_6355.html/ [Дата обращения: 10 марта 2016]

II. Обзор современных языков программирования роботов

Среды программирования роботов можно разделить на три класса: учебные, позволяющие программировать небольших роботов; промышленные, обладающие богатым инструментарием для создания систем управления роботов, различных моделей; академические, которые реализуют какие-либо интересные идеи, однако часто недоступны для скачивания либо их стабильность оставляет желать лучшего.

К учебным, к примеру, можно отнести среду разработки EV3 Software[7] для конструктора Lego Mindstorms EV3, среда разработки NXT-G[11] и конструктор LEGO MINDSTORMS NXT и ROBOLAB[12], конструктор TRIK и среда программирования TRIK Studio[13]. Упомянутые среды программирования позволяют с легкостью решать типовые задачи управления роботом: найти выход из лабиринта, проехать по линии, используя сенсоры, создавая примитивные системы управления для обучения пользователя основам программирования и управления роботами. Но своей простотой они обязаны слабой гибкости языка, по сути они предоставляют последовательную модель потока управления роботом, описанную наглядными графическими моделями.

В инженерной индустрии весьма популярна среда общего назначения среда LabVIEW компании National Instruments и визуальный потоковый язык программирования G, а также среда для моделирования динамических моделей и различных систем управления Simulink. Данные программные продукты предоставляют пользователю огромный набор моделей и библиотек для создания любых систем управления, испытательных стендов, систем реального времени, используя модельно-ориентированный подход. В частности, на LabVIEW возможно программирование небольших роботов. Хотя известны применения LabVIEW в образовательных целях[8], большинство времени в образовательном процессе уходит на изучение самой среды, а не алгоритмов и подходов робототехники. Следует отметить, что эти среды распространяются под коммерческой лицензией.

Другой пример промышленной системы — Microsoft Robotics Developer Studio (MSRDS)[9], бесплатная для академического использования, удобную для программирования распределенных робототехнических систем в терминах потоков данных, компоненты которых представляются веб-сервисами. MSRDS официально поддерживает множество робототехнических платформ, среди которых есть LEGO NXT[10], для которого, однако отсутствует поддержка автономного режима. MSRDS обладает возможностью интеграции пользовательских робототехнических платформ, однако сама среда не поддерживается с 2014 года.

В данной области ведется множество исследовательских работ, к примеру, в диссертации[1] описан визуальный язык программирования в терминах расширенных машин Мура, [3],[4] описывают визуальный язык в терминах потоков данных с привязкой к языку программирования *occam-π* и инструментарию *Transterpreter*, а также применение к управлению «роем» роботов. Работа[5] описывает визуальный потоковый язык и среду для обучения новичков программированию роботов, среда предоставляет удобный интерфейс, однако полученная технология обладает слишком «слабым» функционалом и требует значительной доработки, а также недоступна для скачивания.

Исходя из результатов обзора ясно, что ни одна из упомянутых сред не подходит в том виде, в котором она есть для программирования TRIK в терминах потоков данных. При этом самым близким к желаемому является TRIK Studio, так как это единственная среда в открытом доступе с легко масштабируемой архитектурой, обладающей возможностью расширения новым визуальным языком программирования роботов и переиспользования кодовой базы таких «рутинных» операций, как взаимодействие с роботом.

III. Архитектура Брукса

С выходом статьи Родни Брукса[11], описывающей возможность декомпозировать задачу управления роботом горизонтально на уровни ответственности, стало возможно проще и эффективнее решать "составные"задачи управления, как, например, задачи телеоператорского контроля балансирующего робота с автоматическим избеганием столкновений с препятствиями. Языки потоков данных удачно подходят под эту модель, что подтверждают неоднократные публикации на эту тему: [2], [1], [4], [12]. В этой модели, система управления представлена как набор уровней ответственности, которые отвечают за различные модели поведения робота и строятся последовательно. При этом, по очевидным причинам, упрощается масштабируемость системы и увеличивается ее отказоустойчивость.

У архитектуры Брукса есть множество альтернатив, к примеру, «Колония» Джонотана Коннеля, архитектура «выбора-действия» Патти Мэйса, «схема двигателя» Рональда Аркина[13], модель управления для системы Телеробота[14]. Однако среди упомянутых подходов, подход Брукса является наиболее популярным, поэтому основополагающей концепцией разрабатывамого языка было решено сделать поддержку именно его.

IV. Описание языка

Развитие модельно-ориентированного (*DSM*) подхода[15] позволило быстро создавать достаточно сложные визуальные языки программирования. Среда

[11] http://www.legoengineering.com/program/nxt-g/ [Дата обращения: 10 марта 2016]

[12] http://www.legoengineering.com/program/robolab/ [Дата обращения: 10 марта 2016]

[13] http://www.trikset.com/ [Дата обращения: 10 марта 2016]

программирования TRIK Studio является примером системы, созданной с применением DSM-подхода на базе платформы QReal[16][17]. Основываясь на промышленном опыте разработчиков TRIK Studio, было решено создавать потоковый язык программирования роботов на базе платформы QReal. Первый прототип языка был создан в течение недели. На момент написания статьи язык и средства его поддержки находятся в активной разработке. Опишем некоторые особенности разрабатываемой технологии.

- Архитектура Брукса легко выражается средствами языка.
- Диаграммы поведения роботов, так же как и в TRIK Studio, имеют возможность быть проинтерпретированными на двумерной имитационной модели робота.
- В ближайших планах стоит генерация диаграмм потоков данных в текстовые языки, используемые для программирования TRIK — в первую очередь, JavaScript, F#[18] и Kotlin.

Элементы языка

Элементы языка можно разделить по назначению на несколько групп.

- *Элемент потока данных.* Связь, реализующая поток для передачи данных.
- *Блоки управления приводами.* Блоки принимающие числовые значения, отвечающие за подачу импульсов на приводы (силовые моторы, сервомоторы и т.д.).
- *Блоки считывания данных с сенсоров.* Блоки, генерирующие соответствующие значения, которые подлежат дальнейшей обработке.
- *Блоки синхронизации и фильтрации.* Позволяют временно блокировать передачу данных, устанавливать количество наборов пропускаемых данных и время между отправками.
- *Блоки поддержки устройств ввода.* Считывают данные с устройств операторского контроля (в данный момент поддержан только джойстик, в планах — компьютерная мышь и клавиатура).
- *Блоки рисования на экране.* Отвечают за векторное и растровое рисование на экране.
- *Блоки видеозрения.* Предоставляют доступ ко всем возможностям видеозрения контроллера TRIK.
- *Блоки взаимодействия между роботами.* Отвечают за групповую координацию роботов.
- *Блок текстового программирования.* Блок, позволяющий произвести обработку входных данных на текстовом языке (статически типизируемом диалекте Lua, поддержка которого «унаследована» от кодовой базы TRIK Studio). Чаще всего такой блок будет использоваться для задания математических операций над данными.
- *Блоки управляющих конструкций.* Включают циклы, условные развилки, множественный выбор, распараллеливание, подавление и ингибицию, блок «Подпрограмма» для переиспользования кода. Последние четыре блока обеспечивают поддержку архитектуры Брукса в языке.

V. Пример решения задачи

Рассмотрим задачу управления движением робота при помощи джойстика, при условии, что робот сам избегает лобовых столкновений с припятствиями. Предполагается, что программа пишется для двухколесного мобильного робота, оборудованного спереди инфракрасными датчиками расстояния, для управления колесами используются силовые моторы.

Разобьем задачу на два уровня поведения. Первый будет отвечать за обслуживание запросов пользователя. Второй будет ответственен за избегание столкновений: если робот близок к столкновению, то он должен уклониться от препятствия вне зависимости от того, что нажимает пользователь на пульте.

Рассмотрим первый уровень (рис. 1). Пользователь управляет роботом посредством джойстика. Джойстик генерирует данные, соответствующие нажатиям кнопок или манипулирований рычагом направления. Для простоты считаем, что нажатие любой кнопки завершит программу управления роботом. Данные с рычага преобразуются блоком текстового программирования в соответствующие импульсы моторов робота, которые в данном случае передаются «заглушкам», которые связаны с выходными портами блока «Подпрограмма».

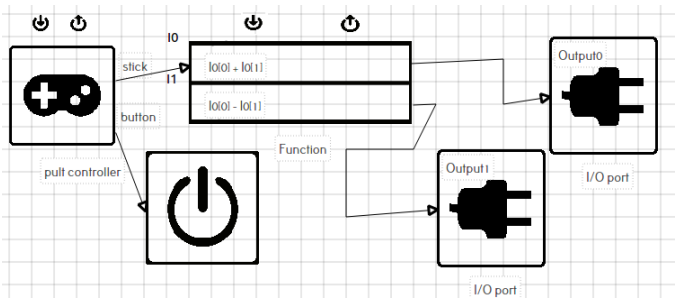

Рис. 1: Уровень управления с пульта.

Рассмотрим второй уровень (рис. 2): данные с датчиков расстояния собираются в вектор и передаются фильтру, который при опасности столкновения отправляет данные дальше в блок математической обработки (если условие не выполнилось, управление может быть передано по потоку «ошибки», в данной программе этот поток не указан; также между проверками условия приходящие данные не обрабатываются (теряются) в течение установленного пользователем времени). В блоке математической обработки вычисляется мощность, которую необходимо подать на 2 мотора. Вычисленные значения передаются выходным портам.

Имея два поведения, создаем управление роботом в

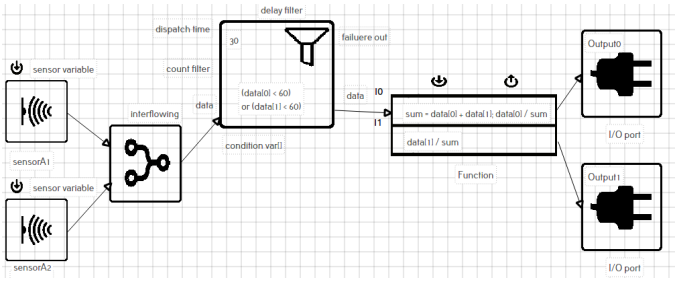

Рис. 2: Уровень избегания столкновений.

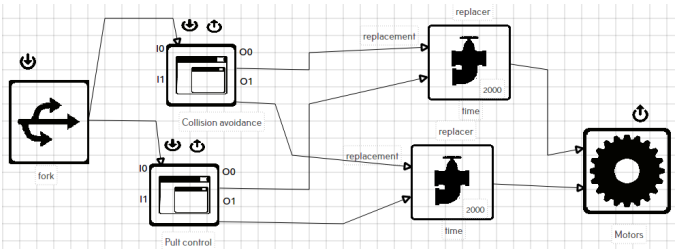

Рис. 3: Программа управления роботом.

модели Брукса (рис. 3). С помощью блока распараллеливания запускаем уровни поведения. Каждый уровень генерирует данные, соответствующие мощностям моторов, передаваемые на блок управления силовыми моторами. Так как второй уровень ответственности должен не позволить столкнуться с препятствием, его значения подавляют значения, полученные с первого уровня, с помощью блоков «Подавления».

VI. Заключение

На момент написания статьи был реализован прототип технологии програмирования роботов TRIK в терминах потоков данных. Система предоставляет возможность интерпретации диаграмм на двумерной имитационной модели робота и (в ближайшем будущем) генерации кода из диаграмм в текстовые языки, используемые для программирования TRIK (JavaScript, F# и Kotlin). Также технология предоставляет поддержку архитектуры Брукса на уровне языка, что продемонстрировано в статье на примере операторского управления роботом с автоматическим избеганием столковений.

Полученная система может рассматриваться как платформа для дальнейших научных исследований. К примеру, интересной представляется автоматическая генерации метамодели языка по спецификациям ПО промежуточного уровня на роботе (к примеру, ROS[14]). Другим направлением возможной работы является описание строгой семантики языка для применения различных формальных методов анализа программ, выраженных в нем.

[14]http://www.ros.org/ [Дата обращения: 10 марта 2016]

Список литературы

[1] Banyasad Omid. A Visual Programming Environment for Autonomous Robots. 2000.

[2] Simpson Jonathan, Jacobsen Christian L, Jadud Matthew C. Mobile robot control // Communicating Process Architectures. 2006. С. 225.

[3] Simpson Jonathan, Jacobsen Christian L. Visual Process-Oriented Programming for Robotics. // CPA. 2008. С. 365–380.

[4] Process-Oriented Subsumption Architectures in Swarm Robotic Systems. / Jeremy C Posso, Adam T Sampson, Jonathan Simpson [и др.] // CPA. 2011. С. 303–316.

[5] Diprose James P, MacDonald Bruce A, Hosking John G. Ruru: A spatial and interactive visual programming language for novice robot programming // Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on / IEEE. 2011. С. 25–32.

[6] Johnston Wesley M, Hanna JR, Millar Richard J. Advances in dataflow programming languages // ACM Computing Surveys (CSUR). 2004. Т. 36, № 1. С. 1–34.

[7] Rollins Mark. Beginning LEGO Mindstorms EV3.

[8] Using LEGO NXT Mobile Robots With LabVIEW for Undergraduate Courses on Mechatronics / Jesús M. Gomez-de Gabriel, Anthony Mandow, Jesús Fernandez-Lozano [и др.] // IEEE Trans. Educ. 2011. Т. 54, № 1. С. 41–47.

[9] Jackson Jared. Microsoft robotics studio: A technical introduction // Robotics & Automation Magazine, IEEE. 2007. Т. 14, № 4. С. 82–87.

[10] Kim Seung Han, Jeon Jae Wook. Programming LEGO Mindstorms NXT with visual programming // Control, Automation and Systems, 2007. ICCAS'07. International Conference on / IEEE. 2007. С. 2468–2472.

[11] Brooks R. A robust layered control system for a mobile robot // IEEE J. Robot. Automat. 1986. Т. 2, № 1. С. 14–23.

[12] Proetzsch Martin, Luksch Tobias, Berns Karsten. The behaviour-based control architecture iB2C for complex robotic systems // KI 2007: Advances in Artificial Intelligence. Springer, 2007. С. 494–497.

[13] Simpson Jonathan, Ritson Carl G. Toward Process Architectures for Behavioural Robotics. // CPA. 2009. С. 375–386.

[14] Albus James Sacra, McCain Harry G, Lumia Ronald. NASA/NBS standard reference model for telerobot control system architecture (NASREM). National Institute of Standards and Technology Gaithersburg, MD, 1989.

[15] Кознов Дмитрий Владимирович. Основы визуального моделирования // М.: Изд-во Интернетуниверситета информационных технологий, ИНТУИТ. ру, БИНОМ, Лаборатория знаний. 2008.

[16] Средства быстрой разработки предметно-ориентированных решений в metaCASE-средстве QReal / АС Кузенкова, АО Дерипаска, КС Таран [и др.] // Научно-технические ведомости СПбГПУ. С. 142.

[17] QReal DSM platform-An Environment for Creation of Specific Visual IDEs / Anastasiia Kuzenkova, Anna Deripaska, Timofey Bryksin [и др.] // ENASE 2013 — Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering. Setubal, Portugal: SciTePress, 2013. С. 205–211.

[18] Kirsanov Alexander, Kirilenko Iakov, Melentyev Kirill. Robotics reactive programming with F#/Mono // Proceedings of the 10th Central and Eastern European Software Engineering Conference in Russia / ACM. 2014. С. 16.