Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
“Санкт-Петербургский государственный университет”

Кафедра Системного Программирования

Агапова Татьяна Юрьевна

Построение дискретного плана управления по спецификации в виде LTL-формулы

Курсовая работа 1 курса магистратуры

Зав. кафедрой системного программирования:
д.ф.-м.н., профессор Терехов А.Н.

Научный руководитель:
ст. преп. Литвинов Ю.В.

Санкт-Петербург
2015

Введение

С каждым годом вычислительные машины играют всё большую роль в жизни людей. В связи с этим естественно развитие способов человеко-машинного взаимодействия, которые со временем становятся более интуитивными и простыми в использовании.

Одним из наиболее естественных для человека средств поставить задачу является естественный язык. К сожалению, большинство существующих на данный момент систем с голосовым управлением разработаны таким образом, что принимают фиксированный набор простых команд, которые напрямую могут быть отображены в действия. Такие системы требуют постоянного контроля со стороны человека.

В идеале хотелось бы иметь возможность описывать желаемое поведение или ставить задачу машине на естественном языке. При этом такая задача может быть достаточно сложной, требовать наличия модели мира, в котором надлежит её исполнить, а также взаимодействия с этим миром. Для этого необходимы методы преобразования высокоуровневых описаний поведения в низкоуровневый план управления системой, который может быть исполнен ей напрямую.

В данном направлении ведутся исследования [1, 2], и среди прочих следует особо выделить подход [3], основанный на темпоральных логиках [4]. В отличие от многих других, он позволяет формально удостовериться в возможности выполнения задачи в заданном мире и гарантировать корректность низкоуровневого плана, построенного по описанию задачи в виде формулы темпоральной логики. К тому же, темпоральные логики, являясь формальным инструментом, всё же позволяют описывать поведения в форме, весьма схожей с естественными языковыми конструкциями. Таким образом, текст на естественном языке преобразуется вначале в такую формулу, а затем по ней генерируется корректный по построению план управления.

Целью моей магистерской диссертации является разработка системы с голосовым управлением, допускающей высокоуровневую постановку задач на естественном языке, а также реализация её в реальном устройстве, в качестве которого планируется использовать кибернетическую платформу ТРИК[1]. Кроме того, ввиду упомянутых выше достоинств темпоральных логик, планируется использовать их в качестве формального аппарата представления задачи.

Данная курсовая работа является первым шагом в этом направлении. В её ходе планируется разработать модель простого дискретного исполнителя, способного выполнять задачи, описанные в виде формул темпоральной логики. Эта модель в дальнейшем будет использоваться в качестве среды для апробации способов улучшения реализуемого подхода.

---

[1] URL: http://www.trikset.com/, дата обращения: 18.12.2015.

Используемый подход

В используемом подходе преобразование постановки задачи на естественном языке в план управления осуществляется в три этапа:

1. Сначала текст на естественном языке преобразуется в формулу линейной темпоральной логики (LTL) с помощью описаний, позволяющих построить предикаты LTL по конструкциям естественного языка.
2. Затем по этой формуле и по модели мира, заданной также в формате LTL, генерируется конечный автомат, представляющий собой желаемое поведение.
3. Наконец, данный автомат преобразуется в некоторое непрерывное управление (к примеру, выраженное в виде кода на некотором языке программирования). Здесь используется информация, позволяющая переводить предикаты LTL в управление, осуществляющее соответствующее действие.

Общая схема описанного процесса изображена на Рис. 1.

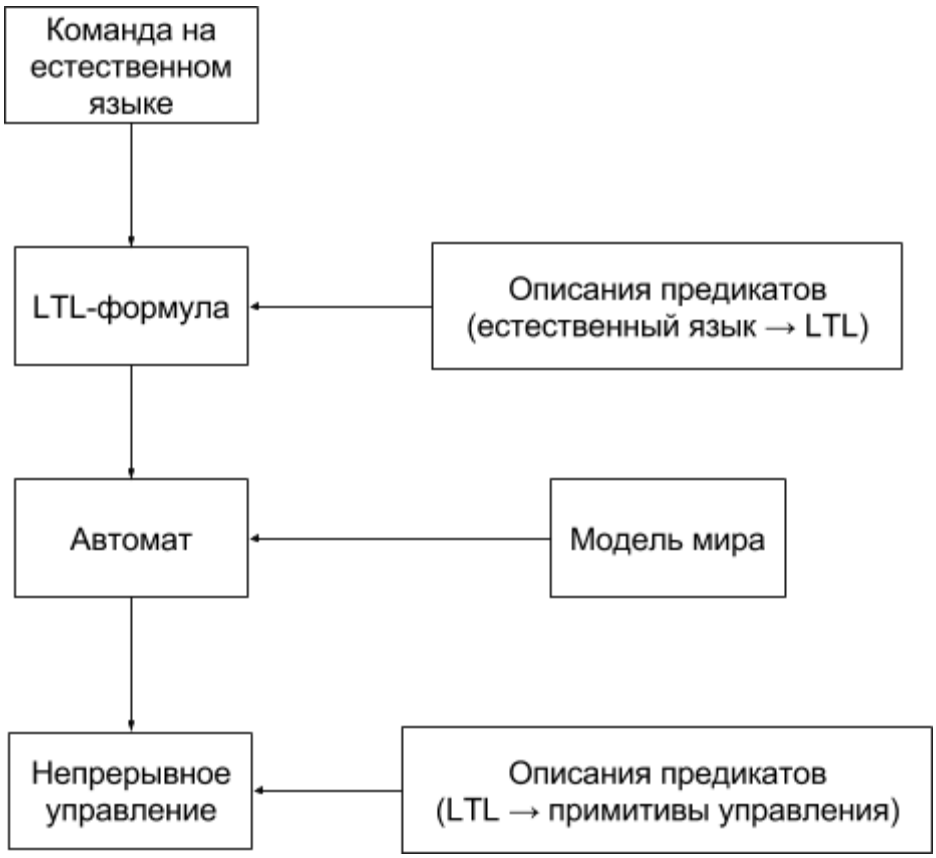

Рис. 1. Процесс получения непрерывного плана управления по постановке задачи на естественном языке.

В настоящей курсовой работе планируется реализовать шаг 2, используя идеи, представленные в [5, 6, 7].

Обзор теоретической базы

Прежде чем перейти к описанию самого подхода к планированию, рассмотрим некоторые теоретические понятия, необходимые для его понимания.

Темпоральные логики

В классической логике истинность утверждений не зависит от времени, ввиду чего с её помощью сложно описывать постоянно изменяющийся реальный мир, обладающий причинно-следственными связями, упорядоченными во времени. По этой причине классическая логика плохо подходит в качестве формализма для описания сложных динамических систем. Для этой цели широко применяются темпоральные логики — класс модальных логик, расширяющих классическую логику операторами, позволяющими выразить зависимость значения логической формулы от момента времени, в который вычисляется это значение.

Существует множество различных способов расширить классическую логику до временной. Здесь мы остановимся на варианте под названием темпоральная логика линейного времени (Linear Temporal Logic, LTL [10]). Именно она используется авторами описываемого подхода для спецификации задач исполнителю, и её же планируется применить в нашей реализации. Кроме того, для некоторого специального подмножества формул LTL существует полиномиальный алгоритм проверки выполнимости, что весьма ценно при построении реальной системы, использующей эту логику.

LTL вводит всего два новых оператора, характеризующих упорядоченность утверждений во времени:

- X (*next*, в следующий момент времени) — утверждение Xq истинно в момент времени t, если q истинно в момент $t+1$;
- U (*until*, до того как) — pUq истинно, если когда-нибудь в будущем будет верно q, а до того времени, начиная с текущего момента, верно p.

Также можно ввести дополнительные операторы — F (*future*, когда-нибудь в будущем) и G (*globally*, всегда). Они очевидным образом выражаются через U: $Fq = trueUq$ и $Gq = \neg F \neg q$.

Таким образом, множество формул LTL задаётся грамматикой:

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

где p — атомарный предикат.

Для того чтобы производить какие-либо утверждения о системе, необходимо иметь её формальную модель. Такая модель называется структурой Крипке и определяется следующим образом: это пятёрка (S, S_0, R, AP, L), где:

- S — множество возможных состояний системы;
- $S_0 \subseteq S$ — непустое множество начальных состояний;
- R — множество переходов, тотальное отношение на $S \times S$: $\forall s \in S \; \exists s' \in S : (s, s') \in R$;
- AP — конечное множество атомарных предикатов;
- $L : S \to 2^{AP}$ — функция пометок, сопоставляющая каждому состоянию системы множество истинных в нём атомарных предикатов.

Вычислением называется последовательность состояний, в которой первое состояние принадлежит начальному множеству, а каждая пара соседних состояний принадлежит множеству переходов.

Теперь можно определить семантику формул LTL: истинность формул рассматривается на вычислениях. Формула верна на вычислении, если она верна в его начальном состоянии.

Model Checking

На темпоральных логиках основан такой раздел формальной верификации программ, как model checking. В этом методе динамическая система описывается в виде модели, представляющей собой формулу темпоральной логики, формализующую инварианты и возможное поведение системы. Алгоритм model checking проверяет, будет ли заданная логическая формула истинна на данной модели системы. При этом в случае, если формула не выполняется, алгоритм выдаёт последовательность состояний системы, на которой эта формула ложна. Подробнее с этим методом верификации можно ознакомиться в [8, 9].

Обзор используемого подхода к планированию

В статьях [5, 6, 7] авторы представляют подход к планированию движений мобильного робота с помощью построения непрерывного плана движения по формуле LTL. Рассмотрим по порядку развитие этого подхода в каждой из данных работ.

Планирование движения мобильного робота

В работе [5], положившей начало целому разделу в планировании действий в робототехнике, предлагается способ постановки задач роботу в виде формулы LTL и генерации по этой формуле и модели мира корректного плана управления, формально удовлетворяющему спецификации.

В качестве исполнителя используется робот в двумерном непрерывном полигональном мире. Некоторые части мира представляют собой области интереса: комнаты, в которые нужно попасть, или препятствия, которые необходимо обходить.

Рассмотрим пример спецификации с помощью LTL. Допустим, у нас имеется среда с четырьмя регионами: π_1, π_2, π_3 и π_4. В начале работы робот находится в регионе π_1. Тогда формула

$$\varphi = F(\pi_2 \wedge F(\pi_3 \wedge F(\pi_4 \wedge (\neg\pi_2 \wedge \neg\pi_3) U \pi_1)))$$

задаёт следующее поведение: “посетить π_2, затем π_3, затем π_4, затем вернуться в π_1, избегая областей π_2 и π_3”.

Таким образом, необходимо построить такое непрерывное управление, которое позволяло бы решить поставленную исполнителю задачу в данной модели мира.

Для решения этой проблемы авторы используют механизмы model checking, для чего им требуется перевести модель в дискретное пространство и построить структуру Крипке по описанию мира. Это происходит следующим образом:

1. Непрерывный мир разбивается с помощью триангуляции на регионы. Они станут состояниями структуры Крипке.
2. Начальное состояние — регион, в котором оказалась начальная точка.
3. Переход между двумя состояниями возможен, если регионы смежны.
4. Предикаты соответствуют местонахождению робота в какой-нибудь области интереса.
5. Функция пометок отображает регион на область интереса, внутри которой данный регион находится.

К построенной структуре Крипке можно применять любые существующие инструменты model checking. Поскольку они генерируют контрпример, на котором формула не выполняется, всё, что нужно сделать, это:

1. По спецификации задачи φ построить её отрицание $\neg\varphi$.
2. Запустить на модели мира и $\neg\varphi$ верификатор.
3. Если он не найдёт контрпримера, значит, в системе может выполняться отрицание спецификации, а значит, среда не допускает выполнение задачи.
4. Если же верификатор выдаст контрпример, это и будет траектория, позволяющая успешно решить поставленную задачу.

Всё, что остаётся после этого — перевести полученную дискретную траекторию обратно в непрерывное пространство и построить соответствующее управление. Этот шаг производится с использованием методов, описанных в [11].

Планирование с учётом значений датчиков

В статье [6] подход расширяется использованием датчиков, сообщающих роботу информацию о состоянии среды. Таким образом, множество атомарных предикатов предметной области теперь состоит не только из предикатов принадлежности региону, но и из предикатов, соответствующих датчикам робота. Это также позволяет естественным образом моделировать мультиагентные системы, поскольку в этом случае остальные роботы являются частью состояния среды, воспринимаемого с помощью датчиков.

Поскольку робот теперь должен учитывать состояние среды, не известное изначально, его поведение теперь не описывается траекторией на карте, а скорее представляет из себя автомат с переходами, зависящими от показаний датчиков. В связи с этим использование model checking уже не является целесообразным.

Вместо этого авторы предлагают описывать спецификации в виде LTL формул класса Generalized Reactivity(1) (GR(1)) [12]. Несмотря на ограничение множества допустимых формул, этот класс достаточно выразителен для описания подавляющего числа спецификаций. Кроме того, он имеет важное преимущество: для проверки выполнимости формул GR(1) известен полиномиальный алгоритм, в то время как в общем случае для LTL доказана двойная экспоненциальная сложность этой задачи.

Формула GR(1) имеет вид $\varphi = \varphi_e \Rightarrow \varphi_s$, где φ_e — формула, описывающая требования к окружению, φ_s - требования к системе. Обе части имеют вид

$$\varphi_e = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e \qquad \text{и} \qquad \varphi_s = \varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s,$$

где

- φ_i^e, φ_i^s — ограничения на начальные значения датчиков и начальное состояние системы;
- φ_t^e, φ_t^s — описание переходов среды и системы — возможных последующих состояниях среды и системы при данных текущих;

- φ_g^e, φ_g^s — описание целей, которые необходимо достичь среде и системе; для системы описание цели представляет собой спецификацию задачи, которую необходимо решить.

Работа [12] описывает алгоритм, строящий за время $O(n^3)$ по формуле GR(1) автомат, все траектории которого формально удовлетворяют LTL-спецификации. В контексте планирования действий мобильного робота этот автомат представляет из себя стратегию поведения робота — выбор следующего состояния по текущему состоянию и значениям сенсоров. При этом, если среда не удовлетворяет описанным ограничениям (φ_e ложно), то импликация $\varphi_e \Rightarrow \varphi_s$ становится истинной вне зависимости от значения φ_s. В этом случае нет никаких гарантий относительно того, как будет вести себя система.

Реактивное планирование действий

Работа [7] расширяет подход, описанный в предыдущем разделе, возможностью формализовать действия робота в общем виде. К предикатам, описывающим состояния системы, добавляется также множество предикатов, свидетельствующих о том, какие действия выполняются в данный момент (например, происходит перемещение из одного региона в другой или нет, включена или выключена камера и т.д.). Это позволяет планировать с помощью данного подхода более сложное поведение, нежели перемещение из одного региона в другой.

Пример применения подхода

Рассмотрим модель робота, оперирующего в дискретном пространстве, представляющем из себя двумерный мир из $m \times n$ клеток. За один шаг робот может переместиться в соседние клетки слева, справа, сверху или снизу. Некоторые клетки представляют собой препятствия, которые робот не может пересекать, а некоторые — особые выделенные области, которые могут являться целью для передвижения робота. Также в некоторых клетках располагаются предметы, которые робот может поднимать и перетаскивать из клетки в клетку. При этом робот не знает, где находятся предметы изначально, но может узнать, есть ли в данной клетке предмет, с помощью сенсоров. Пример подобной среды изображён на Рис. 2.

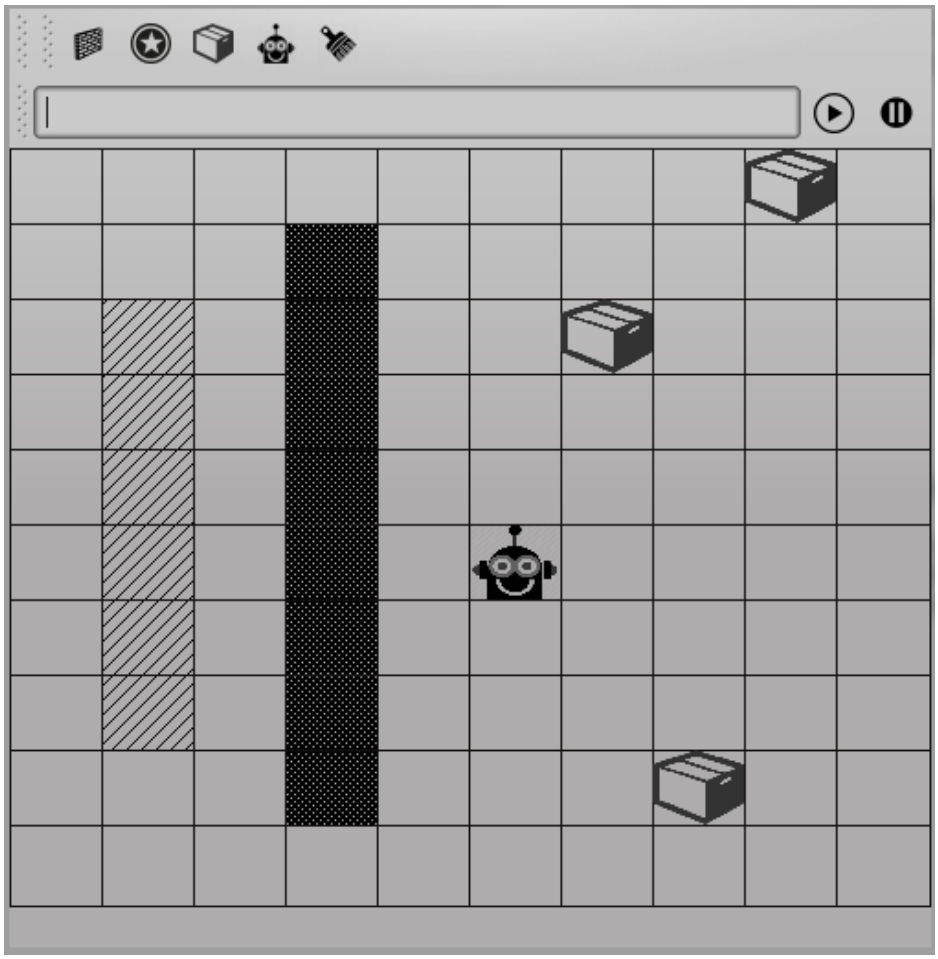

Рис. 2. Клеточный мир робота.

Задача робота — обходить мир, не сталкиваясь с препятствиями, и перетаскивать предметы в отмеченные области. При отпускании предмета в отмеченной области он исчезает, поскольку после его доставки в нужное место больше нет необходимости принимать его во внимание.

Довольно естественным образом можно формализовать эту задачу в терминах LTL:

- $X = \{x_{ij}\} \cup \{Carry\}$ — переменные системы:
  - x_{ij} — показывает, находится ли робот в клетке (i, j);
  - $Carry$ - истинно в случае, если робот держит предмет;
- $Y = \{a_{ij}\} \cup \{Item\}$ — переменные среды:
  - a_{ij} — помечает выделенные области;
  - $Item$ — значение датчика, которое истинно, если в клетке с роботом находится предмет;
  - заметим, что можно было бы ввести набор переменных среды, которые отвечали бы за наличие препятствий в каждой клетке, но, поскольку карта препятствий известна заранее и не изменяется, её можно учесть при генерации допустимых переходов между клетками, упрощая таким образом схему и уменьшая пространство состояний.

Напомним, что описание поведения робота и среды задаётся формулой $\varphi = \varphi_e \Rightarrow \varphi_s$, при этом $\varphi_e = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e$ и $\varphi_s = \varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s$.

Вначале обратим внимание на спецификацию среды. Так как мы не накладываем никаких ограничений на наличие предметов в начальной клетке робота или на его местонахождение в отмеченных областях, то $\varphi_i^e = True$. Что касается изменений среды, то требуется явно указать, что отмеченные области не двигаются: $\varphi_t^e = \wedge G(a_{ij} \Rightarrow Xa_{ij})$. Помимо этого, мы больше не делаем никаких предположений о среде, поэтому $\varphi_g^e = GFTrue$.

Вторая часть импликации касается поведения робота. В начале работы он может находиться в любой клетке, в которой нет препятствия, не держа никакого предмета. Поскольку все препятствия известны заранее, нет необходимости явно моделировать условие отсутствия препятствий в начальном положении — можно построить формулы только для тех клеток, в которых препятствий нет: $\varphi_i^s = \vee_{(i,j)}(x_{ij} \wedge_{(k,l)\neq(i,j)} \neg x_{kl} \wedge \neg Carry)$, если в клетке (i, j) нет препятствия. Кроме этих условий, данная формула также отражает условие взаимного исключения: если робот находится в какой-то клетке, он не может находиться ни в какой другой.

Следующая подформула описывает допустимые действия робота в зависимости от его текущего состояния и значений датчиков. Она состоит из нескольких блоков формул:

- первый блок состоит из формул, описывающих допустимые переходы между клетками; формулы этого блока строятся на основании информации о препятствиях и того факта, что робот может перемещаться строго вверх, влево, вниз или вправо: $G(x_{ij} \Rightarrow \vee_{(k,l),\text{ если переход } (i,j)\to(k,l) \text{ допустим}} Xx_{kl})$;
- второй блок описывает условие взаимного исключения аналогично начальному состоянию;

- ещё две подформулы описывают обращение робота с предметами:
  - если мы видим предмет в текущей клетке и ещё не держим никакой другой — схватить его, при этом на следующем шаге мы останемся в той же клетке:
    $$G(Item \wedge \neg Carry \Rightarrow (XCarry \wedge_{(i,j)}(x_{ij} \Leftrightarrow Xx_{ij})))\,;$$
  - если мы проносим предмет над выделенной областью, его нужно отпустить, при этом на следующем шаге также нужно остаться в той же клетке:
    $$G(\wedge_{(i,j)}(Carry \wedge x_{ij} \wedge a_{ij} \Rightarrow X(\neg Carry \wedge x_{ij})))\,.$$

Наконец, опишем бесконечный поиск предметов во всех клетках: $\varphi_g^s = \wedge_{(i,j)} GF x_{ij}$. Теперь робот будет бесконечно блуждать по всем доступным клеткам мира. Именно эта формула является спецификацией задачи, остальные описывают допустимое поведение робота и среды и, таким образом, неизменны при фиксированной модели исполнителя.

По построенной LTL-формуле можно построить автомат, реализующий желаемое поведение, с помощью методов, описанных в [12]. Пример траектории робота, порождаемой таким автоматом, приведён на Рис. 3.

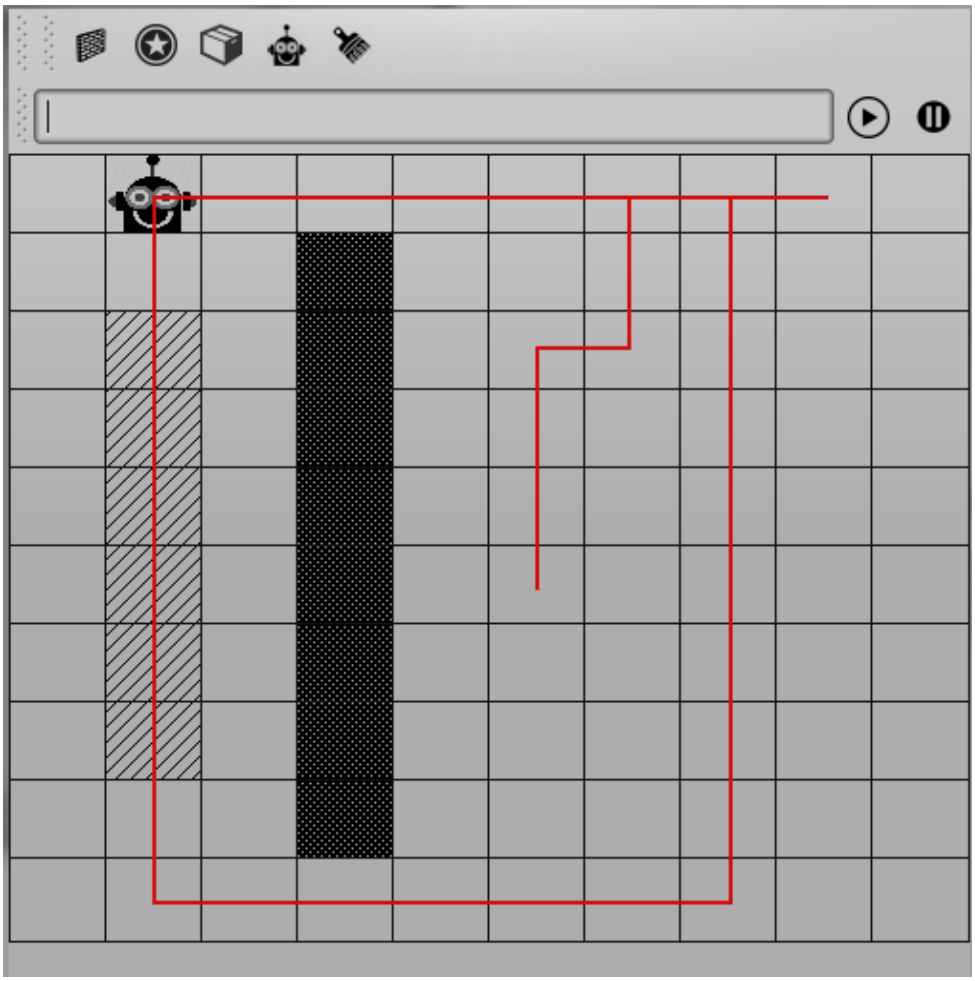

Рис. 3. Траектория робота, разносящего предметы по обозначенным областям.

Заключение и дальнейшие перспективы

В ходе данной курсовой работы на основе работ [5, 6, 7] была разработана модель простого дискретного исполнителя, способного выполнять задачи, поставленные в виде формул LTL, что позволило углубить понимание предметной области и предложенного в вышеупомянутых статьях подхода.

Этот подход обладает неоспоримыми достоинствами: LTL, даже ограниченная до специального подкласса GR(1), достаточно выразительна для описания большинства спецификаций, при этом сами описания довольно естественны и интуитивны, в связи с чем LTL является наиболее перспективным формализмом для спецификации задач, изначально поставленных на естественном языке. Наконец, важным аргументом в поддержку планирования действий с помощью LTL является возможность генерации поведения роботов, которое гарантированно, корректно по построению решает поставленную задачу.

В то же время у рассмотренного подхода присутствуют некоторые существенные ограничения. Во-первых, несмотря на полиномиальность алгоритма построения автомата в зависимости от пространства состояний, размер самого пространства может экспоненциально зависеть от числа датчиков и действий, что приводит к невозможности использовать такой подход в реальном времени. Во-вторых, система полагается на совершенно точное, полное и корректное моделирование среды: если среда ведёт себя хоть немного не так, как описано в спецификации, невозможно получить никаких гарантий относительно поведения робота. По этой причине с использованием только описанных методов невозможно получить по-настоящему реактивную систему, способную учитывать существенные изменения мира в своих действиях.

Существуют работы, посвящённые этим проблемам. К примеру, в [13] вместо полного синтеза автомата вычисляются “выигрышные” состояния системы и выбирается наиболее близкое состояние в смысле пути в графе состояний. Это позволяет существенно ускорить вычисление стратегии и производить его на каждом шаге, а не только в самом начале работы системы, что, в свою очередь, даёт роботу возможность реагировать на изменения в поведении среды.

Помимо расширения модели с целью обеспечить настоящую реактивность, крайне желательным также является её перенос на реальное устройство. Наряду с техническими задачами, связанными с реализацией рассмотренного подхода на конкретной программно-аппаратной платформе, также будет необходимо решить задачу генерации непрерывного плана управления, поскольку представленный в данной работе исполнитель оперирует в дискретном мире, что, конечно, неверно для реальной физической системы.

Литература

1. *S. Lauria, T. Kyriacou, G. Bugmann, J. Bos, and E. Klein.* Converting natural language route instructions into robot-executable procedures. In Proceedings of the 2002 IEEE International Workshop on Robot and Human Interactive Communication, pages 223–228, Berlin, 2002.
2. *A. J. Martignoni III and W. D. Smart*. Programming robots using high-level task descriptions. In M. Rosenstein and M. Ghavamzadeh, editors, Proceedings of the AAAI Workshop on Supervisory Control of Learning and Adaptive Systems, pages 49–54, June 2004.
3. *Kress-Gazit, H., Fainekos, G., Pappas, G.* Translating structured English to robot controllers. Advanced Robotics 22(12), 2008.
4. *M.R.A. Huth and M.D Ryan.* Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, 2000.
5. *Fainekos Georgios E, Kress-Gazit Hadas, Pappas George J.* Temporal logic motion planning for mobile robots // Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on / IEEE. 2005. С. 2020–2025.
6. *Kress-Gazit Hadas, Fainekos Georgios E, Pappas George J.* Where's Waldo? Sensor-based temporal logic motion planning // Robotics and Automation, 2007 IEEE Conference on / IEEE. 2007. С. 3116–3121.
7. *Kress-Gazit Hadas, Fainekos Georgios E, Pappas George J.* Temporal-logic-based reactive mission and motion planning // Robotics, IEEE Transactions on. 2009. Т. 25, No 6. С. 1370–1381.
8. *Карпов Юрий Глебович.* MODEL CHECKING. Верификация параллельных и распределенных программных систем. БХВ-Петербург, 2010.
9. *Clarke Edmund M, Grumberg Orna, Peled Doron.* Model checking. MIT press, 1999.
10. *Emerson E Allen.* Temporal and modal logic. // Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B). 1990. Т. 995, No 1072. С. 5.
11. *Belta Calin, Habets Luc CGJM.* Constructing decidable hybrid systems with velocity bounds // Decision and Control, 2004. CDC. 43rd IEEE Conference on / IEEE. Т. 1. 2004. С. 467–472.
12. *Piterman Nir, Pnueli Amir, Sa'ar Yaniv.* Synthesis of reactive (1) designs // Verification, Model Checking, and Abstract Interpretation / Springer. 2006. С. 364–380.
13. *Livingston Scott C, Murray Richard M.* Just-in-time synthesis for reactive motion planning with temporal logic // Robotics and Automation (ICRA), 2013 IEEE International Conference on / IEEE. 2013. С. 5048–5053.