Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Факультет «Информационных технологий и программирования»

Кафедра «Компьютерных технологий»

С. В. Казаков

Разработка метода построения конечных автоматов верхнего уровня для управления моделью беспилотного самолета на основе обучающих примеров

Научные руководители: Ф. Н. Царев, А. А. Шалыто

Санкт-Петербург

2011 г.

Оглавление

ВВЕДЕНИЕ

Автоматное программирование [1] – парадигма программирования, заключающаяся в построении программ в виде совокупности автоматизированных объектов управления, каждый из которых содержит *систему управления* (взаимодействующие конечные автоматы, в дальнейшем – *управляющие автоматы*) и *объект управления*.

Объект управления характеризуется множеством *состояний*, которые задаются различными *входными параметрами*, и набором *управляющих органов*, позволяющих изменять состояние. Каждый орган управления характеризуется одним или несколькими *управляющими параметрами*,

Управляющий автомат определяется конечным множеством *состояний*, *функцией переходов* и *функцией выходов (действий, воздействий)*. При реализации автоматы могут взаимодействовать за счет вложенности.

Взаимодействие между системой управления и объектом управления выполняется следующим образом: каждый *такт работы* объект управления получает набор значений входных параметров, на основании которых он вычисляет значения функций предикатов, переходов и выходов, выполняет выработанные воздействия над объектом управления и, возможно, изменяет свое состояние с помощью значения функции переходов.

Для многих задач управляющие автоматы удается строить эвристически, но существуют задачи, для которых такое построение невозможно или затруднительно. К этому классу относятся: задача «Умный муравей» [2 – 4], задача «Умный муравей-3» [5], задача о муравьеде и муравьях [6], задача построения автопилота для вертолета [7], задача об управлении моделью беспилотного летательного аппарата [8] и другие.

Последние две задачи являются классическими задачами управления объектом со сложным поведением [1].

Рассмотрим поподробнее последнюю задачу. Существует несколько подходов к ее решению. Один из них состоит в выделении «идеальной» траектории из нескольких полетов, выполненных человеком, и последующее следование ей. Такой подход описан в работе [9].

Другой подход – использование автоматов для управления беспилотным летательным аппаратом и построение таких автоматов с помощью генетических алгоритмов, описанных в работах [10 – 14].

В работе [15] для генерации конечного автомата верхнего уровня, управляющего моделью беспилотного самолета, применялся алгоритм генетического программирования, основанный на использовании метода сокращенных таблиц для представления конечных автоматов. При этом вычисление функции приспособленности базировалось на моделировании поведения самолета во внешней среде, что занимало около пяти минут для одной особи на одном ядре современного компьютера. При этом весь процесс построения занимал несколько недель, что **являлось существенной проблемой этого подхода**.

Еще одним недостатком такого подхода является то, что при его применении для новой задачи необходимо полностью «с нуля» программно реализовывать вычисление функции приспособленности.

С целью устранения указанных недостатков в работе [16] было предложено использовать обучающие примеры как замену моделированию. Такой подход также был основан на алгоритме генетического программирования, однако, в отличие от предыдущей работы, состоял в *построении автоматов нижнего уровня для управления в одном режиме*. В этой работе необходимое время для всего процесса построения уже было 5 – 20

часов. Это исследование было развитием метода построения автоматов, основанного на обучающих примерах, изложенного в работе [17].

Для обеспечения всего процесса полета самолета вместе с построением автоматов нижних уровней необходимо также *построение автомата верхнего уровня, каждое состояние которого соответствует одному из режимов полета*. При этом имеет смысл требовать от такого процесса построения небольшого времени работы, меньшего или сравнимого со временем получения управляющих автоматов нижнего уровня.

Также имеет смысл использовать обучающие примеры, которые будут включать в себя весь процесс полета самолета. Тогда процессы построения системы управляющих автоматов можно будет объединить в один и строить систему автоматов, задавая только обучающие примеры.

Подход с использованием обучающих примеров также примечателен тем, что задавая большое их число, можно, как и в работе [9], избавиться от неточностей, допускаемых человеком при управлении при их записи.

Таким образом, *цель данной работы* – разработка метода построения автомата верхнего уровня для управления моделью беспилотного самолета с использованием обучающих примеров, а также построенных (заданных) автоматов нижнего уровня для каждого из режимов полета, используемых в обучающих примерах.

Предполагается, что в обучающем наборе могут быть примеры, которые задают разное поведение автомата верхнего уровня – имеют разную последовательность используемых режимов. В этом случае полученный автомат будет не просто линейной цепочкой автоматов нижних уровней, а будет иметь сложную логику.

Отметим, что используемый в данной работе объект со сложным поведением – беспилотный самолет в дальнейшем может быть заменен на

любой другой объект, который может быть управляем в нескольких режимах работы. Такое обобщение для управляющих автоматов нижнего уровня уже было сделано в работе [16].

ГЛАВА 1. АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ

В этой главе описываются общие концепции автоматного программирования. Также рассматривается понятие «система со сложным поведением».

1.1. Системы со сложным поведением

В процессе создания программного обеспечения часто возникает необходимость реализации сущностей со сложным поведением. Таким поведением обладают многие устройства управления, сетевые протоколы, сущности, отвечающие за взаимодействие других сущностей между собой и т. д.

Можно сказать, что *объект обладает сложным поведением*, если в ответ на возникновение некоторого события он может совершить одно из нескольких действий. Выбор действия зависит от информации, хранимой внутри и вне данного объекта на текущем шаге вычисления, а также на всех предыдущих шагах.

При традиционной программной реализации таких сущностей возникают избыточные переменные, называемые *флагами*, которым не соответствуют никакие элементы предметной области. Единственное предназначение флагов – участвовать в многочисленных конструкциях ветвления, реализующих логику поведения. Это решение трудно для понимания, подвержено ошибкам и практически не расширяемо.

Вместо этого в последнее время предлагается описывать объект со сложным поведением, приписывая ему некоторое множество управляющих состояний, в каждом из которых поведение объекта является простым и может быть описано непосредственно. Связь управляющих состояний с действиями и механизм переходов между состояниями удобно описывать с помощью

конечных автоматов с выходами [18]. При этом вся логика поведения объекта оказывается сосредоточенной в автоматах. Такой подход к описанию сложного поведения называют *автоматным* [19].

1.2. Автоматное программирование

Парадигма автоматного программирования [1] состоит в представлении систем со сложным поведением в виде автоматизированных объектов управления. *Автоматизированный объект управления* представляет собой объект управления, интегрированный с системой управления в одно «устройство». При этом система управления обычно представляется в виде системы взаимодействующих конечных автоматов.

В случае, когда автоматы описывают системы со сложным поведением, их проектирование является нетривиальной и трудоемкой задачей. Заниматься построением таких автоматов вручную неэффективно. Поэтому возникает естественное желание – автоматизировать процесс построения автоматов, поручив основную работу компьютеру.

ГЛАВА 2. ВЗАИМОДЕЙСТВИЕ СИСТЕМЫ УПРАВЛЯЮЩИХ АВТОМАТОВ С БЕСПИЛОТНЫМ САМОЛЕТОМ

В данной главе описываются общие принципы взаимодействия системы управляющих автоматов и беспилотного самолета. Понимание таких принципов позволяет лучше разобраться в процессе построения автомата верхнего уровня.

При взаимодействии система автоматов рассматривается как **управляющий объект**, без каких либо внутренних деталей. Таким управляющим объектом может оказаться и не система управляющих автоматов.

Беспилотный самолет рассматривается как **объект управления**, но, в отличие от системы автоматов, рассматриваются детали управления. Такой объект управления может быть заменен на любой другой объект со сложным поведением, для которого необходимо управление в нескольких режимах.

2.1. ПАРАМЕТРЫ САМОЛЕТА

Рассмотрим, как может осуществляться взаимодействие с самолетом.

Для анализа состояния самолета используются параметры полета самолета (скорость полета, направление полета, высота и т. д.). Такие параметры далее будем называть *входными параметрами*, так как они являются входными для системы управления.

Самолет характеризуется набором *органов управления*, посредством воздействия на которые им можно управлять. Параметры, соответствующие органам управления, будем называть *управляющими*. Примером органа управления является руль самолета, его параметр – угол поворота.

На орган управления можно воздействовать, изменяя параметр органа управления, что обеспечивает управление самолетом. Такие воздействия будем

называть *выходными воздействиями*, так как они являются результатом работы системы автоматов.

Существуют два типа выходных воздействий – *дискретные* и *непрерывные*.

Дискретные воздействия предназначены для *дискретных органов управления*. Это такие органы, параметр которых может принимать лишь конечное множество значений. Каждое из таких воздействий устанавливает конкретное значение параметра органа управления.

Непрерывные воздействия предназначены для *непрерывных органов управления*. Это органы, характеризующиеся вещественным значением параметра. Непрерывное воздействие изменяет значение параметра на заданную величину.

Например, одним из непрерывных управляющих параметров является угол поворота руля самолета. Непрерывным воздействием на этот орган управления является изменение угла его поворота на некоторое значение.

В свою очередь, примером дискретного органа является стартер. Дискретное воздействие на него – включение или выключение.

2.2. Структура обучающего примера

Исходными данными для построения системы управляющих автоматов является набор обучающих примеров, основная структура которых описана ниже. Обучающие примеры, задающие эталонное поведение, создаются человеком.

Если использовать достаточно большое число обучающих примеров, то дополнительно появляется возможность избавиться от мелких неточностей, которые допускает при управлении человек.

Схема, отражающая структуру обучающего примера, приведена на рис. 1.

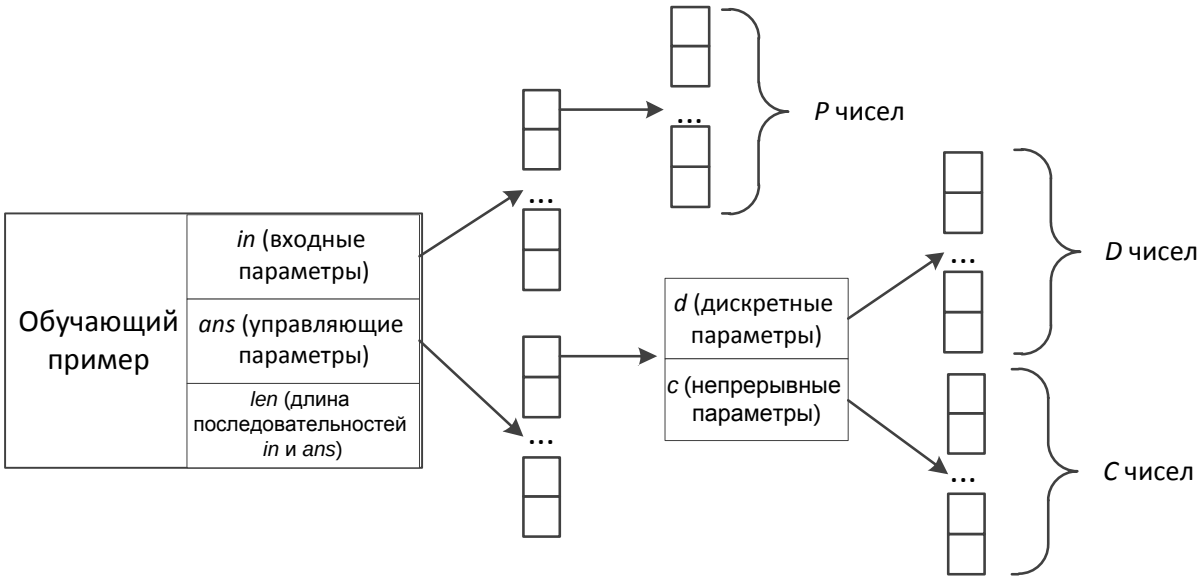


Рис. 1. Схема обучающего примера

Обучающий пример T состоит из двух частей: $T.in$ и $T.ans$. Каждая из них является последовательностью длины $T.len$ – первая из них состоит из значений входных параметров, а вторая – из соответствующих эталонных значений управляющих параметров, которые записываются в ходе экспериментов, проводимых человеком.

Каждый элемент $T.in[t]$ последовательности входных параметров состоит из P чисел – значений этих параметров в момент времени t.

Элемент $T.ans[t]$ включает в себя два набора: $T.ans[t].d$ и $T.ans[t].c$. Последовательность $T.ans[t].d$ состоит из D дискретных, а $T.ans[t].c$ – из C непрерывных управляющих параметров.

2.3. Предикаты о состоянии самолета

Ввиду того, что система автоматов сама по себе не может анализировать численные значения входных параметров, вводится дополнительный шаг их обработки. Благодаря этому появляется набор булевых переменных, используя которые автомат может понимать о происходящем и принимать решение о необходимом действии.

Предикат о состоянии самолета – это функция с множеством значений {*true, false*}, которая вычисляется в каждый такт работы, используя входные параметры самолета. Предикат вычисляется не только по входным текущим (последним пришедшим) параметрам самолета, но и с использованием всех предыдущих параметров.

Предикаты о состоянии самолета записываются в виде исходного кода на языке программирования, реализующего вычисление значения предиката по параметрам. Такой код записывается до работы системы автоматов.

Примером может быть предикат «двигатель включен». Формула такого предиката: *f* = текущее значение входного параметра «состояние двигателя».

Еще одним примером может быть предикат «самолет снижается» (*f* = текущее значение входного параметра «высота самолета» строго меньше предыдущего значения).

В данной работе рассматриваются два типа предикатов:

- предикаты без параметра (примеры таких параметров описаны выше);
- предикаты с параметром.

Предикаты с параметром – это такие предикаты, при инициализации которых можно выбрать одно значение, которое будет участвовать в формуле предиката.

Например, предикат «прошло заданное время» или «высота больше заданной» являются предикатами с параметром. В первом примере выбираемый параметр – время, во втором – высота.

Отметим также, что существуют предикаты, в которых используется некоторое константное значение. Они не являются предикатами с параметром –

выбранное значение нельзя изменить. Пример – предикат «высота самолета большая», который использует константу «большая высота».

Автомат любого уровня может использовать любые предикаты и в любом количестве из тех, которые были описаны до его работы. Используемые им предикаты в начале работы автомата инициализируются. При этом предикаты с параметром инициализируются параметром, который записан в файле автомата. После этого значение параметра предиката не возможно изменить. Уже после инициализации в процессе работы автомат может использовать значения этих предикатов.

2.4. Схема взаимодействия

Схема взаимодействия системы автоматов и беспилотного самолета показана на рис. 2.

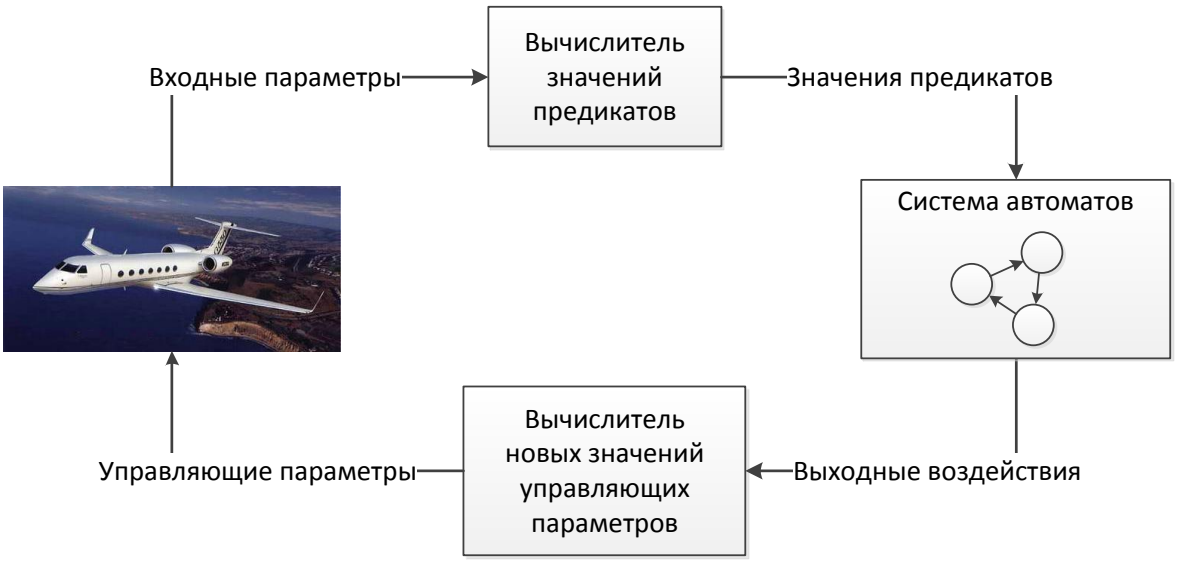


Рис. 2. Схема взаимодействия

Под *тактом работы* системы автоматов понимается один цикл взаимодействия: от получения входных параметров от самолета, до установления новых управляющих параметров.

Каждый такой такт повторяется с определенной частотой. Время между двумя тактами работы называется *величиной такта*. В данной работе величина такта выбрана 0.1 с.

Отметим, что схема взаимодействия с обучающим примером является такой же, как и в случае с самолетом. Отличия состоят только в том, что входные параметры выбираются из обучающего примера, а новые управляющие параметры запоминаются, никак не влияя на следующие входные параметры.

ГЛАВА 3. ПОСТРОЕНИЕ АВТОМАТА ВЕРХНЕГО УРОВНЯ

Для обеспечения всего процесса полета самолета вместе с построением автоматов нижних уровней необходимо также построение автомата верхнего уровня, каждое состояние которого соответствует одному из режимов полета. При этом время построения автомата верхнего уровня должно быть меньше или сравнимо со временем получения управляющих автоматов нижнего уровня.

Предложенный подход основан на обучающих примерах, которые теперь содержат весь процесс полета самолета.

Разработанный метод построения управляющего автомата верхнего уровня использует обучающие примеры, а также построенные (заданные) автоматы нижнего уровня для каждого используемого режима полета.

Предполагается, что в обучающем наборе могут быть примеры, которые задают разное поведение автомата верхнего уровня – имеют разную последовательность используемых режимов. В таком случае полученный автомат будет не просто линейной цепочкой автоматов нижних уровней, а будет содержать сложную логику.

3.1. Обучающий пример для рассматриваемой задачи

Обучающий пример для построения автомата верхнего уровня немного отличается от ранее описанного. Для данной задачи он включает в себя:

- входные параметры для каждого такта работы;
- разделение всего примера на области, соответствующие разным режимам полета.

Заметим, что в отличие от обычного обучающего примера он не содержит управляющие параметры из-за их ненадобности.

Разбиение всего примера на области задается следующим образом. Для каждого режима задается левая граница – самое ранее возможное начало режима, и правая граница – самый поздний возможный конец. Схематично такое разбиение показано на рис. 3.

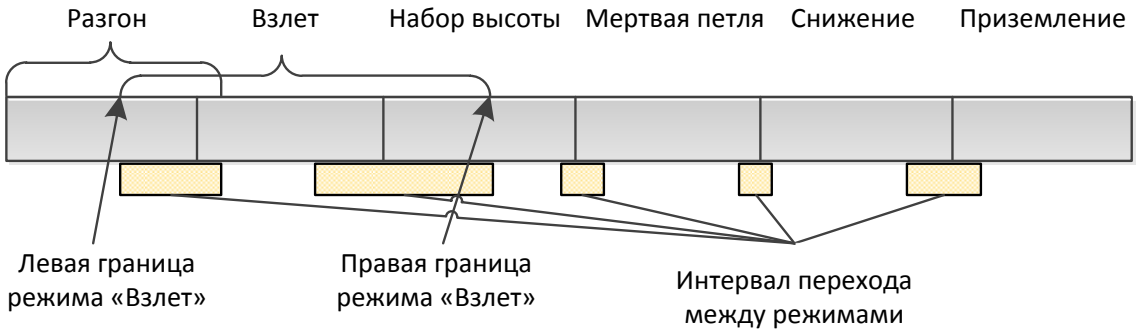


Рис. 3. Разбиение обучающего примера на области

Между правой границей одного режима и левой границей следующего режима появляется область обучающего примера, на которой должен быть выполнен переход в построенном автомате верхнего уровня между заданными режимами. Такая область далее будет называться *интервалом перехода между режимами*. Благодаря ей можно построить условия переходов автомата верхнего уровня.

3.2. Автомат верхнего уровня

Напомним, что автомат верхнего уровня необходим для переключения между режимами полета самолета и что каждое состояние автомата соответствует ровно одному такому режиму. Именно поэтому число состояний автомата должно быть не меньше, чем число используемых режимов, но оно может быть и больше. При этом появляются состояния с одинаковым режимом, но имеющие разный смысл.

В данной работе считается, что автомат верхнего уровня имеет столько состояний, сколько используемых режимов, и при этом нет двух состояний с

одинаковым режимом. Следовательно, для каждого режима существует только одно состояние с таким режимом.

Автомат верхнего уровня представляется в виде обычного конечного автомата с ребрами, которые возможны между любыми состояниями.

Ввиду того, что такой автомат необходим для переключения между режимами полета самолета, а не для непосредственного управления (эту задачу решают автоматы нижнего уровня), ребра в нем содержат только условия перехода, и **не содержат действий**.

Рассмотрим поподробнее вид записи условий перехода.

Во-первых, напомним, что условие на ребре – это булева формула, которая содержит предикаты о состоянии самолета в качестве *атомов*. Под *атомом* понимается наименьшая часть условия, которая считается единым целым.

Из-за того, что условие перехода может быть составным и использовать в своем не формульном описании несколько утверждений, был выбран следующий вид записи условий: «[!]x_1 & [!]x_2 & … & [!]x_k». Здесь x_i – предикаты о состоянии самолета, k – число используемых предикатов. Квадратные скобки означают, что выражение в них может быть, а может и не быть.

Более сложное условие не было выбрано из-за ненадобности такового, и из-за трудностей его автоматического построения.

3.3. Метод построения автомата верхнего уровня

Задача построения автомата верхнего уровня в данной работе разбивается на три подзадачи:

- определение используемых режимов в каждом обучающем примере;

- сопоставление областей обучающего примера и состояний автомата;
- определение ребер и условий переходов по ним.

В результате решения первой задачи для каждого обучающего примера будут найдены режимы работы самолета, и порядок их использования. Применяя эти результаты можно сопоставить каждой области обучающего примера состояние автомата верхнего уровня.

Уже после этого можно определить, какие переходы обязательно должны быть между состояниями для удовлетворения обучающему примеру. Для таких переходов можно выделить из обучающего примера свой обучающий набор, благодаря которому можно построить условие перехода.

3.3.1. Определение используемых режимов в обучающем примере

Задачу определения используемого режима для некоторой области обучающего примера можно решить с помощью выбора автомата нижнего уровня, который лучше всего подходит для управления на этой области. Таким образом, каждой области ставится в соответствие автомат нижнего уровня.

В автомате верхнего уровня каждому состоянию соответствует один режим полета самолета, который тоже задается автоматом нижнего уровня.

Для выбора самого подходящего автомата нижнего уровня сначала для каждого автомата вычисляется значение *функции расстояния*, о которой написано ниже. После этого процесс выбора сводится к задаче определения автомата с наименьшим значением этой функции.

Заметим, что автомат нижнего уровня должен уметь управлять только в одном режиме, но он может хорошо управлять и в нескольких. Поэтому возможен случай, когда будет существовать несколько автоматов нижнего уровня, одинаково хорошо управляющих на области обучающего примера. В данной работе предполагается, что каждой области будет сопоставлен только

один автомат. Поэтому задача выбора лучшего, как уже говорилось ранее, сводится к задаче определения автомата с наименьшим значением функции расстояния, без учета случая с несколько возможными лучшими автоматами. Можно изменить такой подход на выбор с рассмотрением описанного случая, но при экспериментальной проверке такого случая не наблюдалось, и автор не стал этого делать.

Под функцией расстояния понимается функция, которая по автомату и некоторой области обучающего примера выдает значение, которое говорит о «непохожести» поведения автомата и поведения, записанного в области обучающего примера.

В результате исследования было опробовано два способа вычисления функции расстояния:

- вычисление с помощью запуска каждого автомата на области обучающего примера и сравнение сгенерированных действий автомата с действиями в обучающем примере;
- определение подходящего автомата с помощью сравнения заранее заданного его поведения с поведением, описанным в области обучающего примера.

Рассмотрим эти способы поподробнее и укажем их достоинства и недостатки.

3.3.1.1. Вычисление функции расстояния, основанное на сравнении действий

Автомату по порядку подаются входные параметры самолета, которые записаны в области обучающего примера. После подачи каждого набора входных параметров автомат генерирует воздействия на органы управления. Эти воздействия применяются к предыдущим управляющим параметрам, которые выбираются из обучающего примера. Новые управляющие параметры сравниваются с управляющими параметрами, записанными в обучающем

примере. В результате сравнения управляющих параметров на всей области вычисляется значение функции расстояния.

Такой подход прост и требует задания только автомата нижнего уровня, без задания дополнительной информации, но в данном случае существует одна проблема, из-за которой автор отказался от его использования.

Если автомат нижнего уровня при управлении требует немедленного ответа от самолета на свое управление, и, анализируя следующие входные параметры, автомат «понимает» как надо управлять, то при запуске на обучающем примере ответа именно на его действия не последует. Потому такой автомат будет пытаться сделать какой-то шаг с целью получения ответа всё время в процессе его запуска на одной области обучающего примера. Такое поведение сильно мешает понять, какой из автоматов лучше подходит для запуска на данной области. В большинстве случаев лидером оказывался наиболее «пассивный» автомат.

Ввиду того, что при построении автомата верхнего уровня использовались несколько автоматов нижнего уровня, построенных вручную, и требующих немедленного ответа, то данный подход был отвергнут.

3.3.1.2. Вычисление функции расстояния, основанное на сравнении поведения

Второй способ решает проблему, которая была описана выше. В этом случае вместе с автоматом нижнего уровня задается файл поведения автомата. В этом файле содержится последовательность входных параметров самолета. Считается, что такая последовательность задает поведение автомата, а вернее поведение самолета под управлением данного автомата.

Также предполагается, что поведение записано, начиная с тех начальных условий, которые и должны быть для нормального управления автомата нижнего уровня. Например, для автомата, выполняющего мертвую петлю,

самолет в начале выполнения должен находиться в воздухе и нормально лететь, а не быть на земле с выключенным двигателем.

Как уже говорилось, задача вычисления функции расстояния при таком подходе решается сравнением поведения автомата с поведением, описанном в обучающем примере.

Для такого сравнения применяется вычисление **функции редакционного расстояния.**

Ввиду того, что элементом последовательностей является набор входных параметров самолета, для их сравнения вводится своя функция расстояния. Эта функция для каждого входного параметра из набора вычисляет некоторую величину «непохожести» двух параметров. После этого суммирует все такие значения.

При этом для некоторых параметров такая величина «непохожести» равна модулю разности абсолютных значений, нормированная на некоторую константу. Так делается, например, для входного параметра «двигатель включен», или для параметра «крен (угол наклона относительно главной оси)».

Для других же параметров сравниваются относительные значения – изменение относительно начальных параметров. Под начальными параметрами для последовательности понимаются параметры самолета, с которых начинается эта последовательность. Так делается, например, для параметра «направление», так как абсолютное значение направления в большинстве случаев не важно.

Для того чтобы функция редакционного расстояния не пыталась найти сходство для слишком далеко расположенных друг от друга фрагментов, вводится ограничение на максимальное смещение одного фрагмента относительно другого. При экспериментальной проверке максимальное смещение было 10% от длины фрагментов (фрагменты одинаковой длины), что

уменьшало время вычисления более чем в пять раз. Однако число необходимых операции для ее вычисления все же оставалось квадратным относительно длины.

3.3.2. Сопоставление областей обучающего примера и состояний автомата

После определения используемого режима в каждой области обучающего примера такой области необходимо сопоставить состояние автомата верхнего уровня. Ввиду того, что в автомате существует только одно состояние для каждого режима полета, задача сопоставления решается простым поиском требуемого состояния.

Отметим, что возможно сопоставление, при котором первая область будет отображаться не на первое (стартовое) состояние. Тогда появляется задача выбора начального режима при запуске автомата верхнего уровня, для удовлетворения обучающему примеру. Однако такая проблема возникает крайне редко, и в экспериментальном исследовании она не наблюдалась. Поэтому автор работы не стал ее решать.

3.3.3. Определение ребер и условий перехода по ним

Вначале определим, какие ребра будут в автомате, судя по обучающим примерам.

Для каждого из таких примеров уже имеется последовательность состояний автомата, по которой должен «пройти» автомат, если ему на вход подавать входные параметры из обучающего примера. Поэтому, для каждых двух соседних состояний из этой последовательности должно быть ребро между ними.

3.3.3.1. Создание обучающего набора для условия перехода

Рассмотрим ребро между состояниями автомата, соответствующими режимам 1 и 2, а также обучающий пример, из-за которого это ребро появилось. Часть такого обучающего примера изображена на рис. 4.

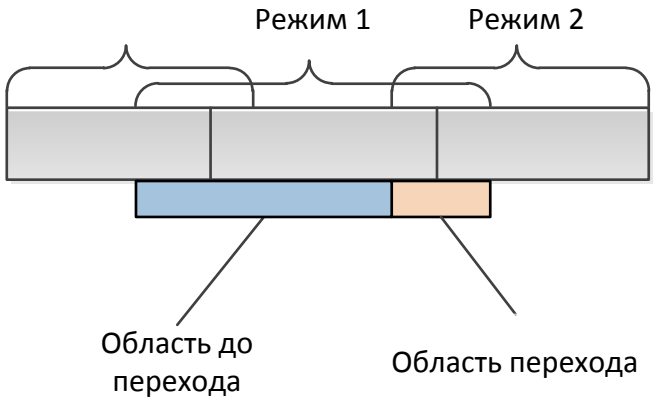


Рис. 4. Область до перехода и область перехода между режимами 1 и 2

Введем понятие *области до перехода* как области обучающего примера, начиная от левой границы режима кончая левой границей следующего режима (область до перехода определена для не последнего режима).

Отметим, что, для того, чтобы переход по ребру осуществился в *области перехода* (интервале перехода между режимами) необходимо, чтобы условие перехода было истинно хотя бы для одних параметров из области перехода, и ложно для всех параметров из области до перехода.

Таким образом, для каждого условия перехода строится свой обучающий набор, используя все обучающие примеры. Такой обучающий набор состоит из областей до перехода и областей перехода.

Заметим, что если ребро является единственным из состояния, то требования к условию перехода, описанные выше, являются окончательными и не требуют дополнений. Если же ребер из состояния несколько, то также необходимо потребовать от условия перехода невыполнимость на всех обучающих наборах других ребер из этого состояния. При этом должна быть обеспечена невыполнимость как на областях до перехода, так и на областях перехода из обучающего набора других ребер. Это гарантирует то, что автомат не перейдет по ребру, по которому не должен быть совершен переход.

Однако такое дополнительное требование можно не ставить одному ребру из состояния. Тогда это ребро будет считаться ребром «по умолчанию» – по нему будет совершен переход в случае невозможности перехода по другим ребрам.

3.3.3.2. Поиск условия перехода

Ввиду того, что задачу поиска можно разбить на две независимых подзадачи, то такой поиск для некоторого ребра осуществляется в два этапа:

- поиск условия, различающего параметры при переходе по данному ребру от параметров при переходе по другим ребрам из того же состояния;
- поиск дополнительного условия, которое будет удовлетворять требованию невыполнимости на областях до перехода обучающего набора самого ребра.

Первая задача состоит в том, чтобы найти условие перехода для некоторого ребра, которое не будет выполняться на всех входных параметрах обучающих наборов других ребер (областей до перехода и областей перехода), но будет выполняться хотя бы на одних параметрах для каждой области перехода из обучающего набора данного ребра.

Такое условие, как уже упоминалось выше, играет существенную роль – оно не позволяет перейти по другим ребрам из данного состояния.

Для решения первой задачи используется алгоритм перебора с проверкой требований. Перебираются условия вида «[!]x_1 & [!]x_2 & … & [!]x_k», которые разрешены на ребрах автомата верхнего уровня. Перебор выполняется в порядке увеличения числа используемых предикатов до некоторого предельного числа. В экспериментальных исследованиях такое предельное число равнялось двум предикатам. При этом используются **предикаты без параметра и предикаты с заданным начальным параметром без выбора**

другого параметра. Последнее условие перебора сделано для того, чтобы выявить гарантированное отличие переходов, а не, возможно, приближенное.

После нахождения такого условия решается вторая подзадача удовлетворения уже только самому обучающему набору рассматриваемого ребра: выполнимости хотя бы для одних входных параметров из области перехода и невыполнимость для всех параметров из области до перехода. Для этого поочередно производятся шаги, в случае успешности одного из которых процесс поиска прекращается:

- проверяется, не является ли уже построенное условие достаточным для выполнения требований;
- производится поиск дополнительного условия, используя предикаты без параметра и предикаты с заданным начальным параметром без выбора другого параметра;
- производится поиск дополнительного условия, используя предикаты с параметром, с выбором параметра с помощью обучающего набора.

На шаге 2 поиск производиться так же, как и в первой подзадаче. В экспериментальной проверке предельное число дополнительных предикатов также равно двум.

Теперь рассмотрим поподробнее шаг 3.

Пусть, например, имеется ребро между состояниями «набор высоты» и «сбалансированный полет», и необходимо выбрать дополнительное условие для перехода по ребру. Рассмотрим, например, предикат «высота больше заданной», и поставим задачу выбрать параметр предиката для удовлетворения обучающему набору. Также рассмотрим область из обучающего примера, состоящую из области до перехода и области перехода.

Заметим, что для того, чтобы было целесообразно выбирать данный предикат и какое-нибудь значение параметра, необходимо, чтобы значение высоты изменялось в одну и ту же сторону на протяжении этой области. Тогда значение параметра имеет смысл взять из области перехода (например, из начала области), как показано на рис. 5.

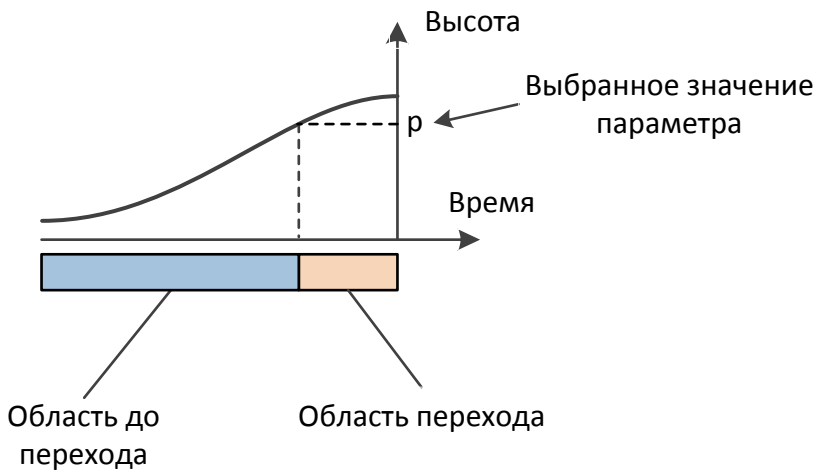


Рис. 5. Выбор значения параметра для одной области

Обобщим на случай многих областей из обучающего набора и любого предиката с параметром.

Для выбора предиката с параметром и некоторого значения параметра необходимо, чтобы значение входного параметра, с которым сравнивается параметр предиката, изменялось в одну и ту же сторону для всех областей, состоящих из области до перехода и области перехода, из обучающего набора ребра.

Под словом «изменялось» подразумевается некоторое глобальное изменение, даже несмотря на возможные локальные изменения в другую сторону. В разработанной программе это условие проверяется следующим образом. Выбирается разность между значением параметра из последних параметров в области и значением параметра в середине области. Эта разность, нормированная на длину половины области, сравнивается с пороговым значением.

При нескольких областях выбирается значение параметра, равное среднему арифметическому значений «рекомендуемого» параметра для каждой области из обучающего набора ребра. При таком выборе происходит усреднение неточностей управления в обучающих примерах, и, считается, что такой выбор будет оптимальным.

Отметим также, что при выборе предиката «прошло заданное время», дополнительное условие точно будет найдено, выбрав этот предикат. Однако имеет смысл сначала попробовать использовать более содержательные предикаты с параметром. Ввиду этого, предикаты с параметром проверяются на возможность выбора подходящего параметра в том порядке, в котором они заданы в конфигурационном файле, что делает поиск дополнительного условия более осмысленным процессом.

ГЛАВА 4. ПРИМЕНЕНИЕ МЕТОДА ДЛЯ ПОСТРОЕНИЯ СИСТЕМЫ АВТОМАТОВ ДЛЯ УПРАВЛЕНИЯ МОДЕЛЬЮ БЕСПИЛОТНОГО САМОЛЕТА

В этой главе описан процесс экспериментальной проверки, а также приведены основные полученные результаты.

Для проверки эффективности и работоспособности предложенного метода была выбрана задача построения системы автоматов управляющей беспилотным самолетом, который должен взлететь, набрать высоту, далее выполнять команды, которые приходят с Земли, и в конечном итоге приземлиться. При этом возможен приход трех команд:

- выполнить мертвую петлю;
- начать снижаться для того, что бы приземлиться;
- начать набирать высоту до необходимой для выполнения трюка.

При этом последние две команды могут отменять друг друга, но команду выполнения мертвой петли отменить нельзя, пока она не будет выполнена.

4.1. Используемый авиационный симулятор

Существуют различные авиационные симуляторы, моделирующие полет самолета. Автором был выбран свободный кроссплатформенный симулятор *FlightGear* (http://www.flightgear.org), который применительно к настоящей работе позволяет осуществлять как ручное, так и автоматное управление моделью самолета.

Этот симулятор позволяет также осуществлять сохранение параметров полета (скорость, направление полета и т. д.) и параметров самолета (положение руля, элеронов, состояние стартера и т. п.). Параметры полета являются входными параметрами для системы управления, а параметры самолета – управляющими, так как за счет их изменений выполняется управление самолетом.

На рис. 6 представлен снимок экрана симулятора *FlightGear* в момент начала разгона.

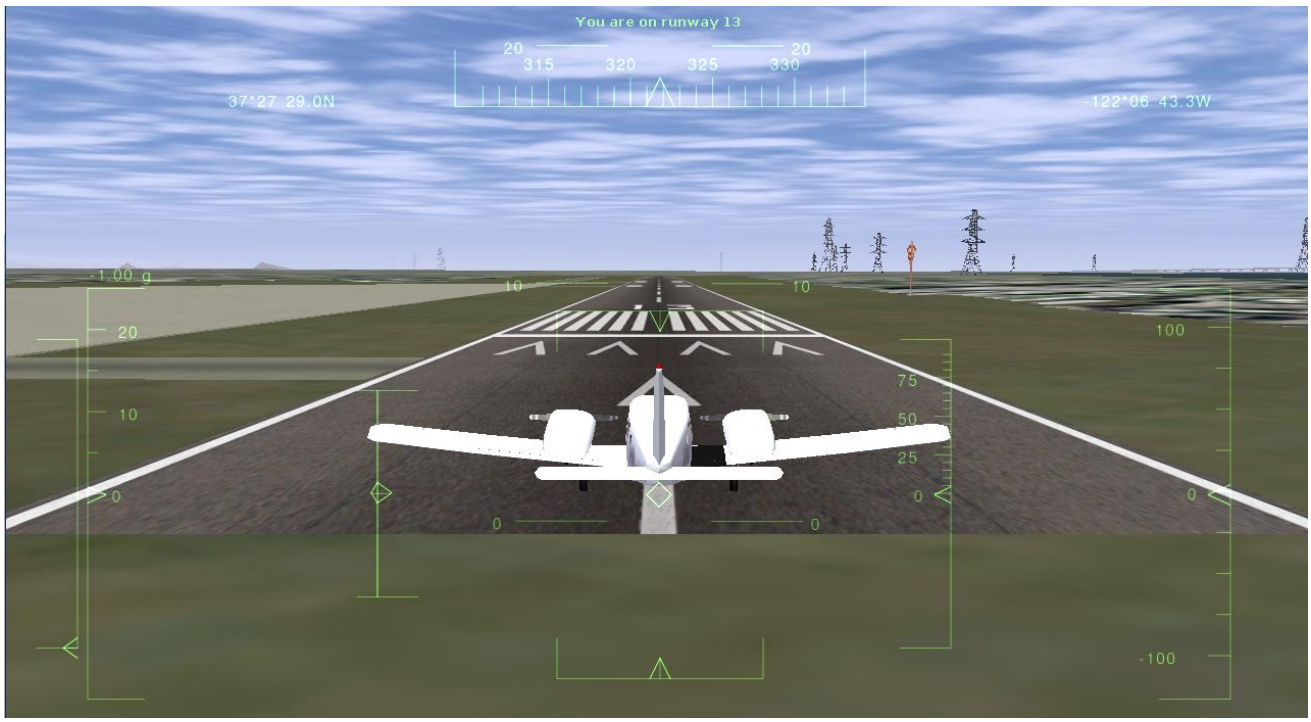


Рис. 6. Снимок экрана симулятора *FlightGear* в момент начала разгона

4.2. Построение автоматов нижнего уровня

Для выполнения поставленной задачи было необходимо осуществлять управление в следующих режимах:

- запуск двигателя;
- разгон;
- набор высоты (взлет);
- сбалансированный полет;
- мертвая петля;
- снижение;
- приземление;
- торможение.

Автомат нижнего уровня для выполнения мертвой петли был получен с помощью подхода построения автоматов нижнего уровня, основанного на

генетическом алгоритме с использованием обучающих примеров, описанного в работе [16].

Автоматы нижнего уровня для всех остальных режимов были построены вручную.

4.3. Построение автомата верхнего уровня

Было записано 13 обучающих примеров, последовательность режимов в которых была одной из следующих:

- запуск двигателя, разгон, набор высоты, сбалансированный полет, мертвая петля, сбалансированный полет, снижение, приземление, торможение;
- запуск двигателя, разгон, набор высоты, снижение, набор высоты, сбалансированный полет, снижение, приземление, торможение;
- запуск двигателя, разгон, набор высоты, сбалансированный полет, снижение, приземление, торможение;
- запуск двигателя, разгон, набор высоты, снижение, приземление, торможение.

Следует отметить, что автомат нижнего уровня для режима «мертвая петля» был построен с помощью обучающих примеров, которые получились вырезанием необходимого режима из обучающих примеров для всего полета.

После этого был апробирован алгоритм построения автомата верхнего уровня.

Сначала был сформирован достаточный набор предикатов, которые программное средство могло использовать для построения условий перехода. Этот набор включал в себя:

- Предикат для работы с двигателем:
    - Двигатель работает длительное время. Длительное время – 0.5 секунды (пять тактов).
- Предикат скорости:

- Скорость относительно земли больше заданной скорости.

- Предикаты высоты:
  - Высота меньше нулевого порога. Нулевой порог задается в файле конфигурации.
  - Высота больше заданной высоты.
- Предикат времени:
  - Прошло заданное время после переключения в данный режим.
- Предикат для «мертвой петли»:
  - Петля была выполнена.
- Предикаты о командах с Земли:
  - Пришла команда выполнить мертвую петлю.
  - Пришла команда начать снижение.
  - Пришла команда начать набирать высоту.

После определения предикатов был запушен алгоритм построения автомата верхнего уровня.

4.3.1. Результаты определения режимов в обучающих примерах

Приведем результаты определения режимов для одного обучающего примера, в котором был выполнена мертвая петля, и для одного, в котором петли не было.

Параметры алгоритма были такими:

- Максимально возможное смещение одного фрагмента относительно другого – 10 % от длины (у сравниваемых последовательностей одинаковая длина).
- «Цена» вставки и удаления символа – 0.2. Именно поэтому значение функции расстояния не может быть больше 0.4 – значения, которое получается при полном поэлементном удалении одной последовательности и вставки другой.

В обучающем примере с мертвой петлей было обнаружено девять режимов.

Значения функции расстояния, основанной на сравнении поведения для каждой области обучающего примера и каждого режима, приведены в табл. 1.

Таблица 1. Определение режимов в обучающем примере с мертвой петлей

	Запуск двигателя	Разгон	Набор высоты	Сбалансированный полет	Мертвая петля	Снижение	Приземление	Торможение
Обл. 1	0.00	0.11	0.11	0.40	0.37	0.33	0.40	0.12
Обл. 2	+∞	0.01	0.06	0.40	0.39	0.37	0.40	0.34
Обл. 3	+∞	+∞	0.12	0.36	+∞	0.39	0.40	+∞
Обл. 4	+∞	+∞	0.37	0.08	+∞	0.35	0.40	+∞
Обл. 5	+∞	0.39	0.39	0.37	0.19	0.37	0.39	0.40
Обл. 6	+∞	+∞	0.40	0.16	+∞	0.37	0.40	0.40
Обл. 7	+∞	+∞	0.40	0.39	+∞	0.20	0.36	+∞
Обл. 8	+∞	0.39	0.39	0.40	0.39	0.32	0.16	0.39
Обл. 9	+∞	0.35	0.34	0.40	0.40	0.39	0.40	0.10

В обучающем примере без мертвой петли было обнаружено семь режимов.

Значения функции расстояния для каждой области обучающего примера и каждого режима показаны в табл. 2.

Значение функции равно +∞, если длина последовательности, задающая поведение какого-либо автомата нижнего уровня, намного меньше, чем длина области обучающего примера.

Таблица 2. Определение режимов в обучающем примере без мертвой петли

	Запуск двигателя	Разгон	Набор высоты	Сбалансированный полет	Мертвая петля	Снижение	Приземление	Торможение
Обл. 1	0.00	0.11	0.11	0.40	0.37	0.34	0.40	0.12
Обл. 2	+∞	0.00	0.05	0.40	0.39	0.37	0.40	0.33
Обл. 3	+∞	+∞	0.05	0.38	+∞	+∞	0.40	+∞
Обл. 4	+∞	+∞	0.36	0.08	+∞	0.33	0.40	+∞
Обл. 5	+∞	+∞	0.39	0.38	+∞	0.09	0.35	+∞
Обл. 6	+∞	0.40	0.40	0.40	0.39	0.23	0.12	0.39
Обл. 7	+∞	0.36	0.34	0.39	0.40	0.35	0.39	0.12

В результате выбора наиболее подходящего режима для каждой области получаются сопоставления, показанные в табл. 3 и 4.

Таблица 3. Сопоставление режимов и областей обучающего примера с мертвой петлей

Область в обучающем примере	Выбранный режим	Значение функции расстояния
1	Запуск двигателя	0.00
2	Разгон	0.01
3	Набор высоты	0.12
4	Сбалансированный полет	0.08
5	Мертвая петля	0.19
6	Сбалансированный полет	0.16
7	Снижение	0.20
8	Приземление	0.16
9	Торможение	0.10

Таблица 4. Сопоставление режимов и областей обучающего примера без мертвой петли

Область в обучающем примере	Выбранный режим	Значение функции расстояния
1	Запуск двигателя	0.00
2	Разгон	0.00
3	Набор высоты	0.05
4	Сбалансированный полет	0.08
5	Снижение	0.09
6	Приземление	0.12
7	Торможение	0.12

4.3.2. Построенный автомат

В результате был построен автомат верхнего уровня, который приведен на рис. 7.

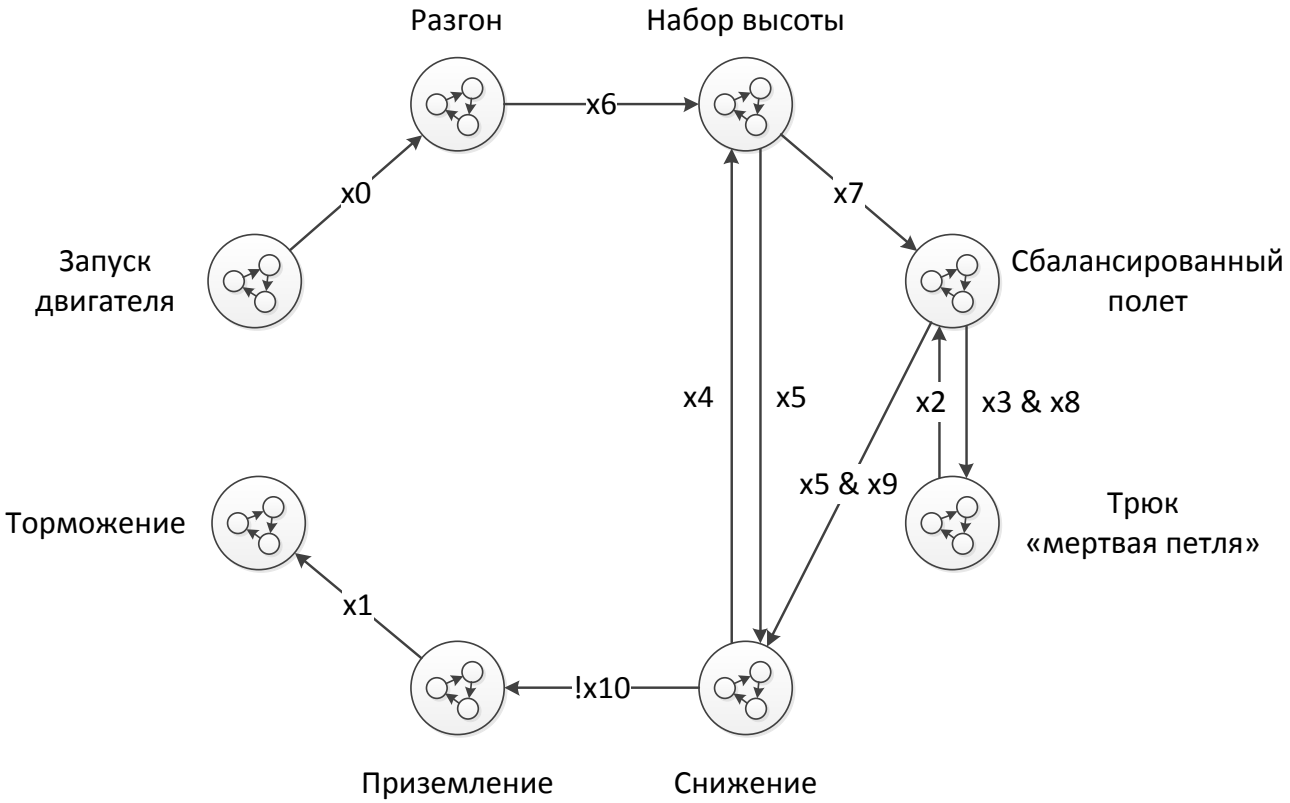


Рис. 7. Построенный автомат верхнего уровня

В данном автомате используются следующие предикаты:

- x0 – двигатель работает длительное время;
- x1 – высота меньше нулевого порога;
- x2 – мертвая петля была выполнена;
- x3 – пришла команда на выполнение мертвой петли;
- x4 – пришла команда начать набирать высоту;
- x5 – пришла команда начать снижаться;
- x6 – скорость относительно земли больше 59 км/ч;
- x7 – высота больше 1381 футов ≈ 400 метров;
- x8 – прошло 50 с после переключения в данный режим;
- x9 – прошло 28 с после переключения в данный режим;
- x10 – высота больше 196 футов ≈ 60 метров.

Отметим, что предикаты x0 – x5 являются предикатами без параметра. Остальные имеют параметр, значения которого для каждого предиката были выбраны разработанным программным средством, используя обучающие примеры.

4.3.3. Оценка эффективности процесса построения

Вычисления производились на одном ядре компьютера с процессором *Intel Core 2 Duo T7250* с тактовой частотой 2 ГГц под управлением операционной системы *Microsoft Windows 7*.

Программное средство полностью написано на языке программирования *Java*.

Время построения автомата верхнего уровня с использованием 13 обучающих примеров составляло менее двух минут. Это намного быстрее, чем описанное в работе [16] построение автомата нижнего уровня с использованием обучающих примеров для одного режима, которое занимало 5 – 20 часов. Заметим также, что в построении автомата нижнего уровня применялся

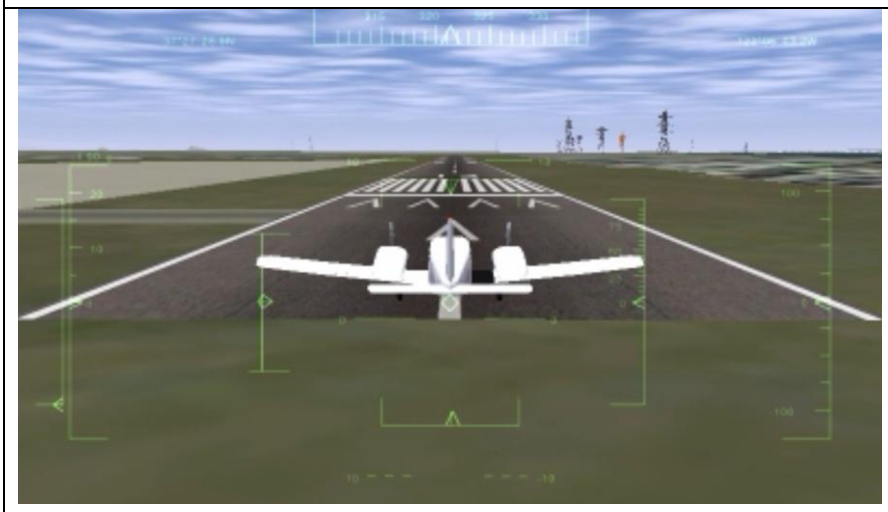
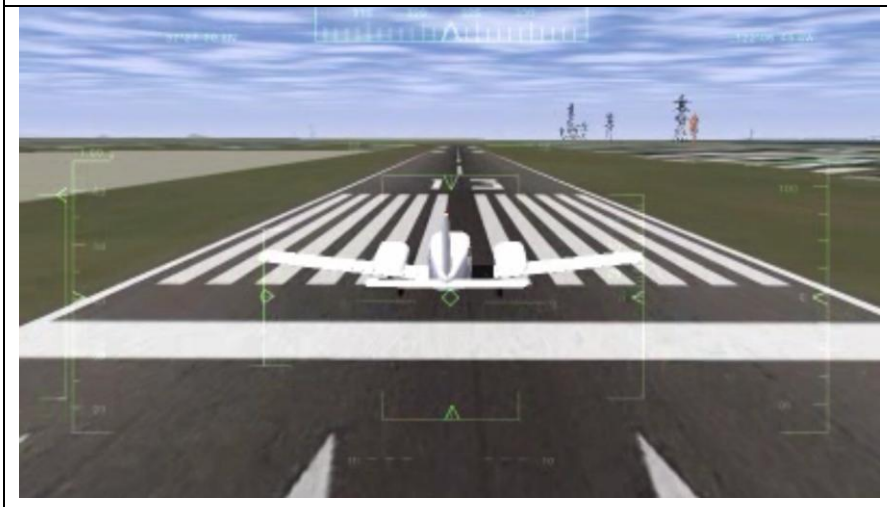
алгоритм генетического программирования, который не используется в настоящей работе и требовал таких больших временных затрат.

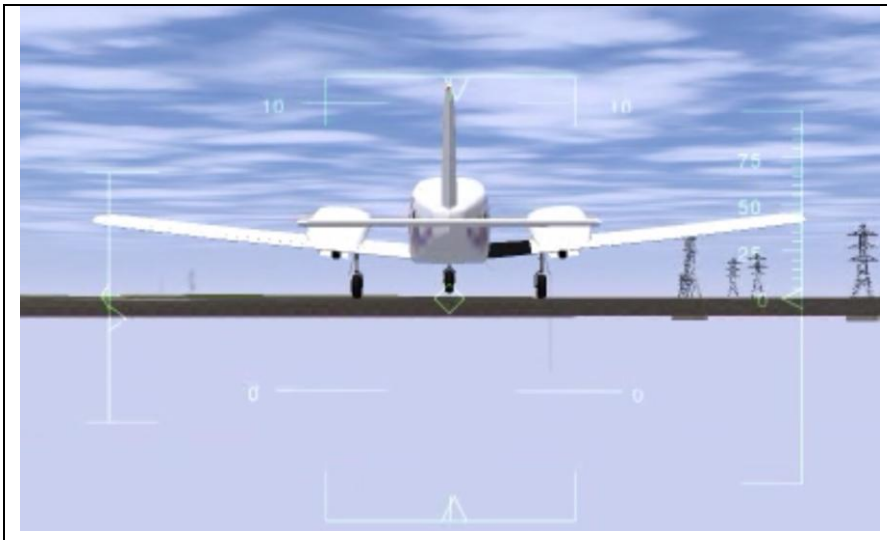
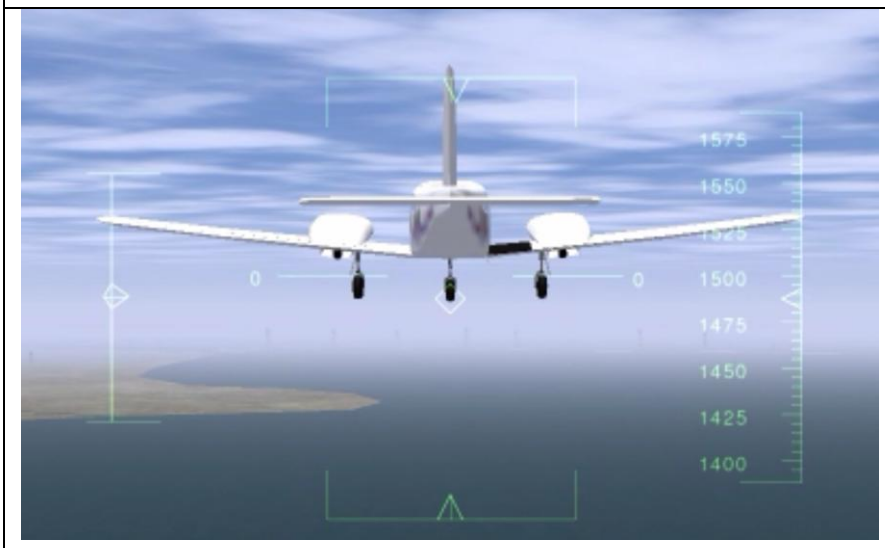
4.3.4. Полет самолета под управлением построенного автомата

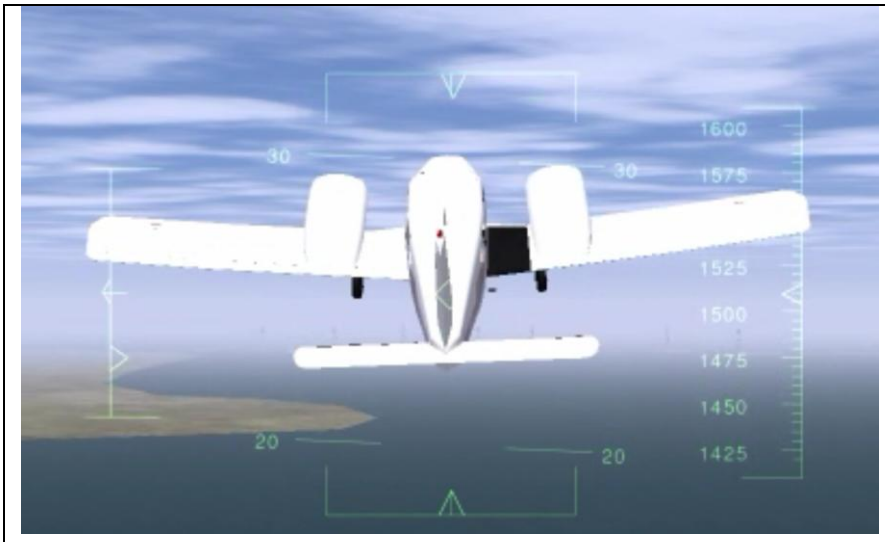
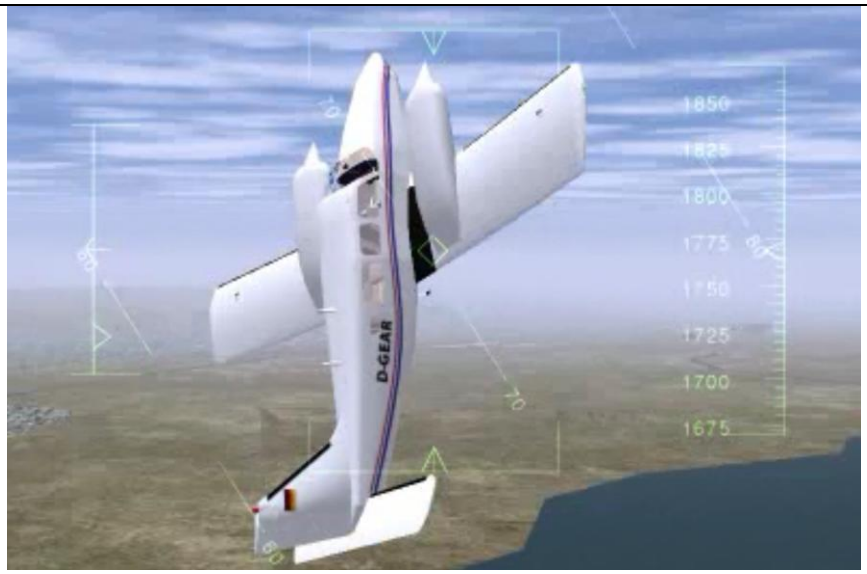
Видеозапись одного из полетов самолета под управлением построенного автомата доступна по адресу http://www.youtube.com/watch?v=dq5AVzqXug0.

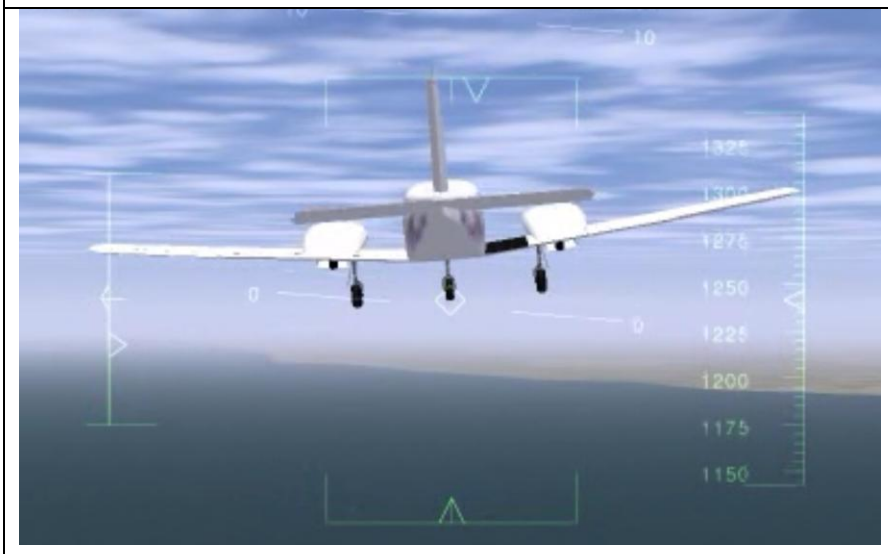
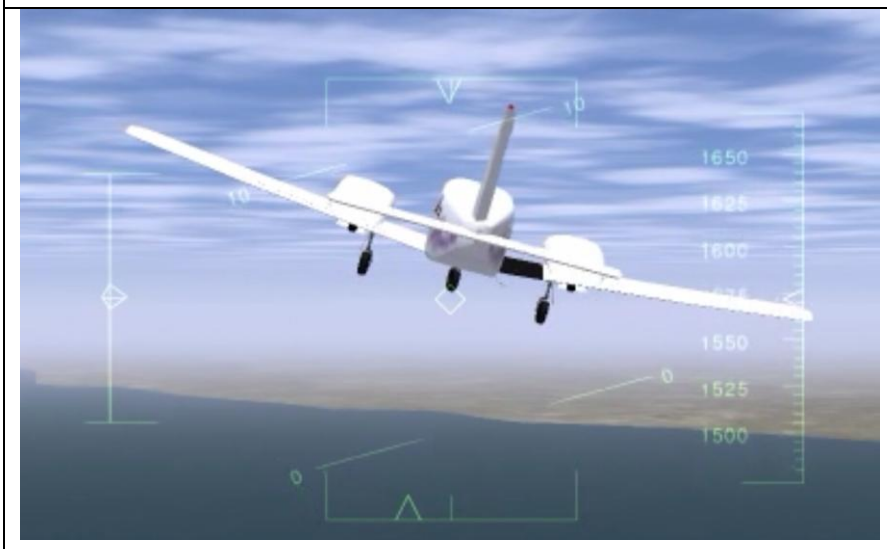
В табл. 5 приведены 15 кадров этой видеозаписи с комментариями.

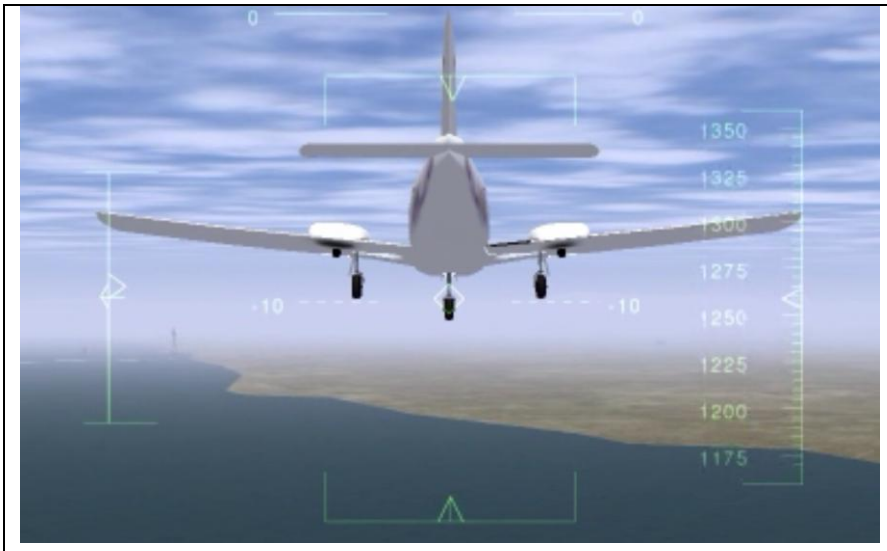
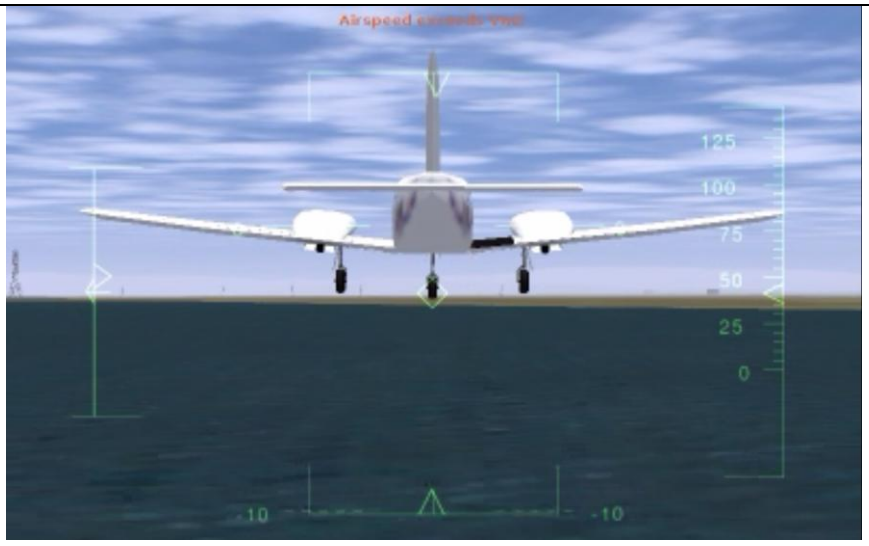
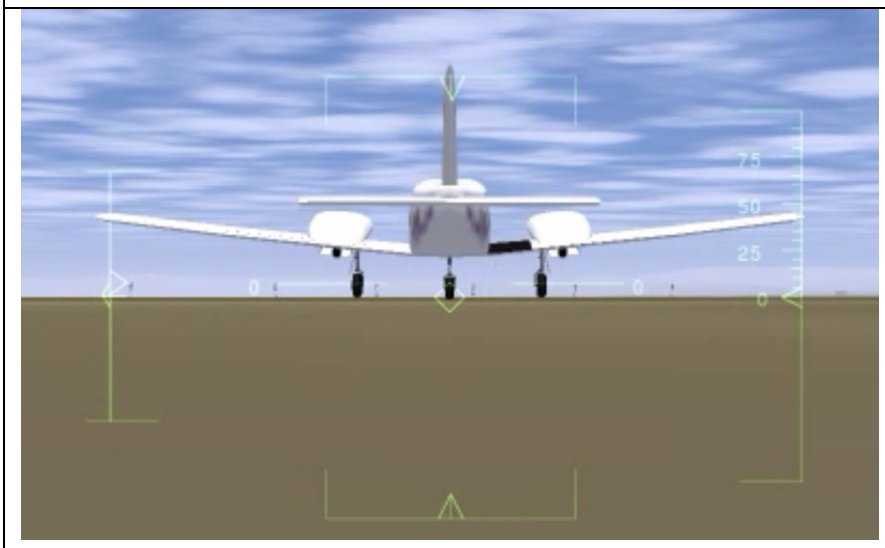
Таблица 5. Кадры полета самолета под управлением построенного автомата

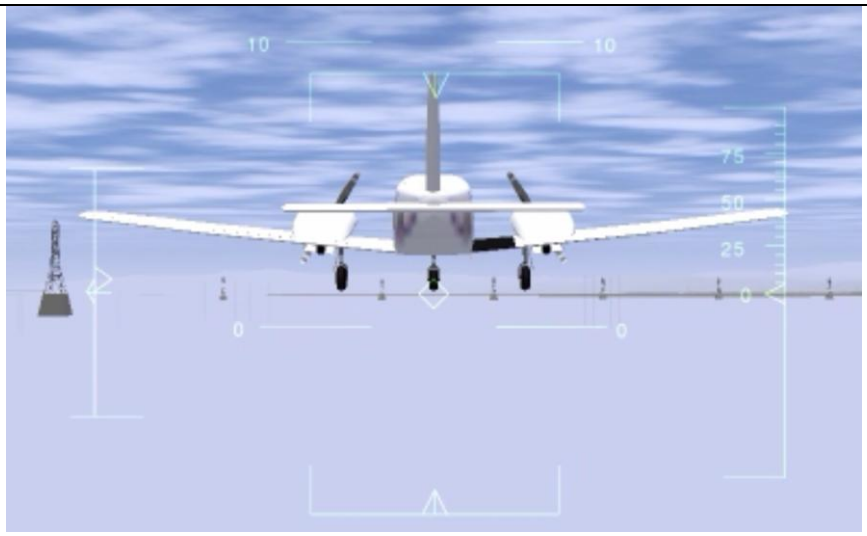
Кадр видеозаписи	Режим автомата	Комментарий
	Запуск двигателя	
	Разгон	

	Набор высоты (взлет)	Отрыв от земли
	Набор высоты (взлет)	Набор высоты
	Сбаланси-рованный полет	

	Трюк «мертвая петля»	
	Трюк «мертвая петля»	
	Трюк «мертвая петля»	

	Трюк «мертвая петля»	
	Трюк «мертвая петля»	Выравнивание после выполнения трюка
	Сбалансированный полет	Выравнивание после режима «мертвая петля»

	Снижение	
	Приземле-ние	
	Торможе-ние	

	Торможе- ние	

4.3.5. Анализ пригодности построенного автомата

В результате многократного наблюдения за полетом самолета в авиационном симуляторе не было обнаружено странностей в его поведении. В большинстве случаев полет проходил гладко без каких-либо больших отклонений от идеальной траектории на всех этапах полета.

Также автором был проведен анализ построенного автомата верхнего уровня, в результате которого все ребра и выбранные условия перехода по ним были признаны логичными и оптимальными.

На основании этого был сделан вывод о пригодности построенного автомата верхнего уровня для решения поставленной экспериментальной задачи.

ЗАКЛЮЧЕНИЕ

В работе предложен метод построения автомата верхнего уровня на основе обучающих примеров и построенных автоматов нижнего уровня для каждого используемого режима.

Такой метод необходим для построения системы управляющих автоматов для обеспечения всего процесса полета самолета.

Благодаря использованию обучающих примеров появилась возможность отказаться от использования моделирования для оценки построенного автомата, которое требует больших вычислительных ресурсов.

Предложенный метод был апробирован на задаче построения системы автоматов для управления беспилотным самолетом во время выполнения всего процесса полета, начиная от запуска двигателя, кончая приземлением, с возможностями управления с Земли и выполнения мертвой петли в полете.

Построенная система автоматов была запущена для управления беспилотным самолетом в авиационном симуляторе. В результате наблюдения за поведением самолета был сделан вывод о пригодности построенного автомата верхнего уровня.

По данной работе был сделан доклад на межвузовской научной конференции по проблемам информатики «СПИСОК-2011», проходившей в СПбГУ в 2011 году.

Таким образом, цели работы достигнуты.

СПИСОК ИСТОЧНИКОВ

1. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. СПб.: Питер, 2010. http://is.ifmo.ru/books/_book.pdf
2. *Angeline P., Pollack J.* Evolutionary Module Acquisition /Proceedings of the Second Annual Conference on Evolutionary Programming. Cambridge: MIT Press. 1993, pp.154 – 163. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
3. *Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A.* The Genesys System: Evolution as a Theme in Artificial Life /Proceedings of Second Conference on Artificial Life. MA: Addison-Wesley. 1992, pp. 549 – 578. www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
4. *Царев Ф. Н., Шалыто А. А.* Применение генетического программирования для генерации автомата в задаче об «Умном муравье» / Сборник трудов IV-ой Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте». Том 2. М.: Физматлит. 2007, с. 590 – 597. http://is.ifmo.ru/genalg/_ant_ga.pdf
5. *Бедный Ю. Д., Шалыто А. А.* Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». http://is.ifmo.ru/works/ant
6. *Попов С. И., Попов Ю. И., Шалыто А. А.* Задача о муравьеде и муравьях /Сборник статей третьей Всероссийской научной конференции «Нечеткие системы и мягкие вычисления». Том II. Волгоград: 2009, с. 57 – 63. http://is.ifmo.ru/works/_popovy_volgograd.pdf
7. *Лобанов П. Г., Сытник С. А.* Использование генетических алгоритмов для построения автопилота для простейшего вертолета /Материалы XV Международной научно-методической конференции «Высокие интеллектуальные технологии и инновации в образовании и науке». СПбГПУ. 2008, с. 298. http://is.ifmo.ru/download/2008-03-01_vertolet.pdf

8. *Паращенко Д. А., Царев Ф. Н., Шалыто А. А.* Технология моделирования одного класса мультиагентных систем на основе автоматного программирования на примере игры «Соревнование летающих тарелок». Проектная документация. СПбГУ ИТМО. 2006. http://is.ifmo.ru/unimod-projects/plates/
9. *Coates A., Abbeel P., Ng A. Y.* Learning for Control from Multiple Demonstrations / Proceedings of the 25th International Conference on Machine Learning. Helsinki: 2008, pp. 144 – 151.
10. *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы. М.: Физматлит, 2006.
11. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход. М.: Вильямс, 2006.
12. *Koza J. R.* Genetic programming: on the programming of computers by means of natural selection. MIT Press, 1992.
13. *Курейчик В. М.* Генетические алгоритмы. Состояние. Проблемы. Перспективы // Известия РАН. Теория и системы управления. 1999. № 1, с. 144 – 160.
14. *Курейчик В. М., Родзин С. И.* Эволюционные алгоритмы: генетическое программирование // Известия РАН. Теория и системы управления. 2002. № 1, с. 127 – 137.
15. *Поликарпова Н. И., Точилин В. Н., Шалыто А. А.* Метод сокращенных таблиц для генерации автоматов с большим числом входных переменных на основе генетического программирования // Известия РАН. Теория и системы управления. 2010. № 2, с. 100 – 117.
16. *Александров А. В., Казаков С. В., Сергушичев А. А., Царев Ф. Н., Шалыто А. А.* Применение генетического программирования на основе обучающих примеров для генерации конечных автоматов, управляющих объектами со сложным поведением // Научно-технический вестник СПбГУ ИТМО. 2011. № 2, с. 3 – 11.

17. *Царев Ф. Н.* Метод построения управляющих конечных автоматов на основе тестовых примеров с помощью генетического программирования // Информационно-управляющие системы. 2010. № 5, с. 31 – 36.
18. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
19. *Непейвода Н. Н.* Стили и методы программирования. М.: Интернет-университет информационных технологий, 2005.