

Логическая модель адаптивной системы управления

Демин¹⁾ А. В., Витяев²⁾ Е. Е.

¹⁾ Институт систем информатики СО РАН, alexandredemin@yandex.ru

²⁾ Институт математики СО РАН, evgenii.vityaev@math.nsc.ru

Аннотация. В предыдущей работе были проанализированы принципы теории функциональных систем П.К. Анохина и информационной теории эмоций П.В. Симонова и дана формализация этих принципов вместе с основными схемами деятельности мозга, объясняющими эти теории. В данной работе, на основании этих схем, разрабатывается логическая модель адаптивной системы управления. На основе данной системы управления разработан логический анимат, решающий задачу фуражирования. Задача фуражирования рассматривается с формированием подцелей и без такового. Проводится экспериментальное сравнение работы логического анимата с известными системами адаптивного поведения, основанными на обучении с подкреплением (Reinforcement Learning): Q-Learning, Q-Lookup Table и Q-Neural Net. Проведенные эксперименты показывают, что логический анимат обучается быстрее и работает эффективнее, чем другие системы, особенно в задаче с формированием подцелей.

1. Введение. Обзор адаптивных систем управления и проблемы их разработки

Классические подходы к управлению строятся на том предположении, что можно получить пусть сложную, но точную аналитически заданную форму функциональной зависимости входных и выходных сигналов системы управления с последующим уточнением значений входящих в нее коэффициентов. Однако при всей изощренности наработанного математического инструментария область применения таких методов управления остаются сравнительно простые объекты управления с очевидными свойствами. На практике же для подавляющего большинства как искусственных, так и естественных объектов управления построение точных математических моделей практически невозможно ввиду их плохой формализуемости. Их свойства априори плохо известны или изменяются в процессе функционирования. К тому же, эти объекты могут функционировать в среде, свойства которой изменяются или же вообще не могут быть определены заранее. В связи с этим вопросы построения систем управления подобными объектами выходят за рамки использования традиционных алгоритмов управления.

В последнее время активно развивается направление исследований «Адаптивное поведение» [12, 20, 25, 26], которое предполагает, что принципы работы систем управления должно следовать принципам, на которых основано поведение живых организмов. Одним из основных подходов этого направления является создание и исследование агентов (компьютерных программ или роботов), способных приспосабливаться к внешней среде. Подобные агенты были названы «аниматами» (animal + automat = animat).

От классического подхода к управлению, а также от традиционных разработок в области искусственного интеллекта разработки в рамках направления «Адаптивное поведение» отличаются не только ориентацией на принципы по-

ведения животных, но и тем, что разрабатываемые системы управления в конечном итоге ориентированы на решение плохо сформулированных задач в плохо предсказуемой среде – т.е. таких, с которыми приходится иметь дело живым организмам.

В настоящее время в рамках направления «Адаптивное поведение» предложено множество разнообразных подходов к построению систем управления. Попытаемся охарактеризовать основные подходы, следуя обзорным работам [12, 20, 26].

В рамках «Адаптивного поведения» существующие подходы к построению систем управления можно условно разбить на две группы:

- **Инженерные подходы:** поглощающая архитектура, динамический подход;
- **Подходы на основе самообучения:** обучение с подкреплением, системы классификаторов.

Инженерные подходы. Инженерные подходы в большей мере ориентированы на конструирование систем управления вручную человеком-разработчиком. При этом разработчик пытается заранее предусмотреть все возможные формы поведения агента и ситуации, когда их необходимо применять. Как правило, системы управления, основанные на данных подходах, показывают достаточно сложное поведение, что демонстрируется на многочисленных примерах управления реальными роботами. Однако возможности обучения в данных системах обычно либо отсутствуют вообще, либо ограничиваются подстройкой параметров управления.

Рассмотрим основные подходы, относящиеся к этой группе.

Поглощающая архитектура. Данная архитектура была предложена Р. Бруксом [15, 16] во второй половине 80-х годов как альтернатива традиционным системам искусственного интеллекта. Поглощающая архитектура предполагает, что агент состоит из отдельных поведенческих модулей, каждый из которых независимо управляет отдельной формой поведения без какого-либо моделирования среды или планирования действий: действия запускаются в ответ на внешние сигналы или даже просто спонтанно. Эти элементарные формы поведения могут представлять собой ненаправленное блуждание, движение к цели, обход препятствия и т.д. Результирующее адаптивное поведение создается в этом случае конкуренцией поведенческих модулей, которые могут подавлять активность друг друга. Например, пока агент не воспринимает каких-либо специфических сигналов, он блуждает по местности в поисках цели. Когда цель обнаружена, активизируется модуль движения к цели, который подавляет активность модуля ненаправленного блуждания. Если на пути к цели возникает препятствие, то активизируется модуль обхода препятствия, который подавляет все остальные модули и т.д.

Несмотря на простоту подобного рода схем, они показали хорошие результаты при создании систем управления мобильными роботами [17]. Однако, как правило, в системах с поглощающей архитектурой отсутствуют какие-либо механизмы самообучения. Чтобы частично преодолеть эти ограничения, в некоторых работах [21] предлагается использовать нейронные сети для реализации поведенческих модулей. Однако результаты этих работ показывают, что, по крайней мере, в некоторых случаях монолитный нейросетевой контроллер ра-

ботает лучше, чем нейросетевой контроллер, разбитый на поведенческие модули.

Поглощающая архитектура обладает следующими недостатками:

- Поглощающая архитектура предполагает, что декомпозиция системы на поведенческие модули должна быть задана априори человеком-разработчиком. Не предусмотрено никаких возможностей для формирования системой новых поведенческих модулей на основании приобретенного опыта. Схема взаимодействия между поведенческими модулями фиксирована и не может быть изменена в процессе работы системы. Тем самым, возможности обучения для данной архитектуры ограничиваются только подстройкой поведенческих модулей, что ограничивает адаптивные способности агента.
- При создании агента разработчик должен предусмотреть все возможные формы поведения агента и способы взаимодействия между поведенческими модулями, что не всегда возможно сделать, особенно если агент предназначен для функционирования в сложной недетерминированной среде.
- Системы управления на основе поглощающей архитектуры не обладают универсальностью. Отсутствует общий метод построения систем управления данной архитектуры. Для каждого агента приходится разрабатывать систему управления индивидуально и «с нуля».

Динамический подход. В динамическом подходе система управления рассматривается как непрерывная динамическая система [22]. Как и в поглощающей архитектуре, динамическая система состоит из множества работающих параллельно процессов. Каждый из этих процессов посредством дифференциальных уравнений устанавливает непрерывную связь между показаниями сенсоров и параметрами эффекторов. Результирующим действием будет являться комбинация управляющих воздействий всех процессов. К примеру, если один процесс будет предлагать повернуть на 20 градусов по часовой стрелке, а другой – на 20 градусов против часовой стрелки, то результирующим действием будет поворот на 0 градусов.

Сведение всей системы управления агентом к простым комбинациям управляющих воздействий отдельных поведенческих подсистем обычно приводит к более гладкому и непрерывному поведению по сравнению системами с поглощающей архитектурой. Однако это также приводит и к проблемам, связанным со сложностью обеспечения взаимодействия между различными типами поведений (к примеру, когда один тип поведения подавляет активность другого). Для решения этих проблем одни авторы [22] предлагают вводить дополнительные «мотивационные переменные», влияющие на поведенческие подсистемы. Другие авторы [23] предлагают использовать специальные «активационные переменные», которые назначаются для каждой подсистемы и определяют их активность, и «соревновательную матрицу», которая определяет, какие формы поведений могут быть активны в данный момент.

Однако динамический подход в целом обладает теми же недостатками, что и поглощающая архитектура:

- Динамический подход предоставляет мало возможностей для самообучения систем управления. Отсутствуют механизмы автоматического формирования новых поведенческих подсистем. Самообучение ограничивается подстройкой параметров в пределах заданных уравнений.

- При решении практических задач не всегда можно предусмотреть все возможные формы поведения.
- Увеличение количества поведенческих модулей при разработке сложных агентов приводит к существенному возрастанию сложности организации взаимодействия между ними.
- Системы управления на основе динамического подхода не обладают универсальностью.

Подходы на основе самообучения. Подходы на основе самообучения в большей мере ориентированы на разработку систем управления, способных самостоятельно формировать свое поведение, обучаясь на опыте своего взаимодействия с внешней средой. Как правило, системы управления, ориентированные на самообучение, демонстрируют значительно более высокие возможности обучения и адаптации по сравнению с инженерными подходами. Однако из-за недостатков существующих подходов и ряда нерешенных проблем, которые будут рассмотрены ниже, область применения подобных систем пока еще ограничивается достаточно простыми задачами.

Рассмотрим основные подходы, относящиеся к этой группе.

Обучение с подкреплением. Теория обучения с подкреплением (Reinforcement Learning) была разработана в работах Р. Саттона и Э. Барто [24]. Идея данного подхода заключается в том, что за каждое совершенное действие агент получает подкрепление – некоторое действительное число, которое может быть положительным (награда) или отрицательным (наказание). Задачей агента является максимизация суммарной награды, которую он может получить с течением времени. Для этого он в процессе обучения пытается сформировать внутреннюю оценку величин суммарной награды, которую он получит, если в текущей ситуации выполнит то или иное действие. Во время своей работы агент совершает те действия, которым соответствует максимальная оценка суммарной награды. Для аппроксимации оценки величины суммарной награды обычно используются нейронные сети.

На данный момент теория обучения с подкреплением и нейронные сети являются наиболее популярными подходами в области разработки адаптивных самообучающихся агентов: по количеству работ данный подход значительно превосходит все остальные направления. Преимущество обучения с подкреплением – его простота, поскольку для обучения агента достаточно получать из внешней среды только сигнал подкрепления. Таким образом, агент обучается непосредственно на своем опыте в процессе взаимодействия с внешней средой. Другое преимущество заключается в том, что данный подход обладает значительно более универсальными свойствами по сравнению с описанными выше инженерными подходами, т.е. системы управления для разных агентов отличаются лишь наборами сенсоров и действий, а также выбранной архитектурой нейронной сети.

Однако данный подход также не лишен недостатков:

- Сходимость алгоритмов обучения с подкреплением доказана только для Марковских сред и для бесконечного количества тактов работы. Однако в реальных задачах условия Марковских сред выполняются достаточно редко. На практике серьезными проблемами данного подхода являются плохая сходимость – алгоритм может вообще не сойтись, либо сойтись к локальному

минимуму, и нестабильность — достигнув оптимального поведения, система может легко «разобучиться».

- Нейронная сеть, обучаемая методом обучения с подкреплением, на практике способна хорошо обучиться только одной форме поведения, т.е. достигать только одну цель. Для решения этой проблемы в некоторых работах предлагается разбивать систему управления на отдельные модули, по аналогии с поглощающей архитектурой, а также использовать иерархический метод обучения с подкреплением (Hierarchical Reinforcement Learning) [14]. Однако данные подходы в основном страдают теми же недостатками, что и поглощающая архитектура: разбиение на модули задается априори человеком-разработчиком, отсутствуют механизмы автоматического формирования новых модулей, схема взаимодействия между модулями фиксирована и не может быть изменена самой системой в процессе ее работы.
- Существенной проблемой для алгоритмов обучения с подкреплением является медленная скорость обучения, которая очень сильно зависит от размера пространства состояний и от частоты получения подкрепления. В реальных задачах, когда количество возможных состояний велико, а также когда достижение конечной цели требует выполнения длительных и разнообразных серий действий, скорость обучения системы, как правило, становится неприемлемо низкой.

Системы классификаторов. Системы классификаторов (Classifier Systems) были впервые предложены Холландом [19]. Данный подход может быть рассмотрен как специальный случай теории обучения с подкреплением, однако поскольку существует множество работ, посвященных данному подходу, а также, поскольку в этих работах системы классификаторов, как правило, не рассматриваются с точки зрения обучения с подкреплением, то системы классификаторов обычно выделяют в качестве самостоятельного направления.

Системы классификаторов работают на основе множества правил (классификаторов), имеющих вид:

Если $\langle \text{условие}_1 \rangle, \langle \text{условие}_2 \rangle, \dots, \langle \text{условие}_n \rangle$, то $\langle \text{действие} \rangle$

где условие правила описывает ситуацию, к которой применим данный классификатор, а заключение — действие, которое необходимо совершить. Каждый классификатор имеет свой вес (силу классификатора), показывающий его полезность для системы. При совершении действий предпочтение отдается тем классификаторам, которые имеют больший вес.

Система классификаторов обучается в результате двух процессов: адаптации и эволюции. Как и в теории обучения с подкреплением, система стремится оптимизировать награду, которую она получает при совершении действий. В процессе адаптации модифицируются веса классификаторов с помощью специального алгоритма, называемого «алгоритмом пожарной бригады», при этом поощряется не только тот классификатор, который привел к успешному действию, но и его предшественники. В процессе эволюции при помощи генетического алгоритма формируются новые классификаторы. Эволюционное обучение применяется очень редко, например, один раз в 1000 итераций, или при стабилизации весов классификаторов в популяции. Это дает алгоритму пожарной бригады время, чтобы подстроить веса классификаторов, после чего они могут быть использованы генетическим алгоритмом.

Приведем основные недостатки данного подхода:

- Классическая система классификаторов, так же как и монолитная нейронная сеть в теории обучения с подкреплением, на практике способна хорошо обучиться только одной форме поведения. Для решения этой проблемы в некоторых работах предлагается разбивать систему управления на отдельные модули [18]. Однако все существующие работы в данном направлении предполагают, что иерархия модулей должна быть изначально задана человеком-разработчиком и не может изменяться в процессе функционирования системы. При этом данные подходы сталкиваются с теми же проблемами, что и системы с поглощающей архитектурой.
- Серьезной проблемой для систем классификаторов является сходимость популяции классификаторов при использовании генетического алгоритма к оптимальному набору правил. Популяция классификаторов может легко сойтись к популяции незначительно отличающихся друг от друга копий лишь нескольких видов правил. Это приводит к «фрагментарному» поведению агента, т.е. в определенных ситуациях агент показывает оптимальное поведение, в то время как в других – наоборот.
- Система классификаторов обладает низкой скоростью обучения и адаптации. Во-первых, это связано с тем, что адаптации весов классификаторов при помощи алгоритма пожарной бригады требует много времени, особенно для обучения длинных цепочек правил. Во-вторых, поиск новых классификаторов осуществляется крайне медленно, поскольку после каждого шага генетического алгоритма следует длительный процесс адаптации весов правил. В результате чего, при функционировании в динамично меняющейся среде, а также при работе в реальном времени скорость адаптации системы может стать неприемлемо низкой.

Итог. Резюмируя обзор, можно выделить следующие основные проблемы, связанные с разработкой автономных адаптивных систем управления:

1. Проблема координации различных форм поведения и организации взаимодействия между ними.
2. Проблема планирования действий. Система управления должна не просто реагировать на текущую ситуацию, но и прогнозировать последовательность действий, которые должны привести к намеченной цели.
3. Проблема обучения на опыте своего взаимодействия с внешней средой. Данная проблема включает не только вопросы обучения отдельной форме поведения, но и вопросы обучения взаимодействию между различными формами поведения.
4. Проблема формирования новых типов поведения. Система управления должна уметь самостоятельно формировать новые типы поведения, направленные на достижение новых целей, и включать их в общую структуру контроля.
5. Проблема универсальности. Системы управления различными объектами должны следовать единым принципам управления и основываться на единой модели.

Существующие на данный момент подходы к построению систем управления решают только некоторые из вышеперечисленных проблем. Таким

образом, становится актуальной задача разработки такой модели системы управления, которая бы решала указанные проблемы.

В данной работе предлагается новая модель адаптивной системы управления, которая решает указанные проблемы за счет использования теории функциональных систем П.К. Анохина и разработанного логико-вероятностного метода обучения.

2. Теория функциональных систем

Архитектура предложенной нами системы управления основана на Теории функциональных систем П.К. Анохина [1-5], Информационной теории эмоций П.В.Симонова [13] и анализе принципов этих теорий [6-8]. Согласно теории функциональных систем, единицей деятельности организма являются функциональные системы, формирующиеся для достижения полезных для организма результатов (например, удовлетворение потребностей). Организация функциональных систем при целенаправленном поведении осуществляется в соответствии с двумя правилами: последовательностью и иерархией результатов. Последовательность результатов выстраивается по принципу «доминанты»: доминирующая потребность возбуждает доминирующую функциональную систему и строит поведенческий акт, направленный на удовлетворение потребности. По отношению к доминирующей функциональной системе все остальные функциональные системы выстраиваются в иерархию по принципу «иерархии результатов»: когда результат деятельности одной функциональной системы входит в качестве компонента в результат деятельности другой.

Центральные механизмы функциональных систем, обеспечивающих целенаправленные поведенческие акты, имеют однотипную архитектуру (рис. 1). Начальную стадию поведенческого акта любой степени сложности составляет *афферентный синтез*, включающий в себя мотивационное возбужде-

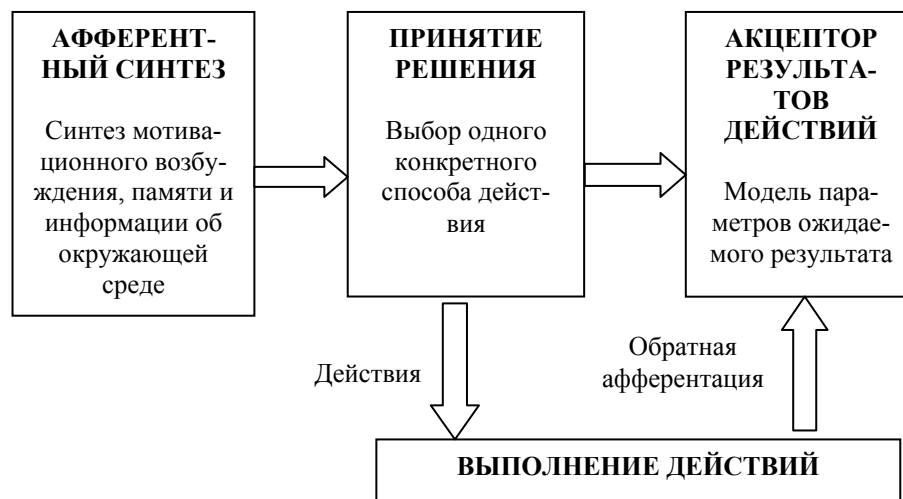


Рис. 1. Схема работы функциональной системы

ние, память и информацию об окружающей среде. В результате афферентного синтеза из памяти извлекаются все возможные способы достижения цели в данной ситуации. На стадии *принятия решений* в соответствии с исходной потребностью выбирается только один конкретный способ действий и формируется *акцептор результатов действий*, представляющий собой модель параметров ожидаемого результата. Выполнение каждого действия сопровождается сигналами о достижении результата, называемыми *обратной афферентацией*. Действия по достижению цели продолжаются до тех пор, пока параметры результата действия, поступающие в центральную нервную систему в форме соответствующей обратной афферентации, не будут полностью соответствовать свойствам акцептора результатов действия.

Отдельная ветвь общей теории функциональных систем – теория системогенеза, изучающая закономерности формирования функциональных систем. В данной работе мы также рассмотрим механизм формирования новых функциональных систем на основе выявления подцелей.

3. Архитектура логической системы управления

В соответствии с теорией функциональных систем будем считать, что моделируемая система управления аниматом имеет иерархическую архитектуру, в которой базовым элементом системы управления является отдельная функциональная система. При такой архитектуре функциональные системы верхнего уровня ставят цели системам нижнего уровня. При этом можно считать, что каждая функциональная система решает задачу достижения цели, используя те же методы, что и остальные функциональные системы. На рис. 2 приведена архитектура системы управления аниматом.

Задача отдельной функциональной системы:

- При заданной цели (подцели) и известной информации об окружающей среде и состоянии функциональной системы найти наилучший способ достижения цели.
- Если на основе прогноза найдено действие, обеспечивающее дости-

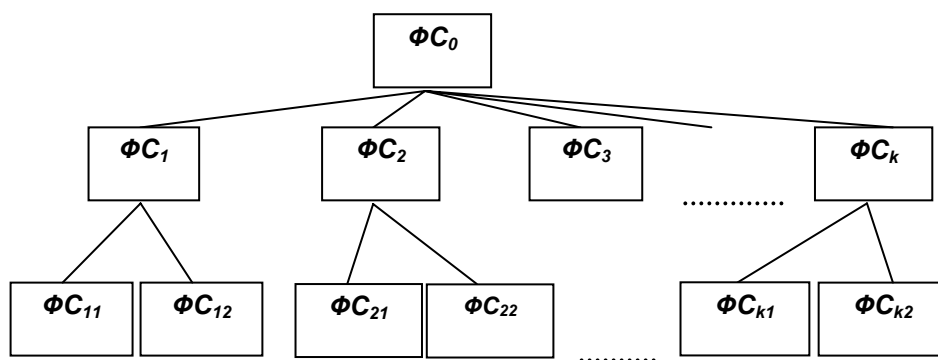


Рис. 2. Архитектура системы управления

жение цели, то выполнить это действие.

- Осуществить контроль правильности выполнения действия, т.е. проверить соответствие параметров достигнутого и желаемого результатов.

Будем предполагать, что система управления аниматом функционирует в дискретном времени $t = 0, 1, 2, \dots$. Пусть анимат имеет некоторый набор сенсоров $S = \{S_1, S_2, \dots, S_n\}$, характеризующих состояние внешней и внутренней среды, и набор возможных действий $A = \{A_1, A_2, \dots, A_m\}$. Считаем, что история деятельности анимата хранится в таблице данных $X = \{X_1, X_2, \dots, X_t\}$, где t -я строка таблицы содержит показания сенсоров и совершенное действие в момент времени t : $X = \{S'_1, S'_2, \dots, S'_n, A_t\}$, S'_1, S'_2, \dots, S'_n – значения сенсоров S_1, S_2, \dots, S_n в момент времени t ; A_t – действие, совершенное в момент времени t . Чтобы таблица X в процессе деятельности анимата не разрасталась до бесконечности, можно ограничить её размеры, сохраняя в ней только данные последних M тактов работы.

В каждый момент времени t на вход системы управления подается информация об окружающей среде в виде набора предикатов $P = \{P_1, P_2, \dots, P_k\}$, описывающих состояние сенсоров в момент времени t . Будем называть эти предикаты *сенсорными предикатами*. Таким образом, на вход системы управления поступают не сами показания сенсоров, а описания их состояний в виде набора сенсорных предикатов. Отметим, что набор сенсорных предикатов может включать не только предикаты, описывающие текущее состояние сенсоров, но и предикаты, описывающие состояние сенсоров в предыдущие моменты времени $((t-1), (t-2), \dots$ и т.д.), тем самым мы позволяем системе обладать некоторой «памятью».

Каждой функциональной системе ΦC соответствует некоторая цель G , достижение которой является задачей данной функциональной системы. Будем предполагать, что каждая цель может быть описана некоторой конъюнкцией сенсорных предикатов $PG = P_1 \& P_2 \& \dots \& P_l$, где $P_i \in P$ (см. рис. 3). Будем называть предикат PG *предикатом-целью*. Когда предикат-цель принимает значение «истина», то это означает, что соответствующая цель достигнута, в противном случае цель не достигнута, т.е., по сути, данный предикат «сигнализирует» системе о достижении цели.

Каждая функциональная система ΦC содержит также набор предикатов $G = \{PG_1, \dots, PG_n\}$, где PG_i – предикаты-цели, соответствующие целям нижестоящих по иерархии функциональных систем, подчиненных данной функциональной системе.

Каждая функциональная система ΦC содержит множество закономерностей PR вида:

$$P_1 \& \dots \& P_n \& PG_1 \& \dots \& PG_m \& A_1 \& \dots \& A_k \rightarrow PG, \quad (1)$$

где $P_i \in P$, $PG_i \in G$, $A_i \in A$. Эти закономерности предсказывают, что:

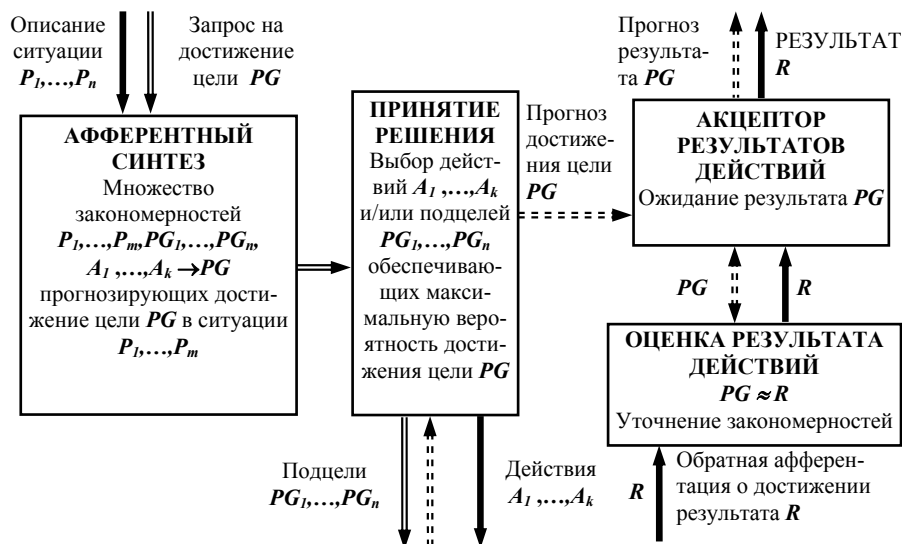


Рис. 3. Модель работы функциональной системы

- если анимат находится в определенной ситуации, описываемой сенсорными предикатами P_1, \dots, P_n ;
- и в этой ситуации он последовательно достигнет цели, заданные предикатами-целями PG_1, \dots, PG_m ;
- и затем последовательно выполнит серию действий A_1, \dots, A_k ;
- то он с некоторой вероятностью¹ p достигнет поставленную цель G (т.е. предикат-цель PG примет значение «истина»).

Дополнительно отметим, что очередность достижения целей и очередность выполнения действий соответствует тому порядку, в котором эти цели и действия записаны в закономерности. Приведем пример. Закономерности $PG_1 \& PG_2 \& A \rightarrow PG$ и $PG_2 \& PG_1 \& A \rightarrow PG$ – различны, поскольку они определяют различный порядок достижения целей G_1 и G_2 , аналогично закономерности $A_1 \& A_2 \rightarrow PG$ и $A_2 \& A_1 \rightarrow PG$ – также различны.

Таким образом, формально мы можем представить функциональную систему ΦC следующим набором: $\Phi C = \langle PG, G, PR \rangle$, где PG – предикат-цель, характеризующий цель, достижение которой является задачей данной функциональной системы; G – множество предикатов-целей, соответствующих

¹ Как и в работе [6] примем интерпретацию вероятности, введенную К. Поппером, как предрасположенность ситуации к появлению некоторого события с определенной вероятностью. Определение события и метод вычисления оценок вероятностей приведены в разделе 6.

функциональным системам, подчиненным данной системе; PR – множество закономерностей, принадлежащих данной функциональной системе вида (1).

Теперь мы можем формально представить систему управления (Control System (CS)) в виде набора $CS = \langle S, A, P, F \rangle$, где S – набор сенсоров системы управления; A – набор возможных действий; P – множество сенсорных предикатов, описывающих состояние сенсоров; F – иерархия функциональных систем.

Иерархия функциональных систем F определяется следующим образом. На вершине иерархии (уровень $k = 1$) располагается одна функциональная система $\Phi C^{(1)} = \langle PG^{(1)}, G^{(1)}, PR^{(1)} \rangle$, соответствующая доминирующей цели $G^{(1)}$. На уровне иерархии $k \geq 2$ располагается множество функциональных систем $F^{(k)} = \{\Phi C_i^{(k)}\}$, таких, что для любой системы $\Phi C_i^{(k)} \in F^{(k)}$ существует надсистема $\Phi C_j^{(k-1)} = \langle PG_j^{(k-1)}, G_j^{(k-1)}, PR_j^{(k-1)} \rangle$, такая, что $PG_i^{(k)} \in G_j^{(k-1)}$ и $PG_i^{(k)} \notin G_j^{(l)}, l \neq (k-1)$.

4. Модель работы функциональной системы

Рассмотрим, каким образом происходит работа отдельной функциональной системы [8, 9], представленной на рис. 3².

Предположим, что в некоторый момент времени t перед функциональной системой $\Phi C = \langle PG, G, PR \rangle$ ставится цель PG . Цель ставится в виде запроса к функциональной системе – достичь эту цель. На вход функциональной системы подается информация об окружающей среде в виде набора значений сенсорных предикатов $P_1(t), P_2(t), \dots, P_k(t)$, описывающих текущую ситуацию.

В процессе афферентного синтеза из памяти извлекается вся информация, связанная с достижением этой цели. Поскольку информация об окружающей среде задана в виде описания ситуации $P_1(t), P_2(t), \dots, P_k(t)$, то из памяти автоматически извлекается только та информация, связанная с достижением цели, которая может быть применена в данной ситуации. Это достигается использованием (извлечением из памяти) только тех закономерностей из множества PR , в которых свойства ситуации $P_1(t), P_2(t), \dots, P_n(t)$ $P_i(t)$, входящие в условие закономерности (1), выполнены.

Далее все извлеченные из памяти закономерности передается в блок принятия решения, где для каждой закономерности R рассчитывается оценка вероятности достижения цели $f(R)$ следующим образом.

² Взаимосвязь вводимых формальных понятий с физиологическими понятиями теории функциональных систем П.К. Анохина и информационной теории эмоций П.В. Симонова приведены в работе [6]. В работе [6] функциональные системы также определяются множеством закономерностей, которые обнаруживаются формальными нейронами. Модель функциональной системы рис. 3 в работе [6] представлена отдельными блоками действий и получения результатов (рис. 5 работы [6]).

Если условная часть закономерности содержит только сенсорные предикаты и действия, т.е. $R = P_1 \& \dots \& P_n \& A_1 \& \dots \& A_k \rightarrow PG$, то оценка вероятности достижения цели равна оценке условной вероятности данной закономерности $f(R) = p(R)$ ³.

Если условия закономерностей содержат не только свойства ситуации, но и подцели, достижение которых необходимо для достижения цели, то оценка вероятности достижения цели рассчитывается по следующей формуле:

$$f(R) = p(R) \cdot f(PG_1) \cdot f(PG_2 | PG_1) \cdot \dots \cdot f(PG_m | PG_{m-1}), \text{ где}$$

- $f(PG_1)$ – оценка вероятности достижения подцели G_1 в текущей ситуации;
- $f(PG_i | PG_{i-1})$ – оценка вероятности достижения подцели G_i из ситуации, определенной предикатом-целью PG_{i-1} , т.е., оценка вероятности достижения подцели G_i после достижения подцели G_{i-1} .

Расчет оценки вероятности $f(PG_1)$ достижения подцели G_1 из текущей ситуации осуществляется отправкой запроса соответствующей функциональной системе, находящейся ниже по иерархии и реализующей достижение этой подцели. Этот запрос активирует в подчиненной функциональной системе аналогичный процесс афферентного синтеза и принятия решений в той же ситуации. В результате чего будет получена максимальная оценка вероятности достижения подцели G_1 , которая и будет возвращена в ответ на запрос.

Расчет оценки вероятности $f(PG_i | PG_{i-1}), i = 2, \dots, m$ осуществляется аналогичным образом путем отправки запроса соответствующей функциональной системе ΦC_i , реализующей достижение цели G_i . Отличие состоит в том, что на вход функциональной системы вместо текущей ситуации подается ситуация определенная предикатом-целью PG_{i-1} (поскольку нам, в точности, неизвестна ситуация, в которой окажется анимат после достижения подцели G_{i-1}). Однако нам точно известно, что, если анимат достиг подцель G_{i-1} , то будет выполнен предикат PG_{i-1} , а это значит, что в описании ситуации все сенсорные предикаты, входящие в PG_{i-1} , будут выполнены. Поэтому, вместо точного описания ситуации, на вход функциональной системы ΦC_i подается её «приближенное» описание, в котором истинны только предикаты, входящие в состав PG_{i-1} . Обозначим это «приближенное» описание ситуации через $S(PG_{i-1})$ и будем называть его *ситуацией, определенной предикатом PG_{i-1}* . Запросы к функциональной системе ΦC_i также активируют в ней процессы афферентного синтеза и принятия решений в ситуации $S(PG_{i-1})$ (аналогично тому, как это было описано выше). В результате выполнения запросов будут получены оценки вероятности достижения подцелей G_i .

³ Метод подсчета оценки условной вероятности закономерности по истории деятельности анимата приведен ниже в п. 6.3.

Если при вычислении запросов для достижения какой-либо подцели потребуется достижение ещё более низких по иерархии целей, то в этом случае аналогичные запросы будут отправлены ещё ниже по иерархии и т.д. Если какая-то из подцелей не может быть выполнена в данной ситуации (нет закономерностей, предсказывающих достижение подцели в данной ситуации), то в ответ на запрос возвращается отказ и закономерность, инициировавшая запрос, будет исключена из рассмотрения.

После того как будут вычислены оценки вероятности достижения цели для всех закономерностей, блок принятия решения просматривает их и выбирает такую закономерность R_{best} , которая имеет максимальную оценку вероятности достижения цели.

Прогноз ожидаемого результата, заданного предикатом-целью PG , отправляется в акцептор результатов действий. Кроме того, во всех функциональных подсистемах нижнего уровня прогнозы подрезультатов PG_i также отправляется в акцепторы результатов действий соответствующих функциональных подсистем.

В начальной стадии обучения, когда ещё нет правил, либо нет ни одного правила, применимого в данной ситуации, действие соответствующей функциональной системы выбирается методом «проб и ошибок» и прогноз отсутствует.

После выполнения действий A_1, \dots, A_k обновляются показания сенсоров и, соответственно, обновляются значения сенсорных предикатов. Таким образом, анимат окажется в новой ситуации, которая будет являться результатом выполнения действий. Данные о полученном результате (обратная афферентация), т.е. набор значений сенсорных предикатов, описывающих новую ситуацию, поступают в акцептор результатов действий в блок оценки результата. Проводится сравнение спрогнозированного и полученного результатов, т.е. проверяется, будет ли выполнен предикат-цель PG в полученной ситуации. В случае, если предикат-цель PG выполнен, анимат достиг цель G , то оценка условной вероятности закономерности³, выбранной в блоке принятия решений, будет увеличена, в противном случае – уменьшена. После каждого действия производится *уточнение набора правил*⁴ PR . Для получения множества закономерностей PR используется специально разработанный адаптированный вариант семантического вероятностного вывода (см. раздел 6).

5. Схема работы системы управления

Рассмотрим, каким образом происходит работа системы управления в целом. Каждый такт работы системы включает два этапа (рис. 4):

- 1) формирование плана действий;

⁴ Уточнение набора правил может производиться двояким способом: либо переобучением множества закономерностей PR в соответствии с алгоритмом пп. 6.2-6.3, либо путем потактового уточнения «сработавших» закономерностей путем уточнения, в соответствии с п. 6.1, «сработавших» закономерностей и проверкой их алгоритмом пп. 6.2-6.3 на удовлетворение требованиям вероятностных закономерностей.

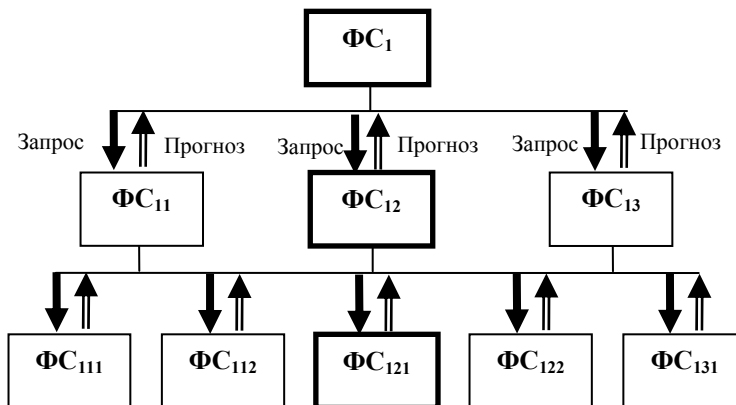
2) выполнение действий и контроль достижения результата.

Во время первого этапа происходит инициация запроса $G^{(t)}$ к функциональной системе $\Phi C^{(t)} = \langle PG^{(t)}, G^{(t)}, PR^{(t)} \rangle$, находящейся на вершине иерархии функциональных систем. Ответом на запрос является оценка максимально вероятного прогноза достижимости цели и выработка соответствующего плана действий. Формирование ответа на запрос происходит путем своеобразного вероятностного «вычисления» достижимости цели, происходящего таким же методом, как и вычисление ответа на запрос в логическом программировании – путем древовидного развертывания вниз, по иерархии всех подцелей, вычисления всех оценок вероятностей и сворачивания их в результирующую оценку вероятности достижения цели (см. рис. 4 этап 1). Подробнее процесс вычисления оценок вероятности запросов был описан в предыдущем разделе.

В процессе вычисления запросов «извлекается из памяти» множество ветвей дерева вывода в виде цепочек закономерностей

$$R^{(k)} \Rightarrow R^{(k-1)} \Rightarrow \dots \Rightarrow R^{(t)},$$

Этап 1. Формирование плана действий



Этап 2. Выполнение действий и контроль достижения результата

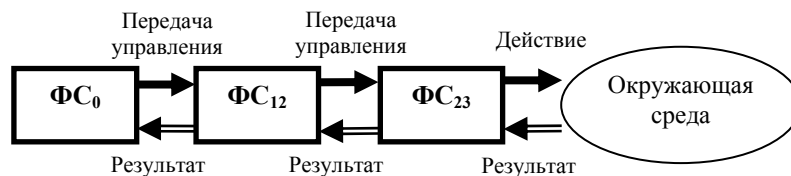


Рис. 4. Один такт работы системы управления

где $R^{(l)}$ – закономерность, принадлежащая функциональной системе $\Phi C^{(l)}$, обеспечивающая достижение цели $G^{(l)}$; $R^{(i)}, i = 2, \dots, k$ – закономерности, принадлежащие функциональным системам $\Phi C^{(i)}$, обеспечивающим достижение подцелей $G^{(i)}$. Совокупность таких ветвей дерева вывода определяет иерархию действий:

$$\underbrace{A_1^{2(k)} \Rightarrow \dots \Rightarrow A_{n_2}^{2(k)}}_{R^{(k)}} \Rightarrow \underbrace{A_2^{(k-1)} \Rightarrow \dots \Rightarrow A_m^{(k-1)}}_{R^{(k-1)}} \Rightarrow \dots \Rightarrow \underbrace{A_1^{(l)} \Rightarrow \dots \Rightarrow A_l^{(l)}}_{R^{(l)}}$$

где $A_1^{(i)}, \dots, A_j^{(i)}$ – серия действий, записанная в условной части соответствующего правила $R^{(i)}$.

На стадии принятия решений доминирующей функциональной системой $\Phi C^{(l)}$ будет выбран один наилучший план действий PL_{Best} , обеспечивающий максимальную оценку вероятности достижения доминирующей цели $G^{(l)}$, включающей выполнение действий на всех уровнях иерархии и по всем ветвям дерева, которые можно представить схемой:

$$\left[\begin{array}{l} \underbrace{A_1^{1\dots l(k)} \Rightarrow \dots \Rightarrow A_{n_{1..l}}^{1\dots l(k)}}_{R^{(k)}} \Rightarrow \underbrace{A_1^{1\dots l(k-1)} \Rightarrow \dots}_{R^{(k-1)}} \dots \Rightarrow \underbrace{A_l^{(l)} \Rightarrow \dots}_{R^{(l)}} \\ \underbrace{A_1^{21\dots l(k)} \Rightarrow \dots \Rightarrow A_{n_{21..l}}^{21\dots l(k)}}_{R^{(k)}} \Rightarrow \underbrace{A_2^{1\dots l(k-1)} \Rightarrow \dots}_{R^{(k-1)}} \dots \Rightarrow \underbrace{A_l^{(l)} \Rightarrow \dots}_{R^{(l)}} \\ \dots \\ \underbrace{A_1^{m1\dots l(k)} \Rightarrow \dots \Rightarrow A_{n_{m1..l}}^{m1\dots l(k)}}_{R^{(k)}} \Rightarrow \underbrace{A_m^{1\dots l(k-1)} \Rightarrow \dots}_{R^{(k-1)}} \Rightarrow \underbrace{A_1^{1\dots l(k-2)} \Rightarrow \dots}_{R^{(k-2)}} \dots \Rightarrow \underbrace{A_l^{(l)} \Rightarrow \dots}_{R^{(l)}} \\ \underbrace{A_1^{121\dots l(k)} \Rightarrow \dots \Rightarrow A_{n_{12}}^{121\dots l(k)}}_{R^{(k)}} \Rightarrow \underbrace{A_1^{21\dots l(k-1)} \Rightarrow \dots}_{R^{(k-1)}} \Rightarrow \underbrace{A_2^{1\dots l(k-2)} \Rightarrow \dots}_{R^{(k-2)}} \dots \Rightarrow \underbrace{A_l^{(l)} \Rightarrow \dots}_{R^{(l)}} \\ \dots \\ \dots \\ \dots \end{array} \right]$$

В этой схеме верхний индекс у действия является кодом цепи дерева.

На этом система управления завершит первый этап работы и перейдет ко второму этапу: выполнению действий и контролю достижения результата.

Во время второго этапа система, в соответствии с планом PL_{Best} , передаёт управление функциональной системе $\Phi C^{(k)}$, которая запустит на выполнение самое первое по плану PL_{Best} действие $A_1^{(k)}$. После совершения действия и обновления показаний сенсоров для всех функциональных систем, вовлеченных в план PL_{Best} , будет произведен контроль достижения результата действий и уточнение набора правил⁴. На этом система управления завершит один такт работы.

Контроль достижения результата действия после выполнения каждого действия необходим, поскольку любой план действий носит вероятностный характер. Система управления на каждом такте своей работы также проверяет, является ли выбранный план действий наилучшим и, если нет, то осуществляет

поиск лучшего плана действий. Тем самым система постоянно корректирует свой план действий с учетом вновь поступающей информации из окружающей среды.

Таким образом, предложенная модель системы управления постоянно на каждом шаге осуществляет прогноз достижения результата и контроль правильности выполнения действий в соответствии с теорией функциональных систем П.К. Анохина.

6. Обнаружение вероятностных закономерностей

Рассмотрим некоторую функциональную систему $\Phi C = \langle PG, G, PR \rangle$, входящую в состав системы управления $CS = \langle S, A, P, F \rangle$. Опишем метод обучения функциональной системы ΦC путем обнаружения множества закономерностей PR . Для обнаружения множества PR был разработан адаптированный вариант семантического вероятностного вывода [6, 9, 10], учитывающий специфику правил PR ⁵.

Приведём формальное определение вероятностных закономерностей⁶ и определим их роль в формировании условных связей в функциональных системах.

6.1. Вероятностные закономерности, уточнение правил.

Обучение происходит путем уточнения правил. Для правила вида (1)

$$R_1 = P_1 \& \dots \& P_n \& PG_1 \& \dots \& PG_m \& A_1 \& \dots \& A_k \rightarrow PG, \text{ где}$$

$$P_i \in P, PG_i \in G, A_i \in A$$

будем называть *подправилом* любое правило

$$R_2 = P_1^2 \& \dots \& P_{n2}^2 \& PG_1^2 \& \dots \& PG_{m2}^2 \& A_1^2 \& \dots \& A_{k2}^2 \rightarrow PG,$$

для которого выполнены нижеследующие условия 1-4. Правило R_1 , относительно своего подправила R_2 , является *уточнением* этого подправила R_2 , поэтому условия 1-4 определяют также уточнение правил следующим образом.

1. $\{P_1^2, \dots, P_{n2}^2\} \subseteq \{P_1, \dots, P_n\}$, $n2 \leq n$. Смысл условия состоит в том, что уточнение правил осуществляется добавлением в посылку правила $P_1^2 \& \dots \& P_{n2}^2$ новых условных сигналов ситуации из числа предикатов P_1, \dots, P_n , которые могут улучшить предсказательную способность правила. Это и означает, что условный сигнал образовал условную связь на уровне нейрона. Проверка улучшения предсказательной способности правила, при добавлении новых условных сигналов, осуществляется проверкой правила на удовлетворение требованиям вероятностных закономерностей п. 6.3.
2. $\{PG_1^2, \dots, PG_{m2}^2\} \subseteq \{PG_1, \dots, PG_m\}$, $m2 \leq m$,

⁵ Как отмечалось в работе [6], семантический вероятностный вывод формализует обнаружение условных связей на уровне нейрона. Условные связи и их роль в процессе выработки условного рефлекса рассмотрены в [6].

⁶ Как отмечалось в работе [6] вероятностные закономерности формализуют выработку элементарных условных связей на уровне нейрона.

либо $m2 = m$ и $PG_i^2 = PG_i$, $i \neq j, i = 1, \dots, m$ и $PG_j^2 \neq PG_j$, $PG_j^2 \in G$ для некоторого j . Смысл первой части условия состоит в том, что уточнение правила может осуществляться путем добавления в правила достижение новых подцелей, уточняющих процесс достижения цели PG и строго увеличивающих вероятность её достижения. Это означает, что новые подцели приобрели сигнальное значение, и образовалась условная связь между ними и достижением цели. Проверка строгого увеличения вероятности достижения цели PG осуществляется проверкой требований вероятностной закономерности (см. п. 6.3). Смысл второй части условия состоит в том, что можно заменить достижение одной из подцелей $PG_j^2 \neq PG_j$ на новую подцель $PG_j \in G$, так, что достижение новой подцели делает вероятность достижения цели строго больше. Известно, что более вероятные условные связи быстрее срабатывают во времени, поэтому, если условная связь с менее эффективной подцелью выработана нейроном (другим или тем же), то всё равно, быстрее по времени сработает более вероятная условная связь и затормозит не успевшую сработать менее вероятную связь.

3. $\{A_1^2, \dots, A_{k2}^2\} \subseteq \{A_1, \dots, A_k\}$, $k2 \leq k$
либо $k2 = k$ и $A_i^2 = A_i$, $i \neq j, i = 1, \dots, k$ и $A_j^2 \neq A_j$, $A_j^2 \in A$ для некоторого j . Интерпретация этого условия аналогична предыдущему случаю.
4. При этом хотя бы одно из неравенств $n2 \leq n$, $m2 \leq m$ и $k2 \leq k$ должно быть строгим, либо должно выполняться одно из дополнительных условий для случаев $m2 = m$ или $k2 = k$.

Вероятностной закономерностью будем называть правило вида (1), удовлетворяющее условию: оценка условной вероятности правила определена

$$p(P_1 \& \dots \& P_n \& PG_1 \& \dots \& PG_m \& A_1 \& \dots \& A_k) > 0$$

и строго больше оценок условных вероятностей каждого из его подправил.

6.2. Алгоритм обнаружения вероятностных закономерностей. Опишем алгоритм обнаружения множества вероятностных закономерностей PR для функциональной системы ΦC .

Обозначим через $Spec(RUL)$ – множество уточнений всех правил из RUL , где RUL – произвольное множество правил вида (1). Пусть пользователем задано натуральное число d – глубина базового перебора.

- 1) На первом шаге генерируем множество RUL_1 всех правил единичной длины, имеющих один из следующих видов:

- 1) $PG_i \rightarrow PG$, $PG_i \in G$;
- 2) $A_i \rightarrow PG$, $A_i \in A$.

Все правила RUL_1 проходят проверку на выполнение условий вероятностных закономерностей (см. п. 6.3). Правила, прошедшие проверку, будут вероятностными закономерностями. Обозначим через

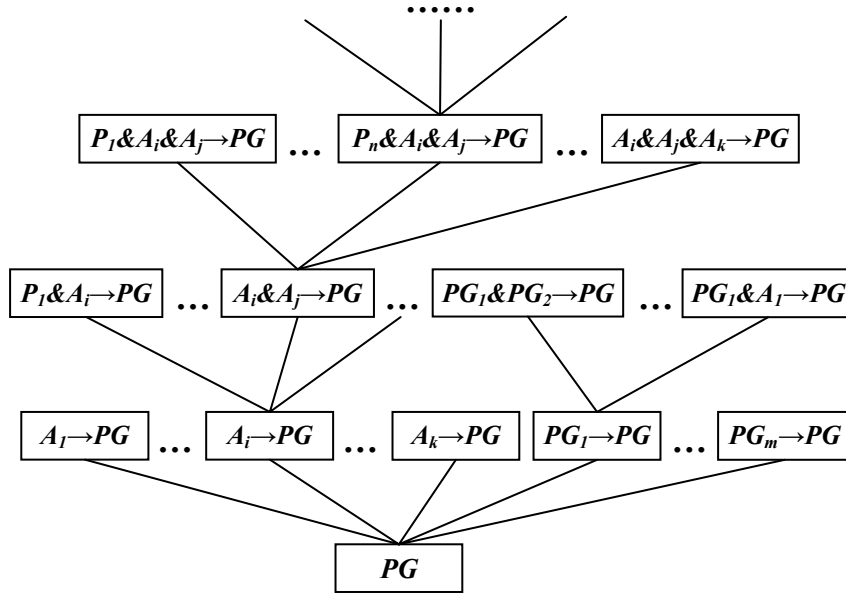


Рис. 5. Дерево семантического вероятностного вывода

REG_l множество всех вероятностных закономерностей, обнаруженных на первом шаге.

- 2) На шаге $k \leq d$ генерируется множество $RUL_k = Spec(RUL_{k-1})$ всех уточнений правил, сгенерированных на предыдущем шаге. Все правила из RUL_k проходят проверку на выполнение условий вероятностных закономерностей. Полученное множество вероятностных закономерностей обозначим через REG_k .
- 3) На шаге $l > d$ генерируется множество $RUL_l = Spec(REG_{l-1})$ уточнений всех вероятностных закономерностей, обнаруженных на предыдущем шаге. Все правила из RUL_l проходят проверку на выполнение условий вероятностных закономерностей. Обозначим через REG_l – множество вероятностных закономерностей, обнаруженных на шаге l . Если новых закономерностей не обнаружено $REG_l = \emptyset$, то алгоритм останавливается.
- 4) Результирующее множество закономерностей PR является объединением всех обнаруженных вероятностных закономерностей $PR = \bigcup_i REG_i$.

На рис. 5 представлено дерево вывода, соответствующее описанному процессу. Шаги алгоритма $k \leq d$ соответствуют базовому перебору, а шаги $k > d$ – дополнительному перебору.

6.3. Проверка условий вероятностных закономерностей. Для статистической проверки любого правила вида (1) на вероятностную закономерность нам достаточно иметь статистику $n(C)$ – число повторений каждого события, порожденного некоторыми из атомарных формул $P_1, \dots, P_n, P_\theta$ правила, которую можно извлечь из таблицы данных X .

Представим правило (1) для простоты в виде $P_1 \& \dots \& P_n \rightarrow PG$.

1. Проверка первого условия вероятностной закономерности. Надо проверить, что условная вероятность правила определена и $p(P_1 \& \dots \& P_n) > 0$. Для этого достаточно проверить, что $n(P_1 \& \dots \& P_n) > 0$. Из определения вероятности следует, что если $n(P_1 \& \dots \& P_n) > 0$, то вероятность не равна нулю.

2. Проверка второго условия вероятностной закономерности.

2а. Рассмотрим сначала правила вида $P_1 \rightarrow PG$, когда в посылке правила стоит только один предикат P_1 . Подправилom правила $P_1 \rightarrow PG$ является правило $\rightarrow PG$. Поэтому вероятность правила $\rightarrow PG$ (с пустой посылкой) должна быть строго меньше условной вероятности правила $P_1 \rightarrow PG$, т.е. $p(PG | P_1) > p(PG)$.

Последнее неравенство можно переписать в виде $p(PG \& P_1) > p(PG)p(P_1)$.

Для проверки этого неравенства сформулируем гипотезу H_0 о независимости предикатов PG и P_1

$$H_0 : p(PG \& P_1) = p(PG)p(P_1)$$

против альтернатив

$$H_1 : p(PG \& P_1) \neq p(PG)p(P_1).$$

Эта гипотеза является сложной с одним ограничением и двумя степенями свободы [11]. Если гипотеза H_0 верна, то предикатные символы PG и P_1 независимы и неравенство для условной вероятности не выполнено. Тогда правило $P_1 \rightarrow PG$ не является вероятностной закономерностью. Если гипотеза H_0 неверна, то верна одна из альтернативных гипотез H_1 , и тогда предикатные символы PG и P_1 зависимы между собой.

Гипотезу H_0 можно переформулировать следующим образом. Пусть числа $n(P_1)$ и $n(\neg P_1)$ фиксированы, а числа $n(P_1 \& PG)$ и $n(\neg P_1 \& PG)$ являются независимыми случайными величинами. Тогда гипотеза H_0 является гипотезой о равенстве вероятностей в двух совокупностях [11] против альтернатив H_1 .

Если гипотеза H_0 неверна, то верна одна из альтернативных гипотез H_1 , и либо $p(PG | P_1) > p(PG)$, либо $p(PG | P_1) < p(PG)$.

Если верно первое неравенство, то тестируемое правило $P_1 \rightarrow PG$ является вероятностной закономерностью. Если второе – то не является.

По соотношениям

$$n(PG \& P_1) > (n(PG)n(P_1)) / N,$$

$$n(PG \& P_1) < (n(PG)n(P_1)) / N,$$

где N – общее количество случаев, можно определить, какое из неравенств, первое или второе, имеет место.

Чтобы проверить гипотезу H_0 воспользуемся точным критерием независимости Фишера [11, с.739]. Этот критерий является равномерно наиболее мощным, несмещенным критерием как в случае проверки гипотезы о двумерной независимости, так и в случае проверки гипотезы о равенстве вероятностей в двух совокупностях [11, с.742]. Применяв этот критерий с некоторым доверительным уровнем α , мы получим, что либо гипотеза H_0 верна и, следовательно, значения истинности предикатов PG и P_1 независимы, и, значит, нет закономерности, либо H_0 неверна, и мы принимаем одну из гипотез H_1 . Когда гипотеза H_1 означает неравенство $p(PG | P_1) > p(PG)$, то тестируемое правило является вероятностной закономерностью с доверительным уровнем α .

26. Рассмотрим в общем случае произвольное правило $P_1 \& \dots \& P_n \rightarrow PG$. Сведем этот случай к предыдущему. Обозначим $C = \{P_1, \dots, P_n\}$, $D \subset C$ и $\&C = P_1 \& \dots \& P_n$, $\&D$ – конъюнкции атомов C и D .

Для проверки того, является ли правило $P_1 \& \dots \& P_n \rightarrow PG$ вероятностной закономерностью, надо проверить, выполняется ли для любого подмножества D (включая \emptyset) соотношение $p(PG | \&C) > p(PG | \&D)$.

Будем рассматривать конъюнкцию $\&D$ как формулу F_1 , а конъюнкцию литер из $C \setminus D$ как другую формулу F_2 . В случае, когда $D = \emptyset$, $F_1 = \text{true}$, а $p(PG | \&D) = p(PG)$. В результате получим неравенство

$$p(PG | F_1 \& F_2) > p(PG | F_1).$$

Так как

$$p(PG | F_1 \& F_2) = p(PG \& F_1 \& F_2) / p(F_1 \& F_2) = p(PG \& F_2 | F_1) / p(F_2 | F_1),$$

то предыдущее неравенство перейдет в неравенство

$$p(PG \& F_2 | F_1) > p(F_2 | F_1)p(PG | F_1).$$

Так как $n(P_1 \& \dots \& P_n) > 0$, то $p(\&C) > 0$, $p(F_1) > 0$, $p(F_2) > 0$ в силу включений $D \subset C$ и $C \setminus D \subset C$. Отсюда следует, что все проделанные преобразования корректны, т.к. ни одна вероятность в знаменателе не равна нулю.

Для проверки последнего неравенства таким же образом сформулируем гипотезу о независимости

$$H_0 : p(PG \& F_2 | F_1) = p(F_2 | F_1)p(PG | F_1)$$

против альтернатив

$$H_1 : p(PG \& F_2 | F_1) \neq p(F_2 | F_1)p(PG | F_1).$$

Ограничимся рассмотрением только тех событий, для которых формула F_1 истинна. На этих событиях определим вероятностную меру $p'(E) = p(E \& F_1) / p(F_1)$.

Тогда гипотезы H_0 и H_1 примут вид:

$$H_0 : p'(PG \& F_2) = p'(F_2)p'(PG),$$

$$H_1 : p'(PG \& F_2) \neq p'(F_2)p'(PG).$$

Гипотеза H_0 против альтернатив H_1 также проверяется с помощью критерия Фишера с некоторым доверительным уровнем α . Правило будет вероятностной закономерностью с доверительным уровнем α , если гипотеза H_0 отвергается с уровнем α для любого подмножества $D \subset C$ и принимается гипотеза H_1 с неравенством $>$.

7. Метод обнаружения подцелей

Изначально система управления аниматором имеет заданную априори иерархию функциональных систем. В простейшем случае она может состоять всего из одной функциональной системы. Однако для успешной работы система управления должна уметь автоматически выявлять новые подцели и порождать соответствующие функциональные системы. Рассмотрим способ автоматического формирования новых подцелей и функциональных систем.

Следуя работам [6, 8], определим подцель как ситуацию, достижение которой значительно увеличивает вероятность достижения вышестоящей цели, и последующие действия из этой ситуации не могут быть определены однозначно.

Покажем, как с помощью закономерностей можно автоматически выявлять подцели. Первое условие определения подцелей можно проверить, анализируя описания ситуаций, заданных в условиях закономерностей. Если некоторая ситуация увеличивает вероятность достижения цели, то её описание будет автоматически включено в условия закономерностей. Это следует из определения алгоритма семантического вероятностного вывода, согласно которому любой предикат, увеличивающий вероятность прогноза, автоматически добавляется в условие правила. Определить, насколько данная ситуация значима для увеличения вероятности достижения цели, также можно при помощи закономерностей, путем сравнения вероятностей закономерностей, содержащих в своих условиях данную ситуацию, с вероятностями закономерностей, не содержащими эту ситуацию.

Проверка второго условия определения подцели состоит в том, что, если из данной ситуации можно выполнить несколько различных серий действий, приводящих к достижению цели, то, эти серии действий можно обнаружить в условиях различных закономерностей.

Таким образом, метод, позволяющий автоматически формировать новые подцели и функциональные системы, заключается в следующем. Для выяв-

ления подцелей анализируется множество правил PR функциональной системы $\Phi C = \langle PG, G, PR \rangle$. Перебираются различные ситуации, входящие в условия закономерностей из PR .

Ситуация, описываемая предикатом $PG_{New} = P_1 \& \dots \& P_k$, $P_i \in P$, будет являться подцелью G_{New} , если выполнены следующие условия:

1. для любого правила

$$R_1 = P_1^{(1)} \& \dots \& P_{n1}^{(1)} \& \dots \& A_1^{(1)} \& \dots \& A_{m1}^{(1)} \rightarrow PG, R_1 \in PR,$$

такого что $\{P_1, \dots, P_k\} \subseteq \{P_1^{(1)}, \dots, P_{n1}^{(1)}\}$, и для любого правила

$$R_2 = P_1^{(2)} \& \dots \& P_{n2}^{(2)} \& \dots \& A_2^{(2)} \& \dots \& A_{m2}^{(2)} \rightarrow PG, R_2 \in PR$$

такого что $\{P_1^{(2)}, \dots, P_{n2}^{(2)}\} \subset \{P_1^{(1)}, \dots, P_{n1}^{(1)}\}$ и $\{P_1, \dots, P_k\} \not\subseteq \{P_1^{(2)}, \dots, P_{n2}^{(2)}\}$ выполнено условие $p(R_1) - p(R_2) > \delta$;

2. существуют правила

$$R_1 = P_1^{(1)} \& \dots \& P_{n1}^{(1)} \& \dots \& A_1^{(1)} \& \dots \& A_{m1}^{(1)} \rightarrow PG, R_1 \in PR, \text{ и}$$

$$R_2 = P_1^{(2)} \& \dots \& P_{n2}^{(2)} \& \dots \& A_2^{(2)} \& \dots \& A_{m2}^{(2)} \rightarrow PG, R_2 \in PR,$$

такие что $\{P_1, \dots, P_k\} \subseteq \{P_1^{(1)}, \dots, P_{n1}^{(1)}\}$, $\{P_1, \dots, P_k\} \subseteq \{P_1^{(2)}, \dots, P_{n2}^{(2)}\}$ и $A_1^{(1)} \neq A_1^{(2)}$.

Первое условие говорит о том, что добавление данной ситуации в условную часть правил должно значительно увеличивать оценку условной вероятности правил (более чем на δ , где δ – некоторый порог, например $\delta = 0.2$), это означает, что достижение такой ситуации значительно увеличивает вероятность достижения вышестоящей цели. Второе условие говорит о том, что после достижения данной ситуации возможны различные дальнейшие действия.

Таким образом, для каждой функциональной системы ΦC анализируются правила PR и выявляются новые подцели. Для каждой новой обнаруженной подцели G_{New} создается новая функциональная система ΦC_{New} , находящаяся ниже по иерархии системы ΦC и реализующая достижение этой подцели. Для созданной функциональной системы ΦC_{New} при помощи семантического вероятностного вывода порождается множество закономерностей PR . Для этого просматривается все множество данных истории анимата X и выявляются случаи, когда анимат в прошлом достигал подцель G_{New} . Для всех функциональных систем, находящихся на уровень выше ΦC_{New} , набор предикатов-целей обогащается еще одним предикатом PG_{New} и генерируются новые правила. Тем самым, множества закономерностей этих функциональных систем обогащаются закономерностями, содержащими новую подцель G_{New} .

8. Эксперименты

В предыдущих параграфах была описана модель адаптивной системы управления, основанная на теории функциональных систем П.К. Анохина. Для того чтобы экспериментально проверить работоспособность предложенной модели мы реализовали описанную выше систему управления и оценили ее спо-

способности к обучению и автоматическому выявлению подцелей на примере управления некоторым простейшим аниматом. В качестве тестовых задач для анимата нами была использована классическая задача фуражирования и ее усложненный вариант.

Суть задачи заключается в том, что агент, двигаясь по плоскости, должен научиться эффективно собирать пищевые объекты. Недостатком данной задачи является то, что она одноэтапная, т.е. в ней не могут быть выделены подцели. Согласно данному выше определению, подцель должна удовлетворять следующим условиям: во-первых, достижение подцели должно увеличивать вероятность достижения вышестоящей цели, во-вторых, цель должна обладать свойством ветвления: если подцель достигнута, то дальнейшие действия не могут быть определены однозначно. Очевидно, что в данной задаче не могут быть выделены подцели, удовлетворяющие этому определению. Поэтому классический вариант задачи фуражирования нельзя считать тестовым для алгоритма обнаружения подцелей.

В связи с этим мы усложнили классическую задачу, сделав ее двухэтапной. Для этого мы добавили в задачу еще один объект, который мы условно назвали «таблетка». Теперь, чтобы съесть еду анимат должен иметь при себе таблетку, которую он должен предварительно найти и подобрать. Когда он съест еду, таблетка исчезнет и, чтобы съесть следующую еду, он должен опять найти и подобрать таблетку, и т.д. В данной задаче можно выделить очевидную подцель – найти таблетку. Таким образом, при помощи данной задачи можно протестировать способность автоматического обнаружения подцелей.

Для исследования предложенной системы управления нами было поставлено два эксперимента по решению классической одноэтапной задачи фуражирования и ее усложненного двухэтапного варианта. При помощи компьютерной программы был смоделирован виртуальный мир и анимат, основной целью которого в обоих экспериментах являлось собирание специальных объектов виртуального мира – «еды». Анимат должен был научиться эффективно находить и собирать еду.

Мир анимата представляет собой прямоугольное поле, разбитое на клетки, и содержит следующие объекты: пустые клетки («трава»), препятствия («препятствие»), и еду («еда»). Объекты «препятствие» располагаются только по периметру виртуального мира, образуя тем самым его естественные границы. Анимат может перемещаться по полю, совершая три типа действий: шаг на клетку вперед («шаг»), поворот налево («налево»), поворот направо («направо»).

В первом эксперименте по полю случайным образом располагается некоторое количество еды. Чтобы собрать еду, анимату достаточно шагнуть на клетку, содержащую еду. Когда анимат шагает на клетку с едой, считается, что он ее «поедает», клетка, на которой находилась еда, очищается, и новый объект «еда» случайным образом появляется в другом месте поля. Таким образом, количество еды в виртуальном мире всегда остается постоянным.

Для ориентации в виртуальном мире анимат имеет девять сенсоров, восемь из которых расположены по окружности анимата: «впереди-слева», «впереди», «впереди-справа», «слева», «справа», «сзади-слева», «сзади», «сзади-справа» и один – в центре: «центр» (рис. 6). Каждый сенсор информирует ани-

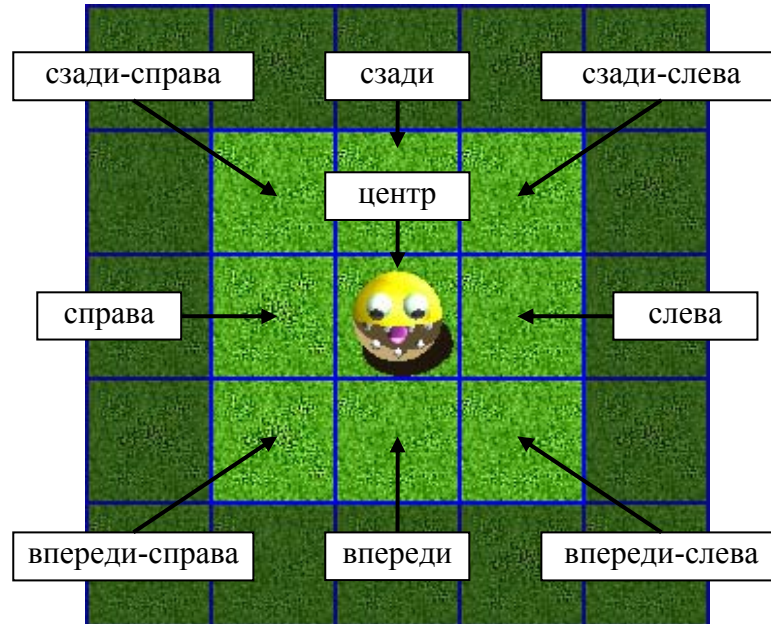


Рис. 6. Сенсорное поле анимата

мат о типе объекта, находящегося в соответствующей клетке, и может принимать следующие значения: «трава», «препятствие» или «еда».

Второй эксперимент является усложнением первого. В этом эксперименте виртуальный мир содержит еще один объект, который мы условно назовем «таблетка». Таблетки, как и еда, случайным образом располагаются по полю. Чтобы съесть еду, анимат должен иметь при себе таблетку, которую он должен предварительно найти и подобрать. Когда он съест еду, таблетка исчезнет и, чтобы съесть следующую еду, он должен опять найти и подобрать таблетку, и т.д. Подбор таблетки происходит так же, как и поедание еды: анимату достаточно шагнуть на клетку, содержащую таблетку. Однако если анимат уже имеет одну таблетку, то пока он не использует ее для поедания еды, он больше не сможет подобрать ни одной таблетки. Когда анимат подбирает таблетку, то клетка, на которой она находилась, очищается, и новая таблетка случайным образом появляется в другом месте поля, т.е. количество таблеток в виртуальном мире также всегда поддерживается постоянным.

Во втором эксперименте анимат имеет десять сенсоров, девять из которых расположены аналогично описанным выше сенсорам из первого эксперимента и принимают значения «трава», «препятствие», «еда» или «таблетка». Еще один сенсор «есть таблетка» информирует анимат о наличии таблетки и принимает значения «да» или «нет».

Для того чтобы оценить эффективность предлагаемой нами системы управления, в экспериментах также проводилось тестовое сравнение с система-

ми, построенными на основании теории обучения с подкреплением (Reinforcement Learning), описанной в работах Р. Саттона и Э. Барто [24].

Для сравнения мы выбрали две системы управления, построенные на основе популярного алгоритма обучения с подкреплением Q-Learning. Суть алгоритма заключается в последовательном уточнении оценок суммарной величины награды $Q(s_t, A_t)$, которую получит система, если в ситуации s_t она выполнит действие A_t , по формуле:

$$Q^{(i+1)}(s_t, A_t) = Q^{(i)}(s_t, A_t) + \alpha(r_t + \gamma \max_A Q^{(i)}(s_{t+1}, A) - Q^{(i)}(s_t, A_t)).$$

Первая из этих двух систем (Q-Lookup Table) основана на использовании таблицы, которая содержит Q-значения для всех возможных ситуаций и действий. Изначально эти значения таблицы заполняются случайным образом. В процессе работы в каждый такт времени система совершает действие и уточняет соответствующие Q-значения.

Вторая система (Q-Neural Net) использует аппроксимацию функции $Q(s_t, A_t)$ при помощи нейронных сетей. При этом для каждого возможного действия A_t используется своя нейронная сеть NN_t . В каждый такт времени система выбирает то действие, чья нейронная сеть выдаст наибольшую оценку Q-значения, после чего действие совершается и происходит адаптация весов соответствующей нейронной сети.

Тестовое сравнение проводилось на поле размером 25 на 25 клеток. Весь период функционирования анимата был разбит на этапы по 1000 шагов (тактов). Оценивалось, какое количество еды соберет анимат с разными системами управления за каждый этап работы. Очевидно, что после того, как система управления полностью обучится и достигнет своего оптимального поведения, анимат начнет собирать примерно одно и то же количество еды за один этап. Таким образом, можно оценить как эффективность каждой системы управления в целом, так и скорость ее обучения.

Результаты первого эксперимента. В первом эксперименте набор предикатов анимата состоит из двадцати семи сенсорных предикатов – по три на каждый сенсор s : (s = «трава»), (s = «препятствие») и (s = «еда»).

Изначально система управления аниматом имеет только одну функциональную систему, целью которой является ощущение наличия еды центральным сенсором, соответствующий предикат-цель имеет вид PG_0 («центр» = «еда»). Когда анимат достигает этой цели, то считается, что он «поедает» еду.

В данном эксперименте системой управления аниматом не было обнаружено никаких подцелей, что вполне естественно, поскольку в этой задаче не существует ситуаций, которые бы удовлетворяли данному выше определению подцели. Основной задачей данного эксперимента являлась оценка эффективности работы отдельной функциональной системы.

На рис. 7 представлены результаты тестового сравнения. Для каждой системы управления рассчитывались средние значения по результатам 20-ти испытаний. Продолжительность каждого испытания составляла 50 000 шагов. Количество еды на поле поддерживалось постоянным и равным 100.

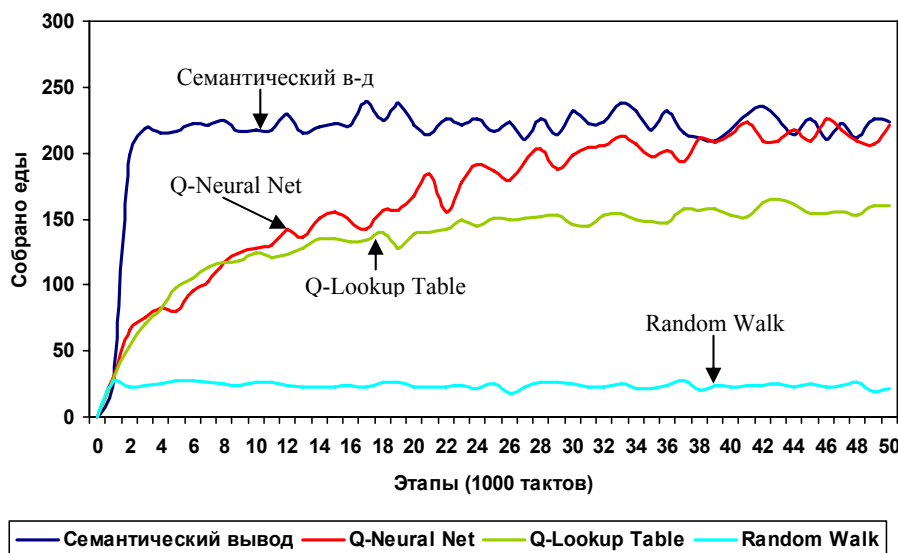


Рис. 7. Количество «еды», собранное аниматором с разными системами управления в первом компьютерном эксперименте

Из графика видно, что система управления на основе семантического вероятностного вывода уже через 1000 шагов полностью обучается и достигает своего оптимального поведения, т.е. начинает адекватно реагировать на все показания сенсоров.

Система управления на основе нейронных сетей (Q-Neural Net) обучается медленнее и достигает оптимального поведения примерно через 10 000 шагов.

Плохая работа системы управления с использованием таблицы Q-значений объясняется достаточно большим количеством возможных ситуаций: с учетом трех действий возможно 2496 различных ситуаций. Результаты экспериментов показывают, что даже после 50 000 тактов работы система управления в среднем просматривает только около 1450 ситуаций. Таким образом, даже после длительного времени обучения возникают ситуации, когда система реагирует неадекватно.

Результаты данного эксперимента показывают, что семантический вероятностный вывод позволяет отдельной функциональной системе работать достаточно эффективно, и, по крайней мере, не хуже, чем на основании подходов Reinforcement Learning.

Результаты второго эксперимента. Данный эксперимент принципиально отличается от предыдущего тем, что задача может быть разбита на два этапа: сначала необходимо найти таблетку, затем – еду. Поэтому одной из основных задач данного эксперимента была демонстрация возможности автоматического формирования иерархии целей и результатов при целенаправленном поведении.

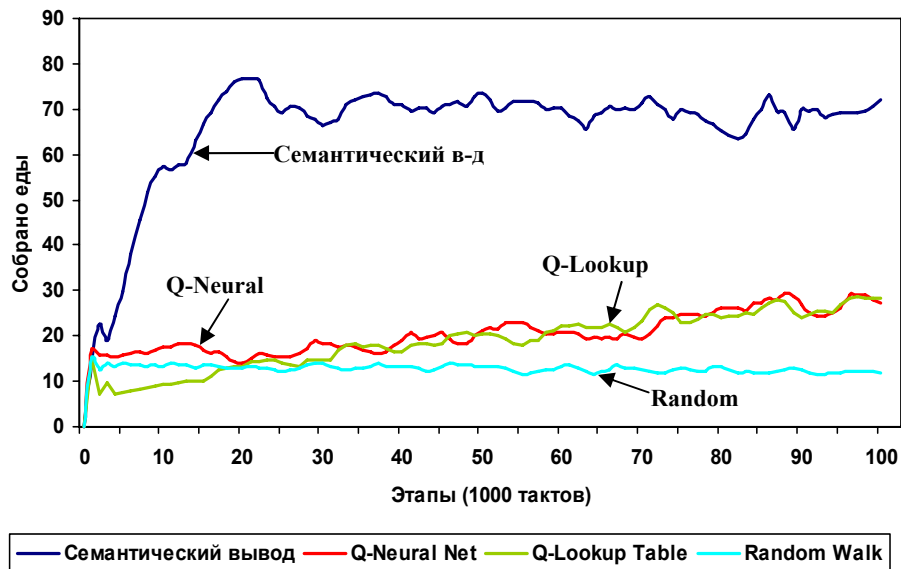


Рис. 8. Количество «еды», собранное аниматором с разными системами управления во втором компьютерном эксперименте

Изначальный набор предикатов анимата состоит из тридцати семи сенсорных предикатов: по четыре предиката на каждый сенсор s , информирующий анимат о состоянии окружающих его клеток: (s = «трава»), (s = «препятствие»), (s = «еда»), (s = «таблетка»), и один предикат, говорящий о наличии таблетки: («есть таблетка» = «да»).

При старте система управления аниматором состоит только из одной базовой функциональной системы, целью которой является достижение ситуации одновременного наличия таблетки и ощущения еды центральным сенсором, соответствующий предикат-цель имеет вид PG_0 = («центр» = «еда» И «есть таблетка» = «да»). Когда анимат достигает этой цели, он «поедает» еду.

В ходе эксперимента система управления аниматором при каждом тестовом запуске стабильно обнаруживала новую подцель, описываемую предикатом-целью PG_i = («есть таблетка» = «да»), и создавала для нее соответствующую функциональную систему. Работа системы управления происходила следующим образом. При отсутствии у анимата таблетки срабатывала закономерность $PG_i \rightarrow PG_0$ как наиболее вероятная в данной ситуации. Она передавала управление нижестоящей функциональной системе, реализующей поиск таблетки. Когда таблетка найдена, у базовой функциональной системы начинали срабатывать правила с более высокой оценкой вероятности, в результате чего она находила еду.

Результаты эксперимента представлены на рис. 8. На графике для каждой системы управления представлены средние значения по результатам 20-ти

испытаний. Для каждого испытания анимату выделялось 100 000 шагов, за это время анимат должен был научиться эффективно решать поставленную задачу. Количество таблеток и еды на поле поддерживалось постоянным: по 100 объектов каждого типа.

Как видно на графике, система управления на основе семантического вероятностного вывода превосходит системы Reinforcement Learning как по скорости обучения, так и по качеству работы.

Системы управления на основе Reinforcement Learning в данном эксперименте показали плохую обучаемость и нестабильную работу. Основная проблема в работе этих систем была связана с тем, что они не могли за приемлемое время научиться стабильно адекватно реагировать на показания сенсоров о наличии таблеток и зачастую проходили мимо таблеток даже после 100 000 шагов обучения.

Дополнительные эксперименты показали, что система управления на основе нейронных сетей (Q-Neural Net) при увеличении длительности обучения до 300 000 – 500 000 шагов в некоторых случаях способна обучиться правильно реагировать на все показания сенсоров. Однако, по нашему мнению, столь длительный срок обучения является неприемлемым для адаптивной системы.

Система управления на основе использования таблицы Q-значений не смогла достичь оптимального поведения даже после 500 000 шагов. Во многом это связано с большим количеством возможных ситуаций: в данной задаче анимат может столкнуться с 137 538 различными ситуациями.

Таким образом, результаты эксперимента показывают, что в условиях усложнения среды умение формировать подцели и достигать их является принципиальным для эффективного достижения конечных целей. Несмотря на то, что в данной модели адаптивной системы управления используется достаточно простой способ формирования подцелей, уже эта возможность дает значительные преимущества в обучении. Как видно из эксперимента, использование иерархии функциональных систем и алгоритма выявления подцелей позволяет предлагаемой нами системе управления эффективно обучаться и решать поставленную задачу. Существующие подходы, основанные на нейронных сетях и Reinforcement Learning, не могут автоматически выявлять подцели и поэтому значительно проигрывают в усложненных экспериментах.

ЛИТЕРАТУРА

1. Анохин П.К. Принципиальные вопросы общей теории функциональных систем // Принципы системной организации функций. – М.: Наука, 1973. – С. 5-61.
2. Анохин П.К. Проблема принятия решения в психологии и физиологии // Проблемы принятия решения. – М.: Наука, 1976. – С. 7-16.
3. Анохин П.К. Принципиальные вопросы теории функциональных систем // Философские аспекты теории функциональных систем. – М.: Наука, 1978. – С. 49-106.
4. Анохин П.К. Опережающее отражение действительности // Философские аспекты теории функциональных систем. – М.: Наука, 1978. – С. 7-27.
5. Анохин П.К. Роль ориентировочно-исследовательской реакции в образовании условного рефлекса // Системные механизмы высшей нервной деятельности: Избр. тр. – М.: Наука, 1979. – С. 338-352.

6. **Витяев Е.Е.** Принципы работы мозга, содержащиеся в теории функциональных систем П.К. Анохина и теории эмоций П.В. Симонова // *Нейроинформатика*, 2008, том 3, № 1, стр. 25-78
7. **Витяев Е.Е.** Формальная модель работы мозга, основанная на принципе предсказания // *Модели когнитивных процессов*. – Новосибирск, 1998. – Вып. 164: Вычислительные системы. – С. 3-61.
8. **Витяев Е.Е.** Объяснение теории движений Н.А.Бернштейна // VII Всероссийская научно-техническая конференция «Нейроинформатика-2005». Сборник научных трудов. – М.: МИФИ, 2005. – Ч.1. – С. 234-240.
9. **Витяев Е.Е.** Извлечение знаний из данных. Компьютерное познание. Модели когнитивных процессов. – Новосибирск: НГУ, 2006. – 293 с.
10. **Витяев Е.Е.** Семантический подход к созданию баз знаний. Семантический вероятностный вывод наилучших для предсказания ПРОЛОГ-программ по вероятностной модели данных // *Логика и семантическое программирование*. – Новосибирск, 1992. – Вып. 146: Вычислительные системы. – С. 19-49.
11. **Кендал М., Стюарт А.** Статистические выводы и связи. – М.: Наука, 1973. – 899 с.
12. **Редько В.Г.** От моделей поведения к искусственному интеллекту // Серия «Науки об искусственном» / под ред. Редько В.Г. – М.: УРСС, 2006. – 456 с.
13. **Судаков К.В.** Общая теория функциональных систем. – М.: Медицина, 1984. – 222 с.
14. **Barto A., Mahadevan S.** Recent advances in hierarchical reinforcement learning // *Discrete event dynamic systems: theory and applications*. – Kluwer Academic Publishers, 2003. – pp. 41-77.
15. **Brooks R. A.** A robust layered control system for a mobile robot // *IEEE Transactions on Robotics and Automation*. – 1986. – V. 2. – pp. 14-23.
16. **Brooks R.A.** Intelligence without representation // *Artificial Intelligence*. – 1991. – V. 47. – pp.139-159.
17. **Brooks, R. A.** A robot that walks: emergent behavior from a carefully evolved network // *Neural Computation*. – 1989. – V. 1(2). – pp. 253-262.
18. **Donnart J. Y., Meyer J. A.** A hierarchical classifier system implementing a motivationally autonomous animat // *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB94)*. – Cambridge, MA: MIT Press/Bradford Books, 1994. – pp. 144-153.
19. **Holland, J.H., Holyoak K.J., Nisbett R.E., Thagard P.** Induction: Processes of Inference, Learning, and Discovery. – Cambridge, MA: MIT Press, 1986. – p. 416.
20. **Maes Pattie.** Modeling adaptive autonomous agents // *Artificial Life*. – 1994. – Vol. 1. – pp. 135-162.
21. **Sharkey N. E., Heemskerk J. N. H., Neary J.** Training artificial neural networks for robot control // *Solving Engineering Problems with Neural Networks* / Eds. by Bulsari A., Kallio S., Tsaptsinos D. – 1996. – pp. 190-196.
22. **Steels L.** The artificial life roots of artificial intelligence // *Artificial Life Journal*. – 1994. – V.1. – pp. 75-100.
23. **Steinhage A., Bergener T.** Dynamical systems for the behavioral organization of an anthropomorphic mobile robot // *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*. – Cambridge, MA: MIT Press, 1998. – pp. 147-152.
24. **Sutton R., Barto A.** Reinforcement Learning: An Introduction. – Cambridge: MIT Press, 1998. – p. 342.
25. **Wilson S.W.** The animat path to AI // *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. – Cambridge, MA: MIT Press, 1991. – pp. 15-21.
26. **Ziemke Tom.** Adaptive behavior in autonomous agents // *Presence: Teleoperators and Virtual Environments*. – Cambridge: MIT Press. – 1998. – Vol. 7(6). – pp. 564-587.