КОГНИТИВНЫЕ СИСТЕМЫ И АГЕНТЫ

Л. А. Станкевич[1], С. В. Серебряков[2]

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178

[1]<stankevich_lev@inbox.ru>, [2]<sergey_s@iias.spb.su>

УДК 681.3

Станкевич Л. А., Серебряков С. В. **Когнитивные системы и агенты** // Труды СПИИРАН. Вып. 3, т. 1. — СПб.: Наука, 2006.

Аннотация. *Обсуждаются принципы построения когнитивных систем, уровень развития которых в настоящее время характеризует современное состояние информатики и искусственного интеллекта. Описываются методы представления, обработки и формирования когнитивных функций, отношений и процессов. Рассматриваются нейрологические средства для их реализации. Обсуждаются вопросы организации когнитивных агентов и многоагентных систем. Приводятся примеры применения когнитивных агентов и систем в играх и системах управления антропоморфных роботов.* — Библ. 8 назв.

UDC 681.3

Stankevich L. A., Serebryakov S. V. **Cognitive systems and agents** // SPIIRAS Proceedings. Issue 3, vol. 1. — SPb.: Nauka, 2006.

Abstract. *Principles of building cognitive systems that define modern state of informatics and artificial intelligence are formed and discussed. Methods of representation, processing, and forming cognitive functions, relations, and processes are described. Neurological means for their implementation are described. Issues of organization of cognitive agents and systems are discussed. Applications of the cognitive agents and systems in games and control systems of anthropomorphic robots are shown.* — Bibl. 8 items.

1. Введение

Развитие искусственного интеллекта привело к формированию нового подхода, который предполагает создание интеллектуальных обучаемых систем на основе раскрытых в последнее время нейрофизиологических принципов построения нервной системы и когнитивных методов познавательной и мыслительной деятельности человека. Такой подход предполагает развитие нового технического направления когнитивных систем.

Когнитивная система может быть определена как система, которая способна познавать свое окружение и адаптироваться к нему или изменять его за счет накопленных в процессе функционирования знаний и приобретенных навыков [1]. В настоящее время предполагается ([2]), что научное направление, связанное с построением когнитивных систем, будет определять развитие информатики и искусственного интеллекта и способствовать развитию передовых компьютерных технологий в ближайшем будущем [2].

Теоретические основы когнитивных систем составляют когнитивные методы, которые объединяют методы познания, т.е. восприятия и накопления информации, и мышления, т.е. использования этой информации при «рассудительном» решении задач [3]. В основе этих методов лежат понятия когнитивных функций, отношений и процессов.

Современные исследования по нейрофизиологии [4] позволили выявить пути разработки базовых модулей и структур для обработки информации в когнитивных системах. В данной статье предлагается использовать для этих целей нейрологические модули, основанных на принципах обучения в нейронных сетях, а также методы грануляции и ассоциативно-логической обработки ин-

формации. Для некоторых специфичных задач предлагается также использовать более эффективные обучаемые модули, основанные на адаптивных триангуляционных аппроксиматорах.

Когнитивные системы позволяют реализовать сложные поведенческие функции, подобные тем, которые реализует нервная система человека. Поэтому применение когнитивных систем может принести ощутимый эффект, например, при управлении современными роботами или в системах поддержки принятия решений при работе в сложных неопределенных средах (информационных сетях, военных и промышленных комплексах и пр.). В данной статье кратко рассмотрены два успешных применения когнитивных систем: для моделирования поведения игроков-футболистов и для управления роботами гуманоидного класса.

2. Когнитивные методы обработки информации

Обработка информации в когнитивных системах основана на методах грануляции, ассоциативно-логической обработки когнитивных функций, отношений и процессов, а также методах автоматического формирования таких функций, отношений и процессов.

Грануляция. Необходимость грануляции связана с неопределенностью информации в реальных условиях. Грануляция рассматривается как одна из базисных концепций когнитивной обработки информации. Предполагается, что любой составной информационный объект (переменная, отображение, образ) может быть декомпозирован на гранулы. Каждая гранула является набором элементарных объектов, которые связаны вместе неопределенностью, близостью, подобностью и функциональностью. Формально объект Q_c может быть представлен гранулировано, т.е.:

$$Q_c = ins\_g(G_1,...,G_i,...,G_N);$$
$$G_i = has\_a(A_1,...,A_j,...,G_M);$$
$$A_j = has\_v(V_1,...,V_q,...,V_Q),$$

где A_j — j-й атрибут гранулы G_i; V_q — q-е значение атрибута A_j; ins_g — отношение включения для гранул; has_a и has_v — отношение «имеет» для атрибута и значения соответственно. Пример грануляции: объект «голова» включает гранулы: «нос», «глаз», «щека» и пр., имеющие атрибуты: «размер», «форма» и пр. со значениями, например, для первого атрибута: «большой», «малый» и пр. Гранулы могут быть точными (интервалы переменных, выделенные области определения функций и отношений, сегменты образов) и неточными (терм-множества переменных, элементы нечетких или вероятностных графов, нечеткие или вероятностные правила и др.).

Ассоциативно-логическая обработка информации. Обработка гранулированной информации основана на использовании ассоциативных отображений (AM). Отображение (AM) является особой формой отображения функции или отношения в виде сетевой вычислительной структуры, которая должна обладать хорошими адаптивными и аппроксимирующими свойствами. При отображении преобразований образов или процессов возможно использование одиночных (SAM) и множественных (MAM) отображений. Формально (AM), (SAM) и (MAM) могут быть представлены модельными наборами вида:

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$AM = \{X, Y, S, U, F\};$$
$$SAM = \{P_x, P_y, AM\};$$
$$MAM = \{P_x, P_y, \{AM_t, t = 1,...,T\}\}$$

Компоненты, входящие в наборы, имеют следующие назначения: X, Y — множества входных и выходных параметров (векторные); S, U — множества структур и структурных единиц, их составляющих; F — множество базисных функций, реализуемых в узловых элементах; P_x, P_y — входной и выходной векторы; AM_t — SAM для t-го момента времени протекания когнитивного процесса; T — период времени процесса с фиксированными моментами t, в которые реализуются AM_t, составляющие MAM.

Ассоциативно-логическая обработка функций и отношений производится с использованием базовых модулей — клеток и ядер, а процессов — локальных и распределенных сетей, составленных из клеток и ядер.

Клетка является минимальным обучаемым элементом, способным самостоятельно обрабатывать информацию в гранулированном виде. Формализованная информационная модель клетки может быть представлена набором множеств

$$M_C = \{X, W_C, H_X, S_C, H_y, BF, y\},$$

где $X = \{x_0,...,x_n\}$ — множество входных параметров; $W_C = \{w_0,...,w_v\}$ — множество регулируемых весов $(m \geq n)$; $H_X = \{h_{x1},...,h_{xq}\}$ — множество скрытых входных параметров, соответствующих информационным гранулам на входах; $H_y = \{h_{y1},...,h_{yl}\}$ — множество скрытых выходных параметров, соответствующих информационным гранулам на выходах; $S_C = \{s_{c1},...,s_{cr}\}$ — множество связей скрытых входных и выходных информационных гранул, определяющих цепочку преобразований гранул при активизации этой связи; $BF = \{bf_1,...,bf_k\}$ — множество базисных функций активаторных элементов клетки; y_c — выходной параметр клетки.

Такая информационная модель поддерживается структурно-функциональной моделью клетки, представленной на рис. 1. Здесь отображены: информационный гранулятор, который формирует множество H_X; активатор, который состоит из множества активаторных элементов из набора BF, выполняющих преобразования скрытых информационных параметров и формирующий множества H_y в соответствии со связями S_C и весами W_C; информационный дегранулятор, который формирует выходной параметр y; настройщик, который формирует множество связей S_C и весов W_C при настройке клетки на отображение $X \to y$, аппроксимирующее функцию $y = F(X)$.

Клетка является универсальным преобразователем информации (адаптивным аппроксиматором), имеющим n входов и один выход. Она соответствует в модельном плане биологическому нейрону с его сетью синапсов, через которые организуются связи с другими нейронами, а также медиаторными регулирующими средствами.

Ядро содержит «склеенные» клетки. Информационная модель ядра может быть формально представлена набором множеств

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$M_N = \{X_N, W_N, H_{NX}, S_N, H_{NY}, BF, Y\}$$

где $X_N = (X_1 \cup X_2 \cup ... \cup X_M)$ — представляет собой «склеенное» множество входов клеток, входящих в ядро; W_N, H_{NX}, H_{NY} — множества весов и скрытых параметров ядра, S_N — множество связей ядра, объединяющее связи клеток, BF — набор базисных функций активаторных элементов, одинаковый для всех клеток; Y — вектор выходных параметров, объединяющий все выходные параметры клеток, формирующих ядро.

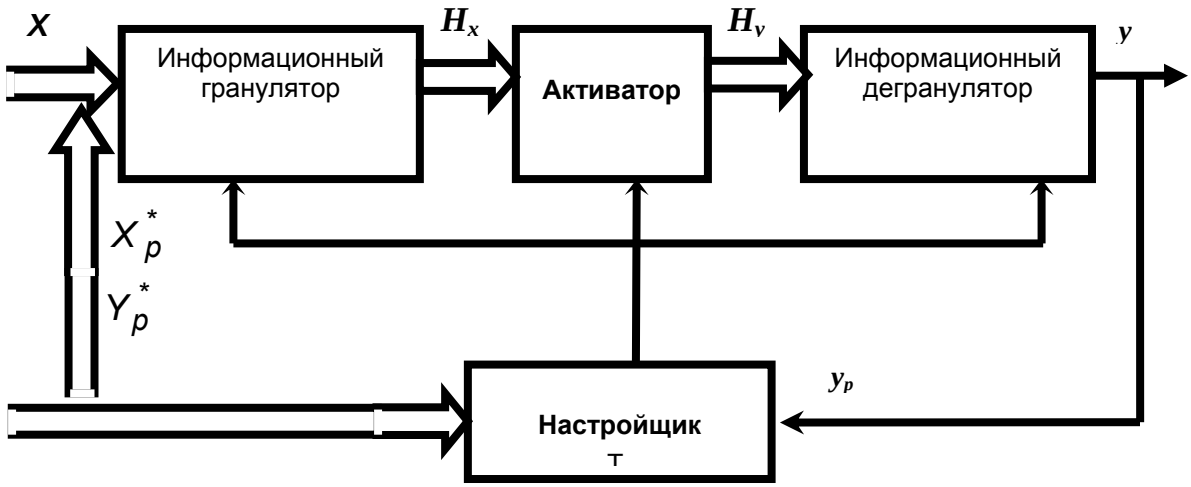

Рис. 1. Структурная модель клетки.

В какой-то мере ядро с частичным объединением клеток можно считать однослойной сетью клеток с входами, параллельно подведенными к каждой клетке (от каждого входа к каждой клетке). Такое представление ядра используется в нейроинформатике для конструирования слоистых и модульных нейросетей. В данной работе оно использовано для моделирования биологических ядер, реализующих поведенческие функции (отношения) с несколькими связанными параметрами, зависящими от ряда общих аргументов.

Клетки и ядра являются базовыми нейрологическими модулями, из которых предлагается строить когнитивные системы. Главными информационными элементами, которыми оперирует когнитивная система, являются образы. Образ формируется как матрица или вектор со связанными координатами. Преобразование образов является основным операционным компонентом при реализации когнитивных процессов, как последовательностей таких преобразований. Преобразования выполняются в структурных компонентах, таких как локальные или распределенные сети.

Локальная сеть содержит связанные клетки и ядра. Локальная сеть рассматривается как самостоятельный модуль обработки образов. Информационная модель локальной сети может быть представлена набором множеств

$$M_{LN} = \{P_X(X, C_X), W_{LN}, S_{LN} P_Y(Y, C_Y)\},$$

где X и C_X — вектор образа P_X и связи между его координатами, т.е. множество входных параметров и связи между ними, определяющие входной (преобразуемый сетью) образ; W_{Ln} — множество регулируемых весов связей между клетками и ядрами, входящими в локальную сеть; S_{LN} — множество связей клеток и ядер, входящих в локальную сеть; Y и C_Y — вектор образа P_Y и свя-

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

зи между его координатами, т.е. множество выходных параметров и связей между ними, определяющих выходной (преобразованный сетью) образ.

Структурно локальная сеть состоит из связанных клеток и ядер, а также обучателя, формирующего множество весов связей W_{LN} элементов сети. Наличие разноуровневых компонентов (клеток и ядер) в локальной сети позволяет строить достаточно сложные преобразующие структуры с полносвязной и слоистой организацией. Локальные сети с полносвязной организацией используются для ассоциативного запоминания образов. Локальные сети со слоистой организацией предназначены для классификации образов или аппроксимации заданных преобразований образов.

Распределенные сети имеют кустовую организацию. Они включают несколько локальных сетей (кустов), обрабатывающих образы, между которыми имеют место слабые (одинарные) связи. Так могут реализоваться процессы, определяющие сложное поведение системы. Распределенные сети могут преобразовывать образы замкнуто: образы, проходящие через кусты, возвращаются преобразованные опять на первый куст. Процесс преобразования образов в распределенной сети определяется цепочками отображений типа:

$$P_{X/t-1} \to F_{K1}(P_{X/t-1}, P_{Z1}) \to P_{X/t} \to F_{K2}(P_{X/t}, P_{Z2}) \to P_{Y/t+1},$$

где $t-1, t, t+1$ — моменты времени, соответствующие смежным циклам преобразований; F_{K1}, F_{K2} — преобразования образов в первом и втором кустах; $P_{X/t-1}, P_{X/t}, P_{Y/t+1}, P_{Z1}, P_{Z2}$ — входной, промежуточный, выходной и внешние образы распределенной сети. При этом имеют место слабые связи между кустами, определяемые малыми весами $w_{i,i+1}$. Такая организация сетей позволяет учесть влияние на процесс управления дополнительных факторов, изменяющих это управление через вводимые в кусты внешние образы.

Формирование функций, отношений, преобразований образов и процессов. В когнитивных системах функции и отношения в форме ассоциативных отображений должны формироваться путем обучения. Для разных вариантов обучаемых модулей, построенных для работы с функциями и отношениями, могут быть использованы различные методы обучения и самообучения. Наилучшие результаты на практике дает применение модулей с локальным обучением. Такие модули быстро обучаются, но расходуют много памяти. В плане расхода памяти более выгодно использование глобального обучения нейронного типа, поскольку здесь требуется значительно меньше памяти. Однако время обучения в этом случае может быть значительно больше.

Представление преобразований образов и процессов в виде цепочек функций и отношений также предполагает обучение. Часто здесь может использоваться обучение с подкреплением при взаимодействии со средой [5]. В этом случае производится обучение тому, что делать, т.е. как отобразить ситуацию в действие, чтобы максимизировать числовой сигнал поощрения от среды. Обучение производится по эпизодам процесса, которые оцениваются специальной функцией поощрения лучших эпизодов.

Топология структур когнитивной системы с обучаемыми компонентами, может быть синтезирована путем самосборки и обучения в соответствии с предварительно определенными целями управления в известной среде. Главным принципом формирования топологии является *самоорганизация*.

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

3. Компоненты когнитивных систем

Однослойная нейронная сеть CMAC[1]. В основе сети лежит модель мозжечка, разработанная Альбусом Дж., 1975. Основным назначением сети CMAC является запоминание, восстановление и интерполяция функций от нескольких переменных. Структура сети представлена на рис. 2.

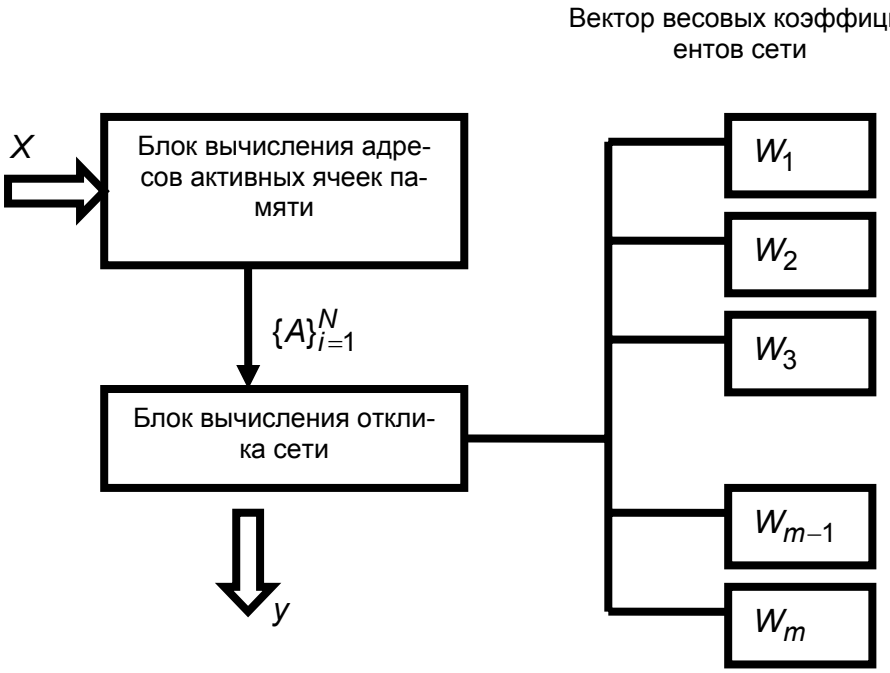

Рис. 2. Структура нейронной сети CMAC.

Сеть функционирует следующим образом. Входной вектор X активирует некоторое множество весовых коэффициентов сети $\{W_{A_i}\}_{i=1}^{N}$, где $\{A_i\}_{i=1}^{N}$ множество номеров активированных ячеек, $\{W_j\}_{j=1}^{m}$ — множество весовых коэффициентов сети, m — общее количество весовых коэффициентов. Мощность N такого множества активных ячеек памяти называется обобщающим параметром сети, и определяет разрешающую способность сети и требуемое количество ячеек памяти. После этого вычисляется отклик сети y как сумма значений активных весовых коэффициентов:

$$Y = \sum_{i=1}^{N} W_{A_i^I}$$

Рассмотрим один из вариантов реализации алгоритма определения множества номеров активируемых ячеек памяти.

Для любого i-го элемента входного вектора X, $i = \overline{1,L}$, L — размерность входного вектора, можно получить так называемый его порядковый номер. Если

[1]CMAC — Cerebellar Model Articulation Controller (суставная модель мозжечкового регулятора ([9])).

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

область определения i-го элемента вектора является интервал $[x^i_{\min}, x^i_{\max}]$, то выбрав шаг квантования

$$\Delta x^i = \frac{x^i_{\max} - x^i_{\min}}{K^i},$$

где K^i — параметр дискретизации, определяющий, на сколько интервалов разбивается исходный интервал (область определения), можно найти порядковый номер как:

$$n_i = \left[\frac{x^i}{\Delta x^i}\right] + 1,\ n_i \in [1, K^i]$$

где [] — оператор взятия целой части числа.

В результате такого преобразования исходный входной вектор преобразуется в вектор, элементы которого представляют собой целые числа. После этого на основании этого вектора и, например, алгоритмов, реализующих псевдослучайное хеширование, можно сформировать множество весовых коэффициентов, соответствующих данному входу.

Обучение нейронной сети происходит следующим образом. Сети подается на вход вектор X. Определяются активные весовые коэффициенты $\{A\}_{i=1}^N$ и отклик сети y. После этого происходит обновление активных коэффициентов по следующей формуле:

$$W_{A_i} = W_{A_i} + \frac{\alpha}{N} * (y^* - y), i = \overline{1,N}$$

где y^* — эталонный отклик на вход X, α — шаг обучения, N — обобщающий параметр

Рассмотренный модуль является простым в реализации и не требующим значительных временных ресурсов для обучения, что позволяет его использовать в задачах реального времени.

Нейрологический модуль с нечетким базисом. Разработан вариант такого модуля использованием нечетко-логического базиса [3]. Модуль реализует нечетко-нейронное отображение входного n-мерного вектора X, в выходной m-мерный вектор Y. Обрабатывающая часть включает в качестве гранулятора фазификатор (процедура Fuz), активатор, состоящий из блока ассоциативного вывода, реализующий процедуру взвешенной агрегации и объединения (процедура Inference), и памяти весов, и в качестве дегранулятора — дефаззификатор (процедура Dfuz). Обучающая часть включает два фаззификатора, блок вычисления текущих весов, реализующий процедуру коррекции весов (процедура W-update), и память образов. Перечисленные процедуры имеют следующее содержание:

Fuz(x_i, μ^q_{xi}) — фазификация, вычисляющая степень принадлежности переменной x_i к q-й нечеткой грануле этой переменной;

Inference$(Ln, w_i, \mu^q_{xi}; i = 1,...,I, q = 1,...,Q)$ — взвешенная агрегация, вычисляющая каждую степень принадлежности переменной y_j к q-й нечеткой грануле этой переменной путем агрегирования степеней принадлежности входных переменных x_i с дальнейшим умножением результата на вес w_i и объедине-

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

нием; I, Q — количество агрегируемых степеней принадлежности и нечетких гранул соответственно; Ln — номер слоя сети, в котором выполняется агрегация;

Dfuz($\mu_{yj}^{q}, y_j; j = 1,...,M, q = 1,...,Q$) — дефазификация, вычисляющая значение выходного параметра y_j, исходя из степеней принадлежности значения этого параметра к нечетким гранулам шкалы параметра;

W-update($w_p, \mu_{yj}^{*}, \mu_{yj}, p$) — коррекция веса при обучении; w_p — текущий вес; μ_{yj}^{q}, μ_{yj} — степени принадлежности правильного и текущего значений выходного параметра y к нечеткой грануле соответственно (правильное значение параметра берется из примера для обучения); p — номер примера; в результате операции находится веса при обработке $p+1$ примера.

Такой модуль является относительно простым для программной и аппаратной реализации и может обучаться в реальном времени.

Адаптивный триангуляционный модуль. Разработан также обучаемый модуль на основе барицентрического аппроксиматора [6]. В нем используется класс линейных функций, известных как барицентрические интерполяторы и специальный метод адаптивной многомерной триангуляции, оперирующий с классами симплексов, позволяющими эффективно определять барицентрические координаты принадлежащих им точек.

Значение любой непрерывной функции $f(x)$ в заданной точке x, лежащей внутри выбранного симплекса с вершинами $x^{(0)},...,x^{(n)} \in \mathbb{R}^n$ аппроксимируется линейной комбинацией значений функции в этих вершинах:

$$f(x) = \sum_{i=0}^{n} \lambda_{x^{(i)}}(x) f\left(x^{(i)}\right), \quad x \in T_{\left(x^{(0)},...,x^{(n)}\right)},$$

где $\lambda_{x^{(i)}}(x)$ — барицентрические координаты x, определяемые как положительные коэффициенты, однозначно задаваемые системой уравнений:

$$\begin{cases} \sum_{k=0}^{n} \lambda_{x^{(k)}} = 1 \\ \sum_{k=0}^{n} \lambda_{x^{(k)}} x^{(k)} = y \end{cases}.$$

В общем случае вычисление барицентрических координат для произвольного симплекса является достаточно трудоемкой задачей, поскольку оно требует обращения матрицы симплекса.

В связи с этим был разработан новый метод многомерной триангуляции, основанный на комбинации триангуляции Куна и периодическом методе бисекции, заключающемся в рассечении самого длинного ребра симплекса, не содержащего вершин, созданных в течение текущего периода. При использовании данного метода триангуляции возникает не более n классов конгруэнтности, следовательно, метод устойчив. Любой симплекс, полученный в результате такой триангуляции, с помощью простых аффинных преобразований масштабирования и инверсии порядка и значений координат может быть приведен

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

к базовой форме, которой соответствует треугольная единичная матрица, обращение которой не составляет никаких проблем.

В результате применения предложенной триангуляции все пространство состояний может быть дискретизовано на сетке переменного разрешения структурированной в виде дерева симплексов.

Предложен простой алгоритм, имеющий сложность $O(N)$, где N — средняя глубина дискретизации. Алгоритм позволяет определить симплекс, которому принадлежит произвольная точка $x \in \mathbb{R}^n$ и, осуществив преобразование координат этой точки в пространство базового симплекса, найти барицентрические координаты $\lambda_{x^{(i)}}(x)$, $1 \le i \le n+1$.

Модуль, построенный на основе такого барицентрического аппроксиматора, может быть настроен на отображение когнитивной функции или отношения путем супервизорного обучения или обучения с подкреплением.

При супервизорном обучении для обновления значений функции в вершинах $f\left(x^{(i)}\right)$ и степеней доверия этим значениям $p_{x^{(i)}}$ на базе примеров, задающих $f(x)$ используется следующая итерационная схема:

$$f_{n+1}\left(x^{(i)}\right) = f_n\left(x^{(i)}\right) + \left(f(x) - f_n\left(x^{(i)}\right)\right)\frac{\lambda_{x^{(i)}}(x)}{p_{n+1}\left(x^{(i)}\right)}$$

$$p_{n+1}{}_{x^{(i)}} = p_n{}_{x^{(i)}} + \lambda_{x^{(i)}}(x).$$

Алгоритм обучения с подкреплением строится на основе методов динамического программирования. Решается задача максимизации значения функционала подкрепления путем выбора оптимального управления $u^*(t)$, т.е.

$$J\left(x;u(t)\right) = \int_0^\tau \gamma^t r\left(x(t),u(t)\right)dt + \gamma^\tau R\left(x(\tau)\right),$$

Где $r(x,u)$ — текущее подкрепление, $r_b(x)$ — граничное подкрепление, γ — коэффициент $0 \le \gamma < 1$.

В процессе решения этой задачи вводиться целевая функция:

$$V(x) = \sup_{u(t)} J\left(x;u(t)\right).$$

Путем аппроксимации уравнения Гамильтона-Якоби-Белмана с помощью конечно-разностной схемы получаем следующее выражение для целевой функции:

$$V^\Sigma(x) = \sup_{u\in U}\left[\gamma^{\tau(x,u)}\sum_{j=0}^{n}\lambda_{x^{(j)}}\left(\eta(x,u)\right)V^\Sigma\left(x^{(j)}\right) + \tau(x,u)r(x,u)\right],$$

где $\eta(x,u)$ — проекция x в направлении, параллельном $f(x,u)$ на противоположную грань симплекса, а $\tau(x,u)$ такого, что:

$$\eta(x,u) = x + \tau(x,u)f(x,u).$$

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Вводятся Q-значения:

$$Q^{\Sigma}(x,u) = \gamma^{\tau(x,u)} V^{\Sigma}\left(\eta(x,u)\right) + \tau(x,u) r(x,u),$$

$$Q^{\Sigma}(x,u) = R(x), \text{при } x \notin X.$$

Для итеративного решения этого уравнения необходимо знание $\eta(x,u)$ и $\tau(x,u)$. Они могут быть определены путем взаимодействия с объектом на основе построенных траекторий изменения его состояния, в зависимости от приложенного управления. Разработан безмодельный подход решения с использованием теоремы Талеса для аппроксимации $\tau(x,u) f(x,u)$ на основе точки входа траектории в симплекс x_1 и точки выхода из симплекса x_2:

$$\tau(x,u) f(x,u) = \frac{x_2 - x_1}{\lambda_x(x_1)}.$$

На основе предыдущих выражений строится итерационная схема вычисления $Q^{\Sigma}(x,u)$:

$$Q^{\Sigma}_{n+1}(x,u) = \gamma^{\frac{\tau_x}{\lambda_x(x_1)}} \left(\frac{V^{\Sigma}(x_2) - V^{\Sigma}(x_2)}{\lambda_x(x_1)} + V^{\Sigma}(x) \right) + \frac{\tau_x}{\lambda_x(x_1)} r(x,u)$$

Здесь τ_x обозначает время движения по траектории от точки x_1 до точки x_2.

Для обеспечения сходимости такая динамика обучения комбинируется со структурной динамикой, заключающейся в уточнении симплексов с нерегулярной целевой функцией.

4. Примеры применения когнитивных систем и агентов

Когнитивные системы могут решать сложные задачи обработки информации и управления. Например, такие системы могут быть эффективны при моделировании поведения или управлении роботами, работающими во взаимодействии с человеком или другими роботами. При этом целесообразно использовать концепцию *когнитивных агентов*, т.е. когнитивных систем, способных взаимодействовать между собой при решении общих задач. На базе концепции агентов могут строиться многоагентные системы управления.

Рассмотрим примеры разработанных с участием авторов когнитивных агентов для игровых приложений и управления антропоморфными роботами.

Когнитивные агенты-футболисты. Примерами когнитивных агентов, построенных с использованием нейрологических модулей, являются *агенты-футболисты*, разработанные для игровой среды футбола роботов (RoboCup Soccer). Агенты-футболисты имеют трехуровневую организацию поведения (навыки, индивидуальное поведение и координация в команде) [7]. Все уровня поведения реализуются с использованием соответствующих баз знаний и средств логического вывода. Нижний реактивный уровень агента реализуют с помощью многослойной структуры простые исполняемые модули поведения. Средний когнитивный уровень генерирует текущее поведение в соответствии с индивидуальным намерением агента. Верхний когнитивный уровень обеспечивает координацию с целью согласования намерений взаимодействующих аген-

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

тов и выработки их общего намерения, которое используется для коррекции индивидуального намерения агента.

Введение на каждом уровне поведения соответствующих баз знаний и средств вывода позволило значительно усложнить поведение агентов. Однако реализовать принятие решений в реальном времени оказалось возможным только путем использования нейрологических модулей, в которые путем обучения были сформированы знания в ассоциативной форме. Процесс создания ассоциативных баз знаний осуществлялся в режиме off-line при настройке агента. После этого агент был способен принимать решения в реальном времени, поскольку использовался ассоциативно-логический вывод. Добавочные средства on-line обучения использовались для оперативного накопления правил правильного поведения, которых не было в базах знаний.

Поведение агента-футболиста определялось набором функций. Главная функция поведения агента-футболиста SBF (Soccer Behavior Function) включала когнитивные (селекция поведений) и актуаторные (управление движениями игрока) функции. Дополнительно использовались функции принятия решения и ряд специальных функций. Первые обеспечивали взаимодействия игроков, обучение поведению, определение позиции партнеров по команде и противников, принятие решений об атаке или обороне. Специальные функции управляли обменом сообщениями, назначением ролей и формаций агентов, обучением с тренером или в игре.

Команда STEP (Soccer Team of ElectroPult), использующая агентов-футболистов, построенных на базе рассмотренной архитектуры, успешно участвовала на Чемпионате Мира RoboCup-2004, проведенном в Лиссабоне (Португалия) в июле 2004-го года, где стала Чемпионом Мира в Симуляционной Лиге футбола роботов.

Рассмотрим систему, основанную на обучении с подкреплением, позволяющую агентам команды STEP обучаться поведению в игре.

Задача обучения с подкреплением — это задача обучения через взаимодействие с окружающей средой с целью достижения некоторой цели. Под термином *окружающей среды* понимается все то, что окружает агента, и с чем он взаимодействует.

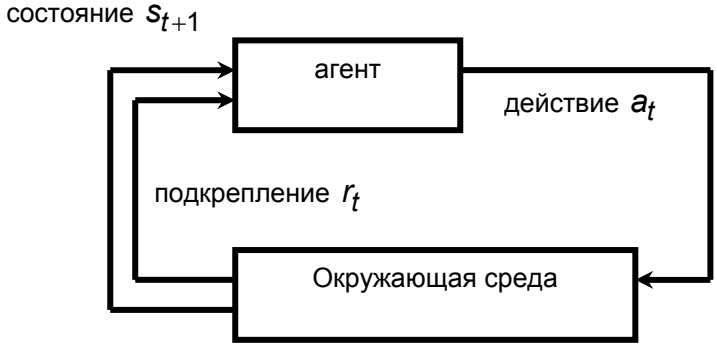

Рис. 3. Взаимодействие агента с окружающей средой.

Это взаимодействие осуществляется непрерывно — агент выбирает и выполняет действия, а среда, реагируя на эти действия в соответствии с определенными правилами, представляет агенту новые ситуации.

Если говорить более точно, то агент и окружающая среда взаимодействуют друг с другом в дискретные моменты времени $t_k, k = 0,1,2,3,\ldots$. В каждый мо-

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

мент времени, k, агент получает некую информацию об окружающей среде $s_k \in S$, где S — множество всевозможных состояний, и на этом основании выбирает действие $a_k \in A(s_k)$, где $A(s_k)$ — множество возможных действий в состоянии s_k. В следующий момент времени t_{k+1} агент оказывается в новом состоянии s_{k+1} и получает или сам вычисляет подкрепление $r_{k+1} \in R$. В каждый момент времени агент осуществляет отображение текущего состояния в вектор, содержащий числовые значения, характеризующие каждое из возможных действий. Действие с максимальным значением считается оптимальным на данном этапе обучения. Такое отображение называется политикой (иногда тактикой) агента, и обозначается как π. Методы обучения с подкреплением указывают, каким образом агент изменяет свою политику в соответствии с опытом. Грубо говоря, цель агента действовать таким образом, чтобы максимизировать суммарное значение подкреплений (наград), которое агент может получить взаимодействуя с окружающей средой в процессе обучения.

Одним из самых распространенных алгоритмов обучения с плдкреплением является алгоритм Sarsa(λ). Рассмотрим подробно версию алгоритма, которая была разработана и использована для обучения поведению игры в сценарных заготовках при игре в виртуальный футбол в среде футбольного сервера RoboCup Soccer Serve*r 2D*.

При описании алгоритма используются следующие обозначения и понятия:

Эпизод — обучение происходит путем проигрывания эпизодических ситуаций достаточно большое количество раз. Эпизод характеризуется начальным состоянием и флагом завершения. Завершение может быть успешным (выполнены некоторые условия, например, был забит гол), или не успешным, например, противники перехватили мяч, или, не был забит гол в отведенное на это время. Остановку обучения производят на основании информации, которая специфична для каждой задачи обучения. Например, критерием остановки может служить отношение успешно сыгранных эпизодов к общему числу проигранных эпизодов, например, учитывая только N последних эпизодов. Если это значение больше определенного порога, то можно останавливать обучение.

CurrentStep — текущий шаг эпизода.

StepLimit — флаг, установленный в случае, если эпизод имеет временные ограничения, например, если задачей является забивание гола нападающими. Если же, например, задачей является удержание мяча, то временных ограничений не существует.

MAX _ STEPS — в случае существования временных ограничений, этот параметр содержит максимальное допустимое значение шагов в эпизоде.

LastAction — предыдущее действие агента. $LastAction \in [1, NumActions]$, где *NumActions* — количество возможных действий агента. Например, возможными действиями агента могут являться ведение мяча ($LastAction = 1$), пас ($LastAction = 2$) и удар по воротам($LastAction = 3$).

NewAction — текущее действие агента, т.е. действие, которое агент собирается выполнить в текущий момент времени.

NumberOfPlayedEpisodes — количество уже сыгранных эпизодов.

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

reward — подкрепление, $reward \in \mathbb{R}$, параметр характеризует последнее действие.

success _ reward — подкрепление в случае успешного завершения эпизода.

failure _ reward — подкрепление в случае неудачного завершения эпизода.

s — текущая ситуация, $s \in S$, где S — множество всех возможных состояний).Число всех возможных состояний может быть как конечно так и бесконечно.

a — действие агента, $a \in A(s)$, в большинстве случаев $A(s) \equiv A$ — множество всех возможных действий агента.

$Q(s,a)$ — функция, определяющая значимость действия a в ситуации s. В реализации алгоритма используется нейронная сеть CMAC , описанная выше, для аппроксимации этой функции (так как сеть CMAC может иметь произвольное число входов и только один выход, то фактически количество используемых сетей равняется количеству возможных действий агента), соответственно, следующие параметры относятся к этой функции:

$\{A\}_{i=1}^{N}$ — множество активных весовых коэффициентов для заданного входа, т.е. состояния s.

W_j — j-ый весовой коэффициент сети.

$\vec{W}$ — вектор всех весовых коэффициентов

α —шаг обучения

$\vec{Q}_a$ — вектор числовых значений, характеризующих действия. Размерность вектора равняется количеству действий. Элементы вычисляются на основании функции $Q(s,a)$.

Q_j — j-ый элемент вектора $\vec{Q}_a$.

ε — вероятность выбора агентом не оптимального действия.

$\vec{e}$ —функция, характеризующая частоту появления того или иного состояния, а также выбора действия. Функция реализуется n векторами, где n — количество доступных действий. Размерность каждого вектора равняется размерности вектора весовых коэффициентов сети CMAC, используемой для аппроксимации функции ценности $Q(s,a)$. Использование этой функции улучшает свойства сходимости рассматриваемого алгоритма.

λ —шаг обновления функции $e(s,a)$

На рис. 4 и рис. 5 представлен алгоритм обучения на основе метода Sarsa(λ). Рассмотрим более подробно представленный алгоритм. Начнем с функции BeginEpisode(). В строке 1 происходит инициализация переменной *CurrentStep*. Этот параметр хранит текущую длительность эпизода. Для некоторых задач терминальным состоянием является израсходования лимита времени, например, удержание мяча, которая уже была упомянута – критерием остановки обучения, например, может быть удержание мяча в течении определенного количества симуляционных тактов. Также этот параметр служит в тех случаях, когда израсходование лимита времени является терминальным — провальным — состоянием. В строке 2 происходит инициализация вектора, хранящего частоты появлений состояний и выборов тех или иных действий.

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

BeginEpisode ()

1 $CurrentStep \leftarrow 1$

2 $\vec{e} = 0$

3 $\{A\}_{i=1}^{N} \leftarrow$ множество активных полей для нового (текущего) состояния s

4 $\vec{Q}_a \leftarrow \{\sum_{i \in A} W_{A_i}\}_j, j = \overline{1,n}$, n - количество возможных действий

5 $LastAction \leftarrow \begin{cases} \arg\max\limits_a \vec{Q}_a & \text{с вероятностью 1-}\varepsilon \\ random(\arg\vec{Q}_a) & \text{с вероятностью } \varepsilon \end{cases}$

6 $Q_{LastAction} \leftarrow \vec{Q}_{LastAction}$

DoEpisodeStep ()

7 $CurrentStep \leftarrow CurrentStep + 1$

8 $If(\ StepLimit\ \&\&\ CurrentStep > MAX\_STEPS\ )\ return\ false$

9 $\{A\}_{i=1}^{N} \leftarrow$ множество активных полей для предыдущего состояния s

10 для всех вариантов действий a

11 для всех A_i

12 если $a == LastAction$

13 $e_{A_i} = 1$

14 в противном случае

15 $e_{A_i} = 0$

16 $\{A\}_{i=1}^{N} \leftarrow$ множество активных полей для нового (текущего) состояния s'

17 $\vec{Q}_a \leftarrow \{\sum_{i \in A} W_{A_i}\}_j, j = \overline{1,n}$, n - количество возможных действий

18 $NewAction \leftarrow \begin{cases} \arg\max\limits_a \vec{Q}_a & \text{с вероятностью 1-}\varepsilon \\ random(\arg\vec{Q}_a) & \text{с вероятностью } \varepsilon \end{cases}$

19 $Q_{NewAction} \leftarrow \vec{Q}_{NewAction}$

20 $\delta \leftarrow reward + \gamma Q_{NewAction} - Q_{LastAction}$

21 $\vec{W} \leftarrow \vec{W} + \frac{\alpha\delta\vec{e}}{N}$, N — обобщающий параметр сети CMAC

22 $Q_{LastAction} = Q_{NewAction}$

23 $\vec{e} \leftarrow \lambda\vec{e}$

Рис. 4. Алгоритм Sarsa(λ) (Начало).

В строке 3–4 алгоритма символически показано отображение входного вектора в активные весовые поля сети CMAC (в текущей версии используется хеширование с разрешением коллизий (открытая адресация с линейным перебором)). Переменной A_i , $i \in \overline{1,N}$, соответствует адрес i-ого активного весового коэффициента.

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

EndEpisode (bool success)

24 $NumberOfPlayedEpisodes \leftarrow NumberOfPlayedEpisodes + 1$

25 $\{A\}_{i=1}^{N} \leftarrow$ множество активных полей для предыдущего состояния s

26 для всех вариантов действий a

27 для всех $A_i \in A, i = \overline{1,N}$

28 если $a == LastAction$

29 $e_{A_i} = 1$

30 в противном случае

31 $e_{A_i} = 0$

32 если $success == true$

33 $reward \leftarrow success\_reward$

34 в противном случае

35 $reward \leftarrow failure\_reward$

36 $\delta \leftarrow reward - Q_{LastAction}$

37 $\vec{W} \leftarrow \vec{W} + \frac{\alpha\delta\vec{e}}{N}$, N — обобщающий параметр сети CMAC

38. $SaveParameters()$

Рис. 5. Алгоритм Sarsa(λ) (Продолжение).

$\vec{W}$ — функция ценности, вектор всех рецепторных полей сети CMAC. В векторе A содержаться адреса активных весовых коэффициентов. Суммируя значения соответствующих активных полей получаем отклик сети — текущее значение ценности для состояния и действия. Вектор $\vec{Q}_a$ имеет размерность n, где n — количество действий. Этот вектор содержит значения ценностей соответствующих действий. Этот вектор является базовым для выбора действий.

В пятой строчке определяется действие, которое будет выбрано. Обычно выбирается действие с максимальным значением (первая подстрока). Параметр ε обычно выбирают из диапазона $(0, 0.05]$. Но иногда выбирается иное, отличное от оптимального, действие (т.е. происходит так называемое исследование, в отличие от первого варианта выбора действия — использование). В строке 6 происходит запоминание параметра ценности для выбранного действия. Это значение будет использоваться в дальнейшем для обновления функции ценности.

Перейдем к следующей функции — DoEpisodeStep(), которая вызывается для обработки шага эпизода. В строке 7 происходит увеличения параметра длительности эпизода на единицу, а в строке 8 проверяется, израсходован ли лимит времени. Если параметр *StepLimit*, говорящий о том, что необходимо останавливать обучение в случае израсходования лимита времени, установлен и лимит превышен, то функция сигнализирует о том, что необходимо начать новый эпизод. В строках 10–15 происходит обновление функции частот e. В 16 строчке происходит определение множества активных рецепторных полей для текущего состояния. В следующих стоках происходит обновление функции цен-

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ности. Параметр step_reward определяет подкрепление за каждый шаг эпизода. В большинстве случаев, этот параметр может задаваться в самом начале процесса обучения, и не меняться. Значение параметра определяется исходя из критерия. Например, если критерий — минимизация времени, то значение этого параметра может быть равно -0.001, а если максимизация, то 0.001. В разработанном модуле, реализующий данный алгоритм, существует возможность задавать подкрепление для каждого шага. В 23 строчке происходит обновление функции частот.

Алгоритм функции EndEpisode(...) аналогичен тому, который был представлен для предыдущей функции. Отличие состоит в том, что при обновлении функции ценности значение параметра $Q_{newAction}$ равняется всегда нулю, а также в том, что используются другие значения наград соответственно для успешного и провального завершения эпизода. В 38 строке происходит сохранение результатов проигранного эпизода в файл для последующего анализа процесса обучения.

Система управления гуманоидного робота. Обучаемые модули были использованы в системе управления первого в России антропоморфного робота АРНЕ (Антропоморфный Робот предприятия Новая ЭРА). Накопленный опыт используется при разработке перспективной системы управления роботом гуманоидного класса, имеющим подобную антропоморфную конструкцию. Систему, названную искусственной нервной системой робота, предполагается строить на основе когнитивных принципов, многоагентной технологии и специально разработанных обучаемых модулей [8].

Основную когнитивную часть системы составляет набор когнитивных агентов, каждый из которых обеспечивает свой набор поведений. Совместная работа агентов позволяет реализовать рациональное поведение робота в текущей ситуации. Выделены когнитивные агенты, реализующие: (1) акустическое взаимодействие с объектами среды (восприятие звуков и голосовое общение с людьми или другими роботами); (2) зрительно-акустическое взаимодействие с объектами среды; (3) целенаправленные манипуляции и перемещения среди объектов среды; (4) координированное поведение при работе в группе роботов; (5) обучение поведению.

Разработана технология для реализации когнитивных агентов на базе рассмотренных обучаемых модулей нейрологического и триангуляционного типа. Такие модули собираются в когнитивные структуры, которые могут быть обучены выбору поведений в соответствии с текущими ситуациями и целями. Разработаны также специальные средства самоорганизации, которые обеспечивают автоматическое конфигурирование и настройку всей системы на решение задач управления конкретным вариантом робота. Они включают модель системы, исходно конкретизированную под текущие цели системы, и средства ее реконфигурации под новые цели. Модель системы должна содержать накопленную ко времени создания системы генетическую информацию о конфигурации системы, среде, поведенческих функциях и процессах, которые она должна реализовать. Эта информация используется в процессе самоорганизации всей системы (самосборки, самообучения и пр.)

Когнитивные агенты в процессе функционирования используют Модель мира, настроенную на среду, Модель поведения, настроенную на поведенческие функции и процессы. Модель мира содержит текущую и прогнозируемую статическую и динамическую информацию о самом роботе, объектах среды и их отношениях. Модель поведения содержит набор поведений и отношений для

Труды СПИИРАН. 2006. Вып. 3. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

их выбора в текущей ситуации. Эта информация необходима для выбора индивидуального и коллективного поведения робота.

Исполнение выбранных поведений, связанное с координированным управлением движениями органов робота, реализуется с использованием Модели робота, настроенной на актуаторные функции и процессы.

5. Заключение

Анализ развития интеллектуальных методов и систем показывает необходимость разработки технических когнитивных систем. Именно на этом пути могут быть созданы будущие интеллектуальные системы с нервно-системной организацией структуры, функций и поведения. Рассмотренные принципы и обучаемые средства могут быть основой таких систем. Их применение в агентах-футболистах показало высокую эффективность при организации сложного поведения. Предполагается, что применение когнитивных принципов в сочетании с многоагентной технологией и разработанными обучаемыми модулями может обеспечить создание искусственной нервной системы современного гуманоидного робота.

Литература

1. *Станкевич Л. А.* Когнитивные нейрологические системы управления // Проблемы нейрокибернетики. Материалы XII Международной конференции по нейрокибернетике (Ростов-на-Дону, октябрь 1999). Ростов-на-Дону, 1999. С. 123–130
2. *Гергей Т.* Когнитивные системы — потребность информационного общества и вызов компьютерным наукам // Девятая Национальная конференция по искусственному интеллекту с международным участием КИИ-2004 (28 сентября – 2 октября 2004, Тверь): Труды конференции в 3-х томах. Т. 1. М.: Физматлит, 2004. С. 3–10.
3. *Станкевич Л. А.* Многоагентные когнитивные нейрологические системы управления // IV Всероссийская конференция «Нейрокомпьютеры и их применение», НКП2000 (Москва, 16–18 февраля 2000). М., 2000. С. 95–103.
4. *Шеперд Г.* Нейробиология. Том 1. М.: Мир, 1987. 297 с.
5. *Barto A., Sutton R.* Reinforcement Learning: An Introduction. Boston: The MIT Press, 1998. 296 с.
6. *Тимакин Д. Л.* Многоагентные когнитивные системы управления динамическими объектами со сложным поведением: Дис. канд. техн. наук. СПб., 2002. 187 с. (СПбГПУ).
7. *Stankevich L. A.* Cognitive Agent for Soccer Game // Proceeding of the First Workshop of Central and Eastern Europe on Multi-Agent Systems CEEMAS'99, St. Petersburg, Russia, 1999. С. 154–160.
8. *Станкевич Л. А.* Искусственная нервная система гуманоидного робота // Труды Юбилейной международной конференции по нейрокибернетике (Ростов-на-Дону, 25–28 сентября 2002). Ростов-на-Дону, 2002. С. 45–52.
9. *Комарцова Л. Г., Максимов А. В.* Нейрокомпьютеры. М.: Изд-во МГТУ имени Баумана, 2002. С. 165–169.