Украины им. Я. Мудрого (г. Харьков) на кафедре информатики и вычислительной техники и может быть рекомендован в качестве альтернативного формата хранения и передачи звуковой информации.

Литература: 1. *Иванов В.Г., Ломоносов Ю.В.* Алгоритм сжатия данных на основе вычислений точек перегиба в структуре сигнала // Вестник ХГПУ. Системный анализ, управление и информационные технологии. 2000. № 94. С. 25-29. 2. *Мюррей Д., Ван Райпер У.* Энциклопедия форматов графических файлов: Пер. с англ. К.: Издательская группа BHV, 1997. 672 с. 3. *Воробьев В.И., Грибунин В.Г.* Теория и практика вейвлет-преобразования. СПб.:Изд-во ВУС, 1999. 208 с.

Поступила в редколлегию 14.06.2000

Рецензент: д-р техн. наук, проф. Куценко А.С.

Иванов Владимир Георгиевич, канд. техн.наук, доцент, заведующий кафедрой информатики и вычислительной техники Национальной юридической академии Украины имени Я.Мудрого. Научные интересы: кодирование и распознавание сигналов различной физической природы. Адрес: Украина, 61024, Харьков, ул. Пушкинская, 77, тел. (8-0572) 19-12-02, 94-80-39, факс: (8-0572) 475-825, E-mail: uracad@kipt.kharkov.ua

Ломоносов Юрий Вячеславович, канд. техн. наук, ассистент кафедры информатики и вычислительной техники Национальной юридической академии Украины имени Я.Мудрого. Научные интересы: методы и алгоритмы компактного представления данных. Адрес: Украина, 61024, Харьков, ул. Пушкинская, 77, тел. (8-0572) 19-12-02.

УДК 519.7

РЕАЛИЗАЦИЯ НЕЙРОНОВ В СЕМАНТИЧЕСКИХ НЕЙРОННЫХ СЕТЯХ

ДУДАРЬ З.В., ШУКЛИН Д.Е.

Рассматривается принцип преобразования виртуальных машин. На его основе реализуется виртуальная машина, которая обеспечивает функционирование семантической нейронной сети, понимающей тексты на естественном языке. Описывается внутренняя структура нейрона и принципы организации хранилища данных нейронов.

1. Введение

Компьютеризация все больше и больше входит в нашу жизнь. Компьютеры находят всё новые и новые области применения: утюги, телевизоры, стиральные машины и большинство других устройств бытовой электроники оснащаются микроЭВМ. Наряду с процессом распространения компьютеров продолжается процесс их совершенствования. Сложность компьютеров увеличивается уже не с каждым годом, а с каждым месяцем. Усложняются задачи, которые могут решать вычислительные машины. Увеличивается объем программного обеспечения, необходимого для выполнения поставленных задач. Размеры программ достигают уже десятков гигабайт.

Параллельно с усложнением программного и аппаратного обеспечения усложняется их использование и сопровождение. Отдельно взятый человек просто физически не в состоянии освоить внутреннюю архитектуру современных компьютерных систем. Для того чтобы приступить к решению своих задач с помощью ПК, рядовой пользователь вынужден изучить огромный объем различной информации сугубо компьютерного назначения.

Для решения этих проблем создаются различные программные комплексы, ориентированные на возможность работы с ними практически без обучения. Однако наличие в таких продуктах чрезвычайно развитого сервиса делает невозможным изучение всех полезных функций за приемлемый промежуток времени. Перед компьютерными специалистами встают задачи по разработке и реализации программного обеспечения всё большей и большей сложности. Широкое распространение нашли так называемые “средства быстрой разработки приложений” (CASE Computer added software engineering). В них процесс создания приложения заключается в общении со специализированной программой “Мастер”, которая по информации, полученной от пользователя, сама вносит изменение в исходные тексты программ. Создаются специальные методы разработки программного обеспечения, такие как структурное, объектно-ориентированное, компонентное программирование, автоматизация программных систем.

Для решения наметившихся проблем во взаимодействии человека с компьютером идеально подходит общение с машиной на естественном языке. Отточенные тысячелетиями правила общения людей друг с другом могут быть применены и для общения человека с компьютером. Каждый человек изо дня в день использует универсальность языка, на котором он общается с другими людьми. На естественном языке мы с вами можем описать практически любые концепции, передать друг другу любую информацию и любые команды. Естественный язык в состоянии полностью удовлетворить требования людей к языку для работы с вычислительной машиной. Общение с компьютером на естественном для человека языке снизит психологический стресс, уменьшит количество ошибок и аварий, значительно снизит стоимость обучения персонала. Трудно полностью предсказать те выгоды, которые сулит нам переход с узкоспециализированных языков программирования на естественный язык.

Для обеспечения полномасштабного перехода на общение с информационными системами на естественном языке необходимо добиться понимания машиной текстов на естественном языке. Понимание естественного языка может быть выполнено

нейронными сетями. Однако на данный момент широко распространены вычислительные системы последовательного принципа действия. На таких системах не существует "родных" средств, реализующих нейронные сети. Последовательные вычислительные системы могут моделировать нейронные сети различным образом. Одним из вариантов моделирования нейронной сети может быть реализация с использованием виртуальной машины.

2. Преобразование виртуальных машин

Для определения того, чем же являются виртуальные машины, рассмотрим функционирование современной вычислительной системы. Основой вычислительной системы являются транзисторы – электронные аналоговые устройства, под которыми понимаются такие устройства, которые обрабатывают непрерывные сигналы в каком-либо диапазоне значений. Важно заметить, что реально существующий прибор - транзистор обрабатывает только непрерывный аналоговый сигнал, принимающий бесконечное число значений в заданном диапазоне. Обычно в вычислительной системе применяются два диапазона: 0-3,3В и 0-5В. Из технологических соображений выбирается диапазон изменения аналогового сигнала, обрабатываемого транзисторами (В ТТЛ: 0-5В). В этом рабочем диапазоне выбираются два непересекающихся поддиапазона, затем этим поддиапазонам назначаются логические значения "Ложь" и "Истина".

Благодаря волевому произволу конструктора, аналоговый сигнал теперь рассматривается как цифровой, т.е., имеющий два значения. Таким образом, "цифровые вычислительные машины можно отнести к классу "машин с дискретными состояниями". Так называются машины, работа которых складывается из совершающихся последовательно одна за другой резких смен их состояния. Состояния, о которых идет речь, достаточно отличны друг от друга, для того чтобы можно было пренебречь возможностью принять по ошибке одно из них за другое. Строго говоря, таких машин не существует. В действительности всякое движение непрерывно. Однако имеется много видов машин, которые удобно считать машинами с дискретными состояниями"[1, с.28].

Следующим этапом является организация на основе транзисторов схем, реализующих операции конъюнкции, дизъюнкции и инверсии. Можно считать, что на этом этапе возникает первая виртуальная машина. На самом деле она уже далеко не первая, существует виртуальная машина, преобразовывающая электроны, протоны и нейтроны в атомы, виртуальная машина, преобразующая атомы в материал, и так далее до уровня транзисторов.

Рассмотрим виртуальную машину, преобразующую транзисторы в логические схемы. Благодаря этой виртуальной машине конструктор получает возможность оперировать не с транзисторами и аналоговыми сигналами, а с более высокоуровневой абстракцией – цифровыми сигналами и логическими схемами. При этом неважно, какой диапазон выбран для изменения аналогового сигнала, какой поддиапазон рассматривается как "Ложь", а какой как "Истина". Уже неважно, из какого материала создан транзистор, и по какой технологии. Тем более стало неважно, куда и какой атом попал при диффузии в эмиттер. Изменилась реальность, с которой работает конструктор. На уровне логических схем реальностью являются "Ложь", "Истина", "Конъюнкция", "Дизъюнкция" и "Инверсия". Затем идет преобразование отдельных логических схем в модули, блоки и наконец перед нами предстает вычислительная система с процессором, памятью и системой ввода-вывода. В абстракции, с которой будет работать программист, и следа не осталось от транзисторов и аналоговых сигналов.

Когда виртуальная машина не полностью скрывает реализацию предыдущего уровня абстракции от следующего, возникают интересные эффекты пересечения уровней виртуальных машин. Ярким примером могут служить химические реакции. Два разных вещества отдельно друг от друга ведут себя согласно физическим законам, а, будучи соединены друг с другом, вступают в химическую реакцию. Этот эффект не следует принимать как опровержение идеи об изоляции уровней с помощью виртуальной машины. Так как виртуальная машина существует только как абстрактное соглашение в сознании субъекта, то пересечение уровней виртуальных машин возникает как осознанное или не осознанное нарушение в реализации виртуальной машины. Существование виртуальной машины как абстракции в сознании субъекта позволяет ему реализовывать новую виртуальную машину, оптимизированную под конкретные цели, как с нарушениями реализации, так и без таких нарушений. Сознательное ограничение экспериментальной ситуации позволяет субъекту либо полностью изолировать один уровень абстракции от другого, либо пользоваться эффектами от пересечения уровней виртуальных машин.

Рассмотренные явления преобразования реальности в теории операционных систем получили название "преобразование виртуальных машин". Как было продемонстрировано, реальность в вычислительных системах может зависеть от уровня абстракции. В связи с этим реальность, порожденную вычислительной системой, принято называть виртуальной реальностью.

Таким образом, виртуальная машина – это система, создающая уровень виртуальной реальности. Функции виртуальной машины заключаются в том, чтобы преобразовать соглашения из более высокого уровня виртуальной реальности в соглашения более низкого уровня виртуальной реальности, и наоборот. Важно заметить, что виртуальная машина

работает только с соглашениями о виртуальной реальности, следовательно, с информацией о виртуальной реальности. Сама же виртуальная машина существует только в сознании субъекта, оперирующего с интерпретацией реальности. Субъектом, в данном случае, может выступать как программист, так и некоторый процесс, развивающийся в вычислительной системе.

В вычислительной системе понятие реальности рассматривается с точки зрения развивающегося процесса, который использует одну или более виртуальных машин, развернутых на более глубоких уровнях абстракции. В то же время процесс реализует виртуальные машины для других процессов, развивающихся на более высоких уровнях абстракции.

Определение реальности для процесса, развивающегося на виртуальной машине, по отношению к ней может иметь вид: реальностью ведомого процесса будут иллюзии, создаваемые виртуальной машиной для этого процесса. Определение реальности для процесса, реализующего виртуальную машину для другого процесса, может иметь вид: реальностью ведущего процесса будут иллюзии, создаваемые этим процессом для ведомого процесса.

Очевидно, что процесс, развивающийся под управлением виртуальной машины, обязан считать иллюзию, создаваемую для него этой виртуальной машиной, реальностью. Несоблюдение этого правила приведет развивающийся процесс к неминуемому краху.

По отношению к вычислительной системе программист выступает в роли виртуальной машины. Это означает, что любая иллюзия может быть навязана системе в качестве реальности. Так же это означает, что любая подсистема вычислительной системы будет воспринимать в качестве реальности любую иллюзию, подготовленную, согласно произволу программиста, ведущей виртуальной машиной. Так как в роли программиста может выступать не только человек, но и процесс, то функцию супервизора, контролирующего виртуальные машины, можно возложить и на любую подсистему вычислительной системы.

При необходимости супервизор может даже нарушать (создавать новые) законы математики или логики. Для того чтобы решить проблему, необходимо просто иметь ее решение в одной из бесчисленного количества виртуальных реальностей. При этом даже не нужно искать ту реальность, в которой это решение существует, нужно просто сконструировать ее. После получения такого виртуального решения, в еще не существующей виртуальной реальности, нужно просто преобразовать уже существующую виртуальную реальность к требуемой, используя принцип преобразования виртуальных машин.

Отрицательным свойством данного подхода является чрезмерная нагрузка на вычислительную систему, выполняющую большое число вложенных друг в друга эмулирующих процессов. Сильно возрастают объемы программы, так как требуется описать не только решение проблемы, но и все виртуальные машины, необходимые для его выполнения. Этот подход более удобен в случае разработки действующих концептуальных моделей вычислительных систем, когда еще не ясны пути решения возникших проблем. Для промышленных систем, в большинстве случаев, более оптимальным является трансляция всех виртуальных машин в одну, выполняющую функции, эквивалентные всем виртуальным машинам, существующим в концептуальной модели.

В качестве примера функционирования указанных положений рассмотрим системы виртуальной памяти и эмуляторы процессоров. “В системе с виртуальной памятью у каждого процесса создается иллюзия, что вся его информация находится в оперативной памяти. Система поддерживает эту иллюзию, размещая некоторые из не используемых в данный момент разделов программы и данных, принадлежащих процессу, на периферийных устройствах. Если процесс пытается обратиться к информации, которая находится не в оперативной памяти, то автоматически вырабатывается прерывание, возбуждающее супервизор. Супервизор перемещает нужную информацию с периферийного устройства в память, и процесс получает доступ к ней. Эта деятельность супервизора незаметна для процесса, поскольку он приостанавливается до тех пор, пока передача информации не будет завершена. По окончании передачи процесс продолжает работу и, насколько ему “известно”, информация, которая была ему нужна, все время находилась в памяти. То, что информация физически находится на одном или нескольких периферийных устройствах, скрыто от процесса”[2, с. 122].

Наряду с предоставлением памяти большего объема система виртуальной памяти обеспечивает защиту процессов друг от друга. Процессам выделяются адресные пространства, изолированные друг от друга. Благодаря этому ошибочный или злонамеренный процесс не в состоянии повредить другому процессу, развивающемуся в одном операционном окружении. С точки зрения виртуальной машины реальность ведомого процесса полностью определяется произволом виртуальной машины. Объем доступной памяти и адресное пространство выбираются и устанавливаются виртуальной машиной.

Рассмотрим ситуацию, в которой некий процесс перестанет функционировать в пределах реальности, организованной для него машиной виртуальной памяти, и попытаемся опираться на “реально” существующие характеристики вычислительной системы. Если процесс начнет игнорировать объемы памяти, предоставленной ему системой вирту-

альной памяти, и начнет ориентироваться на реально доступную системе физическую память, то, во-первых, он может сильно проиграть в быстродействии, так как вытеснение неиспользуемых блоков памяти оптимизируется системой виртуальной памяти, на основе статистических данных, собранных со всех процессов, развивающихся в данное время в вычислительной системе. Во-вторых, процесс проиграет в быстродействии, так как ему придется реализовать виртуальную память собственными программными средствами, в отличие от операционной системы, которая, при возможности, пользуется аппаратными ускорителями. Если процесс начнет игнорировать защиту областей памяти, то это в лучшем случае приведет к краху сам процесс, а в худшем – может привести к краху всю вычислительную систему. Реальность последнего утверждения легко показать на примере процесса, работающего в графическом режиме в многозадачной системе. Допустим, в обход стандартных средств работы с графикой процессу удалось добыть информацию об аппаратном обеспечении, в котором он развивается в данный момент. Также допустим, что это аппаратное обеспечение уже существовало в момент написания программы этого процесса, и разработчик программы прекрасно осведомлен с особенностями работы этой аппаратуры. Допустим, что в программу данного процесса были включены все необходимые драйверы, чтобы работать с аппаратурой напрямую. Допустим, что процессу удалось обойти систему защиты памяти виртуальной машины, не вызвав при этом собственного аварийного завершения. Что произойдет, если этот процесс попытается работать с экранной памятью напрямую, в обход операционной системы? В общем случае, ничего хорошего. Процесс разрушит информацию, сформированную в окнах других процессов. Собственная информация данного процесса также будет разрушена другими процессами, так как будет сформирована вне окна, отведенного для него операционной системой. Попытки процесса изменить режим функционирования графического адаптера приведут к конфликту с операционной системой и, вероятнее всего, полностью выведут вычислительную систему из строя.

Эмуляторы - это программы, которые имитируют собой вычислительную систему. При этом имитируемая вычислительная система может быть как того же типа, что и система, выполняющая имитирующую программу, так и другого типа. Распространена имитация вычислительных систем, которые в виде "железа" никогда не производились. Хорошим примером реализации реально не существующей вычислительной системы служит широко известный язык Java. Этот подход позволяет выполнять программы, разработанные для одного типа вычислительной системы, на совершенно другом её типе.

Процесс, развивающийся под управлением эмулятора, считает, что он работает под управлением аппаратуры, имитируемой эмулятором, и воспринимает эту аппаратуру как реально существующую. В подобных случаях принципы работы эмулируемой аппаратуры настолько сильно отличаются от принципов работы аппаратуры, реально установленной в системе, что это делает невозможным попытку просто предположить выполнение под управлением виртуальной машины, а не под управлением аппаратуры. Несоблюдение процессом правил, установленных виртуальной машиной, в этом случае приводит к тем же последствиям для данного процесса, что и подобное несоблюдение при работе процесса под управлением реальной аппаратуры.

Из проведенного анализа можно сделать вывод о том, что в вычислительных системах соблюдение приведенных соглашений о реальности приводит к успешному развитию процессов в этой системе, а несоблюдение – к краху процесса, а иногда и всей вычислительной системы.

3. Понимание естественного языка

Воспользуемся рассмотренными свойствами виртуальной реальности в разработке формализованного представления смысла текста и системы его обработки. Для решения этой проблемы достаточно разработать любую виртуальную реальность, законы функционирования которой позволят решить поставленную задачу. Практическая реализация найденного решения на последовательных цифровых вычислительных системах будет возможна благодаря трансляции виртуальной машины, реализующей сконструированную виртуальную реальность, в виртуальную машину, работающую в виртуальной реальности, созданной операционной системой.

Прежде всего, необходимо понять, что будем подразумевать под термином "смысл предложения". Так как поиск решения будет проходить в виртуальной реальности, с законами, постулированными в процессе поиска решения, то достаточно определить термин "смысл предложения" способом, позволяющим автоматически извлекать и обрабатывать смысл предложения из текста на естественном языке.

Определим "смысл предложения" в виде следующего утверждения: смысл предложения – это смысл содержания предложения. Содержание предложения, очевидно, является информацией. Но информация является информацией только в контексте системы для ее обработки. Значит, смысл должен быть представлен системе обработки в виде, пригодном для обработки. Таким образом, представление смысла в системе обработки смысла должно быть "естественным" для этой системы и будет зависеть от ее реализации.

Перечислим требования к системе обработки смысла: возможность реализации в парадигме матери-

ального мира, очень высокая степень распараллеленности вычислительных процессов; высокая надежность, способность оперировать нечеткой и неполной информацией, способность оперировать связями между различными понятиями [4]. Исходя из перечисленных требований, построим виртуальную машину, в терминах которой возможно было бы обрабатывать объекты (сущности или концепты), отношения между объектами, нечеткие факторы уверенности и логические величины. Обработка и представление перечисленных понятий не вызывает каких-либо затруднений в виртуальной машине, организованной в виде нейронной сети. Так, отдельные элементарные объекты представляются в виде отдельных нейронов. Более сложные объекты, образованные от элементарных объектов, будут представлены либо совокупностью этих объектов, либо будут абстрагированы до нового элементарного объекта. Элементарные отношения между элементарными объектами представляются в виде связей между нейронами. Сложные отношения или классы отношений между объектами фактически являются сложными понятиями-объектами и могут обрабатываться как совокупности элементарных объектов с элементарными связями между ними. Факторы уверенности представляются в виде градиентных величин, обрабатываемых и передаваемых нейронами [4].

В разработанной виртуальной реальности смыслом обработанной части текста будет мгновенное состояние части нейронной сети, ответственной за извлечение информации из входного потока символов. Мгновенное состояние нейронной сети будет включать в себя мгновенный снимок множества нейронов, множества связей между нейронами и множества внутренних состояний нейронов. Таким образом, формальным представлением символьной информации в нейронной сети может выступать семантическая нейронная сеть, а обработка такого формализованного представления текста в виде нейронной сети будет производиться стандартными для этой нейронной сети способами.

Обычно под нейронной сетью понимается биологическая нейронная сеть[3], либо математическая модель биологической нейронной сети. В рассматриваемой виртуальной реальности у нейрона отсутствуют синаптические контакты. В отличие от биологической нейронной сети нейроны обмениваются аналоговыми сигналами с непрерывно изменяющейся амплитудой, а не пачками импульсов с фиксированной амплитудой. Нейроны выполняют не функции интегрирования сигналов, а функции алгебры логики, что более удобно при реализации на цифровых вычислительных машинах. Следовательно, полученную нейронную сеть также можно рассматривать как некоторую алгебру. Операции алгебры логики представляются в нейронной сети отдельными нейронами, значения предметных переменных представляются в виде градиентных значений, обрабатывающихся нейронной сетью, а последовательность применения операций задается структурой связей между нейронами. Так как отдельные нейроны в рассматриваемой нейронной сети представляют собой элементарные понятия обрабатываемого смысла, а связи между нейронами представляют собой элементарные отношения между понятиями, то такую сеть будем называть семантической нейронной сетью [4].

Текст на естественном языке представлен в виде последовательного потока символов. Различным вариантам состояния входного потока символов в нейронной сети будут соответствовать различные варианты состояния этой сети. Преобразование входного потока символов в состояние нейронной сети может быть выполнено рецепторами. Рецепторы, распознающие однотипные внешние воздействия, такие как символы текста, организуются в специализированные слои. Входной поток символов подается на слой нейронов - рецепторов, при этом каждый рецептор в слое распознает только один символ алфавита входной последовательности, игнорируя все остальные. Если символ распознан, на выходе рецептора устанавливается градиентное значение, соответствующее уровню успешного распознавания символа, например значение “логической истины”. Если в единицу времени рецепторам представить для опознания один символ, то только один нейрон будет иметь логическое значение “истина”, остальные будут иметь логическое значение “ложь”, т.е. в один такт рецепторами будет опознаваться только один символ текста.

Вывод результатов обработки текста из нейронной сети может быть выполнен эффекторами. Эффекторы, реализующие однотипные воздействия на окружающую среду, организуются в специализированные слои. Так, вывод текстов в виде символьной последовательности может быть осуществлен слоем эффекторов, в котором каждому нейрону будет соответствовать один выводимый символ алфавита. В единицу времени только один эффектор будет активирован входным градиентным уровнем, что соответствует одному символу, печатаемому на дамповом терминале. Возможно агрегирование нескольких понятий одним эффектором. Например, при выводе прилагательного эффектор, соответствующий прилагательным, будет находиться в активном состоянии, а существительным – в пассивном.

Важно заметить, что в случае коммутации двух нейронных подсетей друг с другом одни и те же нейроны могут выполнять в одной подсети функции рецепторов, а в другой подсети – функции эффекторов. Поэтому рецепторами и эффекторами будем называть не только нейроны, выполняющие обмен информацией с внешней средой, но и нейроны, выполняющие обмен информацией с любым её источником или приемником.

Применяя метод интроспекции, можно легко установить, что понимание смысла текста не является

статическим, мгновенным процессом. Понимание смысла – процесс, развивающийся во времени. Понимание и уточнение смысла происходит по мере поступления новых данных. Практически любая часть предложения имеет смысл. Это легко проверить, прочитав сначала первое слово, затем первые два слова, затем - первые три, четыре и т.д. Элементарным смыслом обладают даже отдельные буквы. По мере поступления новой информации смысл уже обработанных данных просто уточняется, обычно без повторной обработки уже поступивших данных[4].

Отдельный нейрон, расположенный в слое нейронной сети, извлекающей смысл из текста на естественном языке, будет обозначать элементарное понятие анализируемого языка. Элементарными являются любые понятия естественного языка с законченным смыслом, такие как символ, часть слова, слово, словосочетание, предложение, абзац, весь текст. В случае наличия соответствующего понятия в анализируемом тексте нейрон принимает значение “Истина”, а в случае отсутствия - “Ложь”. Так как процесс анализа текста развивается во времени, по мере поступления в нейронную сеть новых данных, то в нейронной сети возникают волны активности, распространяющиеся от рецепторов к эффекторам. Можно постулировать, что одному фронту такой волны соответствует смысл текста, принятого сетью непосредственно перед началом этого фронта. Тогда окончанием обработки текста можно считать достижение слоя эффекторов фронтом волны, порожденным последним символом обрабатываемого текста. В результате на слое эффекторов, в виде их состояний, будет выставлен смысл текста.

Смысл текста на естественном языке представляется в нейронной сети как мгновенное состояние множества выходных нейронов, находящихся в слое извлечения смысла из входной последовательности. Поэтому преобразование одного смысла в другой представляет собой преобразование состояния одной группы нейронов в состояние другой группы. Состояние нейронов характеризуется значением выходных градиентных данных. Для преобразования смысла в виде градиентных данных могут применяться экспертные системы. Отдельные операции над обрабатываемым смыслом представляются как отдельные правила в базе знаний экспертной системы.

Правила помещаются в базу знаний. Туда же в виде фактов помещаются входные данные и результаты срабатывания правил. Если ситуация, анализируемая экспертной системой, удовлетворяет условиям срабатывания правила, то его следствия считаются верными для данной ситуации.

Для реализации экспертной системы, основанной на правилах, средствами нейронной сети необходимо преобразовать сеть правил в нейронную сеть. В процессе преобразования нужно обеспечить решение следующих проблем: управление вниманием и объяснение полученных результатов. Управление вниманием разбивается на подзадачи определения неизвестного факта, наиболее важного в данный момент, и определение степени завершенности вывода по каждому возможному ответу.

В процессе преобразования правил экспертной системы в нейронную сеть первым этапом является разбиение каждого правила на элементарные операции дизъюнкции, конъюнкции и инверсии. Каждой такой операции назначается группа из нескольких нейронов. Каждая такая группа является элементарной операцией, выполняемой правилом экспертной системы. Группы нейронов связаны с другими группами посредством дендритов и аксонов. В каждой группе существует один главный нейрон, значение на его выходе является значением операции, выполняемой группой. Остальные нейроны группы выполняют служебные функции, такие как управление вниманием. Обработка такого формализованного представления текста в виде нейронной сети будет производиться собственными средствами нейронной сети. Следовательно, формализованное представление текста в виде нейронной сети обладает свойством самовыполнимости.

4. Реализация нейронов в вычислительных системах

Нейронная сеть, как формальный язык описания смысла текста, требует специальной реализации в вычислительной системе. Предлагается реализовать нейронную сеть в виде виртуальной машины, состоящей из нескольких изолированных уровней абстракций. Изоляция уровней может быть обеспечена путем рассмотрения их как черных ящиков с некоторым набором входных и выходных интерфейсов. Такой подход позволит реализовывать внутреннюю структуру каждого уровня, независимо от реализации внутренней структуры остальных уровней. Разрабатываемая виртуальная машина должна обеспечивать трансляцию виртуальной реальности, функционирующей в терминах операционной системы, в виртуальную реальность, функционирующую в терминах нейронной сети. Виртуальная машина, реализующая преобразование абстракций нейронной сети в абстракции операционной системы, будет содержать шесть уровней: уровень операционной системы; уровень хранилища; уровень управления телами нейрона; уровень управления секциями нейрона; уровень логики поведения секции; уровень логики поведения нейрона.

Операционная система представляет собой программу, выполняющую распределение ресурсов вычислительной системы между различными процессами. Уровень операционной системы будет адаптировать функции, выполняемые операционной системой, под требования остальных уровней реализации. В уровне операционной системы будут

реализованы такие функции, как управление файлами, памятью, вводом-выводом и потоками (нитями) выполнения программы,

Хранилище данных представляет собой некоторую структуру долговременного хранения данных, в которой содержатся данные, описывающие состояние нейронов. На уровне операционной системы хранилище данных представляет собой обычный файл. Структура хранилища организована таким образом, чтобы состояние нейронов сохранялось целостным независимо от того, загружены или не загружены в память машины копии данных нейронов. Это достигается путем исключения из хранилища всех контекстно-зависимых данных, таких как файловые указатели и указатели на блоки памяти.

Физически нейрон реализуется в хранилище данных как поток данных (Data Stream). Благодаря этому каждый нейрон можно рассматривать как некоторый файл. В результате работа с нейроном ведется посредством функций работы с файлами, такими как чтение блока данных, запись блока данных, перемещение файлового указателя.

Идентификация, поиск, управление и обработка нейронов в хранилище обеспечивается с помощью идентификаторов нейронов, которые служат основным понятием хранилища данных. Идентификатор представляет собой целое число. При рождении каждому нейрону назначается идентификатор. Он остается неизменным на протяжении всей жизни нейрона. После смерти нейрона этот идентификатор может быть назначен другому нейрону. В одно и то же время в хранилище не может существовать двух и более нейронов с одним и тем же идентификатором. Если в качестве идентификатора нейрона взять 32-битовое целое число со знаком, то хранилище сможет содержать до $2^{31}=$ 2'147'483'648 различных нейронов. Все операции над нейронами осуществляются с указанием их идентификаторов. Удобно ввести специальный идентификатор *NULL*, имеющий нулевое значение. Ни один нейрон не может иметь идентификатор, равный *NULL*. Замена идентификаторов на это значение позволит без дополнительных затрат сигнализировать о том, что идентификатор больше не связан с каким-либо конкретным нейроном.

Уровень управления телами нейронов загружает данные нейрона из хранилища в оперативную память системы. Блок оперативной памяти для тела нейрона выделяется уровнем операционной системы по требованию уровня управления телом нейрона, после чего хранилище заполняет этот блок памяти данными конкретного нейрона из хранилища нейронов. В результате возникают две копии данных нейрона: одна – устаревшая, в хранилище, а другая – актуальная, в оперативной памяти. После завершения работы с нейроном уровень управления телом нейрона сохраняет его данные в хранилище и удаляет из памяти системы блок оперативной памяти. На уровне управления телом каждый нейрон можно рассматривать как файл или поток данных, отображенный в память вычислительной системы, принадлежащий своему хранилищу и имеющий идентификатор, который остается неизменным в течение всего жизненного цикла нейрона-файла. Уровень управления телом нейрона рассматривает его как некоторую целостную совокупность данных без внутренней структуры.

Нейрон рассматривается как объект с внутренней структурой, начиная с уровня управления его секциями. С этого уровня нейрон получает дополнительный идентификатор, представляющий собой целое число, размерность которого равна размерности идентификатора нейрона. Этот идентификатор располагается в теле нейрона и может изменяться в течение его жизни. Он может быть использован для различных целей. Уровень управления секциями не накладывает ограничение на его использование. В дальнейшем этот идентификатор будет определять принадлежность данного нейрона к какому-либо типу или классу и будет соответственно называться либо идентификатором типа нейрона, либо идентификатором класса нейрона, по существу представляя собой идентификатор другого нейрона, выступающего в роли описателя типа или класса данного нейрона. Основной функцией уровня управления секциями нейрона является разбиение тела нейрона на отдельные секции и предоставление другим уровням всех необходимых возможностей по работе с этими секциями. Каждая секция представляет собой блок памяти, расположенный в блоке памяти тела нейрона, и ведет себя как файл, отображенный в память вычислительной системы. Уровень управления секциями не накладывает ограничения на их использование другими уровнями. Каждой секции назначается идентификатор, представляющий собой целое число, размерность которого равна размерности идентификатора нейрона. Этот идентификатор располагается в секции нейрона и не может изменяться в течение ее жизни. Идентификатор секции однозначно определяет каждую секцию в теле нейрона. Один нейрон не может иметь две различные секции с одним и тем же идентификатором. Однако разные нейроны могут иметь по одной секции с одинаковыми идентификаторами секций. Такой идентификатор может быть использован для различных целей. Уровень управления секциями не накладывает ограничение на его применение. В дальнейшем он будет определять принадлежность данной секции к какому-либо типу или классу и будет соответственно называться идентификатором либо типа секции, либо класса секции, по существу представляя собой идентификатор другого нейрона, выступающего в роли описателя типа или класса данной секции.

Возможна реализация уровня управления секциями нейрона так, чтобы каждую новую секцию с ее идентификатором можно было соответственно ин-

терпретировать как тело некоторого нейрона с его идентификатором. В результате можно получить структуру (рис.1), в которой количество вложенных подсекций будет ограничено только физическими ограничениями вычислительной системы.

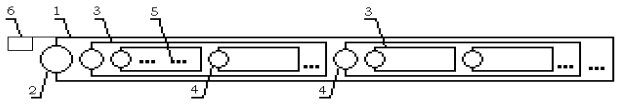

Рис. 1. Структура подсекций: 1 –данные тела нейрона; 2 –идентификатор типа нейрона; 3 – данные тела секции; 4 – идентификатор типа секции; 5 – данные других уровней, расположенные в теле секции; 6 – идентификатор нейрона в хранилище

Каждая секция нейрона может рассматриваться как поток данных (или как файл, отображенный в память) и интерпретироваться другими уровнями любым способом. Уровень логики поведения секции будет рассматривать секцию как табличную структуру, состоящую из неограниченного количества записей по N полей в каждой. Поля в секции, например, нумеруются неотрицательными целыми числами с шагом 1. Каждое поле имеет свой фиксированный тип. Порядок расположения записей в секции нейрона назовем естественным порядком записей в секции. Тогда, например, записи в секции будут нумероваться целыми неотрицательными числами с шагом 1. Благодаря такой организации отдельных нейронов возникает возможность выполнять операции реляционной алгебры, такие как пересечение, объединение и другие для обработки секций нейронов как неупорядоченных данных. Также возникает возможность хранить в отдельной секции блоки упорядоченных данных, например текстовые строки как упорядоченные таблицы из одного поля типа символ. При этом обе возможности реализуются без дополнительных потерь производительности вычислительной машины. Уровень логики поведения секции не накладывает ограничений на использование табличных структур, реализованных в секции нейрона. В дальнейшем основным применением таких табличных структур будет хранение идентификаторов нейронов или идентификаторов секций в виде списков или отношений.

Уровень логики поведения нейрона определяет назначение определенных ранее атрибутов нейрона и алгоритмы их использования. Отношения некоторого нейрона с нейроном - описателем типа нейрона и нейронами - описателями типа секции приведены на рис.2.

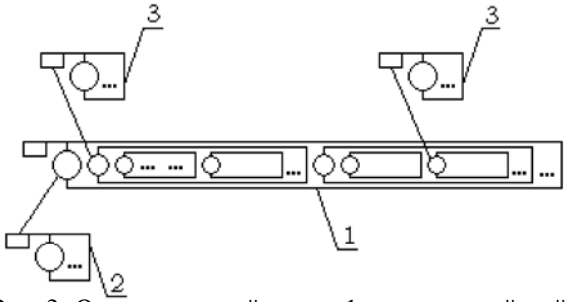

Рис. 2. Отношения нейронов: 1 – некоторый нейрон; 2 – нейрон - описатель типа нейрона; 3 – нейрон - описатель типа секции

Каждый нейрон (рис.2,1) принадлежит типу, от которого он был порожден. Описатель типа также является нейроном (рис.2,2) и имеет свой идентификатор. По аналогии с объектно-ориентированным программированием нейроны-описатели типа можно называть типами классов. Каждый нейрон может выступать в роли класса. Порождение нейрона от класса можно называть наследованием. От одного класса может быть порождено неограниченное количество нейронов. Каждый нейрон подразделяется на секции. Каждая секция имеет свой тип. Носителем типа секции выступает ее нейрон (рис.2,3). Каждая секция будет иметь свой числовой идентификатор, равный идентификатору ее нейрона.

Рассмотренное представление формального языка как нейронной сети будет обладать свойствами как сетевой, так и реляционной баз данных.

5. Заключение

В данной работе была проделана попытка определить, чем является реальность в вычислительных системах. Обнаружено, что термин “реальность” является комплексным, т.е. под ним подразумеваются различные понятия, в зависимости от рассматриваемого контекста. Рассмотрен принцип преобразования виртуальных машин и его влияние на понимание термина “реальность” в вычислительных системах. Обнаружено, что этот термин может выступать в совершенно другом качестве, нежели в повседневной практике. Согласно рассмотренным определениям реальности получен формальный язык представления и обработки смысла в вычислительной системе. Этим формальным языком оказалась семантическая нейронная сеть. Однако полученная нейронная сеть очень сильно отличается от биологической нейронной сети. В полученном нейроне отсутствуют синаптические контакты, в отличие от биологической нейронной сети нейроны обмениваются аналоговыми сигналами с непрерывно изменяющейся амплитудой, а не пачками импульсов с фиксированной амплитудой. Нейроны выполняют не функции интегрирования сигналов, а функции алгебры логики, что более удобно при реализации на цифровых вычислительных машинах.

Литература: 1. *Алан М. Тьюринг* Может ли машина мыслить? М.: Физматгиз, 1960. 112с. 2. *Цикритзис Д., Бернстайн Ф.* Операционные системы. М.: Мир, 1977. 336с. 3. *Мозг:* Пер. с англ./ Перевод Алексеенко Н.Ю./ Под ред. и с предисл. П.В. Симонова. М.:Мир, 1982. 280с.4. *Дударь З.В., Шуклин Д.Е.* Семантическая нейронная сеть, как формальный язык описания и обработки смысла текстов на естественном языке /Радиоэлектроника и информатика. 2000. №. 3. С. 72-76.

Поступила в редколлегию 06.07.2000

Рецензент: д-р техн. наук, проф. Шабанов-Кушнаренко Ю. П.

Дударь Зоя Владимировна, канд. техн. наук, доцент кафедры программного обеспечения ЭВМ ХТУРЭ. Научные интересы: логическая математика, модели языка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-94-46.

Шуклин Дмитрий Евгеньевич, аспирант кафедры программного обеспечения ЭВМ ХТУРЭ. Научные интересы: теория интеллекта, нейронные сети. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-94-46.