СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА.

ВВЕДЕНИЕ

За последние 15-20 лет в области информатики достигнуты значительные успехи. Наравне о известными базами данных, оперирующими с конкретными величинами и фактами, возникли так называемые ***базы знаний***, определяющие и хранящие внутренние отношения и связи между теми или иными явлениями, событиями и фактами, предопределяющие тем самым принципиальный новый подход в обработке информации и проводящие к ее интеллектуализации. Возникла новая специальность "инженер по знаниям", являющаяся составной частью научного направления под общим названием "Системы искусственного интеллекта" (СИИ).

Вопросы описания знаний и модели их представления, модели различного рода интеллектуальных процессов - логического вывода, принятия решений, прогнозирования и т. п., а также разработка на этой основе новых систем и информационных технологий - составляют суть дисциплины СИИ. Курс СИИ является одной из дисциплин, завершающих формирование специалистов по вычислительной технике.

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Студенты по специальности Математическое обеспечение и администрирование информационных систем изучают дисциплину на 5-м курсе. Программа СИИ предполагает предварительное изучение таких дисциплин как "дискретная математика", "математическая логика", "программирование", "теория проектирования ЭВМ", "организация вычислительных процессов".

Цель данного курса - дать основные понятия о моделях представления знаний в новых информационных технологиях, об алгоритмах логического вывода в различных моделях знаний, о подходах к построению моделей предметной области или выделенной сферы деятельности с точки зрения систем ИИ. Другая задача курса - дать представление о новых технологиях обработки информации, основанных на знаниях, а также о системах, объединяемых понятием системы искусственного интеллекта.

В результате изучения данного курса студент должен:

- знать основные отличия систем ИИ от информационных систем традиционных типов;
- уметь строить модели простых предметных областей в логических исчислениях высказываний и предикатов 1-го порядка, а также в продукционных и фреймовых представлениях;
- понимать механизм логического вывода в исчислениях предикатов и уметь осуществлять вывод новых суждений методом резолюций;
- знать основные типы СИИ и принципы их проектирования.

Изучения курса состоит из:

- самостоятельного изучения пособий,
- прослушивания лекций,
- выполнения лабораторных работ.

В заключение курса предусмотрен зачет.

Лабораторная работа 1.

Тема работы: Выбор и концептуальное описание предметной области задачи принятия решений.

Цель работы: Овладение методикой выделения и концептуального описания предметной области задачи; построение на основе этого описания концептуальной модели данной предметной области.

Описание работы.

Данная лабораторная работа соответствует этапу системного анализа задачи, с которого начинается проектирование любой экспертной системы. Это очень важный этап, на который следует обратить особое внимание. Он включает в себя процессы идентификации и концептуализации задачи, т.е. выделения соответствующей предметной области (ПО), сбора необходимой информации,

консультаций с экспертами и начальной формализации задачи в виде концептуальной модели предметной области.

В качестве лабораторной работы выполняется одно задание, которое состоит из 6-ти задач.

1. Построение модели предметной области для данной сферы деятельности человека.

2. Построение продукционной модели заданной предметной области в виде базы фактов (данных) и базы знаний и имитация работы этой модели в течение 5 тактов.

3. Построение фрагмента модели заданной предметной области на языке исчисления высказываний.

4. Построение фрагмента модели заданной предметной области на языке исчисления предикатов путём соответствующего усложнения.

5. Построение фрагмента модели предметной области в виде семантической сети.

6. Построение фрагмента предметной области в виде сети фреймов.

Темы заданий.

Выбирается одна из приведенных ниже тем, и на ее основе выполняются все 6 вышеперечисленных задач. Указанные темы не являются обязательными. Студент вправе предложить свою, близкую ему, тему.

1. Сборка арки с помощью робота. Арка состоит из 2-х вертикальных и одной горизонтальных балок. Первоначально все балки находятся в различных точках стройплощадки.
2. Создание нового текстового файла в редакторе Word. Исходное состояние – компьютер выключен.
3. Запуск в рабочее состояние автомобиля. Начальное состояние – водитель рядом с машиной.
4. Заправка бензобака автомобиля у бензоколонки. Начальное состояние – автомобиль у бензоколонки.
5. Замена картриджа у принтера. Начальное состояние – картридж лежит рядом с принтером.
6. Прием–передача информации на пейджер с помощью пейджинговой компании. В процессе участвуют три человека: 1-й отправляет сообщение, 2-й принимает сообщение, 3-й – оператор – передает сообщение.
7. Установка автомашины в гараж. В начальный момент времени машина стоит у гаража, гараж закрыт.
8. Автоматизированный комплекс (конвейер) проверки наполнения бутылок квасом (водой, молоком и т.п.). В случае обнаружения недолива – доливает необходимое количество жидкости и поочередно снимает бутылки и ставит их в ящик.
9. Диагностика неисправностей блока питания компьютера.
10. Диагностика неисправностей компьютерного дисковода.
11. Ввод документа в компьютер с дискеты и его распечатка.
12. Передача документа по факсимильной связи.
13. Принятие решения о сертификации товара. Решение принимается на основе установления соответствия реквизитов товара (имя, № накладной, фирма, стоимость) реквизитам товаросопроводительных документов. При наличии несоответствий принимается решение об отправке на экспертизу. Если экспертиза подтверждает нарушение, то следует отказ в выдаче сертификата.
14. Диагностика неисправностей компьютера при его включении.
15. Управление лифтом (6 этажей, из двух или более нажатых кнопок выполняется та, которая меньше по номеру, если лифт идет вверх, и та, которая больше по номеру, если лифт идет вниз).
16. Заварка чая по-японски (по-русски, по-китайски и т.п.).
17. Приготовление борща.
18. Управление движением автомобилей на Т-образном перекрестке. Движение допускается в обе стороны. Решения: включить красный светофор, включить зеленый.
19. Управление потоком заданий на компьютер. Задания имеют ранг: 1, 2, 3; время решения соответственно: t1, t2, t3.
20. Обслуживание абонента в библиотеке.
21. Уход за комнатным цветком.
22. Обработка детали на станке, съём и замена детали.

23. Формирование каталога Интернет-ресурсов. Решение о регистрации принимается в случае соответствия заявленных данных о фирме поданным документам (тип деятельности, платежеспособность, оплата сервера и т.п.).
24. Диагностика неисправностей в работе принтера.
25. Форматирование дискеты и запись двух новых файлов.
26. Отправка посылки через почтовое отделение.
27. Ведение и обеспечение функционирования базы данных.
28. Ремонт копировального аппарата.
29. Осуществление модемной связи между двумя компьютерами. Предполагается, что один модем (ожидающий) ждет звонка, второй (инициирующий) осуществляет дозвон.
30. Рыбалка: смоделировать процесс рыбной ловли.
31. Сортировка поездов на сортировочной станции.
32. Уход за цветком в теплице.
33. Управление взлетом самолетов в аэропорту (одна взлетно-посадочная полоса).
34. Игра в "крестики-нолики".
35. Перевозка через реку волка, козы и капусты.
36. Распределение вакансий на бирже труда.
37. Заключение договора на поставку оборудования.

Порядок выполнения работы

Начинать выполнение заданий следует с построения модели предметной области (ПО).

Продукционные модели ПО, кстати говоря, могут использоваться при построении моделей исчисления высказываний, исчисления предикатов, а также моделей сетевого или фреймового типа. Содержательной основой для этого могут служить факты базы данных (БД) и правила базы знаний (БЗ). Факты БД по своей сути являются простыми высказываниями, истинность которых определена. Они истинны, так как все факты, попадающие в БД, по определению истинны. Правила БЗ необходимо интерпретировать как правильно построенные формулы исчисления высказываний (ППФ).

Для иллюстрации сказанного рассмотрим классический пример с обезьяной и бананами.

Задача 1. На основе фактов БД и правил БЗ построим формальную модель этой предметной области на языке исчисления высказываний. Сначала для этого необходимо определить алфавит, т.е. набор символов, обозначающих высказывания, которые мы определим, например, следующим образом:

A – "Обезьяна находится в точке a",
B – "Ящик находится в т. b",
C – "Бананы находятся в т. c",
D – "Обезьяна находится на Ящике",
E – "Обезьяна держит Бананы",
F – "Обезьяна, Ящик, Бананы находятся в разных точках",
G – "Обезьяна находится рядом с ящиком".

Используя логические связки: $\neg, \wedge, \vee, \rightarrow, \equiv$, мы можем строить более сложные умозаключения. Например:

1). $(A \wedge B \wedge C) \rightarrow F$.

(Если все предметы находятся на своих исходных позициях, т.е, в точках a, b, c соответственно, то справедливо сказать: "Обезьяна, Ящик, Бананы находятся в разных точках").

2). $F \rightarrow \overline{G}$.

(Если "Обезьяна, Ящик, Бананы находятся в разных точках", то "Обезьяна находится не рядом с Ящиком").

3). $F \rightarrow \overline{D}$.

(Если "Обезьяна, Ящик, Бананы находятся в разных точках", то "Обезьяна не находится на Ящике").

4). $F \rightarrow \overline{E}$.

(Если "Обезьяна, Ящик, Бананы находятся в разных точках", то "Обезьяна не держит Бананы").

Высказывания 2, 3, 4 можно объединить в одно с помощью логической связки "И":

5). $F \to (\overline{G} \wedge \overline{D} \wedge \overline{E})$.

6). $\overline{F} \to (D \wedge E)$.

(Если "Обезьяна, Ящик и Бананы находятся не в разных точках", (т.е. в одной), то "Обезьяна стоит на Ящике" И "Обезьяна держит Бананы").

7). $\overline{D} \to \overline{E}$.

(Если "Обезьяна не на Ящике", то "Обезьяна не держит Бананы").

Таким образом, фрагмент предметной области на языке исчисления высказываний (ИВ) представляет собой следующий набор формул (БЗ):

$$\begin{array}{l} (A \wedge B \wedge C) \to F; \\ F \to (\overline{G} \wedge \overline{D} \wedge \overline{E}); \\ \overline{F} \to (D \wedge E); \\ \overline{D} \to \overline{E}. \end{array} \qquad (1)$$

Для простоты введем обозначение $S = A \wedge B \wedge C$.

Теперь, используя механизм вывода ИВ, мы можем доказать выводимость любой другой формулы, структура которой соответствует синтаксису ИВ (в рамках данной модели). Докажем, например, что формула

$$(\overline{S} \wedge D) \to E \qquad (2)$$

выводима. Ее смысл: если "Обезьяна, Ящик и Бананы не в исходных точках" И "Обезьяна на Ящике", то "Бананы в руках у Обезьяны".

Логический вывод можно сделать несколькими способами. 1). Можно, например, использовать таблицу истинности, которая дает исчерпывающую картину значений переменных. При этом формулы (1) и (2) образуют выражение логического вывода в виде (см. [1], стр. 59-61):

$$\begin{array}{l} ((A \wedge B \wedge C) \to F) \wedge (F \to (\overline{G} \wedge \overline{D} \wedge \overline{E})) \wedge (\overline{F} \to (D \wedge E)) \wedge (\overline{D} \to \overline{E})) \blacksquare \\ ((\overline{S} \wedge D) \to E) \end{array} \qquad (3)$$

Далее следует вычислить значения выражений слева и справа от знака $\blacksquare$. Если окажется, что формула справа принимает значение И как только все формулы (1), образующие левую часть, одновременно примут значение И, то логическая выводимость (2) из (1) доказана.

Но нелегкое это дело – считать по таблице истинности. Формула (3) имеет 7 переменных. Это значит, что таблица будет содержать 2^7, т.е. 128 строк! Есть ведь и другие методы.

2). Можно попытаться использовать метод вывода, основанный на системе аксиом исчисления высказываний. Но это тоже очень трудоемкий процесс.

3). Лучше всего попробовать метод опровержения, основанный на принципе дедукции и реализуемый посредством резолюций (см. [1], стр.65-71). Согласно этому методу, выводимость формулы (2) из системы (1) доказывается от противного: формула (2) выводима из (1), если присоединение её отрицания к системе (1) делает вновь образованную систему противоречивой. Другими словами, следует доказать противоречивость системы

$$\begin{array}{l} 1)\ S \to F; \\ 2)\ F \to (\overline{G} \wedge \overline{D} \wedge \overline{E}); \\ 3)\ \overline{F} \to (D \wedge E); \\ 4)\ \overline{D} \to \overline{E}; \\ 5)\ \overline{(\overline{S} \wedge D) \to E}, \end{array} \qquad (4)$$

где под номером 5 как раз и стоит отрицание формулы (2). Далее применяем метод резолюций. Для начала все пять предложений следует представить в виде конъюнкции элементарных дизъюнктов (КНФ) ([1], стр. 67 – 74). Для простоты дизъюнкцию будем обозначать "+".

Для первого предложения имеем: 1'. $\overline{S}+F$.

Для второго: 2'. $\overline{F}+(\overline{G} \wedge \overline{D} \wedge \overline{E})$.

Для третьего: 3'. $F+(D \wedge E)$.

Для четвертого: 4'. $D + \overline{E}$.

Для пятого приведем цепочку преобразований:

$$5'.\ \overline{(\overline{S} \wedge D) \to E} = \overline{\overline{(\overline{S} \wedge D)} + E} = \overline{\overline{(\overline{S} + \overline{D})} + E} = \overline{S} \wedge D \wedge \overline{E}.$$

Пользуясь правилом раскрытия конъюнкции по дизъюнкции ([1], стр. 52, 56), имеем для 2': $(\overline{F} + \overline{G}) \wedge (\overline{F} + \overline{D}) \wedge (\overline{F} + \overline{E})$;

для 3': $(F+D) \wedge (F+E)$.

Конъюнкция 5' распадается на три одночленных дизъюнкта: $\overline{S}$, D, $\overline{E}$.

Теперь у нас имеется система элементарных дизъюнктов, на основе которой проводим вывод методом резолюции.

1. $\overline{S}+F$.
2. $\overline{F} + \overline{G}$.
3. $\overline{F} + \overline{D}$.
4. $\overline{F} + \overline{E}$.
5. F+D.
6. F+E.
7. S.
8. D.
9. $\overline{E}$.

Резольвенты

10. F (6, 9).
11. $\overline{D}$ (3, 10).
12. "Л" (8, 11).

Мы получили "пустой" (ложный) дизъюнкт. Это значит, что выводимость формулы (2) из системы (1) доказана, т. е. утверждение (2) истинно.

Отметим, что приведенная резолюция не единственно возможная. Можно и по-другому. Например:

10'. $\overline{F}$ (3, 8).
11'. E (6, 10').
12'. "Л" (9,11'). И т.п.

Задача 2. Рассмотрим теперь методику построения формальной модели ПО на языке исчисления предикатов 1-го порядка. Напомним, что предикатная форма есть дальнейшее развитие исчисления высказываний. К ней также приводятся отношения, полученные при построении продукционной модели. Подробно исчисление предикатов (ИП) рассматривается в [1], гл. 6.

Берем всё тот же пример с обезьяной и бананами. Считаем, что модель предметной области уже построена ([1], стр. 11-12). Следует отобразить ее в терминах ИП. Алфавит А ИП, как известно, кроме символов $\neg, \wedge, \vee, \to, \sim, \forall, \exists$, содержит также:

- индивидные константы,
- предметные переменные,
- функциональные константы,
- высказывания,
- предикатные константы.

Очевидно, что роль индивидных констант здесь будут играть элементы множества X МПО, т.е. множество имен объектов. У нас это: Обезьяна (О), Ящик (Я), Бананы (Б). Роль предметных переменных – элементы множества C (множество имен свойств объектов). В нашем примере это координаты Обезьяны, Ящика и Бананов – соответственно x, y, c. Область определения для них – множество D точек комнаты. В качестве функциональных переменных, как правило, выступают операторы (действия). В нашем случае – множество G = {g_1 – подойти, g_2 – перенести, g_3 – взобраться, g_4 – схватить}. Областью определения функций является множество состояний предметной области, которое возникает в результате выполнения этих действий. Это состояния, являющиеся предусловиями и постусловиями применения действий ([1], стр. 14-16). Обозначим это множество состояний через S = {s_1, s_2, ….s_n} (см. также [2], стр. 23 – 27).

Множество высказываний можно оставить прежним (из вышеприведенного примера 1), но мы возьмем другое.

Множество R – множество имен отношений. Например:

НА(О,Я) – Обезьяна НА Ящике;

У(О,Я) – Обезьяна У Ящика;

$\overline{В}$(О,Б) – Бананы НЕ-В руках Обезьяны;

и т.п. – готовая предикатная форма записи.

Координаты объектов введем через одноместные предикаты.

О(х) – Обезьяна находится в т. х;

Я(у) – Ящик находится в т. у;

Б(с) – Бананы висят в т. с. Здесь (x,y,c∈ D).

Трехместный предикат делает семантику более гибкой:

В(О, Я, х) – Обезьяна и Ящик находятся в т. х.

У(О,Б,с) – Бананы у Обезьяны в т. с.

Из сказанного видно, что роль термов у нас играют индивидные константы (О,Я,Б) и предметные переменные (x,y,c). Но этого недостаточно, т.к. они описывают лишь отдельные состояния предметной области. Переход из одного состояние в другое осуществляется под действием операторов g_i, прилагаемых в данной точке х к конкретному состоянию s: $g_i(x,s)$. Это тоже терм. Следовательно, допустимы такие выражения (будем помнить, что выполняются условия x,y,c ∈ D; s∈ S):

В(О, Я, у, $g_1(x,s)$) – "Обезьяна и Ящик находятся в т. у в результате применения оператора подойти к Ящику к состоянию s в т. х";

НА(О, Я, у, $g_2(y,s)$) – "Обезьяна сидит на Ящике в т. у в результате применения оператора взобраться на Ящик к состоянию s".

Используя таким образом язык исчисления предикатов, запишем нашу модель предметной области в виде следующих аксиом.

1. $\forall x \forall s[(\overline{РЯДОМ}\ (О,Я,x,s)) \rightarrow В(О,Я,y,g_1(x,s))]$ –

"для всех х и s: если в состоянии s Обезьяна и Ящик не находятся рядом в т. х, то Обезьяна может оказаться в т. у, где находится Ящик, путем применения к ситуации s оператора g_1 (подойти к Ящику) из точки х в точку у".

2. $\forall y \forall s[(В(О,Я,c,g_2(y,s)) \rightarrow НА(О,Я,c,g_3(c,s))]$ –

"для всех у и s: если Обезьяна и Ящик под действием оператора g_2 (перенести Ящик) оказались в точке с, то Обезьяна обязательно заберется на Ящик под действием g_3 (взобраться на Ящик)".

3. ($\overline{У}$(О,Б,с,Sн))$\rightarrow$НА(О,$\overline{Я}$,с,Sн) –

"если в точке с у Обезьяны нет Бананов, то она НЕ-НА Ящике".

4. РЯДОМ (О,Я,b,Sн) –

" в начальном состоянии Обезьяна и Ящик не находятся рядом".

5. $\forall y \forall s$(В(О,Я,с,$g_2(y,s)$) –

"для всех у и s: Обезьяна и Ящик находятся в точке с в результате применения оператора g_2 (перенести Ящик)" к состоянию s.

(В выражениях 3–4 кванторы $\forall$ отсутствуют, т.к. это конкретные высказывания, характеризующие начальное состояние Sн).

Поставим теперь вопрос: существует ли такое состояние s∈ S в некоторой точке x∈ D, при котором Бананы находятся у Обезьяны? Формально вопрос запишется так:

6. $\exists$s(У(О,Б,x,s)),

т.е. мы как бы говорим: "Да, существует такое состояние s". Требуется, таким образом, доказать, что это утверждение истинно. С точки зрения формальной логики это случится, если выражение 6 окажется логическим следствием пяти предыдущих посылок:

(1. 2. 3. 4. 5) ▪ 6 .

Как и прежде, будем использовать метод опровержения и, как следствие, метод резолюций. Для начала приведем выражения, имеющие кванторы, к предваренной нормальной форме ([1], стр. 81 – 86), затем возьмем отрицание от формулы 6. В последнем случае получим цепочку преобразований:

$$\exists s(\overline{У}(О,Б,x,s)) = \forall s(\ \overline{У}(О,Б,x,s)) = \ \overline{У}\ (О,Б,x,s).$$

Добавим полученное выражение к пяти вышеприведенным формулам, представленным в предваренной форме ([1], стр. 85). Получим следующую систему предложений:

1'. РЯДОМ(О,Я,x,s)∨В(О,Я,y,g1(x,s));

2'. $\overline{\text{В}}$(О,Я,c,g2(y,s)) ∨НА(О,Я,c,g3(c,s));

3'. У(О,Б,c,Sн) ∨ $\overline{НА}$(О,Я,c,Sн);

4'. $\overline{РЯДОМ}$(О,Я,b,Sн);

5'. В(О,Я,c,g2(y,s));

6'. $\overline{\text{У}}$(О,Б,x,s).

Система из пяти предложений 1'–5' непротиворечива по построению (можно проверить). Если теперь путем подстановок и резолюций мы определим пустой дизъюнкт в расширенной системе 1' – 6', то это будет означать, что добавление 6' приводит ее к противоречию. Но 6' есть отрицание выводимого нами утверждения 6, и поэтому само выражение 6 приводить к противоречию уже не будет (что-нибудь одно!). И, следовательно, выводимо (см. [1], стр. 90 – 96).

Делаем подстановки, строим резольвенты (напомним: подстановки касаются всего предложения в целом). В скобках обозначены номера предложений, участвующих в резолюции.

Резольвенты	Подстановки	замены
7'. НА(О,Я,c,g3(c,s)),	(2', 5')	
8'. $\overline{НА}$(О,Я,c,Sн),	(3', 6')	c/x; Sн/s.
9'. "Л".	(7', 8')	Sн/g3(c,s).

Нашли "пустой" дизъюнкт. Это значит, что наше утверждение

∃s(У(О,Б,x,s)) верно: "Да, существует такое состояние s, при котором Бананы в руках у Обезьяны".

Показанный путь проведения логического вывода, очевидно, не единственный. Возможны другие пути. Например:

7". (1', 4'), 9". (3', 8"),

8". (7", 2') 10". (6', 9"). И т.п.

В качестве упражнения предлагаем раскрыть эти резольвенты, разумеется, с учетом подстановок.

В данном случае мы показали, как получить ответ на самый простой вопрос – "да" или "нет". Могут быть и более сложные выводы. Например: какие действия необходимо выполнить для того, чтобы Обезьяна находилась на Ящике? Или: что Обезьяне надо сделать, чтобы Бананы находились у нее в руках? И т. п.

Ответы на такие вопросы мы сейчас рассматривать не будем, хотя для этого есть все возможности. Такого рода выводы называются дедуктивными и являются основой работы с дедуктивными базами данных. Это мощнейшее и интереснейшее приложение к логике исчисления предикатов, которое в данном пособии не рассматривается из-за недостатка места.

Задача 3. Рассмотрим теперь методику построения модели предметной области, основанной на семантических сетях (СС). В разделе 2 ([2], стр. 30 и далее) приводятся несколько типов СС: предикативные, атрибутивные, иерархические и другие. Мы рассмотрим пример построения предикативной сети. (Студент вправе построить сеть любого другого типа). Вначале следует дать словесное описание ПО. Обозначим через Pi различные ситуации (предложения), возникающие в пространстве состояний.

P1. Обезьяна, Ящик и Бананы находятся в различных точках комнаты – соответственно a, b, c.

P2. Обезьяна видит Бананы И пытается их достать, но Бананы висят высоко.

P3. Обезьяна подходит к ящику и переносит его в точку c, затем залезает на Ящик.

P4. Обезьяна достает Бананы.

Мы описали состояние ПО "крупными блоками". Для построения сети следует разбить их на простые предложения. Мы видим, что первое предложение, по существу, соединяет три простые предложения; 2-е предложение – также состоит 3-х простых, но два из них соединены связкой И. Третье предложение, в свою очередь, распадается на три простые предложения. Союз "и" здесь не

представляется связкой И, т.к. соединяет последовательность действий, а не их одновременность. Нам надлежит еще связать понятие "точка" с понятием "комната", ведь точка, по существу, – часть комнаты. В итоге получаем систему предложений:

	Предикатная запись
Р111. Точка a есть часть комнаты (И)	Часть(т. a, комната),
Р112. Точка b есть часть комнаты (И)	Часть(т. b, комната),
Р113. Точка c есть часть комнаты.	Часть(т. c, комната),
Р11. Обезьяна находится в т. a.	Наход (Обезьяна, т. a),
Р12. Ящик находится в т. b.	Наход(Ящик, т. b),
Р13. Бананы находятся в т. c.	Наход(Бананы, т. c),
Р21. Обезьяна видит Бананы (И)	Видеть(Обезьяна, Бананы),
Р22. Обезьяна пытается достать Бананы.	Пыт(Обезьяна, Бананы),
Р23. Бананы висят высоко.	Висеть(Бананы, высоко),
Р31. Обезьяна подходит к ящику.	Подход(Обезьяна, Ящик),
Р32. Обезьяна переносит Ящик в т. c.	Перенос(Обезьяна, Ящик),
Р33. Обезьяна влезает на Ящик.	Влезть(Обезьяна, Ящик),
Р4. Обезьяна достает бананы.	Достать(Обезьяна, Бананы).

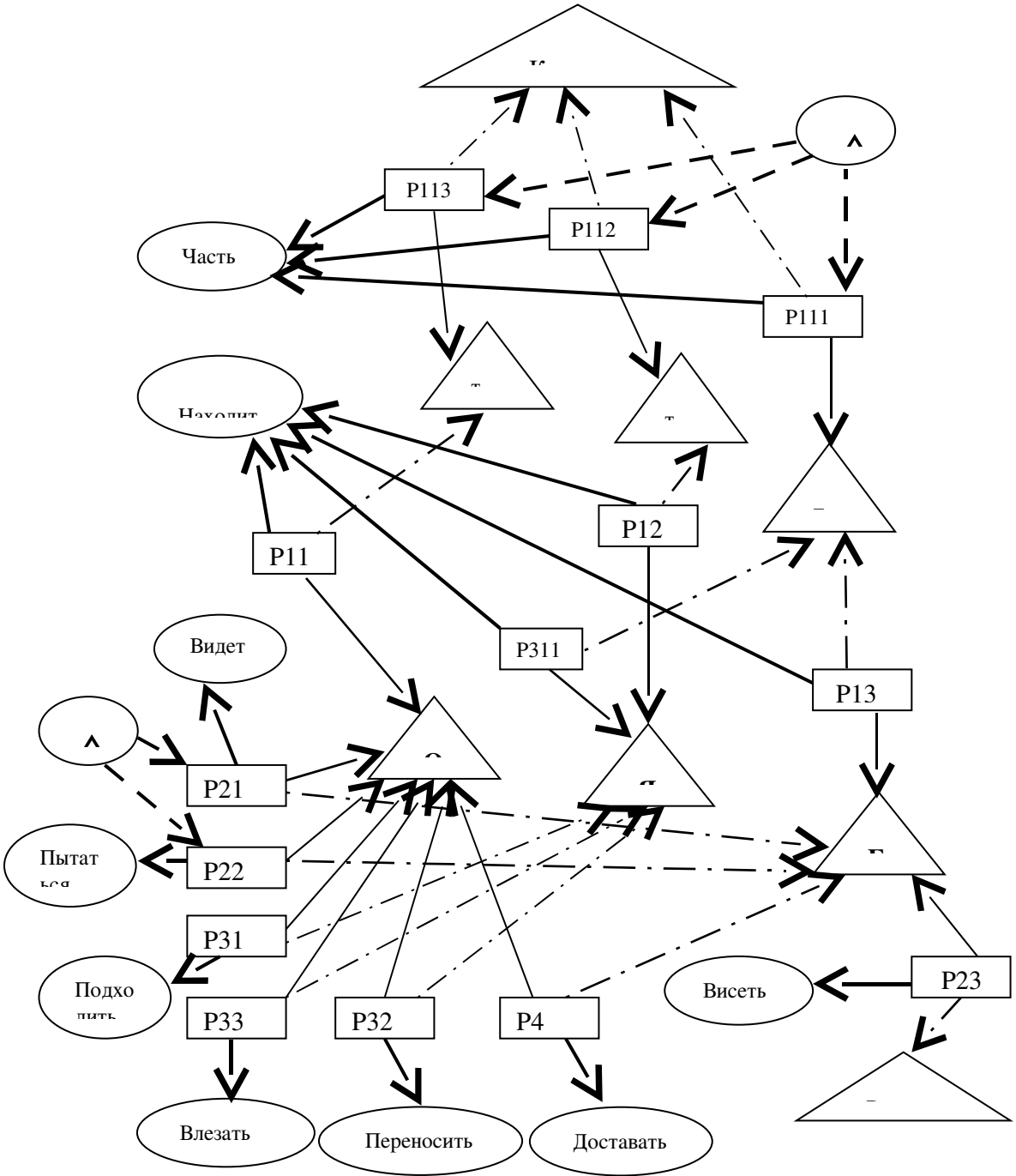


Рис. 1. Семантическая сеть для задачи 3.

Остается построить саму сеть. Для удобства объекты будем обозначать треугольником, предложения – прямоугольником , предикатные вершины – овалом. Стрелку предиката выделим жирной линией, стрелки предикатов логических связок – пунктиром, стрелку первого терма предиката сплошной, стрелку второго терма – штрих-пунктирной линиями. Полученная сеть показана на рисунке 1.

Данная семантическая сеть может дать ответы на множество самых разных вопросов. Например:

Какие предметы находятся в комнате?

Какие объекты могут быть переносимы из одной точки комнаты в другую?

Кто является субъектом действий?

Какие действия может выполнять обезьяна? И т.п..

Методику вывода на сетях мы здесь рассматривать не будем. Пример на эту тему рассмотрен в [2], стр. 41 - 42. Скажем только, что представление текста в виде СС предикатного типа позволяет относительно легко привести содержание текста к формальному виду. Сам текст может быть различным, т.е. один и тот же смысл может быть описан разными текстами. Описывая смысл текста в виде конкретной семантической сети, мы обеспечиваем как бы единственность его представления, что как раз и дает возможность использования СС для построения автоматизированных информационно-справочных и поисковых систем.

Задача 4. Рассматриваемую задачу об обезьяне и бананах представим теперь в виде сети фреймов. Это можно сделать по-разному, применяя в качестве фрейма различные понятия: либо объекты – О, Я, Б, К, либо действия, либо состояния ([1], стр. 37 – 39; [2], гл. 3). Для примера построим один из вариантов однородной СС, вершины которой являются действиями (процессами), а дуги этой сети обозначим как отношения "следовать за". В этом случае мы получаем фреймы – вершины с соответствующим описанием: ПОДОЙТИ (F1), ПЕРЕНЕСТИ (F2), ВЛЕЗТЬ (F3), СХВАТИТЬ (F4), СЛЕЗТЬ (F5), ОТОЙТИ (F6) и т.п.. Так как у нас выбрано только одно отношение – отношение следования, граф переходов будет иметь элементарный вид (рис. 2).

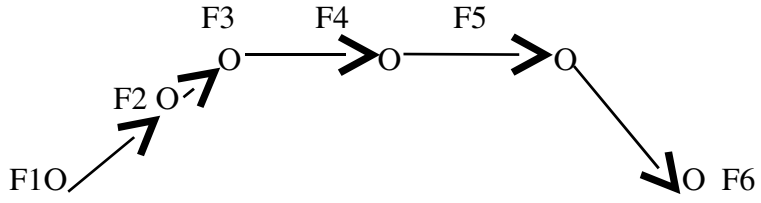
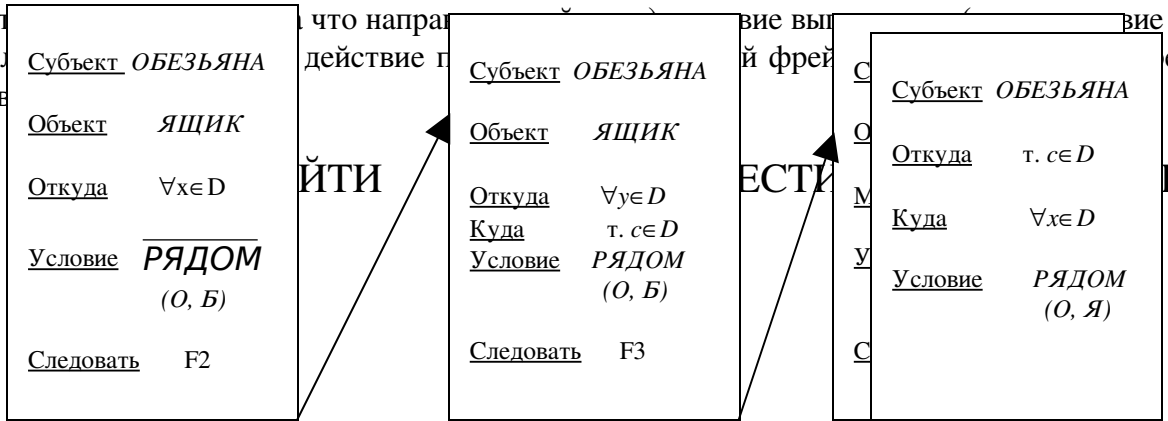


Рис. 2. Граф перехода действий к задаче на фреймах

В нашем примере фреймы имеют следующие слоты:

субъект (тот, кто производит действия), откуда, куда, или место (все рассматриваемые точки комнат что напра вие вы вие должно выпо действие п й фрей ходит действ



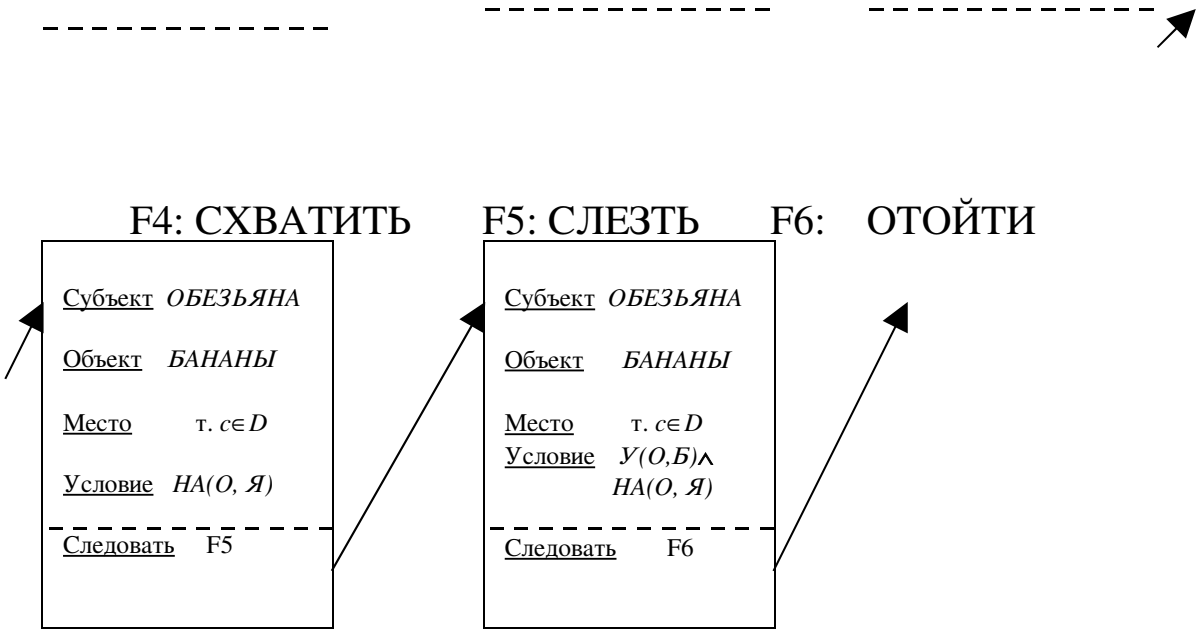


Рис. 3. Фрагмент сети фреймов

Как только выполняется слот условие, активизируется следующий по графу фрейм. Например, если был активизирован фрейм F4, то в нем проверяется выполнение условия "Обезьяна на Ящике". Если оно истинно, то срабатывает слот следовать и активность передается фрейму F5 (по стрелке). Здесь проверяется своё условие "Обезьяна у Бананов И Обезьяна на Ящике". И т.д.

В разделе 3 ([2], стр. 44 – 52) приводятся несколько других примеров описания предметной области в виде фреймов. Студент может выбрать тот метод, который более подходит в соответствии со смыслом задачи.

Задания выполняются индивидуально. После готовится отчёт по лабораторной работе и проводится защита.

Лабораторная работа 2.

Тема работы: изучение нейронных сетей.

Цель работы: Овладение методикой разработки нейронных сетей.

Описание работы.

В лабораторной работе рассмотриваются основы теории нейронных сетей, позволяющие в дальнейшем обратиться к конкретным структурам, алгоритмам и идеологии практического применения сетей в компьютерных приложениях.

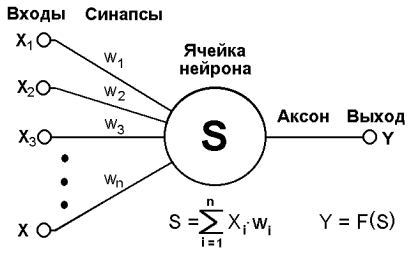
В последние десятилетия в мире бурно развивается новая прикладная область математики, специализирующаяся на искусственных нейронных сетях (НС). Актуальность исследований в этом направлении подтверждается массой различных применений НС. Это автоматизация процессов распознавания образов, адаптивное управление, аппроксимация функционалов, прогнозирование, создание экспертных систем, организация ассоциативной памяти и многие другие приложения. С помощью НС можно, например, предсказывать показатели биржевого рынка, выполнять распознавание оптических или звуковых сигналов, создавать самообучающиеся системы, способные управлять автомашиной при парковке или синтезировать речь по тексту. В то время как на западе применение

НС уже достаточно обширно, у нас это еще в некоторой степени экзотика – российские фирмы, использующие НС в практических целях, наперечет [1].

Широкий круг задач, решаемый НС, не позволяет в настоящее время создавать универсальные, мощные сети, вынуждая разрабатывать специализированные НС, функционирующие по различным алгоритмам.

Модели НС могут быть программного и аппаратного исполнения. В дальнейшем речь пойдет в основном о первом типе.

Несмотря на существенные различия, отдельные типы НС обладают несколькими общими чертами.



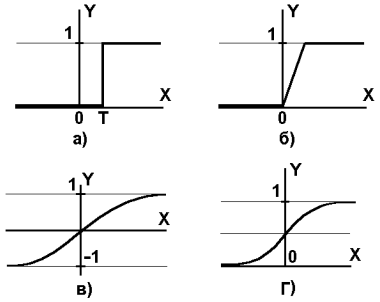
Во-первых, основу каждой НС составляют относительно простые, в большинстве случаев – однотипные, элементы (ячейки), имитирующие работу нейронов мозга. Далее под нейроном будет подразумеваться искусственный нейрон, то есть ячейка НС. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов – однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Общий вид нейрона приведен на рисунке 1. Каждый синапс характеризуется величиной синаптической связи или ее весом w_i, который по физическому смыслу эквивалентен электрической проводимости.

Текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i \cdot w_i \qquad (1)$$

Выход нейрона есть функция его состояния:

$y = f(s)$ (2)



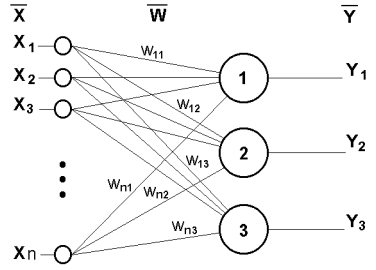
Нелинейная функция f называется активационной и может иметь различный вид, как показано на рисунке 2. Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид (т.е. функция S-образного вида)[2]:

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \qquad (3)$$

При уменьшении ▪ сигмоид становится более пологим, в пределе при ▪ =0 вырождаясь в горизонтальную линию на уровне 0.5, при увеличении ▪ сигмоид приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Одно из ценных свойств сигмоидной функции – простое выражение для ее производной, применение которого будет рассмотрено в дальнейшем.

$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x)) \qquad (4)$$

Следует отметить, что сигмоидная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.



Возвращаясь к общим чертам, присущим всем НС, отметим, во-вторых, принцип параллельной обработки сигналов, который достигается путем объединения большого числа нейронов в так называемые слои и соединения определенным образом нейронов различных слоев, а также, в некоторых конфигурациях, и нейронов одного слоя между собой, причем обработка взаимодействия всех нейронов ведется послойно.

В качестве примера простейшей НС рассмотрим трехнейронный перцептрон (рис.3), то есть такую сеть, нейроны которой имеют активационную функцию в виде единичного скачка* . На n входов поступают некие сигналы, проходящие по синапсам на 3 нейрона, образующие единственный слой этой НС и выдающие три выходных сигнала:

$$y_j = f\left[\sum_{i=1}^{n} x_i \cdot w_{ij}\right], j=1...3 \qquad (5)$$

Очевидно, что все весовые коэффициенты синапсов одного слоя нейронов можно свести в матрицу **W**, в которой каждый элемент w_{ij} задает величину i-ой синаптической связи j-ого нейрона. Таким образом, процесс, происходящий в НС, может быть записан в матричной форме:

Y=F(XW) (6)

где **X** и **Y** – соответственно входной и выходной сигнальные векторы, **F(V)** – активационная функция, применяемая поэлементно к компонентам вектора **V**.

Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется НС. Чем сложнее НС, тем масштабнее задачи, подвластные ей.

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные, на сегодняшний день, конфигурации, описанные, например, в [2],[3],[4] и других изданиях, перечисленных в конце статьи. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами: возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных слоев (влияние числа слоев на способность сети выполнять классификацию плоских образов показано на рис.4 из [5]); введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети; сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тромозящих и др.) также способствует усилению мощи НС. Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

Очевидно, что процесс функционирования НС, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой НС, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

Этот этап называется обучением НС, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации. На этапе обучения кроме параметра качества подбора весов важную роль играет время обучения. Как правило, эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса.

Обучение НС может вестись с учителем или без него. В первом случае сети предъявляются значения

Порядок выполнения работы

1. Необходимо подготовить к выполнению на ПК тестовый пример на языке C++ и Visual Basic (Приложение, листинг 1 и 2), моделирующий работу нейронной сети.
2. Ознакомится с работой алгоритмами обучения искусственных нейронных сетей без учителя и сделать соответствующие выводы о работе нейронных сетей.
3. Подготовить отчёт.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК.

Основной.

1. Болотова Л.С., Комаров М.А., Смольянинов А.А. Системы искусственного интеллекта// Теоретические основы СИИ и формальные модели представления знаний: Учебное пособие – М.: МИРЭА, 1998 – 108 с.
2. Болотова Л.С. Смольянинов А.А. Неформальные модели представления знаний в системах искусственного интеллекта: Учебное пособие – М.: МИРЭА, 1999 – 100 с.
3. Нильсон Н. Принципы искусственного интеллекта: Пер. с англ. - М.: Радио и связь, 1985. – 376 с
4. Попов Э. В. Экспертные системы. - М.: Наука, 1987.
5. Построение экспертных систем / Под ред. Ф.Хейес - Рот, Д. Уотерман, Д. Ленат - М.: Мир, 1987.

Дополнительный.

6. Нильсон Н. Искусственный интеллект. - М.: Мир 1973.
7. Лорьер Ж.Л. Системы искусственного интеллекта, - Мир, 1991.
8. Кандрашина Е.Ю., Литвинцова Л.В., Поспелов Д.А. Представление знаний о времени и пространстве в интеллектуальных системах / Под ред. Д.А. Поспелова. - М.: Наука.

Гл. ред. Физ. - мат. Лит. - 328 с.

9. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. - М.: Наука,1983.
10. Уотерман Д. Руководство по экспертным системам. –

М.: Мир, 1989.

11. Поспелов Д.А. Логико - лингвистические модели в системах управления. - М.: Энергия,1981.
12. Минский М. Фреймы для представления знаний. - М.: Мир,1979.
13. Поспелов Д.А. Ситуационное управление: теория и практика. - М.: Наука,1986.
14. Практическое введение в технологию искусственного интеллекта и экспертных систем / Р.Левин, Д.Эделсом: Пер. с англ. - М.: Финансы и статистика, 1991. - 239 с.

Справочный.

15. Логический подход к искусственному интеллекту / Тейз А., Грибомон П. и др. - М.: Мир, 1990.
16. Нейлор К. Как построить экспертную систему: Пер. с англ. - М.:

Энергоатомиздат, 1991. - 286 с.

17. Поспелов Д.А. Моделирование рассуждений. Опыт анализа

мыслительных актов. - М.: Радио и связь, 1989. - 184 с.

Вопросы для самопроверки.

Каковы отличия СИИ от моделей алгоритмического типа? Чем отличаются математические модели от моделей ИИ?

Дайте определение понятию "знания". В чем отличие "знаний" от "данных"?

Дайте определения понятиям: предметная область, модель ПО, состояние ПО, оператор, задача.

Какова структура модели предметной области? Перечислите ее составляющие и укажите их особенности.

Опишите этапы развития представлений о системах ИИ. Какова роль формально-логических моделей в СИИ?

Опишите процесс решения задачи в СИИ. Чем он отличается от традиционных представлений?

Какими недостатками для реализации СИИ обладают машины фон-неймановского типа?

Как строится граф пространства состояний? Приведите пример такого графа.

В чем суть метода поиска решения "в ширину"? Объясните на графе. (То же для метода "в глубину").

Как строится граф редукции ("И-ИЛИ")?

Дайте определение понятиям "высказывание", "интерпретация". Перечислите операции над высказываниями.

Что такое "правильно построенные формулы"? Приведите примеры ППФ.

Какие методы определения истинности высказываний вы знаете?
Перечислите признаки формальной теории применительно к ИВ.
Приведите пример формализации предметной области в понятиях ИВ.
Что значит "решить задачу" в ИИ? Каков механизм вывода в ИВ?
В чем состоит основная проблема при решении логических задач?
Дайте определение теоремы дедукции.
В чем состоит метод резолюций? Приведите теорию, лежащую в основе метода. Что такое резольвента? Поясните работу метода резолюций на конкретном примере.
Приведите пример предиката 1-го порядка.
Из каких элементов состоит алфавит (словарь) ИП?
Дайте определение ППФ в ИП и приведите пример.
Какие новые аксиомы по сравнению с исчислением высказываний вводятся в ИП?
Приведите пример общезначимых и противоречивых (невыполнимых) ППФ.

Приложение.
Листинг 1.

```
// FILE neuro.h
#include <stdio.h>
#define OK 0 // состояния объектов
#define ERROR 1

#define ORIGINAL 0 // типы активационных функций
#define HYPERTAN 1
#define HARDLIMIT 2
#define THRESHOLD 3

#define INNER 0 // тип распределения памяти
#define EXTERN 1

#define HORIZONTAL 1
#define VERTICAL 0

#ifndef max
#define max(a,b) (((a) > (b)) ? (a) : (b))
#define min(a,b) (((a) < (b)) ? (a) : (b))
#endif

// базовый класс нейронов для большинства сетей
class Neuron
{
protected:
float state; // состояние
float axon; // выход
int status; // признак ошибки
public:
Neuron(void){ state=0.; axon=0.; status=OK; };
virtual float Sigmoid(void)=0;
int GetStatus(void){return status;};
};

class SomeNet
{
```

```
protected:
FILE *pf;
int imgfile; // 0 - числа; 1 - 2D; 2 - эмуляция
unsigned rang;
int status;
unsigned learncycle;
int (*emuf)(int n, float _FAR *in, float _FAR *ou);
public:
SomeNet(void)
{pf=NULL;imgfile=0;rang=0;status=OK;learncycle=0;};
unsigned GetRang(void){return rang;};
void SetLearnCycle(unsigned l){learncycle=l;};
int OpenPatternFile(unsigned char *file);
int ClosePatternFile(void);
void EmulatePatternFile(int (*p)(int n,
float _FAR *, float _FAR *))
{emuf=p;imgfile=2;};
int GetStatus(void){return status;};
};

class LayerBP;
class NetBP;

// нейрон для полносвязной сети прямого распространения
class NeuronFF: public Neuron
{
protected:
unsigned rang; // число весов
float _FAR *synapses; // веса
float _FAR * _FAR *inputs;
// массив указателей на выходы нейронов предыд. слоя
void _allocateNeuron(unsigned);
void _deallocate(void);
public:
NeuronFF(unsigned num_inputs);
NeuronFF(void){rang=0; synapses=NULL;
inputs=NULL; status=OK;};
~NeuronFF();
virtual void Propagate(void);
void SetInputs(float *massive);
void InitNeuron(unsigned numsynapses);
virtual void RandomizeAxon(void);
virtual void Randomize(float);
virtual float Sigmoid(void);
virtual float D_Sigmoid(void);
virtual void PrintSynapses(int,int);
virtual void PrintAxons(int, int);
};

class NeuronBP: public NeuronFF
{ friend LayerBP;
friend NetBP;
float error;
```

```
float _FAR *deltas; // изменения весов
void _allocateNeuron(unsigned);
void _deallocate(void);
public:
NeuronBP(unsigned num_inputs);
NeuronBP(void){deltas=NULL; error=0.;};
~NeuronBP();
void InitNeuron(unsigned numsynapses);
int IsConverged(void);
};

class LayerFF
{
protected:
unsigned rang;
int status;
int x,y,dx,dy;
unsigned char *name; // имя слоя
public:
LayerFF(void) { rang=0; name=NULL; status=OK; };
unsigned GetRang(void){return rang;};
void SetShowDim(int _x, int _y, int _dx, int _dy)
{x=_x; y=_y; dx=_dx; dy=_dy;};
void SetName(unsigned char *s) {name=s;};
unsigned char *GetName(void)
{if(name) return name;
else return (unsigned char *)&("NoName");};
int GetStatus(void){return status;};
int GetX(void){return x;};
int GetY(void){return y;};
int GetDX(void){return dx;};
int GetDY(void){return dy;};
};

class LayerBP: public LayerFF
{ friend NetBP;
protected:
unsigned neuronrang; // число синапсов в нейронах
int allocation;
NeuronBP _FAR *neurons;
public:
LayerBP(unsigned nRang, unsigned nSinapses);
LayerBP(NeuronBP _FAR *Neu, unsigned nRang,
unsigned nSinapses);
LayerBP(void)
{neurons=NULL; neuronrang=0; allocation=EXTERN;};
~LayerBP();
void Propagate(void);
void Randomize(float);
void RandomizeAxons(void);
void Normalize(void);
void Update(void);
int IsConverged(void);
```

```
virtual void Show(void);
virtual void PrintSynapses(int,int);
virtual void PrintAxons(int x, int y, int direction);
};

class NetBP: public SomeNet
{
LayerBP _FAR * _FAR *layers;
// нулевой слой нейронов без синапсов реализует входы
public:
NetBP(void) { layers=NULL; };
NetBP(unsigned nLayers);
NetBP(unsigned n, unsigned n1, ...);
~NetBP();
int SetLayer(unsigned n, LayerBP _FAR *pl);
LayerBP *GetLayer(unsigned n)
{if(n<rang) return layers[n]; else return NULL; }
void Propagate(void);
int FullConnect(void);
void SetNetInputs(float _FAR *mvalue);
void CalculateError(float _FAR * Target);
void Learn(void);
void Update(void);
void Randomize(float);
void Cycle(float _FAR *Inp, float _FAR *Out);
int SaveToFile(unsigned char *file);
int LoadFromFile(unsigned char *file);
int LoadNextPattern(float _FAR *IN, float _FAR *OU);
int IsConverged(void);
void AddNoise(void);
virtual void PrintSynapses(int x=0,...){};
virtual float Change(float In);
};

// Сервисные функции
void out_char(int x,int y,int c,int at);
void out_str(int x,int y,unsigned char *s,unsigned col);
void ClearScreen(void);

// Глобальные параметры для обратного распространения
int SetSigmoidType(int st);
float SetSigmoidAlfa(float Al);
float SetMiuParm(float Mi);
float SetNiuParm(float Ni);
float SetLimit(float Li);
unsigned SetDSigma(unsigned d);

// Псевдографика
#define GRAFCHAR_UPPERLEFTCORNER 218
#define GRAFCHAR_UPPERRIGHTCORNER 191
#define GRAFCHAR_HORIZONTALLINE 196
#define GRAFCHAR_VERTICALLINE 179
#define GRAFCHAR_BOTTOMLEFTCORNER 192
```

```
#define GRAFCHAR_BOTTOMRIGHTCORNER 217

#define GRAFCHAR_EMPTYBLACK 32
#define GRAFCHAR_DARKGRAY 176
#define GRAFCHAR_MIDDLEGRAY 177
#define GRAFCHAR_LIGHTGRAY 178
#define GRAFCHAR_SOLIDWHITE 219
```

Листинг 2.

Функция increaseLastActivated – Запоминает последний активный нейрон

Тип outputNeuron – описывает «Живой» Нейрон

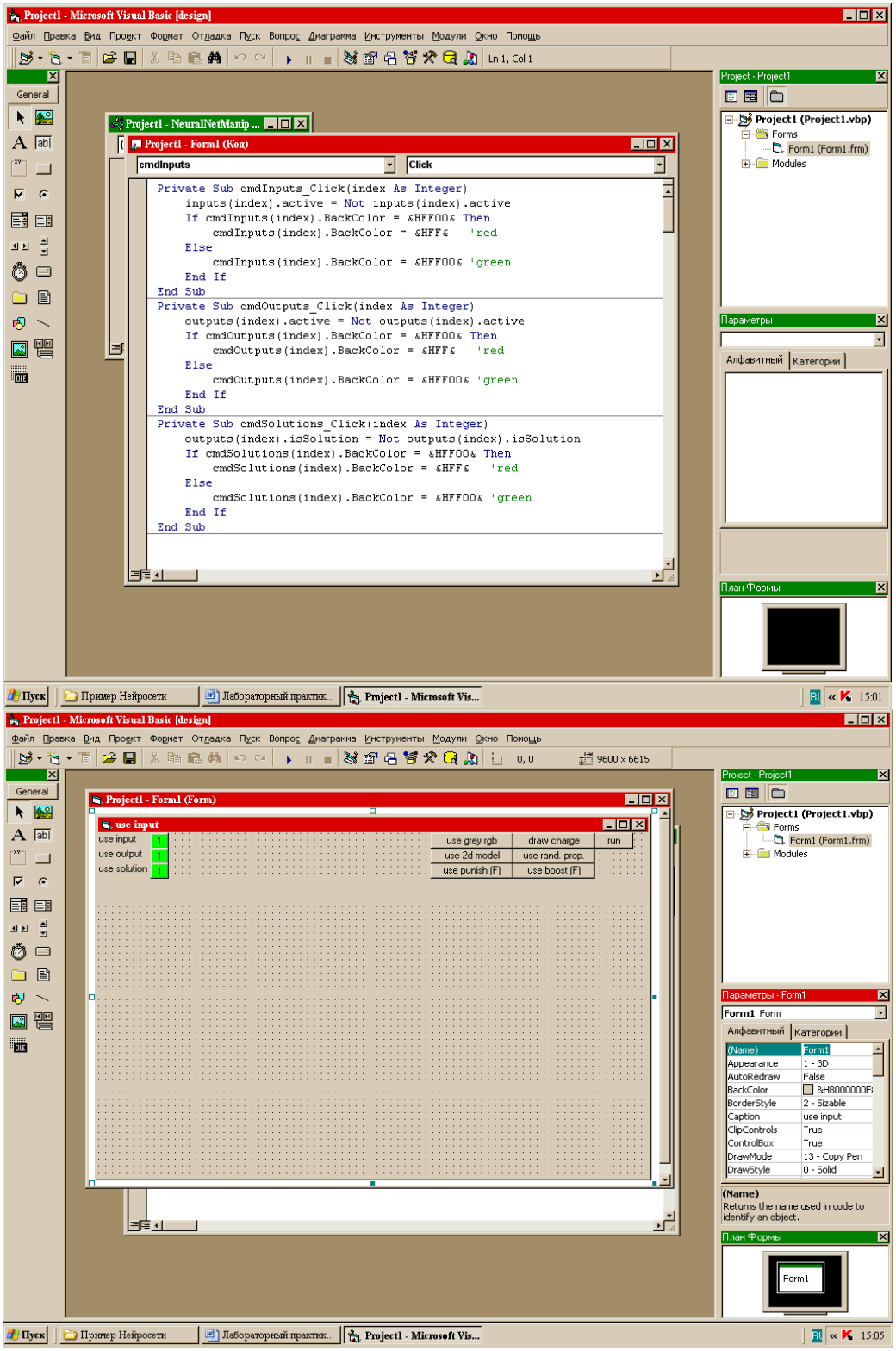
define2DInputLocations – активизирует графическую интерпретацию нейрона.

createOutputPunishTemplate – создание нового выходной группы жизни нейронов.

В программе можно изменять количество входных сигналов (Дендритов 1-4), также изменяется количество аксонов (Выходов 1-4) и также можно выбирать количество решений для жизни нейронов. Сверху расположены кнопки управления входов и выходов. Также имеются инструменты для изменения графической интерпретации жизни нейронов. Внизу отображена сама жизнь. В строке состояния отображается положение активных нейронов.

assignConnectionFromTo – привязывает соединение нейронов

getConnectionIndex – получение информации о нейроне (Жизнь или Смерть)



Пример разработки программы.

Исходный текст программы на Visual Basic, показывающей создание простой нейронной сети:

```
Option Explicit
'the primary objective of this program is to show
'the propagation of a set of input signals.
'and, to attempt the formation of a least resistant
'path from that set of inputs to a set of correct outputs.
'to test it, use the toggle buttons to modify the behaviour
'of the network...
'green means true and red means false
'a 2d/1d neural network model was used instead of 3d or xd
'limits of the neural net
```

```
Global Const maxinputs = 3
Global Const maxoutputs = 3
Global Const maxArrayX = 35            'maximum x size of neural net
Global Const maxArrayY = 35            'maximum y size of neural net
Global Const maxInputConnections = 6    'maximum input connections per node
Global Const maxOutputConnections = 6   'maximum output connections per node
Global Const maxThreshold = maxInputConnections   'maximum resistence
Global Const maxConnections = 6 * maxArrayX * maxArrayY    'is proportional to the size of the net-
work
        'parameters of the neural net
Global rgbmode As Integer
Global drawingmode As Integer
Global usepunish As Boolean
Global usedimension As Integer
Global usepropagationmethod As Integer
Global usesignalboost As Boolean
Global executionmethod As Integer

'modifiers of the neural net
Global Const feedrate = 5          'rate a which the inputs are incremented
Global Const curvature = 7
Global Const lastActivatedCeiling = 2              'Threshold increment rate (every rIncRate that neuralNet
has not been used increase the r)
Global Const ThresholdInc = 10 'Threshold increment (increase neuralNet r by ri for rir
Global randcount As Integer
Global numberOfConnections As Integer
'miscellaneous variables
Global Const pi = 3.1415926535
Global Const e = 2.718281828
Global Const rads = pi / 180
Global Const topOfForm = 73         'top of the form
Global lastmousex As Integer
Global lastmousey As Integer
Global preend As Boolean
Global rgb2(255) As Double          'stores the colors of the palette
Global randomation(10000) As pt
'structures
Type pt
   x As Integer
   y As Integer
End Type
Type connection
   fromX As Integer
   fromY As Integer
   toX As Integer
   toY As Integer
   result As Integer    'either 1 or 0
   multiplier As Double 'will multiply the result by the value (between 0 and 1)
End Type
Type inputNeuron
   x As Integer
   y As Integer
   active As Boolean
```

```
End Type
Type outputNeuron
   x As Integer
   y As Integer
   active As Boolean
   isSolution As Boolean
   punishTemplate(maxArrayX, maxArrayY) As Double
End Type
'the neuron
Type neuron
   Threshold As Double
   lastActivated As Integer   'last activated (reaches maximum)
   tempCharge As Integer      'to keep track of the activity of the neuron
   'connections
   inputConnections(maxInputConnections) As Integer   'index of a connection
   outputConnections(maxOutputConnections) As Integer 'index of a connection
   numberOfOutputs As Integer 'maximum output connections for the node
   numberOfInputs As Integer 'maximum input connections for the node
End Type
Global inputs(maxinputs) As inputNeuron
Global outputs(maxoutputs) As outputNeuron
'the connections are external from the neural net
Global connections(maxConnections) As connection
'the neural network (just a 2d array of nodes)
Global neuralNet(maxArrayX, maxArrayY) As neuron
Public Sub InitializeNeuralNet()
   'initialize neural net
   Dim xx, yy, cc, xx2, yy2
   Dim maxdist, curdist
   'nullify all the connections (-1)
   For cc = 0 To maxConnections
      connections(cc).fromX = -1
      connections(cc).fromY = -1
      connections(cc).toX = -1
      connections(cc).toY = -1
      connections(cc).multiplier = 1
      connections(cc).result = 0
   Next
   'build the neural net
   For xx = 0 To maxArrayX
      For yy = 0 To maxArrayY
         'set initial values
         neuralNet(xx, yy).Threshold = 0
         neuralNet(xx, yy).lastActivated = lastActivatedCeiling + 1       'set last activated
         neuralNet(xx, yy).numberOfInputs = 0       'initialize maximum input connection index
         neuralNet(xx, yy).numberOfOutputs = 0      'initialize maximum output connection index
         neuralNet(xx, yy).tempCharge = 0
         'nullify all connections
         For cc = 0 To maxInputConnections
            neuralNet(xx, yy).inputConnections(cc) = -1
         Next
         For cc = 0 To maxInputConnections
            neuralNet(xx, yy).outputConnections(cc) = -1
```

```
        Next
      Next
    Next
    For xx = 0 To maxArrayX
      For yy = 0 To maxArrayY
        'build output connections between the nodes
        'depending on the number of dimensions (1d or 2d)
        If usedimension = 1 Then
          'only assign connections to the first row of nodes
          'to form the 1dimensional neural net
          If yy = 0 Then
            Call assignConnectionFromTo((xx), (yy), (xx + 1), (yy))
            Call assignConnectionFromTo((xx), (yy), (xx - 1), (yy))
          End If
        Else
          'connect all nodes to three neighbors on the right
          'Call assignConnectionFromTo((xx - 1), (yy - 1), (xx), (yy))
          'Call assignConnectionFromTo((xx - 1), (yy), (xx), (yy))
          'Call assignConnectionFromTo((xx - 1), (yy + 1), (xx), (yy))

          Call assignConnectionFromTo((xx), (yy), (xx + 1), (yy - 1))
          Call assignConnectionFromTo((xx), (yy), (xx + 1), (yy))
          Call assignConnectionFromTo((xx), (yy), (xx + 1), (yy + 1))
        End If
        'the output connection counter is too big (reduce by 1)
        'If neuralNet(xx, yy).numberOfOutputs - 1 >= 0 Then
         '   neuralNet(xx, yy).numberOfOutputs = neuralNet(xx, yy).numberOfOutputs - 1
        'End If
        'same for inputs
        'If neuralNet(xx, yy).numberOfInputs - 1 >= 0 Then
        '    neuralNet(xx, yy).numberOfInputs = neuralNet(xx, yy).numberOfInputs - 1
        'End If
      Next
    Next
    'the Threshold of the outputs should be small
    For xx = 0 To maxoutputs
      neuralNet(inputs(xx).x, inputs(xx).y).Threshold = 0
      Call createOutputPunishTemplate((xx))
    Next
    Randomize
End Sub
Private Sub assignConnectionFromTo(fromX As Integer, fromY As Integer, toX As Integer, toY As Inte-
ger)
    'connect two nodes
    'connect the node(fromx,fromy) to the node(tox,toy)
    'to connect two nodes save the outputs for the source node
    'and the inputs for destination node
    Dim aa, found
    'error-check the supplied (x,y)s are valid
    If inbounds((fromX), (fromY)) And inbounds((toX), (toY)) Then
      'save the outputs for the source node
      found = False
      'error-check that the connection does not already exist
```

```
        'For aa = 0 To numberOfConnections
        '    If connections(aa).fromX = fromX And connections(aa).fromY = fromY And
connections(aa).toX = toX And connections(aa).toY = toY Then
        '        found = True
        '    End If
        'Next
        'error-check that the we have enough unassigned connections
        If numberOfConnections < maxConnections And neuralNet(fromX, fromY).numberOfInputs < max-
InputConnections And neuralNet(fromX, fromY).numberOfOutputs < maxOutputConnections Then
          If found = False Then
            connections(numberOfConnections).fromX = fromX
            connections(numberOfConnections).fromY = fromY
            connections(numberOfConnections).toX = toX
            connections(numberOfConnections).toY = toY

            neuralNet(fromX, fromY).outputConnections(neuralNet(fromX, fromY).numberOfOutputs) =
numberOfConnections
            neuralNet(toX, toY).inputConnections(neuralNet(toX, toY).numberOfInputs) = numberOfCon-
nections
            neuralNet(fromX, fromY).numberOfOutputs = neuralNet(fromX, fromY).numberOfOutputs +
1
            neuralNet(toX, toY).numberOfInputs = neuralNet(toX, toY).numberOfInputs + 1
            numberOfConnections = numberOfConnections + 1
          End If
        End If
    End If
End Sub
Public Sub propagateNeuralNet()
    'main propagation routine
    'objective: to form an identifiable path of
    'least Threshold between a set of inputs and a
    'set of correct outputs avoiding the incorrect outputs
    'by trial and error
    Dim xx, yy, xx2, yy2, sum
    Dim retpt As pt
    Randomize
    'preend used to break out of the infinit loop
    preend = False
    randcount = 0
    'loop until prompted otherwise
    While (Not preend)
        'fire the inputs
        Call fireInputs
        'if any outputs have been reached, train the interneurons
        Call trainFromOutputValues
        'select the node to process
        retpt = getnextPT((xx), (yy))
        xx = retpt.x
        yy = retpt.y
        'check if we should propagate the signal
        sum = sumOfInputs((xx), (yy))
        If SumGreaterThanThreshold((xx), (yy), (sum)) Then
            Call propagateFrom((xx), (yy), (sum))
```

```
            Call drawnode((xx), (yy))
        Else
            'record the no-change
            Call increaseLastActivated((xx), (yy))
        End If
        DoEvents
    Wend
End Sub

Private Function SumGreaterThanThreshold(xx As Integer, yy As Integer, sum As Integer) As Boolean
    'is the sum of the inputs values greater than the threshold of the neuron
    SumGreaterThanThreshold = (sum > neuralNet(xx, yy).Threshold)
End Function

Public Function inbounds(xx As Integer, yy As Integer) As Boolean
    'validate the xx and yy for error checking
    inbounds = (xx >= 0 And xx <= maxArrayX And yy >= 0 And yy <= maxArrayY)
End Function

Public Sub fireInputs()
    Dim i, cc, xx, yy
    'increment the inputs
    For i = 0 To maxinputs
        If inputs(i).active Then
            xx = inputs(i).x
            yy = inputs(i).y
            neuralNet(xx, yy).Threshold = 0
            For cc = 0 To neuralNet(xx, yy).numberOfOutputs - 1
                connections(getConnectionIndex((xx), (yy), (False), (cc))).result = 1
            Next
            Call drawnode((xx), (yy))
        End If
    Next
End Sub
Private Sub trainFromOutputValues()
    Dim i, xx, yy
    'based on when the outputs were last activated
    'reward or punish the entire network
    For i = 0 To maxoutputs
        If outputs(i).active Then
            xx = outputs(i).x
            yy = outputs(i).y
            If neuralNet(xx, yy).lastActivated = 0 Then
                If Not outputs(i).isSolution Then
                    Call punishfrom((xx), (yy))
                Else
                    Call rewardall
                    neuralNet(xx, yy).Threshold = 0
                End If
                neuralNet(xx, yy).lastActivated = 1
            End If
        End If
    Next
```

```
End Sub
Public Sub rewardall()
   'globally increase the Threshold of each node by a small percent of its current value
   Dim xx, yy, cb, newvalue
   'For xx = 0 To maxArrayX
   '   For yy = 0 To maxArrayY
   '      cb = (100 - neuralNet(xx, yy).lastActivated) / maxThreshold
   '      newvalue = neuralNet(xx, yy).Threshold * 1.1
   '      Call NeuralNetManip.modifyThreshold((xx), (yy), (-10000), (newvalue))
   '      Call drawnode((xx), (yy))
   '   Next
   'Next
End Sub
Public Sub punishfrom(xxf As Integer, yyf As Integer)
   Dim xx, yy, cb, newvalue, maxdist, curdist
   Dim cb2, newvalue2, maxlastActivated, curlastActivated
   Dim distancequotient, lastActivatedquotient, avgrule
   Dim outputx
   'If isoutput((xxf), (yyf)) Then
   '   outputx = getoutput((xxf), (yyf))
   '   'use a very exhaustive method to train the nodes
   '   'create a mole hill around the output site
   '   'doesnt work too well
   '   For xx = 0 To maxArrayX
   '      For yy = 0 To maxArrayY
   '         curdist = outputs(outputx).punishTemplate(xx, yy)
   '         'increase Threshold
   '         If curdist > 0 Then
   '            'make sure the Threshold is atleast 1
   '            If neuralNet(xx, yy).Threshold < 0.01 Then
   '               neuralNet(xx, yy).Threshold = 1
   '            End If
   '            newvalue = (curdist * maxThreshold) + neuralNet(xx, yy).Threshold
   '         Else
   '            newvalue = -10000
   '         End If
   '         Call NeuralNetManip.modifyThreshold((xx), (yy), (-10000), (newvalue))
   '         Call drawnode((xx), (yy))
   '      Next
   '   Next
   'End If
End Sub
Private Function getnextPT(xx As Integer, yy As Integer) As pt
   'used to return the x,y of the node to process next
   'based on which processing mode is being used
   getnextPT.x = xx
   getnextPT.y = yy
   If usedimension = 0 Then
      'use the full two dimensional model
      If usepropagationmethod = 0 Then
         'totally randomly select some x,y
         getnextPT.x = Int(Rnd * maxArrayX)
         getnextPT.y = Int(Rnd * maxArrayY)
```

```
      ElseIf usepropagationmethod = 1 Then
         'select the next preassembled random set of x,y
         If randcount + 1 > 10000 Then
            randcount = 0
         Else
            randcount = randcount + 1
         End If
         getnextPT.x = randomation(randcount).x
         getnextPT.y = randomation(randcount).y
      ElseIf usepropagationmethod = 2 Then
         'increase y based on x
         'once x reaches maxArrayX increase it by 1
         'once y reaches maxArrayX reset both
         'same as using one while loop in another
         getnextPT.x = xx + 1
         If getnextPT.x > maxArrayX Then
            getnextPT.x = 0
            getnextPT.y = yy + 1
            If getnextPT.y > maxArrayY Then
               getnextPT.y = 0
            End If
         End If
      End If
   Else
      'use the reduced (simpler) one dimensional model
      If usepropagationmethod = 2 Then
         'y is always 0
         'x loops once it reaches maxArrayX
         getnextPT.y = 0
         getnextPT.x = xx + 1
         If getnextPT.x > maxArrayX Then
            getnextPT.x = 0
         End If
      ElseIf usepropagationmethod = 1 Then
         'y is always 0
         'use the preassembled random x's only
         If randcount + 1 > 10000 Then
            randcount = 0
         Else
            randcount = randcount + 1
         End If
         getnextPT.x = randomation(randcount).x
         getnextPT.y = 0
      ElseIf usepropagationmethod = 0 Then
         'y is always 0
         'x is totally random
         getnextPT.x = Int(Rnd * maxArrayX)
         getnextPT.y = 0
      End If
   End If
End Function
Private Sub increaseLastActivated(xx As Integer, yy As Integer)
   If neuralNet(xx, yy).lastActivated + 1 = lastActivatedCeiling Then
```

```
      'the counter has just reached its maximum
      neuralNet(xx, yy).lastActivated = lastActivatedCeiling + 1
   ElseIf neuralNet(xx, yy).lastActivated + 1 < lastActivatedCeiling Then
      neuralNet(xx, yy).lastActivated = neuralNet(xx, yy).lastActivated + 1
   End If
End Sub
Private Sub modifyThreshold(xx As Integer, yy As Integer, changeby As Double, newThreshold As Dou-
ble)
   Dim newval
   If xx >= 0 And yy >= 0 And xx <= maxArrayX And yy <= maxArrayY Then
      newval = -1
      If changeby <> -10000 Then
         newval = neuralNet(xx, yy).Threshold + changeby
      ElseIf newThreshold <> -10000 Then
         newval = newThreshold
      End If
      If newval >= 0 And newval <= maxThreshold Then
         neuralNet(xx, yy).Threshold = newval
         Call Form1.updateLabel((lastmousex), (lastmousey))
      ElseIf newval > maxThreshold Then
         neuralNet(xx, yy).Threshold = maxThreshold
      ElseIf newval < 0 Then
         neuralNet(xx, yy).Threshold = 0
      End If
   End If
End Sub
Public Function dist(xx As Double, yy As Double, xx2 As Double, yy2 As Double)
   'just return the distance from the first point to the second point
   dist = Sqr((xx - xx2) ^ 2 + (yy - yy2) ^ 2)
End Function
Public Function propagateFrom(xx As Integer, yy As Integer, sum As Integer) As Boolean
   'the propagation will fire a 1 into the result value of the output connections
   Dim i, tempx, tempy, cc
   Randomize
   'error check (somewhat redundant for confident code)
   If inbounds((xx), (yy)) Then
      'clear the input values
      For i = 0 To neuralNet(xx, yy).numberOfInputs - 1
         cc = getConnectionIndex((xx), (yy), (True), (i))
         If cc <> -1 Then
            connections(cc).result = 0
         End If
      Next
      'set the output values
      For i = 0 To neuralNet(xx, yy).numberOfOutputs - 1
         cc = getConnectionIndex((xx), (yy), (False), (i))
         If cc <> -1 Then
            connections(cc).result = 1
         End If
      Next
      neuralNet(xx, yy).lastActivated = 0
      'Call NeuralNetManip.modifyThreshold((xx), (yy), (-(sc / neuralNet(xx, yy).Charge)), (-10000))
      neuralNet(xx, yy).tempCharge = sum
```

```
    End If
End Function
Public Sub define2DInputLocations()
    Dim row, col, xx, yy
    'put the inputs in the same column
    col = 1
    row = Int(maxArrayY / maxinputs)
    For xx = 0 To maxinputs
        'define input locations
        inputs(xx).x = col
        inputs(xx).y = row * xx
        'activate the inputs
        If inputs(xx).active = False Then
            'Call runcommand((xx))
        End If
        Call Form1.setToolTipText((xx), (inputs(xx).x), (inputs(xx).y))
    Next
End Sub
Public Sub define2DOutputLocations()
    Dim row, col, xx, yy
    'put the inputs in the same column
    col = maxArrayX - 1
    row = Int(maxArrayY / maxoutputs)
    For xx = 0 To maxoutputs
        'define input locations
        outputs(xx).x = col
        outputs(xx).y = row * xx
        'activate the inputs
        If outputs(xx).active = False Then
            Call Form1.runcommand((xx))
        End If
        Call Form1.setToolTipText((xx), (outputs(xx).x), (outputs(xx).y))
    Next
End Sub
Public Sub define1DInputLocations()
    Dim row, col, xx, yy
    'define input locations
    'use only the first input
    inputs(0).x = Int(maxArrayX / 2)
    inputs(0).y = 0
    'deactivate other inputs
    'put the inputs in the same column
    For xx = 0 To maxinputs
        'define input locations
        inputs(xx).x = col
        inputs(xx).y = row * xx
        'activate the inputs
        If xx > 0 And inputs(xx).active = True Then
            inputs(xx).active = True
            Call Form1.runcommand((xx))
        End If
        Call Form1.setToolTipText((xx), (inputs(xx).x), (inputs(xx).y))
    Next
```

```
End Sub
Public Sub define1DOutputLocations()
   Dim row, col, xx, yy
   'define output locations
   'use only the first two outputs
   outputs(0).x = Int(maxArrayX / 2)
   outputs(0).y = 0
   oututs(1).x = Int(maxArrayX / 2)
   outputs(1).y = 0
   'deactivate other outputs
   'put the outputs in the same column
   For xx = 0 To maxinputs
      'define input locations
      outputs(xx).x = col
      outputs(xx).y = row * xx
      'deactivate other outputs
      If xx > 1 And outputs(xx).active = True Then
         outputs(xx).active = False
         Call Form1.runcommand((3 + xx))
      End If
      Call Form1.setToolTipText((3 + xx), (outputs(xx).x), (outputs(xx).y))
   Next
End Sub
Public Function isinput(xx As Integer, yy As Integer) As Integer
   isinput = -1
   'return the number of the corresponding input
   Dim i
   For i = 0 To maxinputs
      If xx = inputs(i).x And yy = inputs(i).y Then
         isinput = i
         Exit Function
      End If
   Next
End Function
Public Function isoutput(xx As Integer, yy As Integer) As Integer
   isoutput = -1
   'return the number of the corresponding output
   Dim i
   For i = 0 To maxoutputs
      If xx = outputs(i).x And yy = outputs(i).y Then
         isoutput = i
         Exit Function
      End If
   Next
End Function

Public Sub drawnode(xx2 As Integer, yy2 As Integer)
   Dim ch, rh
   Dim fh, fw
   Dim sw, sh
   Dim fromX, fromY, toX, toY, fromx2, fromy2, tox2, toy2
   Dim color
   fromX = 0
```

```
    fromY = 0
    toX = 0
    toY = 0
    fromx2 = 0
    fromy2 = 0
    tox2 = 0
    toy2 = 0
    If usedimension = 1 Then
        'If maxCharge > maxThreshold Then
        '    fw = Form1.ScaleWidth / maxArrayX
        '    fh = (Form1.ScaleHeight - topOfForm) / maxCharge
        'Else
        '    fw = Form1.ScaleWidth / maxArrayX
        '    fh = (Form1.ScaleHeight - topOfForm) / maxThreshold
        'End If
    Else
        fw = Form1.ScaleWidth / maxArrayX
        fh = (Form1.ScaleHeight - topOfForm) / maxArrayY
    End If
    sw = Form1.ScaleWidth
    sh = (Form1.ScaleHeight - topOfForm)
    If usedimension Then
        'Form1.Cls
        'ch = neuralNet(xx2, yy2).Charge * fh
        'rh = neuralNet(xx2, yy2).Threshold * fh

        'fromx = xx2 * fw
        'fromy = sh - ch + topOfForm - 10
        'tox = xx2 * fw + fw - (fw / 2)
        'toy = sh + topOfForm - 10
        'fromx2 = xx2 * fw + fw - (fw / 2)
        'fromy2 = sh - rh + topOfForm - 10
        'tox2 = xx2 * fw + fw
        'toy2 = sh + topOfForm - 10

        'Form1.Line (fromx, fromy)-(tox, toy), rgb2(neuralNet(xx2, yy2).Threshold), BF
        'Form1.Line (fromx2, fromy2)-(tox2, toy2), rgb2(neuralNet(xx2, yy2).Charge), BF
    Else
        If drawingmode = 0 Then
            color = rgb2(((neuralNet(xx2, yy2).tempCharge / maxThreshold) * 255))
        ElseIf drawingmode = 1 Then
            color = rgb2(((neuralNet(xx2, yy2).Threshold / maxThreshold) * 255))
        ElseIf drawingmode = 1 Then
            color = rgb2(((neuralNet(xx2, yy2).lastActivated / lastActivatedCeiling) * 255))
        End If
        Form1.Line (xx2 * fw, yy2 * fh + topOfForm)-(xx2 * fw + fw, yy2 * fh + fh + topOfForm), color,
BF
    End If
End Sub
Public Sub createOutputPunishTemplate(i As Integer)
    Dim maxdist, curdist, xx, yy
    'find the greatest distance from input1 and some outter edge of the array
    maxdist = 0
```

```
    For xx = 0 To maxArrayX
        For yy = 0 To maxArrayY
            curdist = dist((outputs(i).x), (outputs(i).y), (xx), (yy))
            If curdist > maxdist Then
                maxdist = curdist
            End If
        Next
    Next
    'create a punishment template for later
    For xx = 0 To maxArrayX
        For yy = 0 To maxArrayY
            'get the distance from the current node to the index node
            curdist = dist((xx), (yy), (outputs(i).x), (outputs(i).y))
            outputs(i).punishTemplate(xx, yy) = ((maxdist - curdist) / maxdist) ^ curvature
        Next
    Next
End Sub
Public Function getConnectionIndex(xx As Integer, yy As Integer, isinput As Boolean, branch As Inte-
ger) As Integer
    If isinput Then
        getConnectionIndex = neuralNet(xx, yy).inputConnections(branch)
    Else
        getConnectionIndex = neuralNet(xx, yy).outputConnections(branch)
    End If
End Function
Public Function sumOfInputs(xx As Integer, yy As Integer) As Integer
    Dim sum, i, cc
    sum = 0
    'calculate the sum of the values
    For i = 0 To neuralNet(xx, yy).numberOfInputs
        'get the location of the connection in the connections array
        cc = getConnectionIndex((xx), (yy), (True), (i))
        If cc <> -1 Then
            sum = sum + connections(cc).result * connections(cc).multiplier
        End If
    Next
    sumOfInputs = sum
End Function
```