ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Баймухамедов М.Ф.

ЖАСАНДЫ ИНТЕЛЛЕКТ: ҚАЗІРГІ ЗАМАНҒЫ ТЕОРИЯ ЖӘНЕ ТӘЖІРИБЕ

1 Бөлім

ARTIFICIAL INTELLIGENCE: MODERN THEORY AND PRACTICE

Volume 1

Алматы, 2020

ӘОЖ 004.8(075.8)
КБЖ 32.923я73
Б 20

Академик З.Алдамжар атындағы Қостанай әлеуметтік-техникалық университетінің Ғылыми кеңесі шешімімен басып шығаруға ұсынылады

Рецензенттер:

Құрманов А.К. – техника ғылымдарының докторы, Ахмет Байтұрсынов атындағы Қостанай мемлекеттік университетінің профессоры

Медетов Н.А. – физика-математика ғылымдарының докторы, профессор, Қостанай мемлекеттік педагогикалық университетінің ғылым жөніндегі проректорі

Баймухамедов М.Ф., Джаманбалин Қ.Қ., Акгул М.К.

Б 20 **Жасанды интеллект: қазіргі заманғы теория және тәжірибе / Artificial Intelligence: Modern Theory and Practice:** оқу құралы / М.Ф. Баймухамедов, 1 бөлім – Алматы: «Бастау», 2020. – 248 бет.

ISBN 978-601-7991-32-6

Осы оқу құралының бірінші томы оқырмандарды жасанды интеллект тарихымен, білім беру үлгілерімен, сараптамалық жүйелермен және нейрондық желілермен таныстырады. Жасанды интеллект жүйесінің негізгі ұғымдары, әдістері және алгоритмдері қарастырылады. Логикалық және функционалдық бағдарламалау негіздері баяндалады. Білім беру модельдері және олармен жұмыс істеу әдістері, сараптама жүйелерін жасау және жасау әдістері қарастырылған. Кітап оқырманға пәндік саладағы деректер қорын логикалық жобалау және ProLog тілінде бағдарламалау дағдыларын меңгеруге көмектеседі. Материалды баяндау көп иллюстрациялар мен практикалық мысалдармен сүйемелденеді. Оқулықтың екінші томы жасанды интеллектті практикалық қолдануға арналған.

The first volume of this tutorial introduces readers to the history of artificial intelligence, knowledge representation models, expert systems, and neural networks. The basic concepts, methods and algorithms of artificial intelligence systems are considered. The basics of logical and functional programming are outlined. Models of knowledge representation and methods of working with them, methods of development and creation of expert systems are considered. The book will help the reader to master the skills of logical design of databases of the subject area and programming in the ProLog language. The presentation of the material is accompanied by a large number of illustrations and practical examples.

The second volume of the manual is devoted to the practical application of artificial intelligence.

The textbook is intended for students and undergraduates of technical specialties.

ӘОЖ 004.8(075.8)
КБЖ 32.923я73

ISBN 978-601-7991-32-6

© Баймухамедов М.Ф., 2020
© «Бастау», 2020

МАЗМҰНЫ
CONTENTS

ЖАСАНДЫ ИНТЕЛЛЕКТ: ҚАЗІРГІ ЗАМАНҒЫ ТЕОРИЯ ЖӘНЕ ТӘЖІРИБЕ

Аббревиатуралардың толық жазылуы

ЖИ – жасанды интеллект
ЭЕМ – Электронды есептеу машинасы
КСРО – Кеңестік Социалистік Республикалар Одағы
АҚШ – Америка Құрама Штаттары
БНЖ – білімге негізделген жүйелер
ККП – көп қабатты перцептрон
MNN – көп қабатты нейрондық желі
RNN – қайталанатын нейрондық желі
URL – Uniform Resource Identifier
ЖНЖ – жасанды нейрондық желіні
ХЖ – Хопфилд желі
СЖ – сараптамалық жүйе
ПБ – процедуралық білім
ДБ – декларативті білім
ББББЖ – білім базасын басқару жүйесі
ББ – білім базасы
FRL – Frame Representation Language
МБМ – модель басқаратын модуль
ҮБМ – үлгілермен басқарылатын модельдері
АҚ – аспаптық құралды
АЖ – ақпараттық жүйелер
ТТ – табиғи тіл

Профессор
Баймухамедов Малик Файзуллаұлының
75-жылдығына
арналып баспадан шығады

АЛҒЫСӨЗ

Соңғы уақытта ақпараттық жүйелерге қойылатын талаптардың артуынан туындаған жасанды интеллектке қызығушылықтың өсуі байқалады. Адамзат Интернеттің дамуымен салыстыруға болатын жаңа ақпараттық революцияға үнемі жылжиды.

Жасанды интеллект – информатика бағыты, оның мақсаты қолданушыға-бағдарламашы емес адамға табиғи тілдің шектеулі жиынтығында ЭЕМ-мен қарым-қатынас жасай отырып, дәстүрлі түрде интеллектуалды міндеттерді қоюға және шешуге мүмкіндік беретін аппараттық-бағдарламалық құралдарды әзірлеу болып табылады.

Жасанды интеллект тарихы жаңа ғылыми бағыт ретінде XX ғасырдың ортасында басталады. Осы уақытқа дейін оның пайда болуының көптеген алғышарттары қалыптасты: философтардың арасында адам табиғаты мен әлемді тану процесі туралы даулар бұрыннан жүрді, нейрофизиологтар мен психологтар адам миы мен ойлаудың жұмысына қатысты бірқатар теорияларды әзірледі, экономистер мен математиктер оңтайлы есептер және формальды түрде әлем туралы білім беру мәселелерін қойды; ақырында есептеулердің математикалық теориясының – алгоритмдер теориясының іргетасы пайда болды және алғашқы компьютерлер құрылды.

Осы құралдың мақсаты жасанды интеллектте қолданылатын негізгі бағыттар мен әдістерді баяндау, сондай-ақ оларды адам қызметінің түрлі салаларында пайдалану мүмкіндігін анықтау болып табылады. Бұл оқу құралы алты бөлімнен тұрады. Бірінші кезеңде жасанды интеллектке қысқаша кіріспе келтіріледі, оның ғылыми бағыт ретінде даму тарихы қарастырылады, жасанды интеллекттің негізгі салалары көрсетіледі, жасанды интеллект саласындағы зерттеулердің тарихы мен негізгі бағыттары қарастырылады.

Екінші бөлім Пролог тілінде бағдарламалау негіздеріне арналған, мұнда Прологтың декларативті тіл ретінде сипатталуы, Пролог тіліндегі бағдарламалардың құрылымы мен негізгі элементтерінің сипатталуы, предикат ұғымы беріледі, Прологиядағы рекурсия мен кесу фактілер мен ережелер қарастырылды.

Үшінші тарауда ықтималдық пайымдаулар қарастырылды, анық емес логика теориясының негіздері беріледі, Байесов желілері, Монти Холл парадокс сипатталған.

Төртінші тарауда нейрондық желі ұғымы берілген, нейрондық желілерді құру принципі сипатталған, нейрондық желілерді пайдалану ерекшеліктері қарастырылған, тану есептерінде нейрондық желілер сипатталған, сондай-ақ нейрондық желілерді жобалау, оқыту және бейімдеу.

Бесінші тарауда сараптамалық жүйелерді әзірлеудің теориялық және практикалық мәселелері қарастырылады, алгоритмдік модельдер, білімді ұсынудың логикалық модельдері, үлгілермен басқарылатын өнім модельдері мен модульдер сипатталған. Алтыншы тарау семантикалық желілерге арналған.

Қаралды типтері семантикалық желілер, мәліметтер қорын құру, сипатталған фактілер мен ережелер семантикалық желілер, құрастыру мәселелері семантикалық желілер, түсінік берілген семантикалық тор қаралды қағидаттары мен ағымдағы жағдайы семантикалық тор.

Оқу құралында көптеген мысалдар мен жаттығулар қамтылған. Материалды оқуға ыңғайлы болу үшін глоссарий ұсынылған.

Авторлар Қостанай мемлекеттік педагогикалық университетінің аға оқытушысы Айткенова Аян Алтаеваға оқулықтың мәтінін қазақ тіліне аудару кезінде көрсеткен көмектері үшін алғыс білдіреді.

Кіріспе

Игнаси Белда өзінің «Ақыл-ой, машиналар және математика» деген жұмысында былай деп жазды: «Жасанды интеллект (ЖИ) бірте-бірте біздің өмірімізге енді. Адам сияқты креативтілік, сезім және эмоциялық интеллект деңгейіне ие машиналар пайда болған күні ерте ме, кеш пе кетеді. Бұл күні біз жалғыз емес екенін түсінеміз».

Біздің заманымыздың ең көрнекті белгілерінің бірі – адам қызметінің барлық салаларына енген компьютерлер. Компьютерлердің өмір сүруінің барлық кезеңінде олардың өнімділігінің тұрақты өсуі тіпті ұялы құрылғылардың зияткерлік функцияларын жүзеге асыруға мүмкіндік берді. Ұялы телефонның камерасы көріністі танып, фокус пен экспозицияны сәйкесінше реттейтініне біз бұрыннан үйреніп қалдық. Көптеген көлік жүргізушілері кептелістерден аулақ болуға көмектесетін навигациялық жүйесіз қала ішінде қозғалуды елестете алмайды. Егер 20-25 жыл бұрын Интернет ақпараттық кеңістікті жаулап ала бастағанда, қажетті ресурстарға қол жеткізу үшін сіз URL мекен-жайын және тіпті интернет-ресурстарға («сары беттер») жарияланған нұсқаулықтарды біліп алуыңыз керек болса, енді іздеу жүйесінде фразаны теру жеткілікті ол тек тиісті ресурстарға сілтемелер тізімін ғана емес, сонымен қатар кейбір жағдайларда қолданушыны қызықтыратын сұрақтарға жауап береді. Осының барлығы зияткерлік жүйелердің мысалдары.

Қазіргі уақытта жасанды интеллект бойынша зерттеулерде алты негізгі бағыт бөлінген [1]:

1. Білімді ұсыну. Осы бағыт шеңберінде ЖИ жүйесінің жадында білім беру және формализациямен байланысты міндеттер шешіледі. Ол үшін білімді ұсынудың арнайы үлгілері мен білімді сипаттау тілдері әзірленеді, білімнің түрлі түрлері енгізіледі. Білім беру мәселесі ЖИ жүйесі үшін негізгі проблемалардың бірі болып табылады, өйткені мұндай жүйенің жұмыс істеуі оның жадында сақталатын проблемалық сала туралы білімге сүйенеді.

2. Білімді манипуляциялау. Тапсырманы шешу кезінде білімді қолдануға болатындай, ЖИ жүйесін үйрету және олармен жұмыс істеу керек. Осы бағыт шеңберінде білімді толық емес сипаттамалар негізінде толықтыру тәсілдері әзірленеді, қолда бар білімнің негізінде шынайы және шындыққа ұқсас қорытынды жасау әдістері жасалады, білімге сүйенетін және адами пайымдаулардың ерекшеліктерін имитациялайтын пайымдаулар модельдері ұсынылады. Білімді манипуляциялау білім берумен тығыз байланысты және бұл екі бағытты тек шартты түрде бөлуге болады.

3. Қарым-қатынас. Бұл бағыттағы міндеттер шеңберіне мыналар кіреді: табиғи тілдегі байланысты мәтіндерді түсіну және синтездеу

мәселесі, сөйлеуді түсіну және синтездеу, адам мен ЖИ жүйесі арасындағы коммуникация модельдерінің теориясы. Зерттеу негізінде бұл бағытта лингвистикалық процестерді, сұрақ-жауап жүйелерін, диалогтық жүйелерді және ЖИ басқа да жүйелерін құру әдістері қалыптасады, олардың мақсаты адамның ЖИ жүйесімен қарым-қатынасы үшін қолайлы жағдайларды қамтамасыз ету болып табылады.

4. Қабылдау. Бұл бағыт білім базасында көру көріністері туралы ақпаратты ұсыну әдістерін әзірлеуді, көру көріністерінен олардың мәтіндік сипаттамасына көшу әдістерін және кері өту әдістерін жасауды, ЖИ жүйелеріндегі ішкі көріністер негізінде көру көріністерін тудыру үшін құралдар жасауды қамтиды.

5. Оқыту. ЖИ жүйелерінің бұрын кездеспеген міндеттерді шешуге қабілеттілігін дамыту үшін проблемалық жағдайды сипаттау бойынша немесе оны бақылау бойынша есептер шарттарын қалыптастыру әдістері, жеке есептерді (мысалдарды) белгілі шешуден жалпы міндетті шешуге көшу әдістері, бастапқы есептің декомпозициясының неғұрлым ұсақ және ЖИ жүйелері үшін белгілі әдістерін жасау әзірленеді. Бұл бағытта ЖИ әлі де аз жасалған.

6. Мінез-құлқы. ЖИ жүйелері кейбір қоршаған ортада әрекет етуі тиіс болғандықтан, оларға қоршаған ортамен, өзге де ЖИ жүйелерімен және адамдармен барабар өзара іс-қимыл жасауға мүмкіндік беретін кейбір мінез-құлық рәсімдерін әзірлеу қажет. Бұл бағыт ЖИ-де өте нашар әзірленген.

Іс жүзінде ЖИ мүмкіндіктерінің спектрі шексіз: ғарыштық зерттеулер, әскери іс, робототехника, өнеркәсіп, ауыл шаруашылығы, көлік, медицина, білім және т.б.

Мысалы, қазіргі заманғы жасанды интеллект жүйелері әртүрлі ақпарат және құрылғылар датчиктерінің едәуір көп санының арқасында робототехникалық құрылғыларды тиімді басқаруға қабілетті.

Роботтар – бұл адам еңбегін автоматтандыруға арналған электротехникалық құрылғылар.

ЖИ әдістері экстремалды жағдайларда жұмыс істейтін робот техникалық кешендер үшін алгоритмдер мен аппараттық шешімдерді жасауға мүмкіндік береді. Бұл кешендердің айрықша ерекшелігі нейрондық желілер мен анық логика негізінде тәуелсіз зияткерлік басқару жүйесі, экстремалды жағдайға байланысты топографиялық және аппараттық бейімделу мүмкіндігі бар робототехникалық жүйені басқарудың зияткерлік жүйесін құру болады. ЖИ әдістерін қолданудың практикалық нәтижесінің мысалы тұрмыстық және өнеркәсіптік мақсаттағы үй-жайларда жұмыс істеуге арналған, бейнебақылаудан радиациялық бақылауға дейінгі кіру жүйелерінің кең спектрі бар және осы үй-

жайлардың қауіпсіздігін қамтамасыз етуге арналған роботты құру болып табылады. Роботты интеллектуалды басқару жүйесі автоматты түрде үй-жайларды жоспарлауға бейімделеді, қорғалатын үй-жайдың жоспарлануын ескере отырып, роботтың қозғалысын автоматты басқаруды қамтамасыз етеді, онлайн мониторингті қамтамасыз ете отырып, деректерді зияткерлік өңдеуді жүргізеді және басқару шешімін, соның ішінде төтенше жағдайларда телекоммуникациялық жүйе бойынша хабарламаларды беруді дербес қабылдайды.

ЖИ саласындағы соңғы жетістіктерді келесі коммерциялық жобалармен ұсынуға болады [2]:

− NASA-да әзірленген Remote Agent бағдарламасы жер маңындағы орбитадан алыс қашықтықтағы ғарыш аппараттарының жұмысын кешенді басқару үшін, оның ішінде олардың пайда болуына қарай ақауларды диагностикалау және жою үшін пайдаланылады.

− Диагностика. Медициналық диагностикалық бағдарламалар медицинаның бірнеше саласында тәжірибелі дәрігер деңгейіне жетті.

− Жабдықтау жоспарлау. 1991 жылы АҚШ армиясында Парсы шығанағындағы дағдарыс кезінде Dart (Dynamic Analysis and Re-planning) жүйесі өрістетілді. Бұл жүйенің әзірлеушілері бұл бір қолдану жасанды интеллектке олардың 30 жылдық инвестицияларын ақтады деп мәлімдеді.

ЖИ саласындағы соңғы онжылдықтағы заманауи жетістіктер келесі әзірлемелермен ұсынылды:

1. IBM компаниясының Deep Blue бағдарламасы шахмат матчында әлем чемпионы Каспаровты жеңіп алған алғашқы бағдарлама болды.

2. Alvinn компьютерлік көру жүйесі қозғалыс жолағын ұстана отырып, көлік жүргізуге үйретілді. 2850 миль бойы жүйе 98% уақыт ішінде автомобильді басқаруды қамтамасыз етті.

3. Қытайда жасанды интеллект танымал жүргізуші болды. Ол жаңалықтарды ағылшын тілінде оқиды және «Синьхуа» ақпараттық агенттігінің қызметкері Чжан Чжао – нақты адам ретінде көрінеді. Симуляцияны толығымен компьютерде құрастырды: дикторлардың мәтіндерін, мимиканы және нақты адамдардың еріндерінің қозғалысын біріктірді. Бағдарламаға нақты адам бар бейнелерді жүктейді, ал ЖИ машиналық оқыту көмегімен оны одан әрі эфирде ойнату үшін өз бетінше қимылдарды, әңгіме мәнерін және басқа да мәліметтерді талдайды.

4. Солтүстік Каролина Университеті жанындағы фармацевтика мектебінде ғалымдар екі нейрожелілерден ЖИ құрды. Молекулалардың құрылымы мен қасиеттері туралы деректерді, сондай-ақ қажетті әсерді бір жүктейді. Екінші нейрондық желі біріншісінен үйренеді: бұл деректерді игереді және мүмкін шешімдерді таңдайды.. Қазір ЖИ 1,7 миллионнан астам молекуламен жұмыс істейді. Бұл жаңа дәрі-дәрмектерді әзірлеу

процесін айтарлықтай жеделдетуге көмектеседі, ал табысты нәтижелер, мысалы, жаңа антибиотиктер жасау үшін негіз бола алады.

5. Шығыс Азиядағы Microsoft Application and Services Group сарапшылары эмоцияны «сезініп» және адамдармен «Адамша» сөйлесе алатын жасанды бағдарлама жасады. Xiaoice атты ЖИ 17 жасар қыз сияқты сұрақтарға жауап береді. Егер ол тақырыпты білмесе, ол әкелуі мүмкін. Егер оны өтірікке шақырса, олар ашуланады немесе ұялады. Xiaoice саркастикалық, шірік және шыдамсыз болуы мүмкін – бұл қасиеттер бізге белгілі. Xiaoice күтпеген адаммен қарым-қатынас жасауға өте ұқсас етеді. Мұндай ЖИ диковинкаға дейін және қытайлықтар онымен көңіл көтергісі келген кезде немесе скучно кезде сөйлеседі. Бірақ оны жасаушы Xiaoice жақсарту бойынша жұмыс істейді. Кім біледі, мүмкін Xiaoice Skynettің әжесі болады.

6. Мәскеу мемлекеттік университетінің ғалымдар тобы HautAI OU технологиялық стартапымен бірге photoageclock жасанды интеллектін жасады, ол адамның көз алдында хронологиялық жасын анықтай алады. Көз айналасындағы Ареал өмір салтының немесе басқа да факторлардың әсеріне аз әсер етеді және жасымен табиғи жолмен өзгереді, сондықтан ғалымдар командасы tCheat үшін осы аймақты таңдады. Нейрондық желі көз айналасындағы 8 500 фотосуретті зерттеді және жасы екі жылға дейінгі дәлдікпен анықтауды үйренді.

Егер XX ғасыр коммуникация құралдарының дамуымен және есептеу технологияларын танымал етумен ерекшеленсе, XXI ғасыр жасанды интеллект пен робот техникасын дамытудың жүз жылдығына айналады. Жасанды интеллект ең үздік заңгерлердің бүкіл командасын алмастыра алады, әлемдік чемпиондарда шахматты жеңеді, санаулы минуттарда бейнематериалдардың терабайттарын терең tCheatды жүргізеді және тіпті одан да жетілдірілген жасанды интеллект жасауды өз бетінше үйренеді. Робот техникасы мен адам-орнын алмастыруға келетін болсақ, банк мекемелері кассирлерді терминалдармен ауыстыра отырып, баяғыда жаппай қысқартады, ал жақын арада супермаркеттердегі экскурсоводтар, бармендер, даяшылар, жинаушылар мен кассирлерді ауыстыру күтілуде.

Қорытындылай келе, ЖИ саласы қарқынды дамып келе жатқанын атап өткен жөн. Алдағы жылдары тестілік режимде жұмыс істейтін жобалар іске қосылуы мүмкін: ұшқышсыз такси, суасты қайықтар немесе басқарылатын ЖИ жойғыштар немесе әуежайларда тексеру жүргізетін виртуалды шекарашылар.

1-тарау. ЖАСАНДЫ ИНТЕЛЛЕКТКЕ КІРІСПЕ

1.1. Жасанды интеллект ұғымы

Жасанды интеллект (ЖИ) – бұл информатиканың жаңа бағыты, оны зерттеу пәні алдын ала белгілі заңдарға бағынышты адамның кез келген зияткерлік қызметі болып табылады. Бұл бағытты «информатиканың үлкен ұлы» деп атайды, өйткені ол шешпеген көптеген міндеттер жасанды интеллект аясында өз шешімін бірте-бірте табады. Информатика пәні ақпаратты өңдеу болып табылатыны белгілі. Осы өңдеуден алынған қарапайым және дәл алгоритмдік әдістердің көмегімен орындалуы мүмкін емес және үлкен жиын бар жағдайлар (міндеттер) ЖИ аймағына жатады**.**

ЖИ адам ойлау процесі туралы білімге сүйенеді. Бұл ретте адам миының қалай жұмыс істейтіні белгілі емес, алайда ЖИ элементтері бар тиімді жұмыс істейтін бағдарламаларды әзірлеу үшін бүгінгі күні Ғылым бар адам интеллектінің ерекшеліктері туралы білімдері жеткілікті. Сонымен қатар, ЖИ адам миының жұмысын дәл көшіруге тырыспайды, ал есептеу техникасы құралдарының көмегімен оның функцияларын модельдеуге тырысады.

Өзінің туған сәтінен бастап ЖИ ақпараттармен және кибернетикамен, когнитивті ғылымдармен, логика мен математикамен, лингвистика мен психологиямен, биологиямен және медицинамен өзара әрекеттесетін пәнаралық бағыт ретінде дамиды (1.1-сурет).

Информатика және кибернетика. Көптеген сарапшылар жасанды интеллектке информатика мен кибернетикадан келді. Сондай-ақ, компьютерлік ғылымдағы дәстүрлі әдістермен шешілмейтін көптеген комбинаторлық мәселелер жасанды интеллектке көшті. Сонымен қатар, ЖИ-пен алынған нәтижелер бағдарламалық жасақтаманы жасау кезінде алынады және Computer Science (информатика) бөлігіне айналады.

Когнитивті ғылымдар. Когнитивті ғылымдар – бұл білім туралы ғылым. Жасанды интеллект біліммен де айналысады. Бірақ когнитивті ғылымдар тек қана ақпараттық және нейробиологиялық тәсілдерді ғана емес, сонымен қатар білімді қолданудың әлеуметтік және психолингвистік аспектілерін қарастырады.

Логика және математика. Логика білім берудің барлық белгілі формализмдерінің, сондай-ақ Lisp және Prolg сияқты бағдарламалау тілдерінің негізінде жатыр. Жасанды интеллектін есептерін шешу үшін дискретті математика, ойындар теориясы, операциялар теориясы әдістері қолданылады. Өз кезегінде, жасанды интеллект теоремаларды дәлелдеу үшін, математиканың әртүрлі салаларындағы есептерді шешу үшін қолданылады: геометрия, интегралды есептеулер.

Психология және лингвистика. Соңғы уақытта ЖИ мамандары оны модельдеу мақсатында адамдардың мінез-құлқының психологиялық аспектілеріне қызығушылық танытты.

Психология құнды бағалау моделін құруға, субъективті шешімдер қабылдауға көмектеседі. Қарым-қатынас психологиясы қызығушылық танытады «адам-компьютер», психолингвистика. Компьютерлік лингвистика – бұл бір жағынан табиғи және жасанды тілдерді өңдеудің математикалық әдістеріне және екінші жағынан тілдің феноменологиясына негізделген ЖИ-тің бөлігі.

1.1-сурет. АИ-дің басқа пәндермен байланысы

Биология және медицина мидың, көру жүйесінің, есту жүйесінің және басқа да табиғи датчиктердің жұмысын жақсы оқып, түсінуге және олардың жұмысын модельдеу саласында жаңа серпін беруге мүмкіндік береді.

Бүгінгі күні, жасанды интеллекттің бірыңғай анықтамасы жоқ, табиғи интеллекттің де бірыңғай анықтамасы жоқ.

Осы ғылыми саладағы көптеген көзқарастардың ішінде қазір үшеуі басым [3].

1. ЖИ саласындағы зерттеулер іргелі зерттеулер болып табылады, оның шеңберінде дәстүрлі түрде интеллектуалды болып саналған және бұрын формализация мен автоматтандыруға келмейтін есептерді шешудің модельдері мен әдістері әзірленеді.

2. ЖИ-бұл ЭЕМ-де есептерді шешудің жаңа идеяларымен, бағдарламалаудың принципті басқа технологиясын әзірлеумен, ЭЕМ архитектурасына көшумен, классикалық архитектураны қабылдамайтын, ол алғашқы ЭЕМ-ге дейін өрлеумен байланысты информатиканың жаңа бағыты.

3. ЖИ саласындағы жұмыстардың нәтижесінде есептерді шешетін көптеген қолданбалы жүйелер пайда болады, олар үшін бұрын құрылған жүйелер жарамсыз.

Бірінші тәсілді бейнелеу үшін калькулятормен мысал келтіруге болады. Ғасыр басында көп мәнді сандармен арифметикалық есептеулер аз дарынды тұлғалардың еңбегі мен ақылда мұндай арифметикалық әрекеттерді жасау қабілеті табиғаттың бірегей сыйы болып саналып, ғылыми зерттеулердің объектісі болып табылды. Қазіргі уақытта калькулятордың өнертабысы бұл қабілетті тіпті үшінші сынып оқушысы қол жетімді етті. ЖИ-де бірдей: ол бұрын қалыптаспайтын міндеттерді шеше отырып, адамның зияткерлік мүмкіндіктерін күшейтеді.

Екінші тәсілді көрнекілеу үшін бесінші буын ЭЕМ құру талпынысымен тарихты қарастыруға болады. Жапония 80-ші жылдардың ортасында бесінші буынды ЭЕМ құру бойынша өршіл жобаның басталғаны туралы жариялады. Жоба негізінде ПРОЛОГ тілін аппараттық іске асыру идеясы қойылған. Дегенмен, жоба сәтсіз аяқталды, бірақ бағдарламалау тілі ретінде ПРОЛОГ тілінің дамуы мен таралуына күшті әсер етті. Сәтсіздіктің себебі бір тіл (жеткілікті әмбебап болсын) барлық тапсырмалар үшін жалғыз шешімді қамтамасыз ете алады. Тәжірибе көрсеткендей, барлық есептерді шешу үшін бағдарламалаудың әмбебап парадигмалары ойлап таппады және ол пайда болуы екіталай. Бұл әрбір міндет - мұқият зерттеуді және арнайы тәсілді талап ететін пәндік саланың бөлігі. ЭЕМ жаңа архитектурасын құру әрекеттері параллельді және таратылған есептеулермен, нейрокомпьютерлермен, ықтимал және анық емес процессорлармен байланысты.

Сараптама жүйелерін құру саласындағы жұмыстарды ЖИ-тегі үшінші, неғұрлым прагматикалық бағытқа жатқызуға болады. Сараптамалық жүйелер – бұл арнайы білімді қолдануды талап ететін зияткерлік қызметтің тар салаларында адам-маманды алмастыратын бағдарламалық кешендер. Медицина саласындағы СЖ-ді құру (MYCIN түрі) білімді ең алыс аудандарға таратуға мүмкіндік береді. Осылайша, телекоммуникациялық қолжетімділікпен кез келген ауылдық дәрігер тар мәселе бойынша маманмен қарым-қатынасты ауыстыратын жүйеден кеңес ала алады.

КСРО-да ЖИ пайда болған сәттен бастап өз жақтаушыларын тапты. Алайда, бұл тәртіп бірден ресми мойындалмады. ЖИ «жалған ғылым» деп саналатын кибернетиканың қосалқы саласы ретінде сынға алынды.

Бір сәтке дейін теріс рөлді және «жасанды интеллект» эпатирлеуші атауын ойнады. Сонымен, Ғылым академиясының Президиумында «табиғи жетіспейтіндер» жасанды интеллектпен айналысады деген әзіл айтылды.

Алайда, бүгінгі күні ЖИ-бұл Ресейде ресми танылған ғылыми бағыт, «басқарушы жүйелер мен машиналар» және «ЖИ жаңалықтары» журналдары шығарылады, ғылыми конференциялар мен семинарлар өткізіледі. 200-ге жуық мүшесі бар Ресей қауымдастығы бар, оның президенті доктор техн.ғылым докторы Д.А.Поспелов, ал құрметті Президент РҒА академигі Г.С.Поспелов. Ресей Федерациясы Президентінің информатика және БТ бойынша Кеңесі жанындағы Ресей жасанды интеллект институты бар. РҒА шеңберінде «жасанды интеллект» мәселесі бойынша Ғылыми кеңес бар. Осы Кеңестің қатысуымен ЖИ, аудармалар тақырыбы бойынша көптеген кітаптар шығарылды. Д.А.Поспелов, Литвинцева және Кандрашина – білім беру және өңдеу саласында, Э.В.Попов және Хорошевскийдің – табиғи тіл мен сараптамалық жүйелерді өңдеу саласында, Аверкин мен Мелихов – анық емес логика және анық емес жиындар саласында, Стефанюк – білім алушылар жүйесінде, Кузнецов, Финн және Вагин – логика және білім беру саласында жақсы танымал.

1.2. Ғылым ретінде жасанды интеллект дамуының тарихи бағыттары

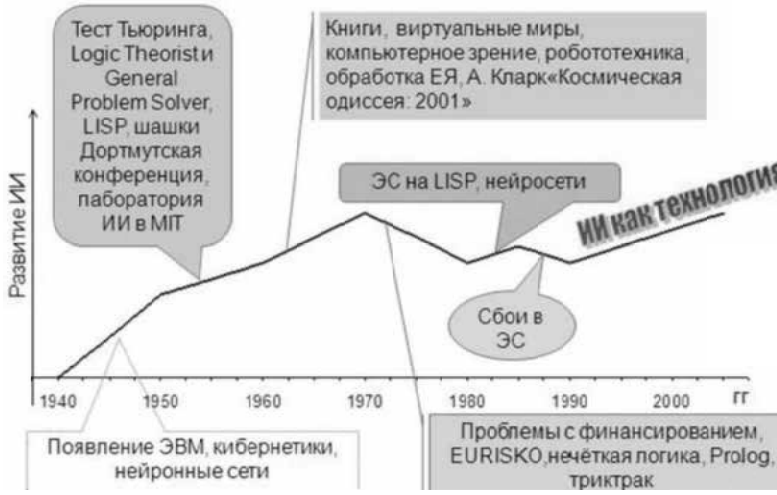
Күрделі есептерді шешу және ежелгі дәуірден ауадағы виталдың ойлау қабілетін модельдеу үшін адам санасының жасанды ұқсастығын құру идеясы.

Ежелгі Египетте Амун құдайының «тірі» механикалық мүсіні жасалды. Гомерде, Илиадада Гефестус құдайы адамоидты тіршілік иелері-автоматтар жасады.

Әдебиетте бұл идея бірнеше рет көтерілді: Галатеи Пигмалионнан Карло Папасы Буратиноға дейін. Алайда жасанды интеллектінің негізін қалаушы ортағасырлық испан философы, математик және ақын Р.Луллий (1235-б. 1315 ж.) болып саналады.

XVIII ғ. Лейбниц (1646-1716) және Р.Декарт (1596-1650) осы идеяны бір-біріне қарамастан, барлық ғылымдарды жіктеудің әмбебап тілдерін ұсына отырып дамытты. Бұл идеялар жасанды интеллект жасау саласындағы теориялық әзірлемелердің негізіне алынды (1.2-сурет).

Ғылыми бағыт ретінде жасанды интеллекттің дамуы ЭЕМ құрылғаннан кейін ғана мүмкін болды. Бұл XX ғ. 40-жылдары болды.



1.2-сурет. Ғылыми бағыт ретінде АИ дамуындағы кезеңдер

Сонымен бірге, Н.Винер (1894-1964) өзінің жаңа жұмысын – кибернетика бойынша іргелі жұмысын жасады.

Жасанды интеллект (artificial intelligence) термині 1956 жылы Станфорд университетінде (АҚШ) осындай атаумен өткен семинарда ұсынылған. Семинар есептеуіш емес, логикалық есептерді әзірлеуге арналды. Жасанды интеллект тәуелсіз ғылым саласы ретінде танылғаннан кейін көп ұзамай екі негізгі бағытқа бөлінді: нейробернетика және «қара жәшіктің» кибернетикасы. Және тек қазіргі уақытта осы бөліктерді қайтадан біртұтас біріктірудің беталыстары байқала бастады.

КСРО-да 1954 жылы ММУ-де профессор А.А.Ляпуновтың (1911-1973) басшылығымен "автоматтар және ойлау" семинары өз жұмысын бастады. Бұл семинарға ең ірі физиологтар, лингвистер, психологтар, математиктер қатысты. Бұл кезде Ресейде жасанды интеллект дүниеге келді деп есептеледі. Шетелде де "қара жәшіктің" нейрокибернетика және кибернетика бағыттары анықталды.

1956-1963 жылдары адам ойлауының модельдері мен алгоритмдерін қарқынды іздеу және алғашқы бағдарламаларды әзірлеу жүргізілді. Қазіргі ғылымдардың ешқайсысы – философия, психология, лингвистика – мұндай алгоритмді ұсына алмайды. Сонда кибернетика өз модельдерін құруды ұсынды. Әр түрлі тәсілдер жасалды және сыналды.

ЖИ саласындағы алғашқы зерттеулер шахматта ойнауға арналған бағдарламаны құрумен байланысты, өйткені шахмат ойнату қабілеті жоғары интеллектінің көрсеткіші болып табылады. 1954 жылы

американдық ғалым Ньюэлл осындай бағдарламаны құруды ойлады. Шеннон ұсынды, ал Тьюринг осындай бағдарламаны құру әдісін нақтылады. Американдық Шоу мен Саймон Амстердамның голландтық психологтар тобымен достастықта де Грооттың жетекшілігімен осындай бағдарламаны жасады. Сонымен қатар, Lisp7 тілінің бастаушысы болған ақпаратты символдық түрде өңдеуге арналған арнайы IPL1 (1956) тілі жасалды.

Алайда, жасанды интеллекттің бірінші бағдарламасы пікірлерді есептеудегі теоремаларды дәлелдеуге арналған Логик-теоретик бағдарламасы болды (9 тамыз 1956). Шахмат ойнауға арналған бағдарлама 1957 жылы (NSS – Newell, Shaw, Simon) құрылды. Оның құрылымы мен Логик-теоретик бағдарламасының құрылымы есептерді әмбебап шешу бағдарламасын (GPS-General Problem Solving) құруға негіз болды. Бұл бағдарлама жағдаяттардың арасындағы айырмашылықтарды талдай отырып және мақсаттарды құрастыра отырып, "Ханой мұнарасы" түріндегі басқатырғыштарды жақсы шешеді немесе белгісіз интегралдарды есептейді. EPAM бағдарламасы (Elementary Perceiving and Memorizing Program) – қабылдау және есте сақтау үшін қарапайым бағдарлама, Фейгенбаум ойлап тапқан. 1957 жылы трансформациялық грамматика бойынша компьютерлік лингвистиканың негізін қалаушылардың бірі Хомскийдің мақаласы пайда болды.

50-ші жылдардың соңында лабиринттік іздеу моделі туды. Бұл тәсіл жай-күй кеңістігін көрсететін кейбір бағандар ретінде тапсырманы білдіреді және бұл бағанда кіріс деректерінен нәтижелеуге оңтайлы жолды іздеу жүргізіледі. Бұл модельді әзірлеу бойынша үлкен жұмыс атқарылды, бірақ практикалық міндеттерді шешуде үлкен идея пайда болған жоқ.

60-шы жылдардың басы – эвристикалық бағдарламалау дәуірі. Эвристика – теориялық негізделген емес, бірақ іздеу кеңістігіндегі аралықтар санын қысқартуға мүмкіндік беретін ереже. Эвристикалық бағдарламалау-белгілі, алдын ала берілген эвристиканың негізінде әрекет стратегиясын әзірлеу.

60-жылдары табиғи тілде сұраныспен жұмыс істейтін алғашқы бағдарламалар құрылуда. БЕЙСБОЛ бағдарламасы (Green және т.б., 1961) өткен бейсбол матчтарының нәтижелері туралы сұрауларға жауап берді, STUDENT (Bobrow,1964) бағдарламасы ағылшын тілінде тұжырымдалған алгебралық есептерді шешу қол жетімді болды.

Машиналық аударма саласындағы жұмыстарға үлкен үміт артылды, олардың басы отандық лингвист Бельскаяның есімімен байланыстырады. Алайда, зерттеушілерге автоматты аударма оқшауланған проблема емес және түсіну сияқты қажетті кезеңнің болуын табысты жүзеге асыру үшін талап ететінін түсіну үшін көп жыл қажет болды.

60-шы жылдары отандық ғалымдар алған ең маңызды нәтижелердің ішінде бейнелерді тану кезінде адам миының қызметін модельдейтін М. Бонгардтың "Кора" алгоритмін атап өткен жөн.

1963-1970 жылдары есептерді шешуге математикалық логика әдістерін қосуға кірісті. 1965 ж. пайда болған ресми логикаға жаңа көзқарас пайда болды (Дж.Робинсон). Бастапқы аксиомалардың жиынтығы болған жағдайда теоремаларды автоматты түрде дәлелдеуге мүмкіндік берген қағидалар әдісінің негізінде 1973 жылы Пролог тілі құрылады.

1954-1964 жж. КСРО-да жеке бағдарламалар құрылып, логикалық есептерді шешу зерттелді. Ленинградта (ЛОМИ – В.А.Стеклова ат. Ленинград математикалық институтының бөлімшесі) теоремаларды автоматты түрде дәлелдейтін бағдарлама (Алиев ЛОМИ) құрылады. Ол Робинсон қарарының әдісіне ұқсас С.Ю.Масловтың бастапқы кері қорытындысына негізделген.

1965-1980 жж. жаңа ғылым-ахуалдық басқару дамиды (Батыс терминологиясында білім беруге сәйкес келеді). Осы ғылыми мектептің негізін қалаушы – профессор Д.А.Поспелов, оның білім беру – жағдайларды ұсынудың арнайы модельдері өңделген.

Шет елдерде ЖИ саласындағы зерттеулер жаңа буынды бағдарламалау тілдерін әзірлеумен және барынша жетілдірілген бағдарламалау жүйелерін (Lisp, Prolg, Plannar, QA4, Macsyma, Reduce, Refal, ATNL, TMS) құрумен сүйемелденеді.

Алынған нәтижелер робототехникада, роботтарды басқаруда, нақты үш өлшемді кеңістікте әрекет ететін қозғалмайтын немесе мобильді роботтарды қолдана бастайды. Бұл ретте жасанды қабылдау органдарын құру проблемасы туындайды.

1968 жылға дейін зерттеушілер негізінен жекелеген "микрокеңістіктермен" жұмыс істеді, олар ойындар, евклидов геометриясы, интегралдық есептеу, "кубиктер әлемі", қарапайым және қысқа фразаларды шағын сөздік қоры бар өңдеу сияқты қосымшалардың ерекше және шектеулі салаларына жарамды жүйелер жасады. Барлық осы жүйелерде бірдей тәсіл қолданылды - дұрыс мән негізінде баламаларды қажетті іріктеуді азайтуға, бағалаудың сандық функцияларын және әртүрлі эвристиканы қолдануға негізделген комбинаториканы оңайлату. 70-ші жылдардың басында жасанды интеллект бойынша зерттеулерде сапалы секіріс болды. Бұл екі себеппен түсіндіріледі.

− Біріншіден. Барлық зерттеушілер бұрын құрылған барлық бағдарламаларға сәйкес саладағы ең маңызды – терең білім жетіспейтінін біртіндеп түсінді. Сарапшы мен қарапайым адам арасындағы айырмашылық сарапшының осы салада тәжірибесі бар, яғни жылдар бойы жинақталған білімі бар.

− Екіншіден. Нақты мәселе туындайды: егер бұл білімді тікелей құрушы осы білімге ие болмаса, бұл білімді бағдарламаға қалай беруге болады. Жауап анық: бағдарламаның өзі оларды сарапшыдан алынған деректерден бөлуге тиіс.

Тапсырмаларды шешу және табиғи тілді түсіну бойынша зерттеулер бір ортақ мәселе – білім беру. 1970 жылға қарай осы идеяларға негізделген көптеген бағдарламалар құрылды. Олардың біріншісі DENDRAL бағдарламасы. Ол масс-спектрометрден түсетін ақпарат негізінде химиялық қосылыстардың құрылымдық формулаларын құруға арналған.

Бағдарлама Нобель лауреаты Д.Ледербергтің қатысуымен Стенфордта жасалды. Ол өз жұмыс істеу барысында тәжірибе жинақтады. Сарапшы оған жекелеген ережелер түрінде ұсынылған мыңдаған қарапайым фактілерді салды. Қарастырылып отырған жүйе алғашқы сараптамалық жүйелердің бірі болып табылады және оның жұмысының нәтижелері таңқаларлық. Қазіргі уақытта жүйе тұтынушыларға спектрометрмен бірге жеткізіледі.

1971 жылы Терри Виноград текшелермен жұмыс жасайтын роботты модельдейтін SHRDLU жүйесін жасады. Роботпен ағылшын тілінде сөйлеуге болады. Жүйе фразалардың синтаксисімен ғана емес, сонымен қатар оның "кубиктер әлемі" туралы семантикалық және прагматикалық білімдерінің арқасында олардың мағынасын дұрыс түсінеді.

80-жылдардың ортасынан бастап, шетелде жасанды интеллект коммерциализация жүреді. Жыл сайынғы күрделі қаржы өсуде, өнеркәсіптік сараптама жүйелері құрылуда, өзін-өзі оқыту жүйесіне қызығушылық артып келеді.

Біздің елімізде 1980-1990 жылдары білім беру саласында белсенді зерттеулер жүргізіліп, білім беру тілдері, сараптамалық жүйелер (300-ден астам) әзірленуде. Мәскеу мемлекеттік университетінде РЕФАЛ тілі құрылады. 1988 жылы ЖИҚ – жасанды интеллект қауымдастығы құрылды. Оның мүшелері – 300-ден астам зерттеуші. Қауымдастық Президенті – Д.А.Поспелов. Ең ірі орталықтар – Мәскеуде, Петербургте, Переславл-Залесскте, Новосибирскте.

1.3. Жасанды саладағы зерттеулер тарихы интеллект және осы саладағы негізгі ұғымдар

Жасанды интеллект (ЖИ) тарихы біздің дәуірге дейін басталады. Аристотель алғашқылардың бірі болып «дұрыс ойлау» заңдылықтарын немесе анықталмайтын ойлау процестерін анықтады. Орта ғасырларда механикалық есептеу құрылғыларын жасау әрекеттері замандастарына

қатты әсер етті. 1642 жылы Блез Паскаль салған ең танымал автомобиль. Паскаль «арифметикалық машина кез-келген жануардың әрекетінен гөрі ойлауға жақын болатын әсер береді» деп жазды.

ЖИ-ті практикалық іске асыру мүмкіндіктері электрондық компьютерлер пайда болғаннан бері пайда болды. Осы уақытта «Машина ойлана ала ма?» тақырыбында философиялық пікірталас басталды. Бұл талқылаудың нәтижесі 50-жылдары Алан Тюринг ұсынған сынақ болды XX ғасыр [4]. Тест мыналардан тұрады: екі телетайп бар (басқа терминалдық құрылғылар жоқ, қазір ICQ ұсынар еді).Телетиптердің бірі автомобильге, екіншісі аппаратқа қосылады, оның артында адам отырады. Бірнеше маман кез-келген телетип бойынша кезек-кезек диалог жүргізеді. Егер көптеген сарапшылар бес минут ішінде көлікті әңгімелесушілердің бірінен таба алмаса, Тюринг сынағы сәтті өтті деп саналады. Тюринг сынағы жасанды интеллекттің дамуында, оның ішінде сынақтың өзін сынға алуда маңызды рөл атқарды.

Мұнда авиациямен ұқсастық жасауға болады. Тюринг сынағының логикасы бойынша жақсы ұшақтарды құстардан ажыратуға болмайтындай етіп қарастырған жөн, тіпті құстар да оларды өздері алады. Авиацияның дамуы дизайнерлер құстарды көшіруді тоқтатып, аэродинамика, материалтану және күш теориясын бастағаннан басталды. Робототехника адам анатомиясын көшіруді тоқтатқаннан кейін индустрияға айналды. Сол сияқты, жасанды интеллект субъектілері адамдар сияқты ойлайтын және әрекет ететін ЖИ жүйелерін құруды тоқтатқаннан кейін өмір сүру құқығына ие болды, бірақ олар жақсы нәтижеге қол жеткізетін ұтымды әрекет ететін және ойлайтын жүйелер құра бастады.

Жасанды интеллект (ЖИ) саласындағы зерттеулердің пайда болуы екі бағытта жүрді: логикалық және нейробернетикалық. Ерте зерттеулер 50-60 жылдары жүргізілді (Н.Винер, Тюринг, Маккаллок, Розенблатт, Саймон, Маккарти, Слейдж, Сэмюэль, Гелернер, Н.Амосов): алғашқы көрініс ЖИ жүйелерін құру үшін LISP бағдарламалау тілі жасалды; 60-жылдардың соңында интеграцияланған (ақылды) роботтардың және алғашқы сараптамалық жүйелер пайда болды.

80-жылдардың басындағы нейробернетикадағы жаңа бум (Хопфилд моделі) пайда болды [5].

Логикалық бағдарламалаудың пайда болуы және ПРОЛОГ тілі. 5-буын компьютерлік бағдарлама. Стратегиялық АҚШ-тың компьютерлік бастамасы. КСРО мен Ресейдегі ЖИ зерттеуі.

Ойлау процесін модельдеу саласындағы зерттеулердің басынан бастап (40-жылдардың соңы) екі уақытқа дейін дерлік тәуелсіз бағыттар пайда болды:

- логикалық,
- нейробернетикалық.

Болашақта 80-жылдардың басында «сараптамалық жүйелер» (ТЖ) ұғымдарының пайда болуымен бұл бағыт «білімге негізделген жүйелер» (Knowledge Based Systems) құрумен айналысатын информатиканың ғылыми-технологиялық бағыты – «білім беру инженері» болды. бұл бағыт әдетте «жасанды интеллект» терминімен байланысты. информатика бағыты – «білім беру инженері», оны құрумен айналысады.

Екінші бағыт, нейробернетикалық, мидың нейрондарына функционалды түрде ұқсас көптеген элементтерден тұратын өзін-өзі ұйымдастыратын жүйелердің құрылысына негізделген. Бұл тенденция Маккаллок-Питтс формальды нейрондық тұжырымдамадан және Розенблаттың әртүрлі перцептрондық модельдерімен зерттеуінен – өрнекті тануды үйренетін жүйеден басталды. Логикалық бағытта салыстырмалы түрдегі жетістіктер мен микроэлектроникадағы төмен технологиялық деңгейге байланысты нейробернетикалық бағыт жаңа сәтті теориялық модельдер пайда болған кезде (мысалы, «Хопфилд моделі») және ультра ірі интегралды схемалар пайда болған кезде 60-жылдардың аяғы мен 80-жылдардың басына дейін ұмытып кетті.

Логикалық бағытты сана деңгейінде немесе ауызша немесе логикалық (мақсатты) ойлау деңгейінде ойлауды модельдеу ретінде қарастыруға болады [6]. Оның артықшылықтары:

- жүйені салыстырмалы түрде оңай түсіну мүмкіндігі;
- жүйені оған негіздеу процесін көрсету жеңілдігі
- пайдаланушылық интерфейс табиғи тілде немесе кез келген
- ресми тіл;
- жүйелік мінез-құлықтың бірегейлігіне бірдей
- жағдайлар.

Логикалық тәсілдің кемшіліктері:

- бұлыңғыр кейіпкерлердің (кескіндердің) орындалуының қиындығы мен табиғи болмауы;
- белгісіздік жағдайында барабар мінез-құлықты жүзеге асырудың қиындығы (немесе тіпті мүмкін еместігі) (білімнің болмауы, шулы мәліметтер, дұрыс емес мақсаттар және т.б.);
- проблемаларды шешу процесін параллельдеудің қиындығы мен тиімсіздігі.

Нейробернетикалық бағытты (немесе нейроинформатика) бейнелі ойлау мен ойлауды сана деңгейінде модельдеу ретінде қарастыруға болады (түйсігі, шығармашылық қиялын модельдеу, түсіну). Оның артықшылығы –- кемшіліктердің болмауы, қисынды бағытқа тән, кемшіліктер оның артықшылықтарының болмауы. Сонымен қатар, нейрокибернетикалық бағытта қарапайым қарапайым бейімделу алгоритмдері мен жасанды нейрондық желінің құрылымдық ерекшеліктерін белгілей

отырып, мүмкін болатын (мүмкін иллюзиялық), өздігінен күрделі және шешілетін міндет үшін жеткілікті түрде әрекет етуге теңшелген жүйені алуға болады. Және оның күрделілігі нейрондық желі моделінің сандық факторларынан ғана. Тағы бір артықшылығы жағдайда аппараттық іске асыру нейрон болып табылады, оның өміршеңдігі, яғни қабілеті сақтауға қолайлы тиімділігі шешу істен шығуы кезінде желі элементтерін. Бұл нейрондық желілердің қасиеті артықшылық есебінен қол жеткізіледі. Бағдарламалық жасақтаманы іске асыру жағдайында нейрондық желілердің құрылымдық артық болуы оларға толық емес немесе шулы ақпарат жағдайында сәтті жұмыс істеуге мүмкіндік береді [7].

«Білім» ұғымы мен «деректер» немесе «ақпарат» ұғымдарының арасындағы айырмашылық неде? Жақында ғалымдар материя мен энергиямен қатар, ақпарат өзінің реттілігін (гетерогенділігі) немесе құрылымын сипаттайтын материалдық дүниенің объективті бар ажырамас бөлігі болып табылады деген қорытындыға келді. Тірі тіршілік иелерінің энтропияны (біртектілікті) арттырғысы келетін әлемде өз құрылымын (реттілігін) сақтау мүмкіндігі олардың қоршаған әлемнің құрылымын тану және тану нәтижесін (яғни, әлем туралы білім) өмір сүру мақсаттары үшін пайдалану қабілеттеріне байланысты.

Сонымен, білім дегеніміз – тіршілік иесі (субъект) сыртқы әлем қабылдаған ақпарат, ал «ақпараттан, білімнен» айырмашылығы субъективті. Бұл субъектінің өмірлік тәжірибесінің сипаттамаларына, оның сыртқы ортамен қарым-қатынас тарихына, яғни оқу немесе өзіндік білім алу процесінің ерекшеліктеріне байланысты. Абстракцияның бұл деңгейінде білім ерекше болып табылады және адамдар арасындағы білім алмасу жоғалтусыз мүмкін емес, өйткені ақпарат кодталған (гетерогенділік) және таратқыштан қабылдағышқа жоғалтусыз берілуі мүмкін (кедергілерге байланысты бұрмалану мүмкіндігін ескерместен).

Білім пәндер арасында білім берудің кез-келген тілі арқылы беріледі, олардың ең типтік өкілі табиғи тіл болып табылады. Табиғи тілді құру және қолдану арқылы адам, бір жағынан, оны әртүрлі өмірлік тәжірибесі бар адамдардың көпшілігіне бірдей беру үшін білімді қалыптастыруға және біріздендіруге тырысты, екінші жағынан, ол жеке білімнің барлық байлығын беруге мүмкіндік берді.

Бірінші тенденция білімнің әртүрлі салаларында (математика, физика, медицина және т.б.) әртүрлі формальды арнайы диалектілердің пайда болуына әкелді.

Екіншісі, көркем әдебиеттің пайда болуына әкелді, ол тіл арқылы адамның миында ассоциацияларды (тәжірибелерді) тудыруға деген ұмтылысқа негізделген, яғни. оқудан алынған білімге және өзінің біліміне сүйене отырып, оны ойлануға және алаңдатуға мәжбүр етеді. Жалпы,

өнердің барлық түрлері бұған бағытталған – ассоциацияларды қолдана отырып білім беру.

Егер біз абстракцияның осындай жоғары деңгейінен (философиялық) қарапайым өмірге көшетін болсақ, онда әдетте жасанды интеллект бойынша әдебиетте жасалынатын білім мен мәліметтерді олардың формаланған түрінде салыстыруға болады [8]. Сонда біз мәліметтерден келесі білім айырмашылықтарын тұжырымдай аламыз:

- білім неғұрлым құрылымдалған;
- білімде атомдық емес элементтер өте маңызды
- білім (мәліметтердегідей) және олардың арасындағы байланыс;
- білім деректерден гөрі өзін-өзі түсіндіреді, яғни. білімде оларды пайдалану туралы ақпарат бар;
- білім пассивті мәліметтерге қарағанда белсенді, яғни білім алады;
- оларды пайдалану жүйенің уылдырық шашу әрекеттері.

Деректер мен білім арасында ешқандай өткір шекара жоқ екенін есте ұстаған жөн, өйткені соңғы жиырма жыл ішінде дерекқорды басқару жүйесін жасаушылар көбінесе оларды білімге айналдыруда. Мысал ретінде мәліметтер базасын жобалау үшін семантикалық желілерді қолдану (білім беру формализмі), объектіге бағытталған мәліметтер базасының пайда болуы, сақталатын процедуралар (бұл белгілі бір дәрежеде мәліметтерді белсенді етеді) және т.б. Сонымен, жоғарыда аталған білім мен мәліметтер арасындағы айырмашылықтар информатика дамуымен тегістеледі.

Инженерлік білімде семиотика – белгі жүйелері туралы ғылымнан алынған келесі негізгі ұғымдар бөлінеді:

- кеңейтілген білім – үстірт немесе нақты білім;
- интенсивті білім – терең немесе дерексіз білім (үлгіні білу),
- синтаксис – белгілер жүйесінің құрылымы (мәліметтер немесе білім),
- семантика – белгі жүйесінің мағынасы (білім), яғни оның білім берудің басқа парадигмасындағы балама көрінісі (ішкі);
- прагматика – белгілер жүйесімен байланысты мақсаттар (мысалы, табиғи тілдегі сөйлемнің мақсаты немесе тағайындалуы (команда, сұрақ, түсіндіру және т.б.).

1.4. Жасанды интеллект аймағындағы зерттеулердің негізгі бағыттары

Қазіргі уақытта ЖИ – бұл тез дамып келе жатқан және өте тарамдалған ғылыми саласы. Тек қана компьютерлік лингвистика бойынша

әлемде жыл сайын 40-тан астам конференция өткізіледі. Әрбір еуропалық елде, сондай-ақ АҚШ-та, Канадада, Жапонияда, Ресейде, Оңтүстік-Шығыс Азияда ЖИ бойынша ұлттық конференциялар тұрақты түрде өткізіледі. Ресейде бұл іс-шара екі жылда бір рет Ресейдің ЖИ қауымдастығының (РҚЖИ) қолдауымен өткізіледі. Бұдан басқа, екі жылда бір рет ЖИ (IJCAI) бойынша халықаралық біріккен конференция өткізіледі. Осы салада 3 мыңнан астам мерзімді басылымдар ғылыми нәтижелер шығарады.

Қазіргі уақытта ЖИ – қарқынды дамып келе жатқан және жоғары тармақталған ғылыми сала. Компьютерлік лингвистиканың өзінде әлемде жыл сайын 40-тан астам конференциялар өткізіледі.

Жасанды интеллекттің барлық салаларында толық және қатаң классификация жоқ, жасанды интеллект шешетін міндеттерді жіктеуге тырысу 1.3-суретте келтірілген.

1.3-сурет. Жасанды интеллекттің міндеттері

Д.А.Поспелов бойынша ЖИ-да ЖИ саласындағы зерттеудің екі басым тәсілі бар: нейробиондық және ақпараттық.

Алғашқылардың жақтастары адамның миында өтетін процестерді жасанды түрде көбейту мақсатын алға қояды. Бұл бағыт медицина, биология және кибернетика қиылысында орналасқан. Сонымен бірге адамның миы зерттеліп, оның жұмысының жолдары ашылып, биологиялық құрылымдар мен оларда болып жатқан процестерді қайталауға арналған техникалық құралдар жасалады.

Екінші тәсілдің жақтаушылары интеллектуалды ақпараттық жүйелерді зерттеуге сүйенеді; програмаларды шешудің интеллектуалды бағдарламалары; білімге негізделген жүйелер.

Өзін өзі бақылауға арналған сұрақтар

1. Жасанды интеллект дегеніміз не?
2. Жасанды интеллект қандай ғылыми бағыттармен өзара әрекеттеседі?
3. Жасанды интеллект пәнін ғылыми пән ретінде түсінудің тәсілдерін сипаттаңыз.
4. Ресейдегі қазіргі жағдайды сипаттаңыз.
5. Жасанды интеллект дамуының "компьютерлік" кезеңін сипаттаңыз.
6. XX ғ. 70-шы жж. жасанды интеллектінің дамуын сипаттаңыз.
7. XX ғ. 80-ші жж. жасанды интеллектінің дамуын сипаттаңыз.
8. Жасанды интеллекттің негізгі есептерін сипаттаңыз.
9. Жасанды интеллект саласында қандай бөлімдер бар?
10. Адамның ойлауын модельдеу мүмкіндіктерінің дәлелдерін келтіріңіз.
11. Зияткерлік құралдардың қоғамға әсері мәселесіне көшу немен негізделді?
12. Жасанды интеллект жүйесінің қауіпсіздік проблемасы немен және қалай шешілуі мүмкін?

2 тарау. ПРОЛОГ ТІЛІНДЕ БАҒДАРЛАМАЛАУ НЕГІЗДЕРІ

2.1. Пролог декларативті тіл ретінде

Пролог тілінің дамуын 1970 жылы Алан Кулмерое мен Филипп Руссель бастады [9]. Олар берілген мәтін негізінде логикалық қорытынды жасай алатын тіл жасағысы келді. Пролог атауы «LOGic-те бағдарламалау» қысқаруы болып табылады. Бұл тіл Марсельде 1972 жылы дамыған. Пролог – алгоритмге емес, предикаттық логикаға негізделген бағдарламалау тілі. Егер алгоритмдік (процедуралық) тілдегі бағдарлама белгілі бір ретпен орындалатын нұсқаулар тізбегі болса, онда Прологтағы бағдарлама тек тапсырманың сипаттамасын қамтиды, ал Пролог машинасы осы сипаттаманы басшылыққа ала отырып, шешім іздейді. Мысалы, ат қадамымен шахмат тақтасын жабудың логикалық міндеті бар. Кез келген алгоритмдік тілде бұл мәселені шешу өте күрделі алгоритмді құруды талап етеді. Прологта жылқы жүретін ережелерді сипаттау жеткілікті, содан кейін Пролог шешімді өзі басады.

Мұндай қарапайымдылықтың басты жағы – бағдарламалардың ресурстарды тұтынуы. Мысалы, шахмат тақтасына бір-бірін ұрмайтын сегіз патшайымның тағы бір танымал тапсырмасында шешімнің толық ағашында 648 шыңы бар. Мұндай ағаштан шешім табу мүмкін емес ұзақ уақытты талап ететіні анық. Прологта бағдарламалау келесі кезеңдерден тұрады:

- объектілер мен олардың арасындағы қатынастар туралы кейбір фактілерді хабарлау,
- объектілер мен олардың арасындағы қатынастар туралы кейбір ережелерді анықтау;
- нысандар мен олардың арасындағы қатынастар туралы сұрақтарды тұжырымдау.

2.2. Предикат түсінігі

Пролог бағдарламасының негізгі элементі – предикат. Математикалық тұрғыдан алғанда предикат – екілік мәнді (шын немесе жалған) қайтаратын функция болып табылады. Прологта предикат нысандар арасындағы қатынасты білдіреді, бұл да шындыққа айналуы мүмкін. Прологта предикат ұғымын Пугачеваның – Киркоровтың әйгілі бұрынғы отбасы мысалында қарастырайық [10]. Біріншіден, біз ата-ана мен баланың қарым-қатынасын жазамыз. Пролог синтаксисінде «Борис –

Алланың ата-анасы» өрнегі келесідей: *parent(boris, alla*). Мұнда *parent* – предикаттың аты, ал *boris, alla* – дәлелдер. *Boris пен alla* дәлелдері тұрақты, сондықтан олар кіші әріптермен жазылады. Прологта бас әріппен айнымалылар басталады. Нүкте предикаттың соңын, сондай-ақ сөйлемнің табиғи тілде аяқталуын білдіреді. Отбасылық мүшелердің ата-аналық қатынастарын да жазамыз:

parent(bedros, filipp).
parent(kristina, denis).
parent(edmuntas, kristina).
parent(vladimir, denis).
parent(alla, kristina).

Енді біз «жұбай» түсінігін береміз *(spouse):*

spouse(filipp, alla).
spouse(vladimir, kristina).

Алынған предикаттар жиынтығы жұлдыздар отбасы туралы білім базасын құрайды. Реляциялық деректер базасында бірдей мәліметтер қалай ұсынылатынын салыстырыңыз.

Ата-аналар кестесі

Parent	***Child***
Boris	*Alla*
Bedros	*Filipp*
Alla	*Kristina*
Edmuntas	*Kristina*
Kristina	*Deni*

Жұбайлар кестесі

S1	***S2***
Alla	*Filipp*
Kristina	*Vladimir*

Көріп отырғаныңыздай, деректерді ұсыну деңгейінде ұқсастық бар. Бірақ осымен бітеді. Бірақ деректерді шығару деңгейінде үлкен айырмашылық бар. Білімді алу үшін реляциялық деректер базасы, мысалы, SQL-де деректерді таңдау үшін сұрау құруы керек. Мысалы, біз Кристинаның ата-анасы кім екенін білгіміз келеді делік. Біз келесі формадағы сұранысты жазуымыз керек:

SELECT *Parent* FROM Ата-аналар WHERE *Child = "Kristina"*

Прологта білімді алуға сұраныс осы білім ұсынылған кездегідей сипатталады. Егер біз келесі предикатты Прологқа ауыстырсақ (мақсаттық предикат):

parent(alla, kristina).

Бұл мақсатты былай оқуға болады: Алла Кристинаның ата-анасы ма? Осы мақсатты білім қорының мазмұнымен салыстыра отырып, Пролог бұл тұжырымның ақиқат екенін анықтап, оны хабарлайды. Жоғарыдағы SQL сұрауы (Кристинаның ата-анасы кім?) Прологта келесідей:

parent(X, kristina).

Мұнда X – керекті мәндер тағайындалуы керек айнымалы. Пролог-тағы айнымалы зат есім немесе сұрақ сөзінің аналогы болып табылады. Жоғарыдағы білім базасынан Пролог екі жауап шығарады:

X = alla
X = edmuntas

Сұрақты келесідей тұжырымдай аламыз: Кристинаның ата-анасы бар ма?

parent(_, kristina).

Бұл жазба: x ата-анасы Y табу, ол өз кезегінде Кристинаның ата-анасы. Алғы сөздегі үтір "Және" немесе конъюнкцияға ұқсас. Мұндай сұранысқа жауап ретінде Пролог келесі жауап береді:

X = alla
Y = edmuntas

Келесі сұранысты беруге болады:

parent(alla, _).

Енді анонимді айнымалы екінші дәлел ретінде қолданылған. Бұл сұранысты келесідей оқуға болады: Алланың балалары бар ма? Пролог жауап береді: Yes.

Предикат шын немесе жалғанды қайтаратын екілік функция болғандықтан (Прологта мәлімдеме ақиқат немесе жалған болуы мүмкін), Прологтағы бағдарламаның нәтижесі – мақсаттың шын немесе жалған екенін анықтау. Айнымалыларға мәндер беру, нәтижелер шығару және т.б. – бұл жанама әсерлер.

Сонымен, Прологтағы бағдарлама предикаттардан тұрады. Пролог бағдарламасы мен білім базасы – синонимдер. Мақсат предикаттар түрінде де тұжырымдалған. Прологта бағдарламаны іске қосу – бұл мақсатты шешу.

2.3. Прологтың интерпретаторы қалай жұмыс істейді?

Прологта шешім табу процесі – мақсаттық предикатты білім базасының предикаттарымен салыстыру [10]. Бұл процесс *унификация* (бірігу) деп аталады.

Алдыңғы бөлімнен алынған мақсатпен Пролог жүйесін ұсынайық (біз Кристинаның немересі кім екенін білгіміз келеді):

parent(X, kristina), parent(Y, X).

Пролог одан *parent(X, kristina*) алғашқы мақсатын ерекшелейді (және оны білім базасымен салыстыруды бастайды (біріктіруді жүзеге асыру үшін). Білім базасы төменде қайталанады:

parent(boris, alla).
parent(bedros, filipp).
parent(edmuntas, kristina).
parent(alla, kristina).
parent(kristina, denis).

Біріншіден, алғашқы предикат пен қосалқы мақсат салыстырылады:

parent(boris, alla) және *parent(X, kristina)*

boris бірінші аргументі X айнымалысымен салыстырылады.

Пролог айнымалылардың *free* (еркін) және *bound* (байланған) күйлерін ажыратынын ескеру қажет. ***Егер екі айнымалы да жалғанған болса, онда біріктіру кезінде олар салыстырылады. Егер олардың бірі тегін болса, онда тағайындау орын алады. Айнымалы мәндерге қайта тағайындауға жол берілмейді.*** Бұл Прологты басқа тілдерден айтарлықтай ерекшелендіреді.

Осылайша, әлі де бос болатын X айнымалысына *boris* мәні беріледі. Осыдан кейін екінші дәлелдер, *alla* және *kristina* біріктіріледі. Өйткені бұл тұрақтылар және *alla kristina* тең емес болғандықтан, предикаттық *parent(boris,alla)* және *parent(X, kristina)* қосалқы мақсаттар сәтсіздікпен аяқталады (fail).

Білім базасында parent предикатының бірнеше даналары болғандықтан, мұндай предикат *бірмәнді емес* деп аталады (non-derministic). Егер предикат біреу болса, онда ол *бірмәнді* (deterministic) деп аталады. Сәтсіздіктен кейін бірмәнді емес предикат болған жағдайда кері қайтару орындалады – предикаттың келесі данасына ауысу. Бұл ретте, егер орын алған болса, айнымалыларға мән тағайындау да алынып тасталады. Содан кейін предикаттық біріктіру жүзеге асырылады.

parent(bedros, filipp) и *parent(X, kristina)*

Біріктірудің нәтижесі бірдей болатыны анық, сәтсіздік (*fail*). Келесі *parent(edmuntas,kristina)* предикатына оралған кезінде сурет басқаша болады: X-ге edmuntas мәні тағайындалады, ал екінші дәлелдерге сәйкес келу де сәтті болады, өйткені *kristina = kristina.*

Осылайша, алғашқы қосалқы бағдарлама аяқталады. Пролог предикаттың қай данасы жұмыс істегенін есіне түсіріп, келесі предикатқа оралу көрсеткішін орнатады:

parent(boris, alla).
parent(bedros, filipp).

parent(edmuntas, kristina).
> parent(alla, kristina).
parent(kristina, denis).

осыдан кейін ол *parent(Y, X*) екінші қосалқы жиынына өтеді, мұнда X = edmuntas, яғни Пролог өз алдына келесі мақсаттарды қояды:

parent(Y, edmuntas).

Эдмунтастың ата-анасын іздеуде Пролог қайтадан бұл предикатты білім базасынан, *parent(boris, alla*) біріктіруге бастайды.

Бұл жолы барлық предикаттарды іздеу нәтижесіз аяқталатынын көруге болады, яғни мақсат

parent(Y, edmuntas)

оң шешім берген жоқ.

Бұл жағдайда Пролог алдыңғы мақсатқа қарай жылжиды (артқы көріністер тізімі бойынша жылжиды) және *parent(X, kristina)* үшін балама шешім табуға тырысады. Бұл жағдайда X қайтадан бос айнымалыға айналады, ал Пролог оралу нүктесіне оралады:

parent(boris, alla).
parent(bedros, filipp).
parent(edmuntas, kristina).
> parent(alla, kristina).
parent(kristina, denis).

яғни негізгі *parent(alla, kristina*)предикаты *parent(X, kristina*) предикатымен ауыстырылады.

Енді X-ке *alla* мәні меншіктеледі, оралу көрсеткіші негізгі предикатқа *parent(kristina, denis)* орнатылды, қайтадан *parent(Y, X)* қосалқы мақсатына көшу орындалады, мұнда X = alla. Алғы сөз қайтадан бірінші предикаттан бастап *parent(Y, alla*) білім базасымен біріктіре бастайды. Бірінші предикатта тұрақты тұрақты *boris* және *Y* айнымалысы біріктіріледі. *Y=boris* тағайындалады, содан кейін екінші дәлелдер салыстырылады. *Alla = alla* болғандықтан, сәйкестік сәтті аяқталды. Осылайша, шешім табылды: Кристина – Бористің немересі. Әке-шешесі Кристинаны іздеудегі сәтсіздік тек білім базасының толық болмауымен байланысты екенін ескеріңіз.

Сонымен, Prolog аудармашысы автоматты түрде шешім іздейді. Іздеу жүйесі сәтсіздіктен кейін кері қайтаруды қолдана отырып іске асырылады. Қайта оралу келесі мағынаны білдіреді: Прологта бағдарламаны жүзеге асыру (мақсатты шешу) – мақсатты білім базасымен біріктіру [7].

2.4. Прологтағы фактілер мен ережелер

Жоғарыда сипатталған "ата-ана-немересі" типінің қатынасын белгілейтін сұрау одан әрі бірнеше рет талап етілуі мүмкін. Осыған байланысты оны басқа сұрау салуларда одан әрі пайдалану үшін есте сақтаған жөн. Прологтың білім базасында фактілерді ғана емес, шартты қатынастарды да сақтауға болады. "Ата-ана-немересі" типінің қатынасы мынадай түрде жазылуы мүмкін:

grandparent(X,Y) if parent(X,Z), parent(Z,Y).

Мұны келесідей оқып шығыңыз: X - Y-нің ұрпағы, егер X-Z-тің, ал Z-Y-дің ата-анасы.

Grandparent(X,Y) предикаты ереже тақырыбы деп аталады, ал *if* – ереже денесінің оң жағындағы өрнек.

Ескерту: ережедегі "if" бумасының синонимі «:-» символдары болып табылады.

Осылайша, деректер базасындағы сияқты, Прологтың білім базасындағы фактілер түрінде біз бастапқы білімдерді сақтаймыз, ал олардан алынған туындыларды фактілер сияқты жолданатын ережелер түрінде жазамыз.

Факт – бұл белгілі нәрсе.

Ереже – бұл бар фактілердің негізінде жаңа фактілерді тудырудың тәсілі.

Туыстық қарым-қатынас үшін біз қосымша фактілерді енгізу қажеттігінен құтыла отырып, көптеген ережелерді белгілей аламыз, мысалы, ағалары, немере және т.б. кімге тиесілі?):

sibling(X,Y) :- parent(Z,X), parent(Z,Y), X<>Y.

Салыстыру предикаты X<>Y "менің әкемнің ұлы, бірақ менің ағам емес" түріндегі коллизияны шешу үшін қажет. Ағай түрінің қатынасын анықтайтын ереже келесідей:

uncle(X,Y) :- parent(Z,Y), sibling(X,Z).

Мақсатты шешу кезінде Пролог сөзге емес, ережеге тап болған кезде, алдымен ереженің тақырыбын біріктіреді, яғни. байланысты айнымалыларды салыстырады және бос айнымалыларға мәндер тағайындайды. Егер аргументтер сәтті біріктірілсе, Пролог тақырыптағы дәлелдердің мәнін ереженің негізгі предикатына ауыстырады және бұл предикатты білім базасымен біріктіруді бастайтын бағыныңқы ретінде орнатады. Осы кіші бағдарламаны сәтті шешкен жағдайда Пролог ереженің келесі жағдайына көшеді. Егер бұл шартты біріздендіру предикаттық нәтижеге әкелсе, онда Пролог ереженің алдыңғы жағдайына көшеді. Бұл алдыңғы қайтару

екіұштылық болған жағдайда ғана болады. Мұны мысалмен көрсетейік. Біз өзімізге Кристинаның ұрпағы кім екенін білуді мақсат етіп қойдық:

grandparent(Who, kristina).

Осындай мақсатқа қол жеткізген Пролог оны ережемен біріктіре бастайды:

grandparent(X,Y) :- *parent(X, Z), parent(Z,Y).*

Мақсатты предикаттағы *Who* айнымалысы – еркін айнымалы және оны ереже тақырыбындағы X айнымалымен біріктіру әрқашан сәтті болады. Айта кету керек, Прологта барлық айнымалылар жергілікті, яғни. ереже ішінде ғана бар.

Біз X орнына *Who* пайдалана алар еді, және бұл бірдей біріздендіруге болатын түрлі айнымалылар болар еді. Жаһандық айнымалыларды құру қажет болған жағдайда, assert предикаты жасайтын динамикалық фактілерді пайдаланады және retract предикаты немесе retractall жойылады. Ереже тақырыбындағы айнымалылар әрдайым еркін болғандықтан, *y = kristina* беріледі. Ереже тақырыбын біріздендіру сәтті өткендіктен, Пролог ереже денесіне тереңдейді және олар байланысты болса, айнымалыларды ауыстыра отырып, дене бірінші предикат ереже астына қояды:

parent (X, Z).

X және Z айнымалылар еркін болып табылады, сондықтан білім базасынан бірінші parent предикатымен осы мақсатты біріздендіру табысты болады:

X = boris, Z = alla

Осыдан кейін Пролог ереже бойынша екінші предикатқа ауысады, X және Y мәнін қояды:

parent(alla, kristina).

Осы кіші нысанның қарары шындықты береді, ал біріздендіру барысында берілген ауыспалы мәндерді Пролог қайтарады

Who = X = boris.

Осылайша, резолюция барысында негізгі мақсаттарды Пролог білім базасындағы ережелерді басшылыққа ала отырып, өзіне өз бетінше мақсат қояды.

Басқа мысал қарастырайық:

Grandparent (*Who, denis*).

Осы кіші аралықты біріктіруге тырысып, Пролог айнымалыларды (alla, denis) бірінші *parent* предикатына салыстырады, сәтсіздікке ұшырайды, келесі инстанцияға ауысады және т.б.

Білім базасында parent(alla, denis) фактісі жоқ болғандықтан, осы кіші бағдарламаның шешімі сәтсіз болады. Сондықтан ереженің бірінші предикатын X = *boris*, Z = alla мәндерімен біріктіру дұрыс емес. ***Сондықтан Пролог ережелердің алдыңғы жағдайына қарай жылжиды және parent(X, Z) кіші мақсаттарына басқа шешім табуға тырысады.***

Бұл жағдайда айнымалыларды беру тоқтатылады (X = *boris*, Z = *alla*). X және Z айнымалылары қайтадан бос болады. Айта кету керек, мұнда тек бір таңбалы предикатқа апару мүмкін. Егер предикаттар тізбегінде ереже ішінде бір мәнді де, бір мәнді емес предикаттар болса, онда сәтсіздіктен кейін ең жақын бір мәнді предикатқа жылжу керек. Осы предикатты бірінші біріздендіру кезінде Пролог *parent* келесі данасына кері көрсеткішті орнатты:

parent(boris, alla).

> parent(bedros, filipp).

parent(edmuntas, kristina).

parent(alla, kristina).

parent(kristina, denis).

Артқа айналған кезде Пролог бұл фактіні біріздендіруге кіріседі және кері нұсқаға үшінші инстанцияны орнатады:

parent(boris, alla).

parent(bedros, filipp).

> parent(edmuntas, kristina).

parent(alla, kristina).

parent(kristina, denis).

Екінші предикаттық *parent* қосалқы тобымен *parent* (X, Z) біріктіргеннен кейін, Пролог айнымалыларға мәндер тағайындайды:

X = bedros, Z = filipp

және екінші предикаттық *parent*(*Z,Y*) ауысады :

parent(bedros,denis).

және екінші предикаттық ата-анаға ауысады (Z, Y):

parent(bedros,denis).

Бұл кіші бағдарламаның шешімі де сәтсіз болатыны анық. Пролог қайтадан ереженің алдыңғы предикатына және басты ата-ананың үшінші инстанциясына өтеді. Төртінші фактіні біріктіру ғана сәтті болады:

parent(alla, kristina),

нәтижесінде біз аламыз

Who = X = alla.

Білім базасында ережелер болған жағдайда Прологтың интерпретаторы осылай жұмыс істейді.

Сонымен, ***факт – бұл тұрақтыға негізделген білім (өзгермейтін білім). Ережелер – бұл фактілерден алынған білім.***

Фактілер мен ережелер жиынтығында алгоритм жоқ.

Ережелер мен фактілер бір-біріне тәуелсіз.

Нәтижені шығаруға арналған ережелер тіркесімі мақсатты шешу кезінде пайда болады. Ережелер тақырыбындағы айнымалылар осы ереженің ішінде ғана болады.

Ереже ішінде артқа айналған кезде, ол ережедегі алдыңғы түсініксіз предикатқа өтеді.

2.5. Prolog тіліндегі рекурсия

Егер біз X Y-тің ата-бабасы болып табылатындығын сұрасақ, онда дәйекті мақсаттарды орнату керек:

parent(X,Y).

grandparent(X,Y).

grandgrandparent(X,Y).

grandgrandgrandparent(X,Y).

және т.б.

grandgrandparent(X,Y) :- parent(X,Z), grandparent(Z,Y).

grandgrandgrandparent(X,Y) :- parent(X,Z), grandgrandparent(Z,Y).

Оның орнына Пролог осы ережені келесідей жазуға мүмкіндік береді:

predecessor(X,Y) :- parent(X,Y).

predecessor(X,Y) :- parent(X,Z), predecessor(Z,Y).

Мұнда predecessor предикатының рекурсивті шақыруы бар. Прологтағы Рекурсия өте ықшам және тиімді бағдарламаларды құруға мүмкіндік беретін қуатты құрал болып табылады.

Факториал есептеу мысалында рекурсияны пайдалануды қарастырайық.

$$n! = n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot 1$$

Факториалды рекурсивті анықтау: 0! = 1; n! n·(n-1)! Факториалды есептеуді жүзеге асыратын Пролог бағдарламасы келесідей болады:

f (0,1).

f (N, F): - N1=N-1, f (N1, F1), F=F1·N.

Бағдарлама, мәнісінде, рекурсивті математикалық сипаттаудан артық емес, жоғарыда келтірілген факториал функциясы. Бағдарламаны трассалау режимінде іске қосып, F(3, X) мақсатын қоямыз. Пролог мақсат предикатын білім базасымен салыстыра бастайды (CALL предикат шақыруын білдіреді, return – предикат жұмысын аяқтау, FAIL – сәтсіздікті, REDO-кері қайтару):

CALL f (3, X)	мақсаты f (0,1)
Fail	сәтсіз, себебі 3 ≠ 0
REDO f (3, X)	келесі f данасына апарады()
N= 3,	денеге ереже кіреді. N=3 тағайындау
N1 = 2	N1 = 2 табамыз
f (2, X)	Пролог өз мақсатын қояды
CALL f (2, X)	f (0,1) -ге сәйкес келетін
FAIL	сәтсіз, себебі 2 ≠ 0
REDO f (2, X),	келесі f () данасына оралады
N= 2,	тағы да ереже денесіне кіреміз. N=2 тағайындау
N1=1	N1=1 табамыз. Бұл басқа N1
f (1, X)	Пролог өзіне алғашқы мақсат қояды f (1, X)
CALL f (1, X)	f (0,1) -ге сәйкес келетін
FAIL	сәтсіздігі, себебі 1 ≠ 0
REDO f (1, X)	келесі f () данасына оралады
N = 1	қайтадан ереженің денесіне енеміз. N = 1 тағайындаңыз
N1= 0	N1=0 табамыз.
f (0, X)	Алғашқы мақсат қояды f (0, X)
CALL f(0,X)	ол f(0,1) салыстырылады
RETURN X =1	сәтті аяқталды, төменгі рекурсия деңгейінен қайтару
F = 1	1-ге көбейту (1-ден факторлық)
RETURN X=1	рекурсияның келесі деңгейінен қайтару
F = 2	көбейту 2-ден 1-ге дейін (2-ден факторлық)
RETURN X=2	рекурсияның келесі деңгейінен қайтару
F=6	3-ке 2 көбейту (3-тен факториал))
RETURN X=6	бағдарламадан қайтару

Факториалды есептеу бағдарламасы жұмысының логикасы рекурсиядан шығуды анықтайтын предикаттың мәтінде орналасуына байланысты. Біріздендіру бағдарлама мәтінінде предикаттардың жүру тәртібімен орындалады және егер предикат f(0,1) соңында қойылса, рекурсиядан

шығу мүмкін болмаса. Осылайша, Прологтың декларациясы оны пайдалану ыңғайлылығы үшін абсолютті емес. Предикаттар кез келген тәртіппен орналасуы мүмкін бағдарламаның нұсқасы:

```
f (N, F): - n>0, N1=N-1,f(N1, F1), F=F1·N.
f (0,1).
```

Сондай-ақ, бұл рекурсивті предикатта рекурсивті шақырудан кейін денедегі ережелер әрекеті бар екенін атап өтеміз. Бұл қалдық рекурсияның бұзылуы деп аталады (tail recursion).

Құйрықты (қалдық) рекурсияның бұзылуы, сонымен қатар құйрықты емес рекурсия деп аталады, рекурсивті қоңыраудың бүкіл ортасын есте сақтауды талап етеді (және нәтиже ғана емес), сондықтан ол үлкен жадының шығынын әкеледі. Алайда, құйрықты емес рекурсияларды жою әдістері бар [10].

Төменде қалдық емес рекурсияның факториалын есептеу мысалы келтірілген.

```
f (N1, N, F1, F): - % егер N1! = F1, онда N! = F
N2=N1+1,
F2=F1*N2, % (N1+1)! = F2
f (N2, N, F2, F).                    % егер N2! = F2, онда N! = F
f (N,n, F, F).                       % Рекурсиядан шығу шарты
```

3 есептеу мақсаты! мынадай көрінеді: f(0,3,1,F).

Прологтағы Рекурсия бірнеше рет қайталанатын әрекеттерді орындау үшін әрқашан пайдаланылмайды. "Жұлдызды" отбасының білім базасын еске түсіреміз, онда ерлі-зайыптылар қарым-қатынастарын, атап айтқанда *spouse(filipp, alla)* сипаттайтын предикат бар. Ата-ана қатынасына қарағанда ерлі-зайыптылар симметриялы болып табылады. Филипп Алланың жұбайы, Алла Филипптің жары. Бұл ретте алдын ала дәлелдердің жағдайы бекітілген. Басқаша айтқанда, егер білім базасындағы факт келесідей жазылса: *spouse (filipp,alla)*. ал мақсат предикаты: *spouse (alla,filipp)* нәтиже теріс болады. Прологаны көрсету үшін, бұл предикат дәлелдерге қатысты симметриялы болып табылады, біз ережені қолдана аламыз:

spouse(X,Y) :- spouse(Y,X).

Мысал ретінде белгілі коллизияны қарастырайық. Ауылда екі отбасы тұрды: анасы қызы және әкесі ұлымен. Оларға аттарды береміз. Анасы мен қызы Maria мен Dasha, ал әкесі мен ұлы Oleg, пен Sergei болсын. Есімдердің бірінші әріптері бізге кім екенін айтады, әйтпесе біз шатастырамыз. Білім базасында бұл фактілер келесі түрде көрсетіледі:

parent(oleg, sergei).

parent(maria, dasha).

Тағдырдың өзгеруіне байланысты Oleg Dashaға, ал Maria Sergeiге үйленді:

spouse(oleg,dasha).

spouse(sergei, maria).

Үшін симметрия жұбайлық қатынастар қосамыз ереже:

spouse(X, Y) :- spouse(Y, X).

Ауылда мінез-құлық қарапайым, сондықтан әкесінің әйелі ана, ал анасы әке деп аталады. Бұл үшін ережелер келесідей болады:

parent(X,Y) :- spouse(X,Z), parent(Z,Y).

Сергей немересін табуға тырысамыз. Мақсат қоямыз

grandparent(sergei, Who).

Пролог ережені табады *grandparent(X,Y) :- parent(X,Z), parent(Z,Y)* және осы ереженің денесіндегі бірінші предикаттың ішкі бағын белгілейді:

parent(sergei, Z).

Сергейдің балалары жоқ, сондықтан Пролог *parent(X,Y) :- spouse(X, Z), parent(Z,Y)* сілтеме жасайды және мақсат қояды

spouse(sergei, Z).

Пролог *Z = maria* нәтижесін береді. *Parent* ережелері екінші предикат

parent(maria,Y).

Y = dasha мәнін қайтарады. Яғни, *Sergei Mariаның* күйеуі ретінде Дашаның әкесі. Енді Пролог *grandfather* ережесіндегі екінші предикатқа өтеді:

parent(dasha,Y).

Dasha-да балалары жоқ, сондықтан Пролог *parent(X,Y) :- spouse(X,Z), parent(Z,Y)* ережесіне жүгінеді және алдымен Дашаға жұбай табуға тырысады:

spouse(dasha, Z).

Нәтиже: Z = oleg. Осы ереженің екінші предикаты parent (oleg, Y) y = sergei береді. Яғни, sergei немересі! Біз жасаған білім базасы осы коллизияны дұрыс көрсетеді.

2.6. Прологта кесу

Жоғарыда айтылғандардың бәрінен, ақиқат қорытынды жасауға болады: Пролог-аудармашы мақсаттың шешімін орындай отырып, әрқа-

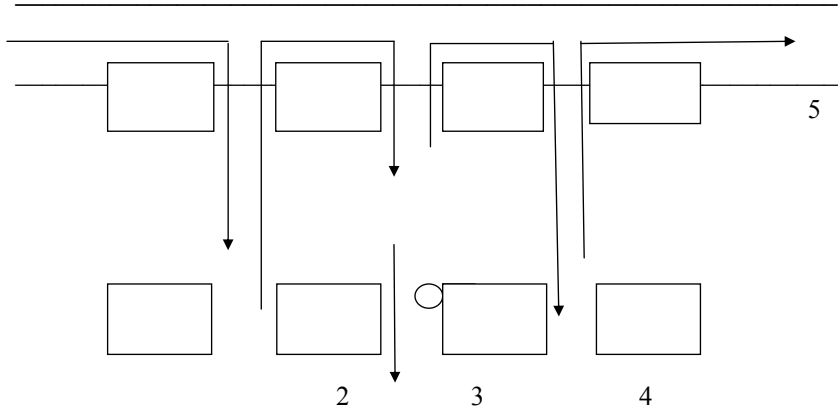
шан шешім ағашының айналасында толықтай жүреді. Ағаштың төмен түсуі ереженің негізгі бөлігінің тереңдеуіне сәйкес келеді, қайтып оралып, келесі тармаққа көшеді – сәтсіздіктен кейін кері айналу. Белгілі бір мысалда кесу ұғымын қарастырыңыз [10]. Елге шақырайық. Сонымен бірге, олар әдетте белгілі бір үйді қалай айдап, табуға болатындығы туралы толық нұсқаулар береді. Мысалы, біз алған нұсқаулар келесідей:

1. Васино ауылына кіріңіз (мүмкін, Ванино да жазылмаған).
2. Оңға бұрылыңыз.
3. Құдыққа барыңыз.
4. Қызыл кірпіштен қалаған үй - оң жағында құдықтан бірінші.

Пролог предикаттары саяжайды іздеу ережесін жазыңыз:

```
dacha1(X) :-
                enter_village(X),
                _a_house.
find_a_house :- turn_right,
                meet_mine,
                see_a_red_brick_house.
enter_village(vasino).
enter_village(vanino).
```

Іздеуді бастаймыз (2.1-сурет).



2.1-сурет. Пролог қиындысының суреттері

Біз ауылға кіріп, оңға бұрамыз (1 көрсеткі). Ауылдың соңына дейін өтіп, құдық таппаймыз. Еңіс орындаймыз, яғни шоссеге ораламыз және келесі оңға бұрылысқа дейін (2 көрсеткі) барамыз, құдыққа дейін жетеміз және оған жақын жердегі ақ кірпіштен жасалған үй екенін көреміз. Көшенің соңына дейін жүреміз және құдықтар жоқ екенін көреміз (3 көрсеткі). Тас жолға жылжып, келесі бұрылысқа дейін жүреміз(4 көрсеткі), барлық көшеден өтіп, жылжып кетеміз, себебі мұнда құдықтар жоқ (5 көрсеткі). Осыдан біз ауылдың атауын дұрыс оқыдық және саяжайды келесі ауылда іздеу керек деген қорытынды жасаймыз. *Vanino* ауылында алдымен бәрін қайталаймыз.

Кесу ағаштың кесінділерінің санын азайтуға мүмкіндік береді. Егер саяжайдың иесі бізге әрбір ауылда бір ғана құдық бар екендігі туралы маңызды қосымша ақпарат берсе (ал егер бағдар ретінде пошта немесе милиция берілсе, онда олар ауылда тек бір данада болуы мүмкін деп өздері болжай аламыз), біз 3 және 4 көрсеткілермен белгіленген әрекеттерді орындамаймыз, ал бірден сол ауылға түспейтінін түсінеміз.

Кесу - леп белгісі бар және ережеге енгізілетін предикат. Пролог оны өткеннен кейін кесу пайда болады. Кесу осы ережені біріздендіру барысында орнатылған откат көрсеткіштерін жояды, яғни осы ереженің алдыңғы предикаттарын бір мағыналы етеді.

Егер біз коттеджді іздеу бағдарламасында ауылда бір құдық бар екенін көрсеткіміз келсе, біз тапқан құдықтан кейін кесуді қою керек:

```
dacha2(X) :-
    enter_village(X),
    find_house.
    find_a_house :- turn_right,
      meet_mine, !,
      see_a_red_brick_house.
enter_village(vasino).
enter_village(vanino).
```

Пролог кесуге (отсечение) жеткенде, *find_house* барлық предикаты бір мағыналы болады, демек, *see_a_red_brick_house* предикаты да кейінгі *find_house* сәтсіздікпен аяқталады. Егер *enter_village* предикаты бірдей болмаса (бағдарламада көрсетілгендей), онда біз басқа ауылда саяжайды табуға мүмкіндігіміз бар, өйткені *find_house* ережесінде кесу *dacha* предикатының қасиеттерін және онда орнатылған құлама көрсеткіштерін қозғамайды.

Егер *enter_village* предикаты бір мағыналы болса, бізге үйге оралуымыз керек. Біз саяжайды таппадық. Егер құдықтың жанындағы үй қызыл кірпіштен шықса, онда кесіндінің болуы іздеу нәтижесінде әсер етпейді. Осылайша, біз шешім не жалғыз, не мүлдем жоқ екені белгілі болған кезде Прологтағы бағдарламаға әрқашанда қосу керек.

Ескерту. Егер саяжайды іздеу бағдарламасы келесідей болса:

```
dacha3(X) :-
        enter_village(X),
        turn_right,
        meet_mine, !,
        see_a_red_brick_house.
  enter_village(vasino).
  enter_village(vanino).
```

көрсетілген жерде салынған кесу *dacha* ережесіндегі барлық алдыңғы предикаттар бір мағыналы, соның ішінде enter_village болады. Басқаша айтқанда, бізге келесі ауылда саяжайды іздеуге тыйым салынады.

2.7. Прологтағы тізімдер

Біз бұрын жұмыс істеген барлық айнымалылар скалярлар болды. Енді Прологта деректер агрегаттарын қарастырайық. Олардың бірі – тізім. *Тізім* -бір типті элементтердің реттелген тізбегі. Тізім элементтері шаршы жақшаларға бөлінеді және үтірлермен бөлінеді:

```
[ 1, 2, 3, 4, 6, 7, 8] –                    integer
[mon,tue, wed, thu, fri, sat, sun] –        symbol
["Иванов", "Петров"] –                       string
[1.5, 2.22, 0.001, 0] –                      real
[] –                                         бос тізім
```

Тізімдер келесідей жарияланады:

```
domains
   sym = symbol*          % символдық мәндер тізімі
   intlist = integer*     % бүтін тізім
   realist =  real*       % нақты сандар тізімі
```

Тізімнің үстіне рұқсат етілетін жалғыз операция – бұл бастың құйрығынан ажырату және бұл операцияны тек предикаттардың аргументтеріне қоюға болады. Тізімдермен жұмыс істеу үшін кейбір пайдалы предикаттарды жазамыз. Атап айтқанда, қалай анықтау болып табылады ма, кейбір ауыспалы элемент тізімнің? Басы мен құйрығына тізімді бөлшектеңіз. Іздеу мәні тізімнің басында немесе құйрықта:
member (H, [H | _]).

member (X, [H | T]) :- member (X, T).

Жоғарыда жазылған амалмен тек бір элементті ғана емес, одан да көп іздеуге болады (мысалы, тізімдегі тізбекті мәндер жұбы)

memb2(H1, H2, [H1, H2 | _]).

memb2(H1, H2, [H | T]) :- memb2(H1, H2, T)

Тізімдегі мәнді қосу үшін (ең алдымен оңай) келесі ережені жасауға болады: *incl (H, T, [H | T]).*

Тізімнен мәнді алып тастау қиын:

excl (H, [H | T], T). % бастан шығарамыз

excl (X, [H,T], [H | TT]) :- excl (X, T, TT). % құйрығынан шығарамыз

Төменде тізімдер бойынша басқа да предикаттар мысалдары бар:

1. Бір тізім элементтерін экранға шығару бағдарламасы:

print_list([]). % рекурсиядан шығару

print_list ([H | T]): - write (H,""), % тізім басы мен бос орын

print_list (T). % құйрықты шығару

Егер мақсат қойылса

Print list (["мен", "есімде", "керемет","сәт"]).

онда нәтиже келесідей болады:

мен керемет сәт есімде

2. Сол бағдарлама, бірақ тізімді кері тәртіппен шығарған:

print_inverse([]). % рекурсиядан шығу

print_inverse ([H | T]) :- print_inverse (T), % құйрықты шығару

write(H, “ “). % бас тізімді және бос орынды шығару

Мақсаты:

write_inverse (["мен", "есімде", "керемет","сәт"]).

Нәтижесі:

есімде сәт керемет менің

3. Тізім элементтерінің сомасын есептейтін бағдарлама:

sum([], 0). % рекурсиядан шығу,

sum([H | T], S) :- sum(T,S1), % S1 – құйрықтағы элементтер сомасы тізім

S = S1 + H. % S-тек басын қосу қалады

сомасы ([], 0). % рекурсиядан шығу,

сомасы ([H | T], S) :- сомасы (T,S1), % S1 – құйрықтағы элементтер сомасы тізім

S = S1 + H. % S-тек басын қосу қалады

Мақсаты:

sum([1,2,3,4,5,6], S), write ("Сома =",S).

Нәтижесі: Сомасы=21

4. Тізім элементтерінің санын есептейтін бағдарлама:

count([], 0). % рекурсиядан шығу,

count([H | T], S) :- count(T,S1), % S1 – элементтер есептегіші тізімнің соңында

S = S1 + 1. % S – -бірлік сомасына қосу қалады единицу

Мақсаты:

count([1,2,3,4,5,6], S), write ("Количество =",S).

Нәтижесі: *Саны = 6*

2.8. Мысал: қасқыр, ешкі және қырыққабат туралы логикалық есепті шешу

Қасқыр, ешкі және қырыққабат туралы белгілі логикалық есепті қарастырайық. Фермер қасқыр, ешкі және қырыққабат өзенінің басқа жағасына өтуі тиіс. Қайықтың жүк көтергіштігі бір рет бортқа бір нәрсе алуға болады: [*wolf, goat, cabbage*]. Қарияның қатысуымен ешкім ешкімді

жемейді. Егер ол кетіп қалса, қасқыр ешкіні жейді, ал ешкі – қырыққабатты.

Бұл мәселені шешу үшін екі тізім ұйымдастырамыз. Бір тізім сол жағалаудың мазмұнын, екіншісі оң жағалаудың мазмұнын көрсетеді. Бастапқыда бәрі сол жағалауда. Сол жағалаудың тізімі: [wolf, goat, cabbage], оң жағалаудың тізімі бос: []. Тапсырманы сипаттайтын предикаттарды анықтаймыз.

```
stuff(wolf                                  % жүктің берілуі

stuff(goat).

stuff(cabbage).

/ * жанжалдың пайда болу шарттары */

conflict(X): - member(wolf, X), member(goat, X).

conflict(X); - member(goat, X), member(cabbage, X).

/ * Қайықтың қозғалысын сипаттайтын предикаттар:

Сол жағалаудан оң жағалауға                 */

go_right([ ], _).          % аяқтау шарты (сол жағалау тізімі бос)

go_right( L, R) : -

stuff(X),                  жүкті таңдаңыз,

member(X, L),              % сол жағалауда бар

excl(X, L, LL),            % сол жағалау тізімінен

not(conflict(LL)),         % сол жағалауда жанжал жоқ

incl(X,R,RR),              % оң жағалау тізіміне қосамыз берега

write(LL,"--",X,"-->",R),  % хабарды шығарамыз go_left(LL,RR).
%  жол артқа
```

/ * Солға қозғалыс екі нұсқада болуы мүмкін. Егер оң жақ жағада жанжал болмаса, онда фермер бір */

```
    go_left(L,R) :- not(conflict(R)),

    write(L,"<-------",R),     % хабарды шығарамыз

    go_right(L, R              % оңға қарай қозғалыс
```

предикатын шақырады

/ * Егер оң жақ жағада жанжал туындаса, біреуді кері алып кету керек. Бұл бағдарламаға қатысты жалғыз Кеңес. Қалған Пролог шешімді өз бетінше іздейді */

go_left(L,R) :-

stuff(X), % жүкті таңдаңыз,

member(X, R), % оң жағалауда бар

excl(X, R, RR), % оң жағалау тізімінен шығарамыз

not(conflict(RR)), % оң жағалауда жанжал жоқ

incl(X,L,LL), % сол жағалау тізіміне қосамыз

write(L,"<--",X,"--",RR), % хабарды шығарамыз

go_right(LL,RR). % жол артқа

Мақсат шақыру келесідей болуы тиіс:

go_right([wolf, goat, cabbage], [ ]).

Егер бұл бағдарламаны іске қоссақ, алғашқы рейспен старик ешкінің оң жағалауына апарады. Екінші рейс – қасқыр. Қасқырды ешкімен оң жақ жағада қалдыру мүмкін емес, сондықтан ол бірінші түскен жүкті алып кетеді, ал ол қасқыр болады. Және де ол қасұырды шексіз алып жүреді.

Бұл бағдарламаның жетіспеуі фермердің тек кімді ғана әкелгенін білмеуі. Оның жадын нығайту үшін *go_left u go_right* предикаттарына соңғы тасымалданған жүктің атауын қосамыз. Бағдарламаның соңғы нұсқасы келесідей:

/ * Қасқыр, ешкі және қырыққабат туралы міндет */

domains % бөлім PDC-Пролог үшін қажет. Бұл SWI-Прологе қажет емес

stuff = wolf; goat; cabbage; nil нөл % сіздің деректер түрін жасаңыз (Nil-бос)

*list = stuff** % деректер түрі тізім

predicates % бөлім PDC-Пролог үшін қажет. Бұл SWI-Прологе қажет емес

member(stuff,list)

incl(stuff,list,list)

```
        excl(stuff,list,list)
        conflict(list)
        go_right(list, list, stuff)
        go_left(list, list, stuff)
    clauses
        stuff(wolf).
        stuff(goat).
        stuff(cabbage).
        member (H, [H | _] ).
        member (X, [H | T] ) :- member (X, T).
        incl (H, T, [H | T]).
        excl (H, [H | T], T ).
        excl (X, [H,T], [H | TT] ) :- excl (X, T, TT).
        conflict(X): - member(wolf, X), member(goat, X).
        conflict(X); - member(goat, X), member(cabbage, X).
        go_right( L, R,Last) :-
            stuff(X),   % жүкті таңдаңыз,
      X <> Last,                    % бірақ тек қана емес
    member(X, L),                   % сол жағалауда бар
    excl(X, L, LL),                 % сол жағалау тізімінен шығарамыз
    not(conflict(LL)),              % сол жағалауда жанжал жоқ
  incl(X,R,RR),                     % оң жағалау тізіміне қосамыз
write(LL,"--",X,"-->",R),           % хабарды шығарамыз
  go_left(LL,RR,X).                 % жол артқа
/ * Егер оң жағалауда жанжал болмаса, біреуі кетеді * /
  go_left(L, R, Last) :- not(conflict(R)),
  write(L,"<-------",R),            % хабарды шығарамыз
  go_right(L, R, nil).              % оң жаққа жылжуды шақырады
                                    % нөл –- ештеңе жоқ
```

/ * Егер оң жақ жағада жанжал туындаса, біреуді кері алып кету керек, бірақ тек біреуді ғана алып келген адам емес*/

```
go_left(L,R,Last) :-          % хабарды шығарамыз
stuff(X),                     % жүкті таңдаңыз
X <> Last,                    % бірақ тек қана емес
member(X, R),                 % оң жағалауда бар
excl(X, R, RR),               % оң жағалау тізімінен шығарамыз
not(conflict(RR)),            % оң жағалауда жанжал жоқ
incl(X, L, LL),               сол жағалау тізіміне қосамыз
write(L,"<--",X,"--",RR),     % хабарды шығарамыз
go_right(LL, RR, X).          % жол артқа өтіп, біз Х-ті түсіргенімізді
                              есімізде сақтаймыз

goal
go_right(([wolf, goat, cabbage], [ ], nil).
```

/ * *Бағдарламаның соңы* * /

Өзін өзі бақылауға арналған сұрақтар

1. Prolg тілін дамыту кім және қашан басталды?
2. Prolog бағдарламалау тілі дегеніміз не?
3. Мұндай білім? Білім түрлері қандай?
4. Білімдердің жіктелуін келтіріңіз.
5. Прологтағы бағдарламаның негізгі элементі не?
6. Предикат ұғымын түсіндіріңіз, мысалдар келтіріңіз.
7. Бұл логика ғылым ретінде? Логиканың Объектілік анықтамасын беріңіз?
8. Ұғым, түсінік көлемі және түсінік көлеміне толықтыру дегеніміз не?
9. Түсініктердің түрлері қандай?
10. Түсініктерді түсінудің негізгі тәсілдерінің мәнін сипаттаңыз.
11. Ұғымды бөлудің мақсаты қандай? Логикалық бөлу ережелері қандай?
12. Күрделі пікірлердің мысалдарын келтіріңіз.
13. Бар умозаключение?
14. Логиканың негізгі заңдарын атаңыз.
15. Логикалық қорытынды дегеніміз не? Мысал келтір.

3-тарау. ЫҚТИМАЛ ОЙЛАР

3.1. Анық емес логика теориясының негіздері

Анық емес логика (ағылш. fuzzy logic) – классикалық логика мен жиын теориясын жалпылайтын математиканың бөлімі, ол 1965 жылы Лотфи Заде алғаш рет енгізілген анық емес жиын ұғымына негізделген, элементтің жиынға тиістілігі функциясы бар объект ретінде, тек 0 немесе 1 ғана емес, [0,1] аралығында кез келген мәндерді қабылдайтын [11,12]. Осы ұғымның негізінде анық емес жиындарға әртүрлі логикалық операциялар енгізіледі және мәні ретінде анық емес жиындар әрекет ететін лингвистикалық айнымалы ұғым тұжырымдалады..

Анық емес логиканың пәні - қарапайым мағынадағы пайымдауларға ұқсас айқындық, шайқалушылық жағдайында пайымдауларды зерттеу және оларды есептеу жүйелерінде қолдану.

Қазіргі уақытта, кем дегенде, анық емес логика саласындағы ғылыми зерттеулердің екі негізгі бағыттары бар:

– кең мағынадағы анық емес логика (жуықтап есептеу теориясы);

– тар мағынадағы анық емес логика (символдық тақ логика).

Символдық анық емес логика

Символдық анық емес логика **t-норманың** ұғымына негізделеді. Кейбір t-нормаы таңдағаннан кейін (оны бірнеше түрлі тәсілмен енгізуге болады) пропозиционалды айнымалылармен негізгі операцияларды анықтау мүмкіндігі пайда болады: конъюнкция, дизъюнкция, импликация, терістеу және басқалар.

Классикалық логикадағы дистрибутивтілік t-норма ретінде Т-Гедель нормасы таңдалғанда ғана орындалады [12].

Сонымен қатар, белгілі бір себептерге байланысты импликация ретінде residium деп аталатын операцияны жиі таңдайды (ол жалпы айтқанда, t-норманы таңдауға байланысты).

Жоғарыда аталған негізгі операцияларды анықтау классикалық булевоз таңбалы логикамен (дәлірек айтқанда, сөздерді есептеумен) көп ортақ логиканың базистік анық емес логикасын формальды анықтауға әкеледі.

Үш негізгі базистік анық емес логика бар: логика Лукасевич, Гедель логикасы және ықтимал логика (ағыл. product logic). Бір қызығы, жоғарыда аталған үш логиканың кез-келген екеуін біріктіру классикалық булдік таңбалы логикаға әкеледі.

Берілген кестедегі үздіксіз логика функцияларын синтездеу

Анық емес логика функциясы әрқашан өз дәлелдерінің бірінің мәнін қабылдайды немесе оны терістейді. Осылайша, анық емес логика

функциясын таңдау кестесіне қоюға болады, онда аргументтер мен терістеуді ретке келтірудің барлық нұсқалары көрсетілген және әрбір нұсқа үшін формулаға орын көрсетілген функцияның мәні. Мысалы, екі аргумент функциясының кесте жолы келесі түрге ие болуы мүмкін:

1) $x_1 \le x_2 \le \bar{x}_2 \le \bar{x}_1 : x_2$ 2) $x_2 \bar{x}_2 \bar{x}_1$ 3) $y(x) = \sum_{i=1}^{N} \phi_i(x) \cdot \theta_i$

Алайда еркін таңдау кестесі әрдайым анық емес логика функциясын көрсетпейді. Жұмыста [12] атты критерий тұжырымдауға мүмкіндік беретін орнату болып табылады функциясы, берілген кестені таңдау функциясы анық емес логика және ұсынылды қарапайым алгоритмі синтездеу негізделген, енгізілген тұжырымдамалары конституент минимум және максимум. Функциясы анық емес логика білдіреді дизъюнкцию конституент минимум, онда конституента максимум – бұл егер айнымалылардың ағымдағы облысының үлкен не тең мәні функциялары осы саладағы (оң жағында маңызы бар функцияларды теңсіздік қоса алғанда, мәні, функциялары). Мысалы, көрсетілген жолдан кестелер конституента минимум түрі бар $x_2 \bar{x}_2 \bar{x}_1$.

Жуықтап есептеу теориясы

Кең мағынада анық емес логиканың негізгі түсінігі – сипаттамалық функцияның жалпыланған ұғымының көмегімен анықталатын анық емес сан. Содан кейін біріктіру, қиылысу және жиындарды толықтыру ұғымдары енгізіледі (сипаттамалық функция арқылы; әр түрлі тәсілдермен қоюға болады), анық емес қатынас ұғымдары, сондай-ақ маңызды ұғымдардың бірі – лингвистикалық айнымалы ұғымдары.

Жалпы айтқанда, анықтамалардың ең аз жиынтығы кейбір қосымшаларда анық емес логиканы пайдалануға мүмкіндік береді, көпшілік үшін шығару ережесін (және импликация операторы) көрсету қажет.

Анық емес логика және нейрондық желілер

Мүшелік функциялары сипатталғандықтан, қарапайым математикалық операциялар арқылы t-нормалар мен k-нормалар нейрондық желі түрінде анық емес логикалық ойлауды елестете алады. Бұл функция үшін керек-жарақтарды нейрондарды активтендіру функциясы, сигналдарды байланыс ретінде беру, ал логикалық t-нормалар және k-математикалық тиісті операцияларды орындайтын нейрондардың арнайы түрлері ретінде нормаларды түсіндіру қажет. Neuro-fuzzy network (ағылш.). Мысалы, ANFIS (Adaptive Neuro fuzzy Inference System) – бейімделген нейро-бұлдыр анықтамалық жүйе [13].

Оны аппроксиматорлардың әмбебап түрінде сипаттауға болады:

$$y(x) = \sum_{i=1}^{N} \phi_i(x) \cdot \theta_i$$

Сонымен қатар, осы формула арқылы нейрондық желілердің кейбір түрлерін, мысалы радиалды негіздік желілерді (RBF), көп қабатты перцептрондарды (MLP), сондай-ақ толқындар мен сплиттерді сипаттауға болады [13].

Информатикадағы анық емес логика

Анық емес логика – радикалды идеялар, интуитивті болжамдар, сонымен қатар тиісті салада жинақталған мамандардың тәжірибесін қойылған мақсатқа жету үшін қолдануға болатын бос ережелер жиынтығы. Анық емес логика қатаң стандарттардың болмауымен сипатталады. Көбінесе ол сараптамалық жүйелерде, нейрондық желілерде және жасанды интеллект жүйелерінде қолданылады. Шындықтың дәстүрлі мәндерінің орнына анық емес логикада шындықтың кең диапазоны қолданылады, олардың арасында шындық, өтірік, мүмкін, кейде есімде жоқ (иә, неге және жоқ, әлі шешпеді, айтпайды...). Қойылған сұраққа нақты жауап жоқ (иә немесе жоқ; "0" немесе "1") немесе барлық мүмкін жағдайлар алдын ала белгісіз жағдайларда анық емес логика жай ғана таптырмас. Мысалы, анық емес логикада «X – бұл үлкен сан» формуласы түсініксіз мағынамен түсіндіріледі, кейбір түсініксіз жиынтықпен сипатталады. "Жасанды интеллект және нейрондық желілер - бұл адамның мінез-құлқын компьютерде модельдеу. Ал адамдар қоршаған ортаны қара-ақ түспен ғана сирек көретіндіктен, анық емес логиканы пайдалану қажеттілігі туындайды".

Күрделі пікірлердің дұрыстығын есептеу ережелері келесі:

$$T(A \wedge B) = \min(T(A), T(B))$$

$$T(A \vee B) = \max(T(A), T(B))$$

$$T(A) = 1 - T(A)$$

3.2. Байесов желілері

Барлық студенттерге жақын қарапайым мысал қарастырайық. Емтихан (Pass) тапсыру үшін оған (Study) дайындалу немесе шпаргалка (Cheat) пайдалану керек. Осылайша, 3 бульдік айнымалы бар. Емтихан тапсыру мүмкіндігін білгіміз келеді.

Қарапайым (атомдық) оқиғалардың ықтималдығы белгілі болған жағдайларда, толық бірлескен бөлу негізінде ықтималдық шығару әдісін қолдануға болады, оның мөлшері 2x2x2 кестемен сипатталады [12].

	Study		¬Study	
	Cheat	¬Cheat	Cheat	¬Cheat
Pass	0,15	0,4	0,04	0,06
-Pass	0,01	0,04	0,05	0,25

Барлық ықтималдықтар сомасы бірге тең. Әрбір торда қарапайым оқиғаның болу ықтималдығы. Бұл ықтималдық нәтиже болып табылады, яғни барлық факторларды ескереді. Осылайша, емтиханды табысты тапсыру ықтималдығы 0,4 емтиханға дайындық ықтималдығын және студенттің шпаргалканы пайдаланбауы ықтималдығын ескереді.

Күрделі оқиғалардың ықтималдығы кестедегі тиісті жолдарды немесе бағандарды қосу арқылы оңай есептеледі. Емтиханға дайындық ықтималдығы кестенің сол жақ жартысындағы клеткалар жиынтығына тең және оқиғаларға сәйкес келеді (оқыды, шпаргалка пайдаланбады және тапсырды; оқыды, пайдаланбады және тапсырды; оқыды, пайдаланбады және тапсырмады; оқыды, пайдаланбады және тапсырмады; оқыды, пайдаланбады және тапсырмады):

$$P\ (Study) = 0,15 + 0,4 + 0,01+ 0,04 = 0,6$$

Шпаргалканы пайдалану ықтималдығы бірінші және үшінші бағандардың сомасына тең:

$$P(Cheat) = 0,15 + 0,01 + 0,04 + 0,05 = 0,25$$

Емтихан тапсыру ықтималдығы бірінші жолдың клеткаларының жиынтығына тең (оқыған, пайдаланған және тапсырған; оқыған, пайдаланбаған және тапсырған; оқымаған, пайдаланған және тапсырған; оқымаған, пайдаланбаған және тапсырған):

$$P\ (Pass) = 0,15 + 0,4 + 0,04 + 0,06 = 0,65$$

Толық ортақ үлестіру негізінде ықтималдықты шығару әдісі практикалық құралдан гөрі ықтималдықты қалыптастыру қағидатының жақсы көрінісі болып табылады, өйткені қарапайым оқиғалардың ықтималдығы әрдайым белгілі емес. Бұл оқиғалардың тең ықтималдығы туралы жорамалдар жиі жасайды.

Әдетте, кейбір айнымалылардың ақиқаттылығын бағалауға болады. Болсын

P (Study) = 0, 6 (40% жағдайда емтиханға дайындалудан гөрі маңызды істер болады),

P (Cheat) = 0, 25 (төрт шпаргалка пайдалануға бір мүмкіндік).

Бұл ықтималдықтар априорлы немесе сөзсіз деп аталады. Олар басқа мәліметтердің жоқтығына пікірдің ақиқаттығына сенімділік дәрежесін білдіреді.

Бір қарағанда, емтихан тапсыру ықтималдығы 0,6 + 0,25 = 0,85 тең. Іс жүзінде бәрі қиын. Study және Cheat оқиғалары бірге болуы мүмкін. Материалды үйренуге болады және сақтандыру үшін шпаргалканы қолдануға болады. Сіз сондай-ақ бәрін үйренуге болады, бірақ тапсыруға болмайды (профессор келеді). Дайындық пен шпаргалкасыз тапсыруға болады (тек бақытты).

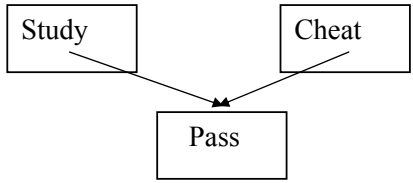
Іздеу ықтималдығын табу үшін шартты ықтималдықтарға ие болу қажет, мысалы, P(Pass|Study) – толық дайындық жағдайында емтихан тапсыру ықтималдығы.

Жалпы жағдайда оқиғаның ықтималдығы а тең

Жалпы, А оқиғасының ықтималдығы тең

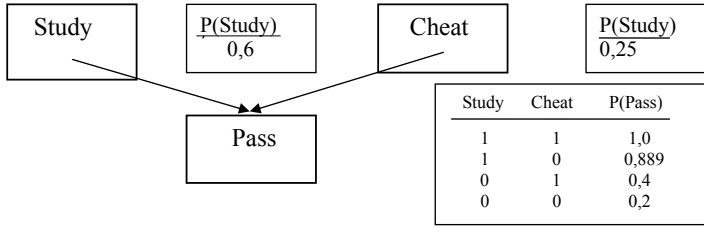
$$P(A) = \sum P(A|Bi)\, P(Bi) \qquad (3.1)$$

мұнда P(Bi) - оқиғаның априорлық ықтималдығы; P (A | Bi) - Bi оқиғасы шын болған жағдайда, А оқиғасының шартты ықтималдығы. Шартты ықтималдылық тәуелді оқиғаларды байланыстырады. Егер біз төртінші айнымалы – күн ауа райын енгізсек (Sunny), онда P (A | Sunny) типті шартты ықтималдықтар қою керек. Нақты жағдайларда біз міндеттің өлшемдігі осыған байланысты тым үлкен болады деп тап бола аламыз. Бұл мәселені шешу үшін байесов желілері қолданылады, олар айнымалы тәуелділікті орнатуға және толық ортақ таралуды есептеуді жеңілдетуге мүмкіндік береді. Төменде қаралған мысал үшін байесовская желісі бар: P(Bi), Bi, P(A|Bi)



Біздің модельде Study және Cheat айнымалылар тәуелсіз болып табылады.

Шпаргалканы пайдалану емтиханға дайындықтың болмауына байланыстыбасқа модель болуы мүмкін. Ол кейінірек қаралады.



Study	Cheat	P(Pass)
1	1	1,0
1	0	0,889
0	1	0,4
0	0	0,2

Желінің әрбір шыңы кездейсоқ айнымалыға сәйкес келеді. Шыңдары бағытталған қабырғалармен қосылады. Егер көрсеткі А-дан В-ға бағытталса, онда в бас шыңы деп аталады. X_i әр шыңы $P(X_i|Parents(X_i)$

шартты ықтималдығының таралуымен сипатталады, ол оның ата-ана шыңының шыңына әсерін сан жағынан бағалайды. Біздің мысалда келесі шартты ықтималдықтар белгілі деп санаймыз: шпаргалкаларды дайындау және сақтандыру кезінде емтихан тапсыру ықтималдығы бірлікке тең; шпаргалкаларды дайындау және пайдаланбау кезінде – 0,889; шпаргалканы емтиханға дайынсыз пайдалану шартымен – 0,4; ал емтихан дайындау және шпаргалкасыз тапсыру ықтималдығы – 0,2 (жай бақытты). Байесов желілерін пайдалану кезіндегі негізгі ұтыс-бұл шыңға байланысты барлық шыңдар емес, тек ата-ана (жақын) шыңдары негізінде ғана кез келген жағдайдың ықтималдығы:

$$P(X_i \mid X_{i-1}, X_{i-2}, \ldots, X_1) = P(X_i \mid \text{Parents}(X_i)) \qquad (3.2)$$

Біздің мысалда біз байланысты оқиғалардың түпкі нәтижесі болуы мүмкін Study шыңына ие: студент дәрістерге қатысып, конспектісі болды, оның компьютерге кіру мүмкіндігі болды, ол уақыт алды, және т.б. осы оқиғалардың нәтижесі ретінде біз емтиханға дайындық фактісі бар. Шпаргалканы пайдалану оқиғасы, сондай-ақ, шпаргалканы жасырын пайдалану үшін киім сияқты уақыттың, техникалық құралдардың болуы ықтималдығы бар бірқатар оқиғалардың салдары болып табылады. Толық ортақ бөлуді есептеу үшін барлық шартты ықтималдықтарды білу қажет, мысалы, дәрістерге келген жағдайда радиотаратқыштың болу ықтимал-дығын. Біз емтихан тапсыру ықтималдығын есептеу үшін P(Оқу) және P(Cheat) ықтималдығын білу жеткілікті.

Жоғарыда келтірілген ереже **тізбекті ереже** деп аталады. Осы Ережені басшылыққа ала отырып, оқиғалар ықтималдығын есептеу жеткілікті. Бұл мысалда біз емтихан тапсыру ықтималдығын есептей аламыз:

$$\begin{aligned} P(\text{Pass}) &= P(\text{Pass} \mid \text{Study}, \text{Cheat}) * P(\text{Study}) * P(\text{Cheat}) + \\ &+ P(\text{Pass} \mid \text{Study}, \neg\text{Cheat}) * P(\text{Study}) * P(\neg\text{Cheat}) + \\ &+ P(\text{Pass} \mid \neg\text{Study}, \text{Cheat}) * P(\neg\text{Study}) * P(\text{Cheat}) + \\ &+ P(\text{Pass} \mid \neg\text{Study}, \neg\text{Cheat}) * P(\neg\text{Study}) * P(\neg\text{Cheat}) = \\ &= 1{,}0 * 0{,}6 * 0{,}25 + 0{,}889 * 0{,}6 * 0{,}75 + 0{,}4 * 0{,}4 * 0{,}25 + 0{,}2 * 0{,}4 * 0{,}75 = \\ &0{,}65 \end{aligned}$$

Біздің қатты жеңілдетілген мысалда есептеу күрделілігінде бұл ұтыстар байқалмайды, өйткені байесов желісінің тізбегі ұзын емес. Екінші фактор – Study және Cheat айнымалыларының тәуелсіздігі. Тәжірибелі студенттер кейбір мүмкіндіктердің дұрыс еместігін байқай алады: емтиханға дайындық кезінде шпаргалка тек қана ұстап алу және емтиханнан шығару қаупін қосады. Логиканы келесідей өзгертеміз. Егер емтиханға дайындалмаса, студент шпаргалка қолдануға тырысамыз деп есептейміз. Байесов желісі төменде көрсетілгендей өзгереді.

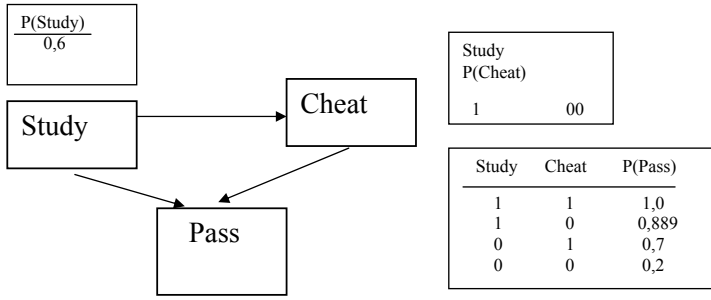
Шпаргалканы пайдалану шартты ықтималдығы емтиханға дайындық жағдайында нөлге тең және дайындық болмаған жағдайда 0,75. ықтималдығы

P(Pass | Study, Cheat) = 0.

Есептеу P(Cheat) және P(¬Cheat):

P(Cheat) = P(Cheat | ¬Study) * P(¬Study) = 0,75 *(1-0,6) = 0,3

P(¬Cheat) = 1- P(Cheat) = 0,7



Енді тізбекті ережені пайдалана отырып, біз емтиханды табысты тапсыру мүмкіндігін есептей аламыз:

3.3. Көрнекіліктер: Монти Холл парадоксы

Байес ережесінің қолданылуын көрсету үшін біз әдебиеттегі Monty Hall парадоксы деп аталатын келесі мәселені қарастырамыз [15]. Сізге теледидар ойынына қатысуға рұқсат етіңіз, оның барысында үш есіктің біреуін таңдау керек. Олардың бірінің артында көлік тұр, қалған екеуінің артында ешкілер тұр. Сіз 1-ші есікке нұсқадыңыз делік. Үй иесі тағы бір есікті ашады, ол №3 есік болсын, оның артында ешкі табылып, сізді кеш болмай тұрып, ойыңызды өзгертуге және №2 есікті ашуға шақырады. Сұрақ: Баяндамашы мен екінші есіктің пікірін тыңдаудың мағынасы бар ма? Бір қарағанда, көліктің қалған есіктердің біреуінің артында болуы ықтималдығы

P (C1) = P (C2) = 1/2, сондықтан шешімнің мағынасын өзгертпейді. Дегенмен, мұқият әрекет ете отырып, үй иесінің №3 есікті ашқанынан пайдалы ақпарат аламыз. Көліктің №1, №2 және №3 есіктің артында тұрғанына априоридің ықтималдығы болсын

$$P(C1) = P(C2) = P(C3) = 1/3.$$

Сөйлесушінің №2 және №3 есікті ашатындығы туралы ықтималдылықты таба аламыз (ол №1 есікті аша алмайды, өйткені біз бұны жоғарыда айтқанбыз). Егер көлік №1 есіктің сыртында болса, онда оның 2 және 3 нөмірлерінің ашылатындығы туралы шартты ықтималдық:

$$P(D = 2 \mid C1) = 0{,}5;\ P(D = 3 \mid C1) = 0{,}5$$

Яғни, жүргізуші қалған есіктердің қайсысын ашқаны маңызды емес. Егер көлік №2 есіктің сыртында болса, онда оның №2 және 3 нөмірлерінің ашылуы ықтималдығы:

$$P(D = 2 \mid C2) = 0;\ P(D = 3 \mid C2) = 1$$

Көлік екінші есіктің артында тұрғандықтан, көшбасшы бұл есікті аша алмайды, бірақ 1 мүмкін болса, үшінші есікті ашады. Керісінше, егер көлік үшінші есіктің артында болса, онда тиісті ықтималдықтар:

$$P(D = 2 \mid C3) = 1;\ P(D = 3 \mid C3) = 0$$

Сонымен, бізде көліктің қай жерде орналасқанына байланысты көшбасшының қай есікті ашатындығы бар. Кері шартты ықтималдылықты табу үшін Байес ережесін қолданамыз.

D = 3 екендігі белгілі болғандықтан, онда

$$P(Ci \mid D = 3) = P(D = 3 \mid Ci) * P(Ci) / P(D = 3)$$

Осы формулаға C1 және C2 алмастыра отырып, біз:

$$P(C1 \mid D = 3) = P(D = 3 \mid C1) * P(C1) / P(D = 3) = ½ * 1/3 / 1/2 = 1/3.$$

$$P(C2 \mid D = 3) = P(D = 3 \mid C2) * P(C2) / P(D = 3) = 1 * 1/3 / 1/2 = =2/3.$$

Осылайша, машинаның екінші есіктің артында болу ықтималдығы екі есе жоғары! «Қаламның ұшында» жасалған тұжырымға логикалық пайымдау арқылы қол жеткізуге болады: егер көлік біз таңдаған бірінші есіктің артында болса, онда көшбасшы қалған есіктердің қайсысын таңдауға мән бермейді. Егер көлік екінші есіктің артында болса, онда көшбасшының №3 есіктен басқа амалы жоқ. Таңдауды өзгерте отырып, нашар жағдайда бізде бірдей мүмкіндік қалады, ал ең жақсы жағдайда көлік аламыз.

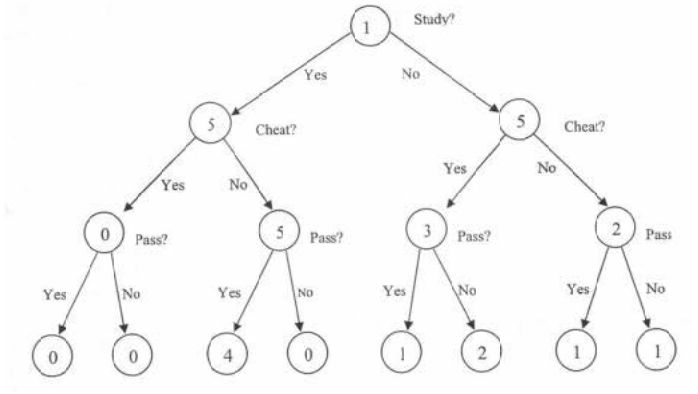
3.4. Бақылау негізінде оқыту

Алдыңғы бөлімде қарастырылған мысалдар ақылға қонымды сұрақ туғызуы мүмкін: шынайы өмірде осы керемет құралдарды қалай пайдалану керек? Барлық осы ықтималдылықтарды қайдан алуға болады? Математиктер бұл сұрақты өте қарапайым түрде айналып өтеді. Барлық пайымдау сөздерден басталады: А оқиғасының ықтималдығы Х-ке тең болсын, осылайша математиктер қажет болғанынша жасайды. Екінші жағынан, инженер мүмкін нәрсені жасау керек. Егер сіз еденге лақтыру немесе карталардың палубасын салу сияқты идеалды мысалдарды

қолданбасаңыз, онда сіз тек бақылауға негізделген ықтималдылықты таба аласыз.

№	Study	Cheat	Pass
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	No	Yes
4	Yes	No	No
5	No	Yes	No
6	No	No	No
7	Yes	No	Yes
8	No	Yes	No
9	Yes	No	Yes
10	No	No	Yes

Осы үлгіден біз емтиханды тапсыру ықтималдығын есептей аламыз, ол P(Pass) = 6 / 10 = 0,6 өткендердің жалпы санына сәтті өткендер санының қатынасына тең. Сол сияқты, біз P (Study) = 5/10 = 0,5 мөлшерін және P(Cheat) = 3/10 = 0.3 парағын қолдана отырып, ықтималдығын аламыз. Сол сияқты P (Pass|Study) = 3/5 = 0.6 шартты ықтималдықтарды аламыз (емтиханды тапсырғандардың емтиханға дайындалғандардың жалпы санына қатынасы) және P (Pass|Cheat) = 1 / 3. Таралуды ағаш түрінде көрсету ыңғайлы (3.1-сурет):



3.1-сурет. Бақылау ағашы

Мұнда, бұрын қарастырылған іздеу мәселелеріндегідей, ең төменгі тереңдіктегі ағашты қалыптастырған жөн. Осыған байланысты, ең алдымен, ең бастысын тарту қажет. Ақпараттық іздеу атрибуттары. Атрибутты таңдаудың бір қолайлы критерийі – жауаптағы ақпараттың мөлшері.

Ақпараттың мөлшері Шеннон формуласымен есептеледі. Егер ықтимал жауаптардың $p(v_i)$,ықтималдығы болса, онда I нақты жауаптың ақпараттық мазмұны битпен өлшенеді.

$$I(p(v_1), p(v_2), \dots, p(v_n) = -\sum p(v_i) \log_2 p(v_i) \quad (3.4)$$

Емтиханға дайындық туралы жауаптағы ақпарат көлемін есептейміз:

$$I(p(Study), p(\neg Study)) = - 5/10 * \log_2 5/10 - 5/10 * \log_2 5/10 = 1 \text{ бит}$$

Жауап парағындағы ақпарат мөлшері:

$$I(p(Cheat), p(\neg Cheat)) = - 3/10 * \log_2 3/10 - 7/10 * \log_2 7/10 = 1{,}533 \text{ бит}$$

Осылайша, чит парағы туралы жауап бір жарым есе көп ақпарат береді, оны іздеу ағашына ертерек қою керек. Тағы бір мәселе - оқу жиынтығындағы ақпараттың жеткіліктілігі. Жоғарыда келтірілген мысал өте өрескел баға береді. Нақты ықтималдықтар айтарлықтай өзгеше болуы мүмкін. Тағы бір экстремал - бұл үлкен популяцияға сәйкес келетін іріктеменің үлкен мөлшері. Егер біз барлық студенттер үшін экскурсияны ұйымдастыратын болсақ, біз барлық сұрақтарға толық жауап аламыз және бұдан былай модельдер қажет болмайды. Үлгіні қолданудың тиімділігі аз ғана үлгі бойынша бүкіл халық туралы қорытынды жасауға болатын кезде қол жеткізіледі. Мысалы, егер сіз студенттердің 10%-ы емтиханды алаяқтық парақтарды қолдана отырып өткізетінін білсеңіз, емтихан форматын өзгерте аласыз, мысалы, кез-келген дереккөздерді пайдалануға рұқсат бере аласыз, бірақ сұрақтарды қиындата аласыз.

Үлгі жеткіліктілігін өлшеудің бір әдісі – әр жаңа итерацияның ықтималдылық өзгерістерін салыстыру арқылы бастапқы зерттеу кестесіне деректерді біртіндеп қосу. Нәтижелердің өзгеруі рұқсат етілген қателіктен аз болған кезде, жаттығуды аяқталған деп санауға болады.

Өзін өзі бақылауға арналған сұрақтар

1. Тақ логика дегеніміз не?
2. Бұл тақ логиканың мәні.
3. Символдық тақ логика неге негізделеді?
4. Үш негізгі базистік тақ логиканы атаңыз.
5. Логиканың негізгі заңдарын атаңыз.
6. Логикалық қорытынды дегеніміз не? Мысал келтір.
7. Лингвистикалық айнымалыға түсінік беріңіз.
8. Тақ жиынға анықтама беріңіз.
9. Керек-жарақтың функцияларына арналған қисық формаларын атаңыз.
10. ANFIS дегеніміз не?
11. Не пікір? Мысал келтір.

12. Пікірдің қандай құрылымдық элементтерін білесіз? Анықтама беріңіз және мысал келтіріңіз.
13. Керек-жарақтың функцияларына арналған қисық формаларын атаңыз.
14. Байесов желілері қандай мәселелерді шешу үшін қолданылады?
15. Монти Холл парадокс дегеніміз не?

4-тарау. НЕЙРОНДЫҚ ЖЕЛІЛЕР

4.1. Нейрондық желі туралы түсінік

Нейрондық желілер биологиялық ағзалардағы жүйке жүйесінің примитивті моделін қалпына келтіруге негізделген. Тірі табиғатта нейрон – бұл электрлік қоздырғыш жасуша, ол синаптикалық байланыстар арқылы электрлік және химиялық сигналдарды пайдалана отырып, ақпаратты өңдейді, сақтайды және береді. Нейронның күрделі құрылымы және тар мамандануы бар. Синапстарды пайдаланып сигналдарды беру үшін бір-бірімен байланысып, нейрондар биологиялық нейрондық желілерді құрады. Адамның миында орта есеппен 65 миллиард нейрон және 100 триллион синапс бар. Шын мәнінде, бұл барлық тіршілік иелерінің оқу және ми белсенділігінің негізгі механизмі, яғни – олардың ақыл-парасаты. Мәселен, Павловтың классикалық экспериментінде, әрдайым итті тамақтандырар алдында, қоңырау соғылды, ит тез қоңырауды тамақпен байланыстыруды үйренді. Физиологиялық тұрғыдан алғанда, оның миындағы эксперименттің нәтижесі есту үшін жауап беретін ми қыртысының бөліктері мен сілекей бездерін басқаруға жауапты аймақтар арасында синапстық байланыстарды орнату болды. Нәтижесінде, қоңырау үнімен қабығы қозған кезде, ит тұздай бастады.

Сонымен, ит сыртқы әлемнен келетін сигналдарға (мәліметтерге) реакция жасап, «дұрыс» қорытынды жасауға үйренді. Бұл жасанды интеллект саласындағы зерттеулердің негізін құрайтын биологиялық жүйке жүйелерінің қателіктерін оқып білу және түзету қабілеті. Олардың алғашқы міндеті мидың төменгі деңгейлі құрылымын жасанды түрде көбейтуге тырысу болды. «жасанды ми» компьютерін жасаңыз. Нәтижесінде «жасанды нейрон» ұғымы ұсынылды – бірнеше кіріс фактілерін бір нәтижеге айналдыратын және оларға әсер ету салмағын беретін математикалық функция [17]. Әрбір жасанды нейрон кіріс сигналдарының салмақты мөлшерін қабылдай алады және егер жалпы кіріс белгілі бір шекті деңгейден асып кетсе, екілік сигналды одан әрі жібере алады.

Жасанды нейрондар желілерде біріктірілген – кейбір нейрондардың шығуын басқаларының кірістерімен байланыстырады. Жасанды нейрондар байланысқан және өзара әрекеттесетін жасанды нейрондық желі жасайды – бағдарламалық немесе аппараттық құралдарда жүзеге асырылатын нақты математикалық модель. Қарапайым тілмен айтқанда, нейрондық желі – бұл деректерді қабылдайтын және жауап беретін қара жәшіктің бағдарламасы. Көптеген қарапайым элементтерден тұратын нейрондық желі өте күрделі мәселелерді шешуге қабілетті.

Бір нейронның (перцептронның) математикалық моделін алғаш рет 1943 жылы американдық нейрофизиологтар мен математиктер Уоррен МакКаллок, Уолтер Питтс ұсынған және олар жасанды нейрондық желі анықтамасын ұсынған. Физикалық тұрғыдан алғанда, компьютерді қолданатын модельді 1957 жылы Фрэнк Розенблат жасаған. Нейрондық желілер ЖИ-ді практикалық іске асырудың ең көне идеяларының бірі деп айта аламыз.

Қазіргі уақытта нейрондық желілерді жүзеге асырудың көптеген модельдері бар. «Классикалық» бір қабатты нейрондық желілер бар, олар қарапайым мәселелерді шешу үшін қолданылады. Бір қабатты нейрондық желі әдеттегі сарапшы модельдерде қолданылатын әдеттегі полиномдық, салмақтық функциясымен математикалық түрде бірдей. Көпмәндегі айнымалылардың саны желілік кірістердің санына тең, ал айнымалыларға дейінгі коэффициенттер синапстардың салмақтық коэффициенттеріне тең.

Математикалық модельдер бар, онда бір нейрондық желінің шығуы екіншісінің енуіне бағытталады, ал қосылыстардың каскадтары құрылады, көп қабатты нейрондық желілер деп аталады (MNN, көп қабатты нейрондық желі) және оның ең күшті нұсқаларының бірі - конвульсиялы нейрондық желілер.

MNN компьютерлерінің үлкен есептеу мүмкіндіктері бар, бірақ сонымен бірге үлкен есептеу ресурстарын қажет етеді. АТ жүйелерін бұлтты инфрақұрылымға орналастыруды ескере отырып, көп қабатты нейрондық желілер көп қолданушылар үшін қол жетімді болды және қазіргі заманғы ЖИ шешімдерінің негізіне айналды. 2016 жылы Америка Құрама Штаттарынан келген сандық ойлау, танымдық есептеу компаниясы 160 миллиард сандық нейроннан тұратын нейрондық желіні құрды және оқыды. Бұл Google-ге (11,2 миллиард нейрон) және Ливермордағы АҚШ ұлттық зертханасына (15 миллиард нейрон) қол жетімді нейрондық желілерге қарағанда әлдеқайда күшті.

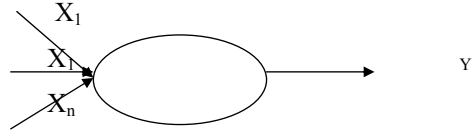
Нейрондық желінің тағы бір қызықты түрі – бұл желілік қабаттан шығыс кірістердің біріне қайтарылған кезде кері байланыс нейронды желісі (RNN, қайталанатын нейрондық желі). Мұндай платформалар «жад эффектіне» ие және олар кіріс факторларының өзгеру динамикасын бақылай алады. Қарапайым мысал – бұл күлімдеу. Адам өзінің эмоцияларын анық көрсетпестен бұрын, оның көздері мен бетінің бұлшық еттерінің байқалмайтын қозғалыстарымен күле бастайды. RNN сізге мұндай қозғалысты ерте кезеңдерде анықтауға мүмкіндік береді, бұл тірі заттың мінез-құлқын уақытылы алдын-ала болжау үшін пайдалы, суреттер сериясын tCheat немесе табиғи тілде тұрақты сөйлеу ағынын құру.

4.2. Нейрондық желілерді құру принципі

Компьютерді қолдана отырып кез-келген мәселені шешудің стандартты әдісі – бұл тапсырма зерттеліп, алгоритм құрастырылып, ол программа тілінде бағдарлама түрінде орындалады. Жөндеуден кейін бағдарлама жұмыс істеуге дайын.

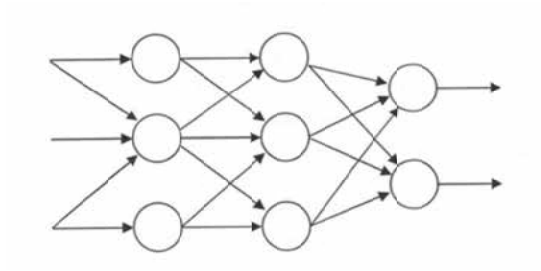
Бізге баскетбол ойнайтын роботты, яғни допты себетке лақтыруға жауап беретін бөлімді жасау жүктелсін. Біріншіден, әр түрлі позициялардан доптың ұшуы үшін дифференциалдық теңдеулерді құрастыру, ауаға төзімділікті түзету, қоздырғышқа қатысты сенсорлардың параллаксы және т.б. Осыдан кейін сіз бағдарламалауды бастауға болады. Сонымен бірге, Шакил О'Нил жоғары білім туралы айтпағанда, ең болмағанда, қарапайым математиканы біледі деп ешкім айтуға келмейді. Адам миы бұл мәселені түбегейлі басқаша шешетіні анық. Бұл әдіс не екенін бәрі біледі – бірнеше рет қайталау.

Алдыңғы бөлімде біз нәтижесі қандай да бір оқиғалардың туындау ықтималдығы болып табылатын оқытуды қарастырдық. Ал бірнеше рет қайталаулар негізінде қандай да бір бағдарламалау жоқ болатын есептеуіш құрылым құру мүмкін бе, есептерді шешуге оқыта алар еді. Мұндай құрылым *нейрондық желі* деп аталады. Идея табиғаттан алынған. Нейрондық желі нейрондардың жиынтығы – есептеу элементтері (кейде персептрондар деп аталады), олардың әрқайсысы бірнеше кіріс-синапс және бір шығу-аксон бар [17].



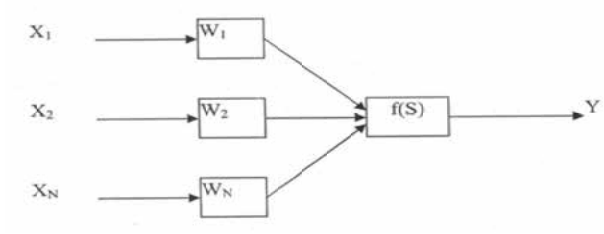
4.1-сурет. Жалғыз нейрон

Бір нейронның интеллектісі төмен (4.1-сурет). Ол N жиынындағы тәуелсіз айнымалылар үшін қарапайым регрессиялық модельді іске асырады деп болжауға болады. Егер біз көптеген нейрондарды желілік құрылымға біріктіретін болсақ, онда орындалатын функция өздігінен күрделі болуы мүмкін. 4.2-суретте көрсетілген желі нақты қабаттарға ие, яғни кіруден (шығудан) тең нейрондардың қатарлары. Басқа құрылымдар, соның ішінде кері байланысы бар (қайталанатын) құрылуы мүмкін.



4.2-сурет. Нейрондық желі

Мұндай желіні іске асыру әрбір нейронды жеке микропроцессорда орындаған кезде немесе бағдарламалық жасақтама болуы мүмкін немесе егер нейрондар электрондық кестелер сияқты арнайы бағдарламаларға енсе. Жеке нейронның құрылымы еркін болуы мүмкін, бірақ көбінесе төмендегілер қолданылады (4.3-сурет):

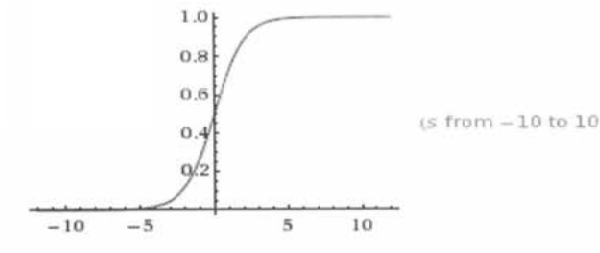


4.3-сурет. Нейронның құрылымы

X_i кіріс сигналдары (айнымалылар) өлшенеді (синаптикалық салмақ деп аталатын W_i коэффициенттеріне көбейтіледі), содан кейін қосылады және алынған өлшенген сома

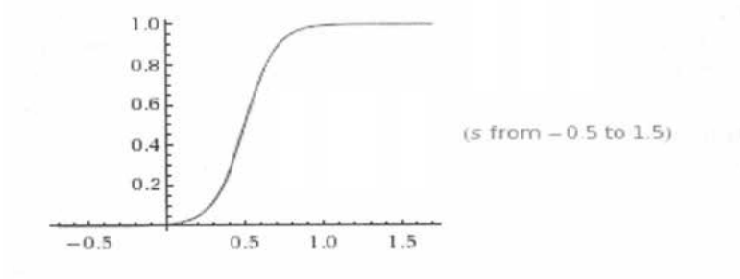
$$S = W_1X_1+W_2X_2+\ldots+W_NX_N \qquad (4.1)$$

f (S) функциясы арқылы өзгертілуі мүмкін, оны активтендіру функциясы деп атайды. Y шығыс сигналы салмаққа (масштабтауға) байланысты болуы мүмкін. Сигмоидты функция $Y = 1 / (1 + \exp(-\lambda S))$, сондай-ақ гиперболалық тангенс, логарифмдік функция, сызықтық және басқалары активация функциясы ретінде жиі қолданылады. Мұндай функцияларға қойылатын басты талап – монотондылық.



4.4-сурет. Логистикалық сигмоидтық функция

Енді бір нейронның есептеу мүмкіндіктерін қарастырайық. Синапстардың саны үшке тең болсын, ал синапстық салмағы W_1, W_2, W_3 1.0-ге тең, ал активтендіру функциясы график түрінде көрсетілген келесі формада болады (бұл X осі бойымен 0,5-ке оңға жылжыған логистикалық сигмоидтық функция: (4.5-сурет):



4.5-сурет. Логикалық функцияларды іске асыру үшін активтендіру функциясының түрі

Кіріс сигналдары 0 немесе 1 мәндерін қабылдасын. Бұл жағдайда, егер кіріс сигналдарының кем дегенде біреуі 1 болса, яғни ажырату функциясы болса, шығыс шамамен 1 болады. Егер W_1, W_2, W_3 синаптикалық салмағы 0,2-ге тең болса, сол нейрон конъюнктура функциясын орындайды.

4.3. Нейрондық желіні оқыту

Миға тәуелділік аналогиясы нейронның құрылымы мен нейрондар желісіне қатысты болмайды. Нейрондық желілерді оқыту идеясы да табиғаттан алынған. Адамның миы өздігінен үйренуге қабілетті екендігі

белгілі және көбінесе орындалатын әрекеттердің негізін құрайтын процестердің сипатын білмей, сәттілікке жетеді. Мысалы, допты баскетбол шеңберіне қосу үшін баскетболшы робот шеңберге және бағытқа дейінгі қашықтықты өлшеп, параболалық траекторияны есептеп, доптың массасы мен ауаның тұрақтылығын ескере отырып лақтыруы керек. Ер адам мұны тек жаттығу арқылы жасай алады. Бірнеше рет лақтырулар жасап, нәтижелерін байқай отырып, ол өз әрекеттерін біртіндеп жетілдіріп, техникасын жетілдіреді. Сонымен қатар оның миында құю техникасына жауап беретін нейрондардың тиісті құрылымдары қалыптасады. Осылайша, жаттығудың ажырамас атрибуты - қайталану және нәтижені дереу бағалау мүмкіндігі. Нейрондық желілер үшін бұл процесті келесі алгоритммен ұсынуға болады (4.6-сурет) [18]:

4.6-сурет. Нейрондық желіні құру және пайдалану процесі

Нейрондық желінің құрылымын таңдау жеке міндет болып табылады және әр нейронның желілік топологиясын және активтендіру функцияларын таңдаудан тұрады. Бастапқыда нейрондардың параметрлері өздігінен орнатылады.

Оқыту арнайы жаттығу мәліметтерін желінің кірісіне, яғни шығысы белгілі болатын кіріс деректерінің берілуінен тұрады. Нәтижесінде алынған мәліметтер шығарылады, нәтижелер күтілгендермен салыстырылады және қателік мәні есептеледі [18,19].

Осыдан кейін, қателік функциясын азайту үшін нейрондық желі параметрлері түзетіледі. Егер қанағаттанарлық дәлдікке қол жеткізілмесе, сіз желінің құрылымын өзгертіп, жаттығулар туралы мәліметтер

жиынтығында қайталауыңыз керек. Желіні оқығаннан кейін тестілеу жүргізіледі, яғни арнайы сынақ деректерінде дәлдікті бақылау. Бұл барлық деректерді екі ішкі топқа бөлу керек дегенді білдіреді: біріншіден, желілік жаттығулар, ал екіншіден – тестілеу. Бұл бөлім кездейсоқ немесе тұрақты болуы мүмкін, мысалы, бастапқы деректер массивінің әрбір екінші жазбасын тестілеуге қолдануға болады. Жаттығудан тестілеудің ерекшелігі, тестілеу деректері тек нақтылықты тексереді, және бұл деректер желінің параметрлерін таңдау үшін қолданылмайтындықтан, олар оқыту сапасының өлшемі бола алады. Тестілеуді адам жаттығуларына ұқсас етіп емтиханға теңеуге болады. Мысал ретінде мәтінді оптикалық тану тапсырмасын қарастырыңыз. Танылатын символға сәйкес келетін таңдалған кескіннің нүктелерінің матрицаларын кіріс сигналдары ретінде желі кірісіне беріңіз (4.7-сурет). Танылатын белгінің нүктелерінің матрицасы желі кірісіне берілсін. Шығу кезінде танылған белгіге сәйкес келетін сигналдар қалыптасады. Желілік жаттығулар кейіпкерлерді жазуға арналған әртүрлі нұсқалардың желісін бірнеше рет «ұсынумен» және дайын «жауаптармен» тұрады.

4.7-сурет. Таңбаларды тану есебіндегі нейрондық желі

Ескереміз, бұл желісі-теңдікті есте сақтау барлық ықтимал нұсқалары дағдылы кескінде әр таңба, ал қалыптастырады, шығыс сигналы

$$Y_j = f(X_1, X_2, \ldots X_n)$$

ретінде функциясын жылғы кіріс айнымалылар. Мәтінді танудың бұл тәсілі басқаларға қарағанда жылдамдықта айтарлықтай артықшылыққа ие екенін ескеру керек. Адам да солай: мәтінді оқу үшін, ол қаріптің атын сұрамайды және барлық мүмкін стильдерді жадында іздемейді. Мидағы ассоциациялар бірден пайда болады. Желіні оқыту міндеті үлкен өлшемге ие. Сонымен, әрқайсысы 3 синапс болатын 10 нейроннан тұратын желіні оқыту үшін кем дегенде 40 параметрдің мәндерін таңдау керек (Wi-синаптикалық салмақтың 30 мәні және λ_i активтендіру функциясының 10 параметрі). Егер әрбір параметр 1/100 шешімімен таңдалса, онда оқу мәліметтерінің жиынтығында желілердің жалпы саны 10040 болады.

Әрине, тіпті суперкомпьютерлер де мұндай тапсырманы орындай алмайтыны анық. Бұл проблема кері таралу (back propagation) алгоритмінің көмегімен қанағаттанарлық түрде шешіледі, ол келесідей. Бастапқыда желінің барлық параметрлері өздігінен орнатылады.

1. Оқу мәліметтері желі арқылы жіберіледі және $E = sum(E_i^2)$ қателіктердің жалпы функциясы есептеледі, мұнда $E_i = Y_i - y_i$, Y_i, – шығыс мөлшерінің есептік мәні, y_i - күтілетін шама.

2. Әр параметрге қатысты қателік функциясының туындыларының мәні есептеледі және олардың негізінде нейрондық желі параметрлеріне түзетулердің есебі есептеледі.

3. Желінің параметрлері түзетулердің мөлшерімен реттеледі, содан кейін 2 және 3-қадамдар басынан бастап қате функциясы алдын-ала белгіленген деңгейге жеткенше қайталанады.

Қарапайым қарапайымдылығына қарамастан, бұл алгоритм көп уақытты қажет етеді және оны жеделдету – кезек күттірмейтін мәселе.

Егер жаттығу нәтижесінде қанағаттанарлық нәтиже болмаса, онда желінің құрылымын өзгерту қажет. Мұны қолмен жасауға болады немесе құрылымды алдын ала анықталған жиыннан таңдауға болады (құрылымдар кітапханасы). Мұндай шешімдерді қолдайтын бағдарламалық өнімдер бар. Бірақ ең сәтті тәсіл желінің құрылымы автоматты түрде құрылатынын мойындау керек. Мысал ретінде BioComp Systems Inc үкіметтік емес ұйымдарының нейрондық желісін келтіруге болады (www. biocompsystems.com).

Бұл тәсіл генетикалық алгоритмдерді осы тапсырмаға қолданудан тұрады. Желіні оқыту барысында «күшті» нейрондар және олардың арасындағы байланыс (параметрлердің өзгеруіне сезімтал) және «әлсіз» нейрондар (параметрлері түпкілікті нәтижеге айтарлықтай әсер етпестен өзгеруі мүмкін) анықталады.

Осы деректерді қолдана отырып, сіз нейрондардың популяциясын басқара аласыз. Әлсіз нейрондар мен синапстар өліп кетуі керек, ал құрылымның дамуы үшін, сондай-ақ жалпыға ортақ «жойылып кетудің» алдын алу үшін желі «мутацияға» ұшырайды: ол кездейсоқ немесе басқа жолмен қосылады, мысалы, «күшті» нейрондарды, жаңа нейрондар мен синаптикалық байланыстарды нығайту үшін қосылады [20]. Осылайша, көптеген ұрпақтар арқылы олардың саны ондаған мыңға жетуі мүмкін, желі оңтайлы құрылымға ие болады.

Осыған байланысты сұрақ туындауы мүмкін: егер қосылыстардың толық жиынтығымен максималды өлшемді желіні оқыту әдейі аз уақытты қажет етсе, неге желілік құрылымды оңтайландыруға мұндай уақытты жұмсайсыз? Бұдан басқа, бұрын оқытылған нейрондық желінің өнімділігі айтарлықтай үлкен деп айтылды.

Мұның екі себебі бар. Біріншісі, көбінесе оқытылған желі кейіннен басқа платформада, атап айтқанда, аппараттық деңгейде жүзеге асырылады. Екінші себеп – оңтайландыру көбінесе түпкілікті нәтижеге әсер етпейтін артық шамаларды алып тастауға байланысты енгізу айнымалыларының санының азаюына әкеледі. Бұл факт нейрондық желілерге сапалы жаңа қасиет береді: сіз қандай деректер маңызды және қайсысы маңызды екендігі туралы алаңдамайсыз – жаттығу кезінде артық мөлшер жойылады. Статистикада ұқсас функция қадамдық сызықтық регрессия, дисперсияны талдау және факторлық талдау арқылы орындалады.

Мәселен, біз доллар бағамын болжау үшін нейрондық желіні қолдануға тырысамыз. Бастапқы мәліметтер ретінде біз бірнеше жыл бойына осы курстың мінез-құлқын орната аламыз (айтпақшы, өткен уақытта олардың мінез-құлқына негізделген құндылықтарды болжаудың бұл әдісі «техникалық талдау» деп аталады), экономикалық индикаторлар мен индекстер (Dow Jones, NASDAQ, тұтыну бағалары), сондай-ақ метеобақылау деректеріне дейін кез келген басқа да мәндерді қоя аламыз. Егер осы мәліметтердің барлығынан жақсы болжамды нейрондық желі алынса, оны қолдану өте шаршататын болады және кейде қымбатқа түседі. Сонымен қатар, маңызды параметрлерді анықтау құнды нәтиже болып табылады. Егер доллар бағамына ауа температурасы әсер етсе, осы феноменнің табиғатын зерттеуге болады. Мен мұндай байланысты ауа температурасы мен доллар бағамының графиктерін салыстыра отырып, табуға болады. Алайда, тікелей салыстыру, негізінен, сызықтық тәуелділікті анықтауға мүмкіндік береді, сонымен қатар мұнда жұмысқа адам интеллекті қосылады.

4.4. Нейрондық желілерді қолдану ерекшеліктері

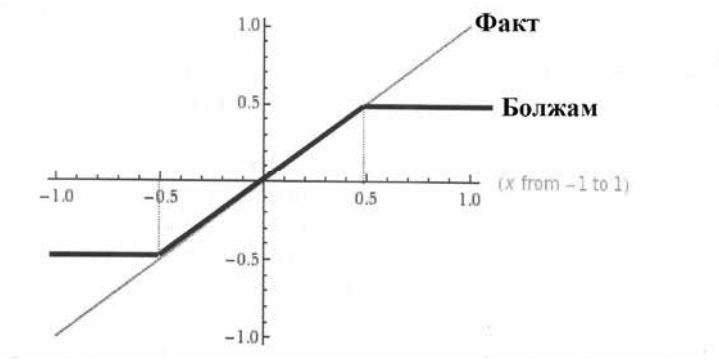
NGO сияқты бағдарламалар пакеттерімен ұсынылатын топология мен нейрожелісінің параметрлерін таңдауды толық автоматтандыру нейрон желісінің көмегімен модельдеу процесін мүлдем басқармауға болатын елесін жасай алады. Алайда, нейрондық желілерді, сонымен қатар статистикалық талдау әдістерін қолдану – бұл құралдың жұмыс принциптерін түсінуді қажет ететін шығармашылық процесс. Нейрожеліні дұрыс құрастырмаудың және оқытудың салдары негізінен жалпылау немесе қайта оқыту болып табылады [20].

Жалпылау дегеніміз желінің шамадан тыс жеңілдеуі деп аталады, онда шағын тәуелділіктер пайда болмайды. Мысалы, ауа-райын болжауға арналған желіні қорытындылау оның белгілі бір маусымда орташа

температураны алуға әкеледі. Бұл проблемадағы жалпылаудың төтенше дәрежесі – орташа жылдық температура. Әрине, мұндай болжамның пайдасы шамалы.

Қайта оқыту дегеніміз тағы бір төтенше жағдай, бұл нейрондық желі қажетсіз күрделі құрылымға ие және нәтижелерін оқу процесінде мұқият реттейді. Нәтижесінде, тренингке қатыспайтын мәліметтер бойынша, желі ең нашар нәтижелерді көрсетеді. *Қайта оқытудың белгісі* – бұл жаттығу кезеңіндегі және тестілеу сатысындағы қателіктер арасындағы айтарлықтай айырмашылық. Желілік топологияны генетикалық түрлендірумен ұштастыра отырып, тым ұзақ оқудың тағы бір қауіпі – тестілеу нәтижелерінің тозуы. Егер бір оқу процедурасында тестілеу мәліметтері оқу процесіне қатыспайтын болса, онда желілік модификациямен тест мәліметтері сәтсіз топологияларды қабылдамау үшін пайдаланылады және сол арқылы соңғы желіге әсер етеді.

Нейрондық желілердің елеулі жетіспеушілігі-экстраполяцияға қабілетсіздігі. Басқаша айтқанда, кіріс айнымалы мәндерінің белгілі бір диапазонында оқытылған нейрондық желі осы ауқымнан тыс жерде болжай алмайды. Іс жүзінде, бұл, әдетте, шығыс айнымалы 4.8-суретте көрсетілгендей, зерттеу аймағынан тыс жерде өзгермейтіндігімен көрінеді. Егер желі -0.5 <X <0.5 диапазонында оқытылса, онда осы диапазоннан тыс шығу функциясының графигі әдетте көлденең орналасады.



4.8-сурет. Оқу саласынан тыс нейрондық желінің әрекеті

Нейрондық желі кез келген, тіпті ең күрделі, кіріс айнымалыларын шығаруға қамқорлық жасайтындығына қарамастан, деректерді алдын-ала өңдеу мағынасыз болмайды. Мәселен, қор нарығында болжамдалған

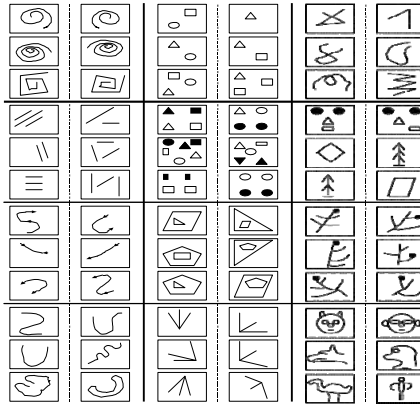
жағдайда, бағамдаудың абсолюттік мәндерінен салыстырмалы мәндерге, яғни бір бақылаудан екіншісіне өсу орынды болуы мүмкін. Бұл жағдайда нейрондық желі үшін қандай деңгейлердегі ауытқуларды жаттығулар үшін қолдану маңызды емес, егер біз тек баға ұсыныстарының бағытын білгіміз келсе (жоғары немесе төмен) [19]. Абсолютті мағынада 1000 немесе 5000 мәндерінің айналасындағы баға белгілеулерінің ауытқуы бойынша оқытылған желі 3000-ға жақын жерде болжау үшін мүлдем пайдасыз.

4.5. Тану есептеріндегі нейрондық желілер

Үлгі, сынып – белгілі бір критерий бойынша объектілердің белгілі бір тобын біріктіретін классификация жүйесіндегі топтастыру. Әлемді бейнелі қабылдау – тірі мидың қасиеттерінің бірі, ол қабылданатын ақпараттың шексіз ағымын түсінуге және сыртқы әлем туралы әр түрлі мәліметтерге бағдар ұстауға мүмкіндік береді. Сыртқы әлемді қабылдай отырып, біз әрдайым ақпаратты жіктейміз, яғни оларды ұқсас, бірақ бірдей емес құбылыстар топтарына бөлеміз. Мысалы, елеулі айырмашылыққа қарамастан, бір топқа әр түрлі жазбалармен жазылған "А" әріптері немесе кез келген октавада және кез келген аспапта алынған бір нотаға сәйкес келетін барлық дыбыстар жатады. Қабылдау тобы туралы ұғымды құру үшін оның өкілдерінің аздаған санымен танысу жеткілікті. Мидың бұл қасиеті сурет сияқты ұғымды қалыптастыруға мүмкіндік береді.

Бейнелер бір жиыннан келген құбылыстардың соңғы санымен танысу оның өкілдерінің кез келген көп санын тануға мүмкіндік береді. Бейнелер әртүрлі бақылау материалында оқитын әр түрлі адамдар көбінесе бірдей және бір-біріне тәуелсіз бір объектілерді жіктейді деген мағынада тән объективті қасиеттерге ие. Бейнелердің дәл осы объективтілігі бүкіл әлемдегі адамдарға бір-бірін түсінуге мүмкіндік береді.

Жалпы бейнелерді тану проблемасы екі бөліктен тұрады: оқыту және тану. Оқыту жекелеген объектілерді көрсету арқылы, олардың қандай да бір бейнеге қатыстылығы көрсетіле отырып жүзеге асырылады. Оқыту нәтижесінде тану жүйесі бір бейненің барлық объектілеріне бірдей реакциялармен және басқа реакциялармен – ерекшеленетін бейнелердің барлық объектілеріне бірдей реакциялармен әрекет ету қабілетін алуы тиіс. Оқыту процесі нысандардың соңғы санын көрсету арқылы ғана аяқталуы керек. Оқыту объектісі ретінде суреттер (4.9-сурет) немесе басқа да көрнекі бейнелер (әріптер, сандар) болуы мүмкін.



4.9-сурет. Оқыту объектілерінің үлгісі

Оқу барысында объектілердің өзі және олардың бейнеге қатыстылығы ғана көрсетіледі. Оқытудан бұрын оқытылған жүйенің әрекетін сипаттайтын жаңа объектілерді тану процесі қажет. Бұл процедураларды автоматтандыру – образдарды тануды оқыту проблемасын құрайды. Адам өзі шешсе немесе ойлап тапса, содан кейін машинамен жіктеу ережесін енгізсе, тану мәселесі ішінара шешіледі, өйткені адам мәселенің (жаттығудың) негізгі және негізгі бөлігін өз мойнына алады.

Объектілердің бастапқы сипаттамасын таңдау бейнелерді тану проблемасының орталық міндеттерінің бірі болып табылады. Бастапқы сипаттаманы (атрибуттар кеңістігін) сәтті таңдаған кезде, тану мәселесі тривиалды болып шығуы мүмкін, ал керісінше, дұрыс таңдалмаған бастапқы сипаттама ақпаратты одан әрі өңдеудің қиынға түсуіне немесе мүлде шешімнің болмауына әкелуі мүмкін.

4.6. Нейрондық желілерді жобалау, оқыту және бейімдеу

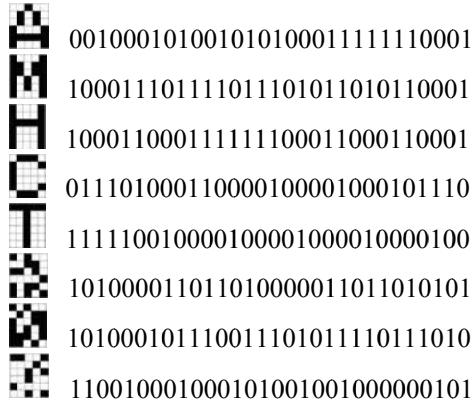
Оқыту - бұл процесс, нәтижесінде жүйе біртіндеп сыртқы әсерлердің белгілі бір жиынтығына қажетті реакциялармен жауап беру қабілетіне ие болады, ал *бейімделу* дегеніміз – бұл сыртқы жағдайлардың үздіксіз өзгеруі жағдайында басқарудың талап етілетін сапасына қол жеткізу мақсатында жүйенің параметрлері мен құрылымын түзету. 2-суретте көрсетілген барлық суреттер оқу тапсырмасын сипаттайды. Осы міндеттердің әрқайсысында дұрыс шешілген есептердің бірнеше мысалдары келтірілген (жаттығулар тізбегі). Егер суреттердің сипатына

да, олардың кескіндеріне де тәуелді болмайтын, бірақ олардың бөліну қабілеттілігін ғана анықтайтын белгілі бір әмбебап қасиетті байқай алсақ, онда үйренудің әдеттегі міндетімен қатар әр объектінің жаттығу жүйесінен белгілі бір кескінге жататындығы туралы ақпаратты қолдана отырып, тағы бір жіктеу мәселесін – мұғалімсіз оқу міндетін туғызуы мүмкін.

Мұндай түрдегі тапсырманы сипаттамалық деңгейде келесі түрде тұжырымдауға болады: жүйеде бір мезгілде немесе дәйекті түрде олардың кескіндерге тиесілігі туралы қандай да бір нұсқауларсыз объектілер ұсынылады. Жүйенің кіріс құрылғысы көптеген объектілерді бейнелейді және оған алдын ала салынған бейнелерді бөлу қасиетін пайдалана отырып, осы объектілердің дербес жіктелуін жүргізеді. Мұндай өзінөзі оқыту процесінен кейін жүйе тек таныс объектілерді (оқыту жүйелілігі бар объектілерді) ғана емес, сонымен қатар бұрын қойылмаған объектілерді тану қабілетін алуы тиіс. Жүйені өздігінен оқып үйрену процесі деп – нәтижесінде бұл жүйе мұғалімді ескертпестен бірдей кескін нысандарының кескіндеріне бірдей реакцияны және әр түрлі кескіндердің кескіндеріне әртүрлі реакцияларды дамыту мүмкіндігін алатын процессті айтады [21].

Бұл жағдайда мұғалімнің рөлі барлық образдарға бірдей болатын және көптеген нысандарды суреттерге бөлу қабілетін анықтайтын кейбір объективті меншіктер жүйесін шақыруда ғана болады. Мұндай объективті сипат – бұл кескіндердің ықшамдылығы. Таңдалған кеңістіктегі нүктелердің салыстырмалы позициясы қазірдің өзінде нүктелер жиынын қалай бөлуге болатындығы туралы ақпаратты қамтиды. Бұл ақпарат кескінді бөлу жүйесін өздігінен үйренуге жеткілікті болатын кескіннің бөліну қасиетін анықтайды.

Суретті машинаға енгізу үшін оны машина тіліне аудару керек, яғни кодтау керек, машинамен жұмыс істей алатын таңбалардың кейбір комбинациясы түрінде ұсыну керек. Жазық фигураларды кодтау әртүрлі жолмен жүзеге асырылады. Суреттерді "табиғи" кодтауға ұмтылған жақсы. Біз суреттерді кейбір өрісте тігінен және көлденең түзулермен бірдей элементтерге – квадратқа бөлеміз. Кескін түсіп кеткен элементтер толығымен қараңғыланады, қалғандары ақ болып қалады. Қара элементтерді бірлік, ақ элементтерді нөл деп белгілеуге келісейік. Біз өрістің барлық элементтеріне дәйекті нөмірлеуді енгіземіз, мысалы, әр жолда солдан оңға және жоғарыдан төмен қарай сызықтар бойымен. Одан кейін осындай өрісте салынған әрбір фигура өрісте элементтер болғандықтан, көптеген сандардан (нөлдерден) тұратын кодпен бір мәнді көрсетіледі.



4.10-сурет. Проекциялау және суретті кодтау мысалдары

Мұндай кодтау (4.10-сурет) "табиғи" деп саналады, өйткені бейнені элементтерге бөлу біздің көру аппаратының жұмысының негізінде жатыр. Торлы қабаттың сезімтал элементтері өзінің жүйке талшықтары арқылы миға сигналдарды береді, олардың қарқындылығы осы элементтің жарықтануына байланысты болады. Осылайша, көздің оптикалық жүйесімен торға жобаланған сурет таяқшалармен және колбочкалармен жекелеген учаскелерге бөлінеді және элементтер бойынша кейбір кодта миға беріледі. Өрістің жекелеген элементтері рецепторлар деп аталады, ал өрістің өзі – рецепторлардың өрісі.

Рецепторлардың өрісінде бейнелеуге болатын барлық жазық фигуралардың жиынтығы көп. Бұл жиынтықтың әрбір нақты фигура осы жиынның нысаны бар. Олардың кез келген осындай объектілері белгілі бір кодқа сәйкес келеді. Сондай-ақ, кез келген кодқа рецепторлардың өрісінде белгілі бір сурет сәйкес келеді. Кодтар мен суреттердің арасындағы бір-біріне сәйкес келу суретті әрқашан оның кодымен шығаруға болатындығын ескере отырып, кодтармен ғана жұмыс істеуге мүмкіндік береді.

ЖНЖ сыйымдылығы – тану үшін ЖНЖ кірістерінде тану үшін берілген суреттер саны. Бірнеше кіріс суреттерін бөлу үшін, мысалы, екі сыныпта бір ғана шығыс жеткілікті. Сонымен қатар, әр логикалық деңгей – «1» және «0» жеке сыныпты білдіреді [20]. Екі шығыс кезінде сіз 4 сыныпты және т.б. кодтай аласыз. Жіктеудің сенімділігін арттыру үшін әр сыныпты шығыс қабатында бір нейронмен немесе одан да жақсырақ бірнеше оқшаулау арқылы артықтықты енгізген жөн, олардың әрқайсысы кескіннің өзінің сенімділік дәрежесімен класқа жататындығын анықтауға

үйретілген, мысалы: жоғары, орта және төмен. Мұндай ЖНЖ бір-біріне кіретін кескіндерді анық емес (бұлыңғыр немесе қиылысатын) жиынтықтарға жіктеуге мүмкіндік береді.

Жасанды нейрондық желіні (ЖНЖ) оқытудың жаңа класына келесі кезеңдер кіреді:

1. Мәселенің тұжырымдамасы жасалып, пәндік аймақты сипаттайтын негізгі параметрлер жиынтығы бөлінеді.

2. Осы класты шешу үшін ең қолайлы нейрондық желінің парадигмасы таңдалады (кіріс деректерінің түрін, шекті функцияны, желі құрылымын және оқыту алгоритмдерін қамтитын модель). Әдетте, қазіргі нейропакеттер, нейроплаттар және нейрокомпьютерлер бір емес, бірнеше негізгі парадигмаларды жүзеге асыруға мүмкіндік береді.

3. Белгілі шығу мәндерімен қауымдасқан кіріс деректер жиынтығы түрінде ұйымдастырылған оқыту мысалдарының кең жиынтығы дайындалады. Оқу үшін кіріс мәндері толық емес және ішінара қарама-қайшы болуы мүмкін.

4. Кезек бойынша кіріс деректері ЖНЖ көрсетіледі, ал алынған шығыс мәні эталонмен салыстырылады. Содан кейін желінің нақты және қалаған шығуының арасындағы қателерді азайту үшін нейронаралық қосылыстардың салмақ коэффициенттерін қайта құру жүргізіледі.

5. Оқу кіріс мәндерінің барлық жиынтығындағы қателік рұқсат етілген деңгейге жеткенге дейін немесе ЖНЖ тұрақты күйге келгенше қайталанады. Нейрондық жүйені оқытудың қарастырылған әдісі «қателіктерді кері қайтару» (error backpropagation) деп аталады және невроматематиканың классикалық алгоритмдерінің қатарына жатады.

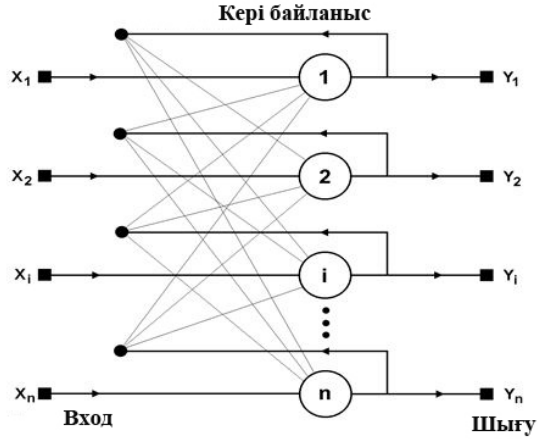
Расталған және дайындалған ЖНЖ-ні нақты енгізілген деректерде қолдануға болады, бұл пайдаланушыны дұрыс шешім қабылдауға итермелеп қана қоймайды, сонымен қатар оның сенімділік дәрежесін де бағалайды.

4.7. Акустикалық сигналдарды тану үшін нейрондық желіні құру

Жасанды нейрондық желілердің әртүрлі конфигурацияларының арасында, оқыту принципі бойынша жіктелуі, қатаң тілмен айтсақ, мұғаліммен оқуға сәйкес келмеуі немесе мұғалімсіз оқуы мүмкін емес адамдар бар. Мұндай желілерде синапстардың салмағы желі жұмыс істей бастағанға дейін бір рет есептеледі және барлық есептеу жаттығулары осы есептеуге келеді. Бір жағынан, априори ақпаратын ұсынуды мұғалімнің көмегі ретінде қарастыруға болады, екінші жағынан – оның кіруіне нақты

деректер келіп түскенше өзінің мінез-құлқын өзгерте алмайды, сондықтан «әлеммен» (мұғаліммен) кері байланыс буыны туралы айтуға тура келмейді. Мұндай логикасы бар желілерден Хопфилд желісі және әдетте ассоциативті жадты ұйымдастыру үшін қолданылатын Хэмминг желісі белгілі.

Хопфилд желісінің блок-схемасы 4.11-суретте көрсетілген. Ол нейрондардың бір қабатынан тұрады, олардың саны бір уақытта желінің кіріс және шығыс саны. Әр нейрон барлық басқа нейрондарға синапс арқылы қосылады, сонымен қатар сигнал кіретін бір кіріс синапсы бар. Шығыс сигналдары, әдетте, аксондарда пайда болады.



4.11-сурет. Хопфилд желісінің құрылымдық схемасы

Бұл желі қауымдасқан жады ретінде шешетін мәселе, әдетте, келесі түрде қалыптасады. Үлгі деп саналатын екілік сигналдардың кейбір жиынтығы белгілі (бейнелер, дыбыс цифрлары, қандай да бір объектілерді немесе процестердің сипаттамаларын сипаттайтын өзге де деректер). Желі оның кірісіне берілген еркін бейнелі емес сигналдан тиісті үлгіні (егер мұндай болса) бөле (ішінара ақпарат бойынша"еске түсіру") немесе кіріс деректері үлгілердің бірде-біріне сәйкес келмейтіндігі туралы «қорытынды бере» алуы тиіс. Жалпы жағдайда кез-келген сигналды X векторымен сипаттауға болады: X = {xi: i=0...n-1}, n – бұл желідегі нейрондардың саны және кіріс және шығыс векторларының өлшемі. Әр x_i элементі +1 немесе -1 болады. K үлгісін сипаттайтын векторды X_k арқылы, ал оның компоненттері тиісінше-x_{ik}, k=0...m-1 белгілейміз, m – үлгілер

саны. Желі ұсынған деректер негізінде қандай да бір үлгіні танғанда (немесе "еске түсіргенде"), оның шығыстарында дәл солай болады, яғни Y = Xk, мұнда y- желінің шығыс мәндерінің векторы: y = { yi: i=0,...n-1}. Әйтпесе, шығыс векторы бір де бір үлгімен сәйкес келмейді.

Желі оған берілген мәліметтерге негізделген кез-келген үлгіні таныса (немесе «есте сақтайды»), оның шығысында оны құрайды, яғни Y = Xk, мұндағы Y - желінің шығыс мәндерінің векторы: Y = {yi: i = 0,. ..n-1}. Әйтпесе, шығу векторы кез-келген үлгіге сәйкес келмейді.

Егер, мысалы, сигналдар кейбір бейнелерді білдірсе, онда графикалық түрде желінің шығуынан деректерді көрсете отырып, үлгілік (табысқа жеткен жағдайда) біріне немесе желінің "еркін импровизациясына" (сәтсіз болған жағдайда) толық сәйкес келетін суретті көруге болады.

Торапты инициалдау сатысында синапстардың салмақтық коэффициенттері былайша белгіленеді [5]:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, & i \neq j \\ 0, & i = j \end{cases} \qquad (4.1)$$

Мұнда і және j – сәйкесінше пресинаптикалық және постсинаптикалық нейрондардың көрсеткіштері; x_{ik}, x_{jk} – k-шы үлгі векторының і-ші және j-ші элементтері.

Желінің жұмыс істеу алгоритмі келесідей (p – итерация нөмірі):

1. Белгісіз сигнал желі кірісіне жеткізіледі. Іс жүзінде оны енгізу аксон мәндерін тікелей орнату арқылы жүзеге асырылады:

$$yi(0) = xi,\ i = 0...n-1 \qquad (4.2)$$

сондықтан, анық синхронды кіріс синапстарының желілік диаграммасында белгілеу тек шартты болып табылады. Y_i-нің оң жағындағы жақшадағы нөл желілік циклде нөлдік итерацияны білдіреді.

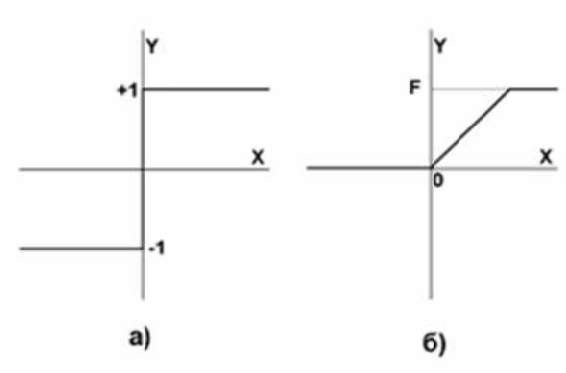
2. Нейрондардың жаңа жағдайы есептеледі

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p),\ j=0...n-1 \qquad (4.3)$$

және аксонның жаңа мәндері

$$y_j(p+1) = f\left[s_j(p+1)\right] \qquad (4.4)$$

мұндағы f – 4.12-суретте көрсетілген секіру түріндегі активация функциясы.



4.12-сурет. Белсендіру функциялары

3. Соңғы итерация кезінде шығыс аксоны мәндерінің өзгергенін тексеріңіз. Егер иә болса, 2-қадамға ауысу, әйтпесе (егер нәтижелер тұрақтанған болса) – соңы. Бұл жағдайда шығыс векторы - бұл кіріс деректерімен жақсы үйлесетін үлгі.

Жоғарыда айтылғандай, кейде желі тануды жүргізе алмайды және шығу кезінде жоқ бейнені береді. Бұл желі мүмкіндіктерінің шектеулі проблемасына байланысты. Хопфилд желісі үшін есте сақталған суреттердің саны шамамен 0,15 • n мәнінен аспауы керек. Сонымен қатар, Егер А және Б екі бейнесі өте ұқсас болса, олар желіде қиылысатын ассоциацияларды шақыруы мүмкін, яғни А векторының желісіне кіруіне ұсыну оның шығуында Б векторының пайда болуына және керісінше әкеледі.

Әлбетте, бір қабаттағы нейрондардың синапстарының барлық салмақтық коэффициенттерін W матрицасына дейін төмендетуге болады, онда әрбір элемент wij jth нейронының ith синаптикалық байланысының мәнін анықтайды. Осылайша, Ұлттық Ассамблеяда болып жатқан процесті матрица түрінде жазуға болады:

$$Y = F(XW) \qquad (4.5)$$

мұндағы X және Y – кіріс және шығыс сигналдарының векторлары, сәйкесінше F(V) – V векторының компоненттеріне элементарлы түрде қолданылатын активация функциясы.

Теориялық тұрғыдан алғанда, әр қабаттағы қабаттардың саны мен нейрондардың саны еркін болуы мүмкін, бірақ іс жүзінде ол НЖ әдетте жүзеге асырылатын компьютердің немесе арнайы микросхеманың ресурстарымен шектеледі. НЖ неғұрлым күрделі болса, соған сәйкес тапсырмалар соғұрлым үлкен болады.

Өзін өзі бақылауға арналған сұрақтар

1. Нейрондық желі түсінігін беріңіз.
2. Көп қабатты нейрондық желілер дегеніміз не?
3. Нейрондық желілерді құру принципі қандай?
4. Нейрондық желіні аппараттық іске асыру қалай жүзеге асырылады?
5. Нейрон құрылымын сипаттаңыз.
6. Жеке нейронның есептеу мүмкіндіктері қандай?
7. Нейрондық желілерді пайдалану ерекшеліктері қандай?
8. Таңбаларды тану міндетінде нейрондық желіні келтіріңіз.
9. Жасанды нейрондық желіні оқыту процесін сипаттаңыз.
10. Қандай сатыларды қамтиды оқу процесі жасанды нейрон желісі.
11. Хопфилд желісінің құрылымдық сұлбасын келтіріңіз.
12. Хопфилд желісі шешетін міндет қалай тұжырымдалады.

5-тарау. САРАПТАМАЛЫҚ ЖҮЙЕЛЕР

5.1. Сараптамалық жүйелер ұғымы

Сараптамалық жүйе (СЖ) – бұл кәсіби қызметінде сарапшы адамды алмастыра алатын бағдарлама. Құрылымдық сараптау жүйесі білім базасынан, шығару машинасынан және пайдаланушы интерфейсінен тұрады.

Білім базасы Прологтағы мағынаға ие, яғни фактілер мен ережелерден тұрады. Шығару машинасы (inference engine) білім базасына (knowledge base) жүгінеді және пайдаланушы интерфейсін пайдалана отырып, қажет болған жағдайда сұрақтар қоя отырып, пайдаланушының сұрауын жауап ретінде өзгертеді. Мұндай құрылым сараптама жүйесін, оған жаңа білім қосу арқылы дамытуға мүмкіндік береді және бұл ретте бағдарламаны қайта жазу талап етілмейді. Бос сараптама жүйесі (білім базасы жоқ) *қабық* (expert system shell) деп аталады және көптеген пән салалары үшін пайдаланылуы мүмкін.

Сараптау жүйесін құру формализациядан тұрады, яғни сарапшы білімін сараптау жүйесінің қабығы үшін талап етілетін нысанға айналдыру. Басқаша айтқанда, білім беруші болып табылатын сарапшы-адам талап етіледі және бұл білімді білім базасына енгізу үшін тұжырымдауға қабілетті. Бұл факт міндетті шешу құралы ретінде сараптамалық жүйені таңдау үшін анықтаушы болып табылады. Сарапшы-адам білім базасының форматын қалай талап ететін түрде өз білімін үнемі жеткізе алмайды. Мұндай жағдайларда сарапшы мен білім базасы арасындағы "аудармашы" болып табылатын білім жөніндегі инженер (knowledge engineer) іске қосылады.

СЖ жұмысын қарапайым мысалда қарастырайық. Біздің білім базасы жануарларды жіктеуге арналсын. Біз жануарды көрдік және оны анықтағымыз келеді. Білім базасы тек екі жануар, зебра және леопард болсын. Бұл жануарларды сипаттау ережелері (нақты білім базасының синтаксисінде емес, орыс тіліне еркін аударғанда) келесідей көрінеді:

"ЕГЕР жануар сүтқоректілер класына жататын болса ЖӘНЕ жануар жыртқыштардың түріне жататын болса ЖӘНЕ жануардың қара дақтары болса, ОНДА жануар – *қабылан (леопард)*".

"ЕГЕР жануар сүтқоректілер класына жататын болса ЖӘНЕ жануар шөптің түріне жататын болса ЖӘНЕ жануардың қара және ақ көлденең жолақтары бар болса, ОНДА жануар – *зебра*".

СЖ гипотезаларды олардың білім базасында орналасуы тәртібімен тексеруді бастайды, бұл жағдайда леопардтан бастап. Бұл гипотезаның шынайылығын анықтау үшін сараптама жүйесі алдымен жануар

сүтқоректілер класына жататынын анықтау керек. Ол үшін білім базасында ереже табу керек:

"ЕГЕР ұрғашының сүт бездері болса, ОНДА жануар сүтқоректілер класына жатады".

Енді малдың сүтқоректілерге жататынын анықтау үшін сүт бездері бар ма анықтау керек. Егер білім базасында тиісті ереже болмаса, пайдаланушы осы сұраққа өзі жауап беруі тиіс. Бұл жағдайда сараптау жүйесі сұрақ қоюы керек: **"Жануарлардың ұрғашысының сүт бездері БАР МА?»**

Егер пайдаланушы оң жауап берсе, онда "жануар сүтқоректілер класына жатады" фактісі белгіленген деп саналады (қарапайымдылық үшін біз еркекті кездестіргенде жағдайды қарастырмаймыз). Осыдан кейін леопардты анықтау ережесі жануардың жыртқыш екенін анықтауды талап етеді. Бұл фактіні тек жеуге бола отырып, дұрыс анықтауға болады, сондықтан адамгершілік ұғымдардан білім базасына "егер жануардың тырнағы болса немесе жануардың қылықтары болса, онда жануардың жыртқыштардың түріне жатады"деген ереже енгізілген.

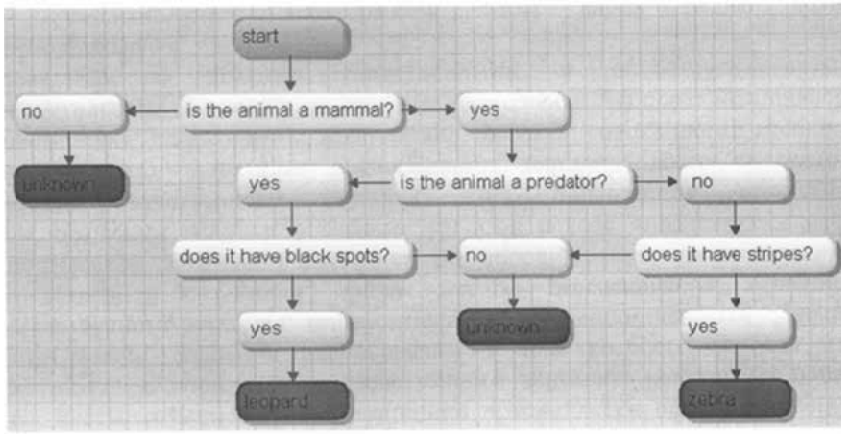
Бұл фактіні сенімді түрде желінген кезде білуге болады, сондықтан адами себептерге байланысты ереже білім базасына енгізілген: **«ЕГЕР жануардың тырнақтары болса, НЕМЕСЕ жануардың тістері болса, ОНДА жануар жыртқыштардың түрлеріне жатады».**

Пайдаланушы ешқандай тістерді немесе тырнақтарды көрмеді, керісінше тұяқтарды байқады. Демек, қабылан гипотезасы қабылданбайды және СЖ келесі зебра гипотезасын тексеруге кіріседі. «Сүтқоректілердің» фактісі бұрыннан белгілі, енді жануардың шөп тұқымдас түрлерге жататындығын анықтау керек. Тиісті ереже келесідей: **«ЕГЕР жануардың мүйіздері болса НЕМЕСЕ жануардың тұяқтары болса, ОНДА бұл жануар шөптесін түрлеріне жатады».**

СЖ алдымен сұрақ қояды: «Жануардың мүйізі БАР МА?»

Пайдаланушы теріс жауап береді, және осы ережедегі жағдайлар НЕМЕСЕ қызметіне байланысты болғандықтан, келесідей сұрақ туындайды: «Жануарлардың тұяғы бар деген РАС ПА?».

Пайдаланушы оң жауап береді, демек, "Жануар шөп тұқымдарының түріне жатады" фактісі белгіленген деп саналады. "Зебра" гипотезасын тексеру үшін соңғы шартты тексеру қалады: "Жануардың қара және ақ көлденең жолақтары БАР МА?". Егер жолақтар болса, онда "зебра" гипотезасы расталады, ал сараптама жүйесінің алдына қойылған міндет орындалды деп саналады. 5.1-сурет LPA компаниясы әзірлеген VISIRULE ортасында іске асырылған қарапайым сараптау жүйесінің үзіндісі бар (http://lpa.co.uk).



5.1-сурет. Ережелерді ұсыну мысалы

СЖ жобалаудың негізгі талабы пайдаланушыға жауап бере алатын сұрақтар қойылуы керек. Басқаша айтқанда, сараптамалық жүйе пайдаланушының білімін сарапшының біліміне түрлендіреді. Жоғарыда келтірілген мысалда жануарларды идентификациялау сәтсіз мәселе сүт бездері туралы мәселе болып табылады (әсіресе, біз кездейсоқ кездескен леопард туралы әңгіме болса). Оның орнына келесі ережені қолдану керек: "**ЕГЕР жануардың жүнднрі болса, ОНДА жануар сүтқоректілер класына жатады**".

Сараптама жүйесінің басқа маңызды қасиеті – алынған қорытынды ақиқаттығының ең жақсы дәлелі болып табылатын қандай да бір тұжырымның қалай алынғанын түсіндіру қабілеті болып табылады. Егер А.Конан-Дойлдың әңгімелерін еске алсақ, онда бізді доктор Ватсон сияқты кез-келген Шерлок Холмстың тұжырымы парадоксалды және ойластырылған болып көрінеді, бірақ бұл тұжырымның қалай жасалғанын түсіндіргеннен кейін, біз бұл туралы өзіңіз ойлаған жоқпыз.

СЖ білімнің әртүрлі салаларында, соның ішінде медицинада (диагностика және емдеу), геологиялық барлауда (кен орындарының перспективаларын анықтау), химияда (органикалық қосылыстардың қасиеттерін болжау) қолданылады. Соңғы уақытта сараптамалық жүйелер тауар таңдауда сатып алушыларға көмек көрсету үшін Интернет-саудада белсенді қолданылады. Ең танымал Yandex-тегі *ГУРУ* сараптамалық жүйесі (http://market.yandex.ru/guru-categories.xml) ең кең тауарлар тізбесі бойынша білікті сатушы-консультантты ауыстыратын заңды тұлға.

5.2. Сараптамалық жүйелерде білім беру модельдері

5.2.1. Деректер мен білім түсінігі

ЭЕМ-дер айналысатын ақпарат процедуралық және декларативті болып бөлінеді. Процедуралық ақпарат есептерді шығару процесінде орындалатын бағдарламаларда, декларативті ақпарат осы бағдарламалар жұмыс істейтін мәліметтерде болады [23].

Процедуралық білім (ПБ) есептерді шығаруда қолданылатын іс-әрекеттер тізбегін сипаттайды (мысалы, компьютерлік бағдарламалар, алгоритмдердің сипаттамасы, кейбір өнімге құрастыру нұсқаулары).

Процедуралық білімді алгоритмдік білімді ұсыну моделінің көмегімен сипаттауға болады.

Декларативті білім (ДБ) - бұл процедуралық емес білімнің бәрі, атап айтқанда: энциклопедиядағы мақалалар, физика, химия, басқа ғылым-дардағы заңдарды тұжырымдау және т.б.

СЖ оқу кезінде дәстүрлі түрде – білім дегеніміз не және олардың әдеттегі деректерден айырмашылығы неде деген сұрақ туындайды.

Деректер – объектілерді, процестер мен құбылыстарды, сондай-ақ олардың қасиеттерін сипаттайтын жеке фактілер.

Білім эмпирикалық жолмен алынған деректерге негізделген. Олар адамның практикалық қызметінің нәтижесінде алынған тәжірибесін жинақтауға бағытталған ойлау әрекетінің нәтижесі болып табылады.

Білім - бұл мамандарға осы саладағы міндеттерді қоюға және шешуге мүмкіндік беретін практикалық іс-әрекет пен кәсіби тәжірибе нәтижесінде алынған пәндік саладағы заңдылықтар (принциптер, байланыстар, заңдар).

Деректерді сақтау үшін ДБ (оларға үлкен көлем және ақпараттың салыстырмалы түрде аздаған меншікті құны тән), ал білімді сақтау үшін – білім қоры (шағын көлем, бірақ тек қымбат ақпараттық массивтер) қолданылады. *Білім базасы* – кез келген зияткерлік жүйенің негізі.

Білімді келесі категорияларға жіктеуге болады.

Беттік (үстіңгі) – пәндік саладағы жекелеген оқиғалар мен фактілер арасындағы көрінетін өзара іс-қимыл туралы білім.

Тереңді – пәндік салада өтетін процестердің құрылымы мен табиғатын көрсететін абстракциялар, аналогиялар, схемалар. Бұл білім құбылыстарды түсіндіреді және объектілердің мінез-құлқын болжау үшін пайдаланылуы мүмкін.

Мысалы.

Беттік: қоңырау батырмасын басу, қоңырау шалуға болады.

Тереңдік: қоңыраудың және сымдардың принциптік электр схемасы.

Қазіргі СЖ негізінен беттік біліммен жұмыс істейді. Бұл қазіргі уақытта білімнің терең құрылымын анықтауға және олармен жұмыс істеуге мүмкіндік беретін әмбебап әдістемелер жоқтығына байланысты.

Білім ерекшеліктері.

1. *Ішкі интерпретациялануы*. Әрбір ақпараттың біртұтас атауы болуы керек, оны СЖ таба алады, сонымен қатар аталған атау сұраныстарға жауап береді. Жадта сақталатын мәліметтер атаулардан айырылған кезде оларды жүйемен сәйкестендіру мүмкіндігі болмады. Бағдарламаны жазған бағдарламашының нұсқауымен жадтан шығарып, деректерді тек бағдарлама ғана шығара алады. Машина сөзінің осы немесе басқа екілік кодының артында жасырынған нәрсе жүйеге белгісіз болды.

Егер, мысалы, ЭЕМ жадына 1-кестеде көрсетілген мекеме қызметкерлері туралы мәліметтерді жазу қажет болса, онда ЭЕМ жадына ішкі интерпретациясыз осы кестенің жолдарына сәйкес келетін төрт машиналық сөздің жиынтығы енгізіледі. Бұл жағдайда осы машиналық сөздерде екілік разрядтардың қандай топтарымен мамандар туралы мәліметтер кодталғаны туралы ақпарат жүйеде жоқ. Олар тек 5.1 кестені қолданатын программистке ғана белгілі. Жүйе " Петров туралы не біледі?" немесе "мамандар арасында сантехник бар ма?" сияқты сұрақтарға жауап бере алмайды.

Кесте 5.1

Тегі	**Туған жылы**	**Мамандығы**	**Еңбек өтілі, жылдар саны**
Попов	1965	Слесарь	5
Сидоров	1946	Токарь	20
Иванов	1925	Токарь	30
Петров	1937	Сантехник	25

ЭЕМ жадына білімге көшкен кезде *ақпараттық бірліктердің кейбір протоқұрылымдары* туралы ақпарат енгізіледі. Бұл мысалда ол тегі, туған жылы, мамандығы және еңбек өтілі туралы ақпарат қай категорияда сақталатынын көрсететін арнайы машина сөзі. Бұл жағдайда жүйелік жадта болатын тегі, туған жылы, мамандығы және еңбек өтілі көрсетілген арнайы сөздіктер көрсетілуі керек. Барлық осы *атрибуттар* кесте жолдарына сәйкес келетін машиналық сөздер үшін атаулар рөлін атқара алады. Олар бойынша қажетті ақпаратты іздеуге болады. Кестенің әрбір жолы протоқұрылымның данасы болады. Қазіргі уақытта ДББЖ деректер базасында сақталатын барлық ақпараттық бірліктердің ішкі түсіндірілуін іске асыруды қамтамасыз етеді.

2. *Құрылымдылық.* Ақпараттық блоктар икемді құрылымға ие болуы керек. Олар үшін «матрешка принципі» орындалуы керек, яғни кейбір ақпараттардың басқаларына екіншісіне рекурсивті ендірілуі керек. Ақпараттық бірліктен оның кейбір құрамдас ақпараттық бірліктерін бөліп көрсетуге болады. Басқаша айтқанда, "бөлік – бүтін", "тегі – түрі" немесе "элемент – класс" типті жекелеген ақпараттық бірліктердің арасында еркін белгілеу мүмкіндігі болуы тиіс.

3.*Байланыстылық.* Ақпараттық бірліктер арасындағы ақпараттық базада әртүрлі түрдегі байланыстарды орнату мүмкіндігі көзделуі тиіс. Ең алдымен, бұл байланыстар ақпараттық бірліктер арасындағы қатынастарды сипаттай алады. Қатынастардың семантикасы декларативті немесе процедуралық сипатта болуы мүмкін.

Мысалы, екі немесе одан да көп ақпараттық бірліктер "бір уақытта" қатынасымен, екі ақпараттық бірлік – "себеп – салдары" қатынасымен немесе "қатар болу" қатынасымен байланысты болуы мүмкін. Келтірілген қатынастар декларативті білімді сипаттайды. Егер екі ақпараттық бірлік арасында "аргумент – функция" қатынасы белгіленсе, онда ол белгілі бір функцияларды есептеумен байланысты процедуралық білімді сипаттайды. Бұдан әрі құрылымдық қатынастарды, функционалдық қатынастарды, каузалды қатынастарды және семантикалық қатынастарды ажырататын боламыз. Біріншісі көмегімен ақпараттық бірліктердің иерархиясы беріледі, екіншісі бір ақпараттық бірліктерді басқалары арқылы табуға (есептеуге) мүмкіндік беретін процедуралық ақпаратты көтереді, үшіншісі себеп – салдарлық байланыстарды береді, төртіншісі барлық қалған қатынастарға сәйкес келеді.

Ақпараттық бірліктер арасында өзге де байланыстар орнатылуы мүмкін, мысалы, жадыдан ақпараттық бірліктерді таңдау тәртібін анықтайтын немесе екі ақпараттық бірліктер бір сипаттамада бір-бірімен үйлеспейтінін көрсететін.

Білімнің аталған үш ерекшелігі білім берудің жалпы моделін енгізуге мүмкіндік береді, оны ақпараттық бірліктері бар иерархиялық желі болып табылатын *семантикалық желі* деп атауға болады. Бұл бірліктер жеке есімдермен жабдықталған. Семантикалық желінің доғалары ақпараттық бірліктер арасындағы Әртүрлі байланыстарға сәйкес келеді. Бұл ретте иерархиялық байланыстар құрылымдаудың қатынастарымен, ал иерархиялық емес байланыстар - өзге түрдегі қатынастармен анықталады.

4. *Семантикалық метрика*. Көптеген ақпараттық блоктарда кейбір жағдайларда ақпараттық блоктардың ситуациялық жақындығын, яғни ақпараттық блоктар арасындағы ассоциативті байланыстың сипатын сипаттайтын қатынасты анықтау пайдалы. Оны ақпараттық бірліктерге қатысты байланыстыру деп атауға болады. Мұндай көзқарас кейбір

ақпараттық жағдайдағы кейбір жағдайларды алып тастауға мүмкіндік береді (мысалы, «сатып алу», «қиылыстағы қозғалысты басқару»). Ақпараттық агрегаттармен жұмыс жасау кезінде сәйкестік коэффициенті бұрыннан бар білімді табуға мүмкіндік береді.

5. *Белсенділік*. ЭЕМ пайда болған сәттен және онда қолданылатын ақпараттық бірліктерді деректер мен командаларға бөлгеннен кейін деректер пассивті, ал командалар белсенді болатын жағдай пайда болды. ЭЕМ-де өтетін барлық үдерістер командалармен бастамаланады,ал деректер қажет болған жағдайда ғана осы командалармен пайдаланылады. АЖ үшін бұл жағдай қолайлы емес. Адам сияқты, АЖ-де қандай да бір әрекеттерді өзектендіруге жүйеде бар білім ықпал етеді. Осылайша, АЖ-де бағдарламаларды орындау ақпараттық базаның ағымдағы жай-күйіне бастама болуы тиіс. Фактілердің немесе оқиғалардың сипаттамаларының негізінде пайда болу, байланыстарды орнату жүйенің белсенділік көзіне айналуы мүмкін.

Ақпараттық бірліктердің аталған бес ерекшелігі деректер білімге айналатын шектерді анықтайды, ал деректер базасы *білім базасына (ББ)* айналады.

Білімдермен жұмысты қамтамасыз ететін құралдардың жиынтығы *білім базасын басқару жүйесін (ББББЖ)* құрады.

Қазіргі уақытта ішкі интерпретациялану, құрылымдау, байланыстылық толық көлемде іске асырылатын, семантикалық шара енгізілген және білім белсенділігі қамтамасыз етілетін білім базасы жоқ.

ез келген пәндік сала өз түсініктері мен олардың арасындағы байланыстардың жиынтығымен, есептерді шешудің ерекше әдістерімен сипатталады. Пәнді білу және ондағы мәселелерді шешу жолдары өте әртүрлі.

Декларациялық және процедуралық білім арасындағы айырмашылықты "НЕНІ БІЛУ" және "ҚАЛАЙ БІЛУ" арасындағы айырмашылық ретінде көрсетуге болады. Процедуралық білім зияткерлік қызметтің бағдарламаға салынған проблемалы ортаны білу, яғни қандай да бір мәндерді қалай қолдануға болатынын білу алғышартқа негізделген. Декларативті білім белгілі бір мәндерді білу ("НЕНІ БІЛУ") осы мәндерді өңдеу үшін қолданылатын процедуралармен терең байланыстары жоқ деген алғышартқа негізделген. Декларативті білімді қолдану кезінде ақыл-ой кез-келген түрдегі фактілерді өңдейтін әмбебап процедуралар жиынтығына және белгілі бір білім саласын сипаттайтын көптеген нақты фактілерге негізделген деп саналады. Процедуралық біліммен салыстырғанда декларативті білімнің басты артықшылығы – декларативті білім белгілі бір білім бөліктерін пайдалану тәсілін көрсетудің қажеті жоқ. Қарапайым мәлімдемелерді бірнеше жолмен қолдануға болады, және бұл

әдістерді алдын-ала бекіту ыңғайсыз болады. Бұл сипат декларативті білімнің икемділігі мен экономикалық тиімділігін қамтамасыз етеді, өйткені дәл сол фактілерді әр түрлі қолдануға мүмкіндік береді.

Декларативті білімде білім тәуелсіз немесе әлсіз тәуелді фактілердің жиынтығы ретінде қарастырылады, бұл мәлімдемелерді жай қосу немесе алып тастау арқылы білім мен оқытуды өзгертуге мүмкіндік береді. Процедуралық білім үшін модификация мәселесі әлдеқайда күрделі, өйткені бұл тұжырымдаманың қалай қолданылғанын қарастыру қажет. Алайда, процедуралар түрінде ұсынуға ыңғайлы және таза декларативті түрде ұсыну өте қиын нысандардың бар екендігі белгілі. Декларативті және процессуалдық білімнің артықшылықтарын пайдалануға ұмтылыс аралас ұсынуды қолданатын формализмнің дамуына әкелді: декларативті қоса берілген процедуралармен (мысалы, кадрлық өкілдіктер немесе тіркелген рәсімдері бар желілер) немесе декларативті үлгілері бар модульдер түріндегі процедуралар. Оның ең дамыған түрінде бұл проблема объектіге бағытталған тәсілмен жүзеге асырылады.

Білімді көрсетудің төрт негізгі моделі бар: өндірістік модель, семантикалық желі, фреймдік және логикалық модель.

Білімді ұсынудың *өндірістік моделі* – интеллектуалды жүйелердегі ең кең таралған заттардың бірі. Модель өндірістік жүйелерге негізделген.

Семантикалық желі – бұл білімді бейнелеудің көрнекі әдісі. Мұндай модельдің негізі кез-келген білімді объектілер (ұғымдар) жиынтығы және олардың арасындағы қатынастар (қатынастар) түрінде ұсыну идеясы болып табылады.

Фреймдік модель де жасанды интеллект жүйелерінде кеңінен қолданылады (мысалы, сараптамалық жүйелерде (СЖ)). *Кадр* – бұл құбылыстың, оқиғаның, жағдайдың, процестің немесе объектінің мәнін ең аз сипаттау.

Білімді ұсынудың *логикалық модельдерінің* негізі формальды жүйе (теория) ұғымы болып табылады.

Формальды теориялардың мысалдары предикативті есептеулер және кез-келген нақты өндіріс жүйесі болып табылады. Логикалық модельдерде, әдетте, бастапқы эволюциялық есептер қолданылады, олар бірқатар эвристикалық стратегиямен толықтырылады. Бұл әдістер дедуктивті типтік жүйелер, яғни. олар үй-жайлардың белгілі бір жүйесінен нәтиже алу үшін үлгіні қолданады.

Предикативті жүйелердің әрі қарай дамуы индуктивті типтегі жүйелер болып табылады, оларда ережелер оқудың көптеген мысалын өңдеуге негізделген жүйе құрылады.

Білімді көрсетудің логикалық модельдерінде білімнің жекелеген бөліктері арасындағы қатынастар қолданылған формальды жүйенің

синтаксистік ережелерімен қамтамасыз етілген өте нашар құралдарды қолдану арқылы көрінеді.

Қарастырылған білімнің әр моделі біліммен жұмыс істеуге бағытталған бағдарламалау тілін құруға негіз бола алады. Мұндай тілдер - бұл кадрлық көріністерге негізделген FRL (Frame Representation Language) және өндірістік ұсыну моделіне негізделген Пролог тілі. Алайда, білім берудің әртүрлі модельдерінің артықшылықтары мен кемшіліктері бар. Сондықтан, 80-жылдардың соңында бірлескен білімді ұсыну тілдерін құру үрдісі байқалды. Көбінесе фреймдік және өндірістік модельдер біріктіріледі.

5.2.2. Алгоритмдік модельдер

Алдын ала қарапайым өмірлік жағдайды қарастырайық: егер оның шешімімен таныс емес адамды мәселені шешуге тарту қажет болса, не істеу керек:

1. Тапсырманы шешу тәсілін (әдісін, тәртібін) таңдайды және оны барлық егжей-тегжейлі зерттейді.
2. Таңдалған әдісті орындаушыға ол үшін түсінікті түрде хабарлайды.
3. Орындаушы есепті әдіске сәйкес қатаң түрде шешеді.

Осы процестің мәнін тереңдете отырып, әрбір кезеңді мұқият қарастырайық.

Бұл процестің бірінші кезеңі әдетте қиындық тудырмайды, өйткені көптеген кездесетін есептер үшін шешім әдісі практикадан белгілі немесе ақылға қонымды мағынамен беріледі немесе әдебиетте сипатталған. Жиі басты қиындық – кейбір талаптарға ең көп жауап беретін бірнеше әдістерден, мысалы: ең аз еңбек сыйымдылығы, ең жоғары тиімділігі және т.б.

Екінші кезең әлдеқайда күрделі. Егер мәселені шешудің әдісі (әдісі) еркін сипатталған болса, онда оны орындаушы дұрыс түсінетіндігіне кепілдік жоқ. Сондықтан әдістің сипаттамасы белгілі ережелерге сәйкес орындалуы керек, атап айтқанда:

- мақсат үшін бастапқы болып табылатын шамаларды бөлу;
- тапсырманы орындау процесін орындаушыға белгілі және кез келген түсіндірусіз орындауға болатын кезеңдерге бөлу;
- кезеңдерді орындау тәртібін көрсету;
- тапсырманы шешу процесінің аяқталу белгісін көрсету;
- барлық жағдайларда, бұл мәселені шешу нәтижесі болып табылады.

Осы ережелерге сәйкес орындалған әдісті сипаттау *есепті шешу алгоритмі* деп аталады. Мұндай сипаттаманы жасау әдетте оңай емес, бірақ оны басшылыққа ала отырып, онда көрсетілген барлық кезеңдерді қажетті тәртіппен механикалық орындай отырып, орындаушы міндетті әрдайым дұрыс шеше алады.

Алгоритм – бұл белгілі бір ережелер бойынша жазылған, бастапқы деректердің барлық мәндерінде (кейбір көптеген мәндерден) оны түсіну мен механикалық орындаудың бір мағынасын қамтамасыз ететін, сонымен бірге алгоритмнің көрсетілген әрекеттерінің дәйектілігіне сәйкес қадамдардың соңғы санына қойылған міндетті шешуге қол жеткізіледі.

Алгоритмнің мысалы тағам пісіру рецепті болуы мүмкін.

Қарапайым алгоритмді қарастырайық – шай қайнату алгоритмі:

1. Бастапқы шамаларды дайындау – шай, су, шәйнек, стакан, қасық.
2. Шайнекке су құю.
3. Суды қайнату және оттан алу.
4. Шәйнекке шай салу.
5. Суды қайнатуға дейін апару (бірақ қайнатпау), оттан алу.
6. Шай дайын. Процесс тоқтатылады.

5.2.3. Білім берудің логикалық модельдері

Логикалық тәсілдің негізгі идеясы қолданбалы міндеттерді шешу және ЭЕМ-нің пайдаланушымен өзара іс-қимылын ұйымдастыру үшін қажетті барлық білім жүйесін фактілер жиынтығы ретінде қарастыру болып табылады.

Фактілер кейбір логикадағы формула ретінде ұсынылады (бірінші немесе жоғары ретті, көп мәнді, модальды, тақ немесе басқа). Білім жүйесі осындай формулалардың жиынтығымен көрінеді. ЭЕМ - де ұсынылған кезде ол білім базасын құрады. Формулалар бөлінбейді және білім базасын модификациялау кезінде тек қосылады және жойылады.

Логикалық әдістер білім базасында анық ұсынылған жаңа фактілерді шығарудың дамыған аппаратын қамтамасыз етеді. Білім манипуляциясының негізгі белгісі – шығару операциясы. Бұл сараптау жүйелері мен міндеттерді шешетін кезде логикалық әдістерді қарқынды пайдалануды анықтайды. Кез келген зияткерлік жүйелер үшін маңызды осы аппаратты басқа қолдану – білім базасының логикалық тұтастығын, яғни оның қарама-қайшы еместігін және белгілі бір алдын ала белгіленген ережелерге (тұтастықты шектеуге) сәйкестігін бақылау мүмкіндігі.

Білім берудің логикалық әдістері нақты анықталған семантикаға ие фактілерді жазу үшін қарапайым және айқын нотацияны қамтамасыз етеді

(ең болмағанда, бірінші тәртіптің дәстүрлі логикасына негізделген әдістер үшін). Әрбір факт болашақта қалай пайдаланылатынына қарамастан білім базасында тек бір рет ұсынылады. Логикалық әдістерді қолдана отырып әзірленген білім базасы, әдетте, түсіну үшін өте қарапайым [23].

Білімді ұсынуда пәндік аймақ пен тапсырма аксиома жиынтығы ретінде сипатталған кезде і-ретті предикаттардың классикалық есептеулеріне негізделген формальды логикалық модельдер ерекшеленеді.

Логикалық модельдердің негізін формальды теорияның төрт ұғымы жатыр:

$$S = <B, F, A, R> \quad (5.1)$$

мұнда *B* – S теориясының негізгі символдарының (алфавит) жинағы;

F – теорияның формулалары деп аталатын S теориясының өрнектерінің жиыны (өрнектер деп S теориясының базалық символдарының соңғы кезектілігі түсініледі);

A – S теориясының аксиомалары деп аталатын формулалар, яғни көптеген априорлы шынайы формулалар;

R – қорытынды ережелері деп аталатын формулалар арасында $\{r_1,r_2,...,r_n\}$ қатынастардың соңғы жиыны.

Әдетте B – формулаларынан синтаксистік тұрғыдан дұрыс өрнектерді құруға мүмкіндік беретін тиімді процедура (көптеген синтаксистік ережелер) бар.

Әрбір r_j үшін j-дің оң саны бар, мысалы j формулаларынан тұратын әрбір жиын үшін және f формула үшін f формуласы бар r_j-ге қатысты j формулаларының деректері тиімді шешіледі ме деген сұрақ туындайды. Егер r_j қатынасы болса, онда f r_j ережесіне сәйкес берілген j формулаларының тікелей салдары деп аталады.

Шығару ережелері осы теорияның шеңберінде шынайы деп саналатын формулалардың көптігін кеңейтуге мүмкіндік береді.

Егер формуланың кез-келген формуланы S-де бар-жоғын білуге мүмкіндік беретін бірыңғай тиімді процедура болса, формалды теория шешілмейтін деп аталады. Егер формула жоқ болса, S формулярлы жүйе деп аталады, егер A формуласы болмаса, онда A және ¬A туындылары S-де алынады.

Білімді ұсыну үшін қолданылатын ең көп таралған ресми жүйе - бұл бірінші ретті предикативті есептеу. Болжалды есептеу алфавиті келесі таңбалар жиынтығынан тұрады:

- тыныс таңбалары «{»,«(», «,», «)», «.», «;», «}»;
- пропозициялық байламдар { ¬, ⊃, ∧, ∨};
- квантор-таңбалары { ∀ , ∃};
- айнымалылар таңбалары x_k, k= 1,2,...;

– *n*-жергілікті функционалды әріптер: f_k^n, $k \geq 1$, $n \geq 0$ (f_k^0 тұрақты әріптер деп аталады);

– *n*-жергілікті предикаттар әріптері (символов): p_k^n, $k \geq 1$, $n \geq 1$.

Болашақта x_k-тің орнына қарапайымдылық үшін, *u*, *v*, *x*, *y*, *z*,...; f_k^0 – *a, b, c, d*,...; орнына f_k^n ($n \neq 0$) – f, *g*, *h*,...; және орнына – p_k^n – *P*, *Q*, *R*, *S*, *T*, *V*, *W*…қолданатын боламыз

Алфавиттің символдарынан әр түрлі өрнектерді салуға болады. Терминдер, қарапайым формулалар (атомдар) және дұрыс салынған формулалар (немесе жай формулалар) ажыратылады. Айнымалы немесе тұрақты әр әріптің әр белгісі – терм. Егер $t_1,...,t_n$ ($n \geq 1$) – терм болса, онда $f_k^n(t_1,...,t_n)$ терм болып табылады.

Егер p_k^n – предикатты әріп, ал $t_1,...,t_n$ – термдер, онда $p_k^n(t_1,...,t_n)$ – қарапайым формула (атом).

Атом – қарапайым бөлінбейтін элементі.

Атом – дұрыс құрылған формула. Егер А және В – дұрыс құрылған формула болса, онда $\neg A$, $A \vee B$, $A \wedge B$, $A \supset B$ бар дұрыс құрылған формулалар. Егер А – дұрыс құрылған формула, х – А-дағы айнымалы болса, онда $(\forall x)A$ және $(\exists x)A$ – дұрыс құрылған формулалар. Білдіру болып табылады дұрыс салынған формуласы, егер ол алынды сақтай отырып жоғарыда келтірілген ережелері.

Формуланың мазмұнын беру үшін ол қарастырылып отырған тақырып саласына қатысты тұжырымдама ретінде түсіндіріледі.

Интерпретация деп бос емес D жиынынан және әрбір предикат p_k^n алдындағы әріпті салыстырмалы қандай да бір сәйкестіктен тұратын кез келген жүйені түсінеді, f_k^n әр функционалды әрпіне – кейбір n – $D^n \to D$ жергілікті функциясын және f_k^0 әрпінің әр әрпіне – D элементінің кейбір элементтеріне. Берілген интерпретация кезінде айнымалыларды осы интерпретацияның D аймағынан барлық мәндерді "жүгіруші" деп ойлайды, ал кез келген элементарлық формулада "ақиқат" (А) немесе "жалған" (Ж) мәні жазылады.

Мән $p_k^n(t_1,...,t_n)$ элементар формуласына келесі ережеге сәйкес тағайындалады: егер предикаттық әріптің шарттары D-дан осы түсінікпен анықталған қатынасты қанағаттандыратын элементтерге сәйкес келсе, онда элементар формуланың мәні дұрыс болады, әйтпесе ол жалған болады.

Элементтік емес формуланың мәні оның құраушы формулаларының мәндеріне сүйене отырып, рекурсивті түрде есептеледі. Таңдалған

интерпретацияға байланысты формулалардың мәні шын немесе жалған болуы мүмкін екені анық.

Предикаттарды есептеу шеңберінде шешілетін негізгі міндет – кейбір интерпретация аймағында берілген формуланың шынайылығын немесе жалған болуын анықтау. Бұл ретте ерекше рөл жалпы мағыналы формулаларға, яғни кез келген интерпретация кезінде шынайы формулаларға және орындалмайтын формулаларға, яғни кез келген интерпретация кезінде жалған формулаларға беріледі. Келесі дедукция теоремасы дұрыс: $B_1,...,B_n$ және A формулалары берілген болсын.

A формуласы – $B_1,...,B_n$ қисынды нәтижесі, егер $B_1 \wedge ... \wedge B_n \supset A$ формуласы дұрыс болса, яғни $\models (B_1 \wedge ... \wedge B_n) \supset A$.

Есіңізде болсын, А формуласы логикалық түрде $B_1,...,B_n$ формуласынан келеді, сонда ғана кез келген интерпретация J қанағаттандыратын $B_1 \wedge ... \wedge B_n$, қанағаттандырады, сондай-ақ Aны қанағаттандырады,

$B_1,...,B_n$ формулалары *жіберулер* деп аталады, ал А – логикалық тізбектің тұжырымы және $B_1,...,B_n \models A$ деп белгіленеді.

Теореманы дәлелдеудің міндеті $B_1,...,B_n$ формулалар жиынтығынан белгілі бір А формуласының логикалық салдарын нақтылау, $B_1 \wedge ... \wedge B_n \supset A$ формуласының дұрыстығын немесе формуласының мүмкін еместігін дәлелдеуге тең ... $B_1 \wedge ... \wedge B_n \wedge \neg A$.

Бірінші ретті предикаттарды есептеу үшін кез-келген формулалардың жарамдылығын анықтайтын жалпы әдіс жоқ екендігі белгілі, яғни бірінші ретті предикаттық есеп шешілмейді. Алайда, егер предикаттарды есептеуге арналған кейбір формула жарамды болса, онда оның дұрыстығын тексеру процедурасы бар, яғни предикативті есептеуді жартылай шешілмейтін деп атауға болады [24].

Кейбір фактіні предикаттарды есептеу формуласы түрінде жазу мысалын келтіреміз:

БЕРУ (МИХАИЛ, ВЛАДИМИРГЕ, КІТАПТЫ)

$(\exists x)$ (ЭЛЕМЕНТ (х, ОҚИҒА - БЕРУ) $\wedge$ ИСТОЧНИК (х, МИХАИЛ) $\wedge$ АДРЕСАТ (х, ВЛАДИМИР) $\wedge$ ОБЪЕКТ (х, КІТАП).

Мұнда сипатталған екі тәсілі жазбалар бір факт: " МИХАИЛ ВЛАДИМИРГЕ КІТАПТЫ БЕРДІ ".

Негізгі артықшылығы пайдалану предикаттар есептеу үлгісі ретінде ұсынылған білім – бұл болуы біркелкі формальды процедура дәлелдемелер теоремалар. Алайда, жоғары дәрежесі бірізділікті білдіреді және негізгі кемшілігі бұл тәсілді – күрделілігі пайдалану дәлелдеу кезінде эвристик ерекшелігін көрсететін нақты проблемалық орта. Көрсетілген кемшілік әсіресе маңызды құру кезінде сараптамалық

жүйелер, есептеу қуаты, негізінен, анықталады білім, мінездемелік ерекшелігін проблемалық орта.Ресми жүйелердің басқа кемшіліктеріне олардың монотондылығы, қолданылатын элементтерді құрылымдау құралдарының болмауы және қарама-қайшылықтардың жол берілмеуі жатады.

Тұсаукесер моделі ретінде пайдаланған кезде формальды жүйелердің кемшіліктерін жоюға деген ұмтылыс семиотикалық жүйелердің пайда болуына әкелді. Семиотикалық жүйені формальды түрде сегізбен беріледі:

$$S = <B, F, A, R, Q (B), Q (F), Q (A), Q (R)>. \quad (5.2)$$

Мұнда алғашқы төрт компонент формальды жүйені анықтаудағы сияқты, ал қалған компоненттер – зияткерлік жүйенің білім базасында осы проблемалық ортадағы субъектілердің құрылымы мен қызметі туралы жинақталған тәжірибенің әсерінен алғашқы төрт компонентті өзгерту ережелері.

Логикалық әдістердің негізгі жетіспеушілігі – бұл білім қорында фактілерді ұйымдастырудың нақты принциптерінің жоқтығы. Мұндай қағидаларды оқшауламай және дәйекті түрде қолданбай, үлкен модель талдауға және өңдеуге қиын болатын тәуелсіз фактілердің көрінбейтін конгломератына айналады.

5.3. Үлгілермен басқарылатын өнім модельдері мен модульдері

Дәстүрлі бағдарламалауда командалар қатаң белгіленген ретпен орнатылады. Әдепкі бойынша, і-ші команда орындалғаннан кейін (і+1) і-ші команда тармақталу командасы болмаса, і-ші команда орындалады. Дәстүрлі бағдарламалаудағы тармақтаудың барлық орындары анық көрсетіледі. Мұндай бағдарламалау тәсілі өңдеу тізбегі өңделетін деректерге аз тәуелді болған жағдайда ыңғайлы, яғни тармақ нормадан емес, ерекшелік болған кезде.

Әрбір модель басқаратын модуль (МБМ) деректердің бір немесе бірнеше құрылымын зерттеу және модификациялау тетіктерінен тұрады. ҮБМ диапазоны қарапайым өнімдік ережеден үлгі бойынша туындайтын еркін күрделілік дәрежесіне дейін кең шектерде ауытқуы мүмкін. Әрбір ҮБМ кезекті қадамда жұмысын талдайды жұмыс жад, барлап болуы құрылымдар, олар салыстырылады, оның үлгісі. ҮБМ негізінде құрылған жүйелер *үлгілермен басқарылатын шығару жүйелері* деп аталады. Бұл жүйелердегі басқару функцияларын интерпретатор жүзеге асырады.

Білім беру тұрғысынан ҮБМ қолданатын тәсіл келесі ерекшеліктерді сипаттауға болады:

– Білім базасында сақталатын тұрақты білімді және жұмыс жадында сақталатын уақытша білімді бөлу;

– СЖ модульдердің құрылымдық дербестігі, жүйенің өзгеруіне және жетілуіне ықпал етеді, бұл СЖ үшін өте маңызды, олардың білімін үнемі өзгертеді. Сонымен қатар, модульдердің тәуелсіздігі әртүрлі авторлар жазған бағдарламалардың интеграциясын жеңілдетеді;

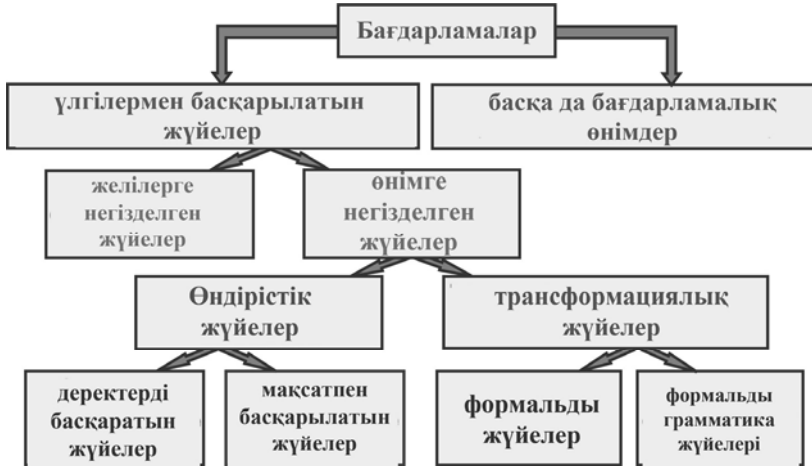
– басқару сұлбасын проблемалық сала туралы білім алатын модульдерден бөлу, бұл басқарудың түрлі сұлбаларын қолдануға мүмкіндік береді.

Үлгілермен басқарылатын жүйелер әртүрлі орындалуы бар және модульдерге салынатын шектеулерге сәйкес жіктеледі (5.2-сурет). Егер мұндай жүйелер желінің жоғарғы жағында орналасқан модульдерден тұратын болса, онда олар *желілерде негізделген жүйелер* деп аталады.

Үлгілермен басқарылатын жүйелердің көпшілігі келесі шектеуді қанағаттандырады: әрбір модульдегі жұмыс жадының барлық зерттеулері біріктірілген және деректерді түрлендіру бойынша барлық іс-қимылдарға алдын алады. Осылайша, модуль екі бөлікке бөлінеді: деректерді зерттейтін алдын-ала шарттар және деректерді түрлендіретін әрекет. Мұндай бөлінуі бар модульдер *ережелер* деп аталады, ал мұндай ережелерді қолданатын жүйелер *ережелерге негізделген жүйелер* деп аталады.

Ережеге негізделген жүйелер ережелердің түрлері бойынша өнімдік және трансформациялық болып бөлінеді. Өнім жүйелері салыстыру және жоспарлау (басқару) – интерпретаторда тіркелген жүйенің айқын функциялары ережелерден құрылған. Трансформациялық жүйелер өнімнен айырмашылығы қағидаларды салыстыру және басқару бойынша айқын функциялары болмауы мүмкін. Трансформациялық жүйелердің мысалдары формальды және формальды грамматика жүйелері болып табылады. Өнім жүйелері деректермен (ережелердің алдын ала қызметтерімен) басқарылатын және мақсаттармен (ережелердің әрекеттерімен) басқарылатын өнім жүйелеріне бөлінуі мүмкін. *Дәстүрлі түрде өнім жүйелері* деп тек деректермен жіберілетін қорытындыны пайдаланатын жүйелер ғана түсінеді.

Әдетте алдын-ала шарттар (антецедент) жұмыс жадының деректері туралы тұжырымдардың логикалық комбинациясы түрінде беріледі, ал әрекет (консервент) жадының модификациясы бойынша кейбір операция болып табылады. Әрекеттің күрделілігі қарапайым тағайындау операциясынан бастап күрделілік дәрежесі функциясына дейін айтарлықтай шектерде ауытқиды.



5.2-сурет. Үлгілермен басқарылатын жүйелердің жіктелуі

Өнім жүйелерінде мақсаттармен басқарылатын алдын-ала шарттар мен әрекеттер – бұл деректер туралы тұжырымдар. Мұнда қорытынды дәлелденуі тиіс тұжырымдардан кері бағытта жүзеге асырылады. Үлгілердің декларативті және процедуралық түрде берілуі мүмкін екенін атап өту қажет [24].

Сонымен, модульдер мен өнімді ережелер үлгілерімен басқарылатын білім беру келесі қасиеттерге ие:

– білімді ұйымдастырудың модульділігі;
– тәуелсіз білім фрагменттерін білдіретін ережелердің тәуелсіздігі;
– білім модификациясының жеңілдігі мен табиғаттылығы;
– бұл әр түрлі басқару стратегияларын қолдануға мүмкіндік береді.;
– есептерді автоматты түрде шешу мақсатында басқару механизмдерінің бірқатар қосымшаларын құру мүмкіндігі.

Жалпы түрде *өнім* деп мынадай түрдегі өрнек түсініледі:

$$(i);\ Q;\ P;\ A \Rightarrow B;\ N. \qquad (5.3)$$

Мұнда і - бұл өнім көптеген өнімдерден бөлінетін өнімнің атауы. Атау ретінде осы өнімнің мәнін көрсететін кейбір лексема (мысалы, "кітап сатып алу" немесе "құлыптың коды") немесе жүйенің жадында сақталатын олардың жиынындағы өнімнің реттік нөмірі болуы мүмкін.

Q элементі өнімнің қолданылу аясын сипаттайды. Мұндай салалар адамның танымдық құрылымдарында оңай бөлінеді. Біздің біліміміз "сөрелерге қарай ыдырайды". Бір "сөреде" тамақты қалай дайындау керек,

екіншісінде – жұмысқа қалай жету керек және т.б. туралы білім сақталады. Сонымен қатар, бұл модельдерді білім беру үшін пайдаланған кезде де, АЖ білім базасында салаларға де бөлу орынды.

Өнімнің негізгі элементі оның ядросы болып табылады: $A \Rightarrow B$. Өнім ядросының интерпретациясы әртүрлі болуы мүмкін және сол және оң жағында секвенция белгісінің $\Rightarrow$ тұрғанына байланысты. Өнім ядросының әдеттегі оқуы келесідей: егер А, онда В, ядроның неғұрлым күрделі конструкциялары оң бөлігінде баламалы таңдауға жол береді, мысалы, егер А, онда В1, әйтпесе В2. Секвенция әдеттегі логикалық мағынада шынайы А-дан қисынды жол жүру белгісі ретінде түсіндірілуі мүмкін (егер шынайы өрнек болмаса, онда ештеңе айтуға болмайды). Өнім ядросының басқа да интерпретациялары мүмкін, мысалы А әрекет жасауға қажетті кейбір шарттарды сипаттайды.

Р элементі өнім ядросының қолданылу шарты бар. Әдетте Р логикалық өрнек (әдетте, предикат). Р "ақиқат" мәнін қабылдаған кезде, өнімнің ядросы белсендіріледі. Егер Р жалған болса, онда өнімнің ядросы пайдаланылуы мүмкін емес. Мысалы, егер өнім «АҚШАНЫҢ БОЛУЫ; ЕГЕР Х ЗАТТЫ САТЫП АЛҒЫҢЫЗ КЕЛСЕ, КАССАҒА ОНЫҢ ҚҰНЫН ТӨЛЕ ЖӘНЕ САТУШЫҒА ЧЕКТІ БЕР» өнім өзегінің қолданылуы шарты жалған, яғни ақша жоқ, сондықтан өнімнің өзегін пайдалану мүмкін емес.

N элементі өнімнің кейінгі шарттын сипаттайды. Олар өнімнің ядросы сатылған жағдайда ғана өзектендіріледі. Өнімнің кейінгі шарттары В енгізілгеннен кейін орындалуы керек әрекеттер мен процедураларды сипаттайды. Мысалы, дүкенде кейбір заттарды сатып алғаннан кейін осы дүкенде бар тауарлар тізімдемесінде қажет, мұндай түрдегі заттардың санын бірлікке азайту қажет. N өнім өзегін сатқаннан кейін бірден орындалуы мүмкін емес.

Егер белгілі бір өнім жиынтығы жүйенің жадында сақталса, онда олар өнім жүйесін құрайды. Өнімдер жүйесінде өнімді басқарудың арнайы процедуралары анықталуы керек, олардың көмегімен өнімдер жаңартылады және жаңартылған саннан өнімді орындау үшін таңдау болады.

Бірқатар АЖ желілік және өндірістік білім модельдерінің тіркесімін пайдаланады. Мұндай модельдерде декларативті білім модельдің желілік компонентінде сипатталады, ал процедуралық білім өндірісте сипатталады. Бұл жағдайда олар өндірістік жүйенің семантикалық желідегі жұмысы туралы айтады.

Кадрлармен қатар, өнімдер АЖ-дағы білімді ұсынудың ең танымал құралы болып табылады. Өнімдер, бір жағынан, логикалық модельдерге жақын, бұл оларға тиімді шығару процедураларын ұйымдастыруға мүмкіндік береді, ал екінші жағынан, классикалық логикалық модельдерге

қарағанда білімді айқын көрсетеді. Оларда логикалық калькуляцияларға қатаң шектеулер жоқ, бұл өндіріс элементтерінің түсіндірмелерін өзгертуге мүмкіндік береді.

Бұл тәсілдің басты кемшілігі – дәстүрлі бағдарламалау әдістерімен салыстырғанда оның тиімділігі төмен. Әр түрлі авторлар өндіріс жүйелерін басқаша жіктейді. Кейбіреулері оларды декларативті өкілдікті, басқалары процедуралық немесе декларативті процедуралық байланыстырады. Айырмашылықтар «өндірістік ереже» ұғымы қаншалықты кең түсіндірілетіндігімен түсіндіріледі. Ең қарапайым өндірістік ережеде (яғни, бекітілген процедуралар жоқ ереже) процедураның элементі бар, өйткені ереже қандай да бір әрекеттерді орындау үшін пайдаланылады деп болжанады.

Процедуралық ұсыныстың декларативтіден ерекшелейтін нәрсе, өйткені декларативті білімде оны пайдалану туралы ақпарат болмайды. Неғұрлым күрделі өндірістік ережелерде «процедуралық» дәрежесі одан да жоғары. Дегенмен, өндірістік ережелерде және тіпті үлгілермен басқарылатын модульдерде де декларативтілік элементі бар, өйткені ережелер мен модульдерді қолдану тәсілі оларда көрсетілмеген.

Осылайша, өндірістік ережелер декларативті және процедуралық өкілдіктердің, сондай-ақ кадрлар мен иерархиялық желілер түріндегі өкілдіктердің қасиеттерін біріктіреді.

Өндірістік модель көбінесе өнеркәсіптік сараптамалық жүйелерде (СЖ) қолданылады. Ол өзінің көрнекілігімен, жоғары модульділігімен, толықтырулар мен өзгертулердің қарапайымдылығымен және қарапайым ықпал ету механизмімен дамытушыларды тартады.

5.4. Сараптамалық жүйемен жұмыс

Сараптама жүйесімен жұмыс істеу мәселесіне тоқталайық. Ең алдымен, осы процеске қатысатын тұлғалар шеңберін (рөлдерді) белгілеу маңызды. Сараптамалық жүйемен жұмыс істеу процесі деп оның іс-әрекетін жасау үшін қажетті, сондай-ақ онда сақталатын сараптамалық білім алу үшін тікелей өзара іс-қимыл түсініледі. Осылайша, СЖ-мен өзара іс-қимыл:

- СЖ міндеттерін шешетін пәндік саладағы сарапшы;
- білім жөніндегі инженер - СЖ әзірлеу жөніндегі маман;
- бағдарламашы-аспаптық құралдарды әзірлеу жөніндегі маман (АЖ);
- соңғы пайдаланушы-шешім алу үшін жүйені пайдаланатын адам немесе тұлғалар тобы.

Сарапшы білімді анықтайды (пәндік саланы сипаттайтын деректер мен ережелер), СЖ-де шығарылған білімнің толықтығы мен дұрыстығын қамтамасыз етеді.

Білім жөніндегі инженер сарапшыға СЖ жұмысы үшін қажетті білімді анықтауға және құрылымдауға көмектеседі, осы пән саласына неғұрлым қолайлы аспаптық құралды (АҚ) таңдауды жүзеге асырады және осы АҚ-да білім беру тәсілін айқындайды, сарапшы енгізетін ережелерде пайдаланылатын стандартты функцияларды (осы пән саласына тән) бөліп, бағдарламалайды.

Сонымен қатар, инженер – білім бойынша инженердің болмауы (яғни оны программистпен ауыстыру) немесе СЖ құру процесі сәтсіздікке әкеп соқтырады немесе оны едәуір ұзартады.

Программист СЖ барлық негізгі компоненттері бар АҚ әзірлейді, АҚ-дың ол пайдаланылатын ортамен түйіндесуін жүзеге асырады.

СЖ екі режимде жұмыс істейді: білім алу және міндеттерді шешу (кеңес беру немесе СЖ пайдалану).

Білім алу тәртібінде СЖ-мен қарым-қатынасты сарапшы білім жөніндегі инженер арқылы жүзеге асырады. Сарапшы заттық саланы деректер мен ережелер жиынтығы түрінде сипаттайды. Деректер сараптама саласында бар объектілерді, олардың сипаттамалары мен мәнін анықтайды. Ереже қарастырылып отырған пән саласына тән деректерді манипуляциялау тәсілдерін анықтайды. Сарапшы білім алу компонентін пайдалана отырып, СЖ-не пәндік саладан тапсырмаларды өз бетінше (сарапшысыз) шешуге мүмкіндік беретін білім жүйесін толтырады. Білім алу тәртібінде түсініктеме компоненті маңызды рөл атқарады. Оның арқасында сарапшы тестілеу кезеңінде СЖ сәтсіз жұмыс істеу себептерін оқшаулайды, бұл сарапшыға ескі білімдерді мақсатты түрде өзгертуге немесе жаңа білімді енгізуге мүмкіндік береді. Әдетте түсініктеме компоненті мынаны хабарлайды: ереже пайдаланушының ақпаратын қалай қолданады; деректер немесе ережелер неге пайдаланылды немесе пайдаланылды; қандай қорытындылар мен т.б. жасалды.

Бағдарламаны әзірлеудің дәстүрлі тәсілінде білім алу режимі бағдарламашы жүзеге асыратын алгоритмдеу, бағдарламалау және күйге келтіру кезеңдеріне сәйкес келеді. Осылайша, дәстүрлі тәсілден айырмашылығы, бағдарламаларды әзірлеуді бағдарламалаушы емес, бағдарламалауды білмейтін маман (СЖ қолдана отырып) жүзеге асырады.

Консультация режимінде СЖ-мен байланыс нәтижеге және (немесе) шешім алу әдісіне қызығушылық білдіретін соңғы пайдаланушы жүзеге асырылады. СЖ мақсатына байланысты қолданушы осы тақырып бойынша маман бола алмауы мүмкін, бұл жағдайда ол СЖ-ке кеңес алу үшін жүгінеді, өзіне қалай жауап алу керектігін білмейді немесе маман

бола алмайды, бұл жағдайда ол нәтижені алу процесін тездету үшін СЖ-ге жүгінеді немесе жоспарлы жұмысты СЖ-ға тағайындаңыз. "Пайдаланушы" термині көп мәнді болып табылады, өйткені соңғы пайдаланушыдан басқа СЖ-ны сарапшы, білім жөніндегі инженер және программист қолдана алады. Сондықтан, СЖ кім үшін жасалғанын атап өткім келеді, "соңғы пайдаланушы" терминін пайдаланады.

Кеңес беру режимінде пайдаланушының міндеті туралы деректер келесі әрекеттерді орындайтын диалогтық компонентпен өңделеді:

1) қатысушылардың (Пайдаланушының және СЖ) рөлін бөледі және олардың бірлескен міндеттерді шешу процесінде өзара іс-қимылын ұйымдастырады;

2) пайдаланушы үшін үйреншікті тілде берілген тапсырма туралы деректерді жүйенің ішкі тіліне түрлендіреді;

3) ішкі тілде берілген жүйе хабарламаларын пайдаланушы үшін үйреншікті тілде (әдетте бұл шектеулі ЖТ немесе графика тілі).

Өңдеуден кейін деректер БН-не түседі. БН кіріс деректері, пәндік сала туралы жалпы деректер және ББ ережелері негізінде шешуші (интерпретатор) есептің шешімін қалыптастырады.

СЖ-нің дәстүрлі бағдарламаларынан айырмашылығы, тапсырмаларды шешу режимінде ол белгіленген операциялар тізбегін орындап қана қоймай, оны алдын-ала қалыптастырады. Егер СЖ-нің жауабы пайдаланушыға түсініксіз болса, ол жауаптың қалай алынғанын түсіндіруді талап етуі мүмкін. дәстүрлі бағдарламаларынан айырмашылығы міндеттерді шешу тәртібінде операциялардың ұйғарылған бірізділігін орындайды ғана емес, сондай-ақ оны алдын ала қалыптастырады. Егер СЖ жауабы пайдаланушыға түсінікті болмаса, онда ол жауап қалай алынғанын түсіндіруді талап ете алады.

Өзін өзі бақылауға арналған сұрақтар

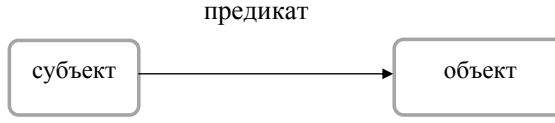
1. Сараптамалық жүйе түсінігін беріңіз
2. Сараптау жүйелерінің негізгі белгілері қандай?
3. ЭС таралуына ықпал ететін себептерді көрсетіңіз?
4. ЭС мақсаты қандай?
5. Кез келген СЖ қандай критерийлер бойынша сипаттауға болады?
6. Іздеу кеңістігін және шешілетін есептің белсенді агенттерінің санын қалай анықтауға болады? Бұл СЖ сипаттамасына қалай әсер етеді?

7. Келесі аспектілерді қолдана отырып шешілетін есептер классы бойынша СЖ сипаттаңыз: кеңейту, толық анықтау, түрлендіру есептері.
8. СЖ ғылым мен техниканың қандай салаларында кеңінен таралған? Мысалдар келтіріңіз.
9. Медицинада СЖ көмегімен шешілетін есептер шеңберін сипаттаңыз?
10. СЖ әзірлеу күрделілігі немен анықталады?
11. Сараптамалық жүйені әзірлеудің қандай кезеңдерін білесіз?
12. СЖ қандай негізгі компоненттерден тұрады? Олардың әрқайсысының мақсатын көрсетіңіз.
13. СЖ жұмысында білім базасы мен жұмыс жадының рөлін түсіндіріңіз.
14. "Кеңес беру" режиміндегі сараптама жүйесінің жұмыс алгоритмін түсіндіріңіз.
15. Сараптамалық жүйемен жұмыс істеуді сипаттаңыз.

6-тарау. СЕМАНТИКАЛЫҚ ЖЕЛІЛЕР

6.1. Анықтама. Тарихи анықтама

Семантикалық желі - бұл доғалармен байланысқан түйіндер түрінде, графикті қолдана отырып білімді көрсетуге арналған құрылым. Түйіндер тұжырымдамаларға сәйкес келеді, ал доғалар олардың арасындағы қатынастарға сәйкес келеді. Қарапайым жағдайда семантикалық желінің элементі келесі түрдегі үштік болып табылады:



Бір тұжырымдама желінің құрылымын анықтайтын бірнеше үштікте болуы мүмкін. Семантикалық қатынастардың басты қасиеті – *арлық* (арность), яғни аргументтер саны. Жоғарыда екілік қарым-қатынас үлгісі, яғни 2 арлығымен қарым-қатынас келтірілген. Егер желінің барлық қатынастары бір типті болса, онда желі *біртекті* деп аталады. Біртекті желінің үлгісі-биологиялық түрлердің жіктелуі. Әртүрлі желіде қатынас түрлерінің саны біреуден артық.

Білімді білдіру үшін семантикалық желілерді қолданудың негізгі мақсаты – тілден тәуелсіздікті қамтамасыз ету, сонымен қатар табиғи тілдерге тән түсініксіздік пен түсініксіздікті жою. *Табиғи тіл* – тірі организм ретінде, сөйлесуге қатысушылардың түсінуін қамтамасыз ету үшін өзінің негізгі мақсатын жақсарту жолында дамиды. Алайда, жергілікті тілшілердің табиғи жалқаулығына байланысты бұл оңтайландыру көбінесе құрылыстарды барынша жеңілдетуге дейін барады, нәтижесінде бір сөйлемнің мағынасын тек контекстік ортадан анықтауға болады.

Хрестоматиялық мысал тіпті екімағыналы емес, "үш мағыналы" деген тіркес «Ол оны гүлдермен далада қарсы алды» деген сөз тіркесін білдіреді. Гүлдер қайда түсініксіз: қызда, ұлда немесе алаңда. Ағылшын тілін қоса алғанда, барлық тілдер әртүрлі. Мәселен, "Tell me, what has four wheels and flies" деген сұрақ бізді қандай ұшу аппараты төрт дөңгелектің бар екенін еске түсіреді. Семантикалық желі математикалық дәл нысандағы білімді қамтуы тиіс.

Тарихи анықтама

Математика әлемдегі құбылыстардың көпшілігін логикалық тұжырымдар түрінде суреттеуге мүмкіндік береді. Семантикалық желілер

математикалық формулаларды бейнелеу әрекеті ретінде пайда болды. Қазіргі семантикалық желілердің ата-бабаларын 1909 жылы Чарльз Сандерс Пирс ұсынған экзистенциалды график ретінде қарастыруға болады. Олар арнайы диаграмма түрінде логикалық мәліметтерді ұсыну үшін органикалық химияда қолданылды. Пирс бұл әдісті «болашақтың қисыны» деп атады.

1913 және 1922 жж. желілерді зерттеудегі маңызды бастама неміс психологы Отто Зельцтің жұмысы болды. Оларда ол графтар мен мағыналық қатынастарды ұғымдар мен ассоциациялардың құрылымын ұйымдастыруда, сондай-ақ қасиеттерді мұрагерлік ету әдістерін зерттеуде қолданды. Зельцтің ғылыми зерттеулері шахмат тактикасын зерттеуге үлкен әсер етті, бұл өз кезегінде Саймон мен Ньюелл сияқты теоретиктерге әсер етті. Зерттеушілер Дж.Андерсон (1973), Д.Норман (1975) және басқалары бұл еңбектерді адамның жадын және зияткерлік қасиеттерін модельдеу үшін пайдаланды.

Тіл біліміне келетін болсақ, Тениер графикалық сипаттамаларды жасауға қатысқан алғашқы ғалым болды. Ол тәуелділік грамматикасы үшін графикалық белгіні қолданды. Тенер Еуропадағы лингвистиканың дамуына айтарлықтай әсер етті. Компьютерлік семантикалық желілерді Ричард Риченс егжей-тегжейлі 1956 жылы машиналық аударма бойынша Кембридж тілін оқыту орталығы жобасының аясында жасады. Машинаны аудару процесі 2 бөлікке бөлінеді: бастапқы мәтінді презентацияның аралық түріне аудару, содан кейін бұл аралық форма қажетті тілге аударылады. Мұндай аралық форма дәл мағыналық желілер болды. Алғашқы осындай жүйені Masterman жасаған, мысалы, ХАЛЫҚ, БІЛІМ, ЖАСАУ, БОЛУ сияқты 100 примитивті ұғымдар болды. Осы тұжырымдамаларды қолдана отырып, ол сипаттамаларды гипертиптен кіші түрге ауыстыру механизмі бар 15000 бірлік сөздікті сипаттады. Кейбір машиналық аударма жүйелері 56 түрлі қатынастардың жиынтығы болған Цеккато корреляциялық желілеріне негізделді, олардың кейбіреулері кейстер, кіші түрлер, мүшелер, бірліктер және тұтас қатынастар болып табылады. Ол тұжырымдамалар мен қатынастардан тұратын желілерді мәтіндік талдау және бір мәнді шешу бағдарламасының іс-қимылдарын басқару үшін пайдаланды. Бұл зерттеулер Роберт Симмонс (1966), Уилкс (1972) және басқа ғалымдармен жалғасырылды [28].

Жасанды интеллект жүйелерінде семантикалық желілер әртүрлі сұрақтарға жауап беру, оқу, есте сақтау және ойлау процестерін зерттеу үшін қолданылады. 70-жылдардың аяғында желілер кең тарала бастады. 80-ші жылдары желілер, жақтау құрылымдары мен сызудың жазбаша нысандары арасындағы шекаралар біртіндеп жойылды. Экспрессивті қуат

енді желілерді немесе жазудың жазбаша түрлерін таңдаудың пайдасына шешуші дәлел бола алмайды, өйткені бір жазба нысанын пайдаланып жазылған идеялар екіншісіне оңай ауыса алады. Және керісінше, оқылымдылық, тиімділік, артсыздық және теориялық талғампаздық сияқты маңызды факторлар маңызды рөл атқара бастады, сонымен қатар компьютерге енгізу, редакциялау және басып шығару оңай.

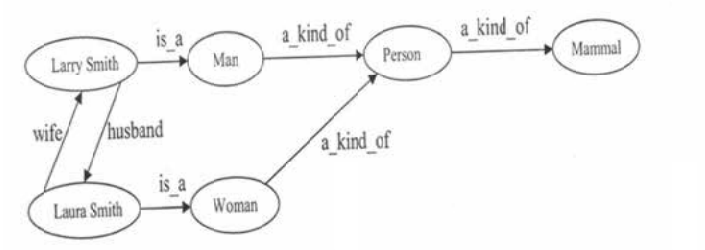
Дрю МакДермоттің семантикалық желілерді қолдану арқылы білім беру проблемаларын түсінуіне 30 жыл бойы айтылған «Жасанды интеллект табиғи ақымақтықты жеңеді» атты эссесі әсер етті.

Мәтіндік ақпаратты талдау және іздеу проблемаларында семантикалық желілік аппараттарды қолданудағы отандық дамудың ең дамыған түрі «Гарант Парк Интернет» компаниясының RCO (Russian Context Optimizer) брендімен бағдарламалық өнімдер жиынтығы болып табылады.

Соңында Семантикалық өрмек тұжырымдамасын атап өткен жөн, оны 6.7 және 6.8 бөлімдерде толығырақ қарастырамыз. *Семантикалық өрмек* – бұл Дүниежүзілік тордың одан әрі дамуы, оның ресурстарында орналастырылған құжаттарда осы құжаттардағы ақпараттың мағынасын алуға мүмкіндік беретін семантикалық түзету бар. Мұндай семантикалық құжаттар негізінен RDF форматында жасалады (Resource Description Framework) – XML тілінің кеңейтімі, тақырыптық-предикаттық-үштікті ұсыну арқылы толықтырылған.

6.2. Семантикалық желілердің түрлері

Барлық қатынастар екілік болатын семантикалық желі реляциялық графты құрайды (6.1-сурет).

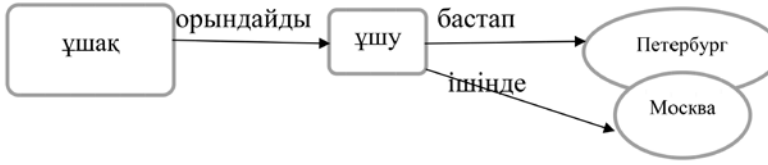


6.1-сурет. Реляциялық граф мысалы

«Larry Smith» және «Man» деген ұғымдар графикте бірдей эллипстермен белгіленгеніне қарамастан, олардың семантикалық сапасы әр түрлі.

«Larry Smith» – бұл ер адамдар тобының үлгісі, ал «Man» – бұл көптеген ер адамдар. Өз кезегінде «Man» және «Woman» – бұл неғұрлым қуатты адамдар жиынтығы («Person»). Сонымен, реляциялық мәліметтер базасында болғандай, бір-біріне, біреуіне және көпке қатынасы бар.

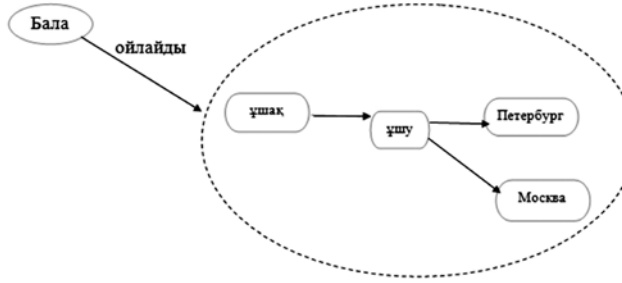
Егер арития екіден өзгеше болса, онда графикалық түрде семантикалық желіні бейнелеу қиынға соғады. Бірақ, деректер базасындағыдай, сіз бәрін қалыпқа келтіріп, екілік қатынастарға келтіре аласыз. Мысалы, «қозғалтқыш» сияқты біртұтас қатынасты «мотор» -> «күй» -> «жұмыс» деп қысқартуға болады. 3 типтегі арлықпен байланыс «Ұшақ Петербургтен Мәскеуге ұшады» үш ұғыммен жұмыс істейді: тақырыбы – ұшақ, ал объектілері – Мәскеу мен Петербург. Предикат – ұшуды орындау («ұшу»). «Ұшу» қосымша ұғымын енгізе отырып, бұл қатынасты семантикалық желінің келесі үзіндісімен ұсынады (6.2-сурет):



6.2-сурет. Тұжырымдамалық граф анықтамасына

Көптеген жағдайларда, қарым-қатынас тұжырымдамасы ретінде, ол қатысатын жай объект немесе субъект емес, бірақ басқа қатынас немесе семантикалық желінің тұтас бөлігі. Мысалы, бала аспанда ұшақты көріп, бұл ұшақ Петербургтен Мәскеуге ұшып жатыр деп ойлайды.

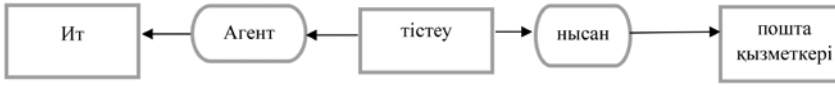
Мұндай желілерді *пропозициялық желілер* деп атайды, ал мұндай желінің графын *тұжырымдамалық граф* деп атайды (6.3-сурет).



6.3-сурет. Концептуалды граф

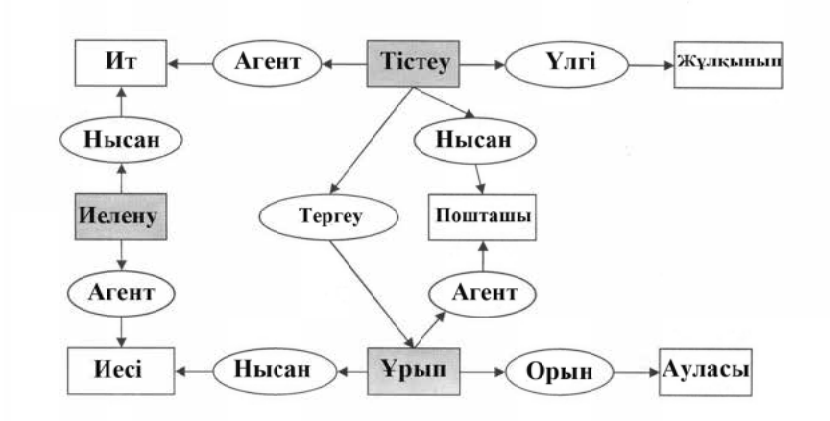
Ұсыныс желілеріндегі ұялы байланыстар еркін түрде үлкен болуы мүмкін. Мысалы, баланың ата-анасы егер ол ұшақ Санкт-Петербургтен Мәскеуге ұшып бара жатыр деп ойласа, ол қателескен деп сенуі мүмкін, ал олардың атасы мен әжесі өз кезегінде ата-анасы немеренің дедуктивті қабілеттерін және т.б. бағаламайды деп сенуі мүмкін.

Оқиғаларды ұсыну үшін ортасында етістігі бар граф немесе Растье графы қолайлы. Мысалы: ит пошташыны тістейді (6.4-сурет).



6.4-сурет. Растье графы

Негіз ұғым емес, әрекет немесе оқиға, бұл жағдайда тістеу жатыр. Зат немесе тістейтін агент – ит, ал зат – пошта қызметкері. Етістіктің ортасында орналасқан граф сізге графты салмай-ақ жасауға мүмкіндік береді. Біз осы оқиға туралы білім базасын жай ғана кеңейте аламыз (6.5-сурет):



6.5-сурет. «Пошташы» семантикалық желісі

Пошташыны ит тістегендіктен, пошта қызметкері ит иесінің ауласында ұрып-соққан.

6.3. Семантикалық желілердегі қатынастар түрлері

Семантикалық желілердегі қатынастардың ең көп таралған түрі – элементтердің, жиынтықтардың және объектілердің бөліктері арасындағы қатынасты сипаттайтын иерархиялық тип. Оларға мыналар жатады:

1) **ISA классификациясының байланысы** (ағылшын тілінен аударғанда “is a” болып табылады). Көпшілік (сынып) оның үлгілерін жіктейді (мысалы, «Сократ – 69л адам»). Бұл қатынасты кейде “member of” деп те атайды. Орыс тілінде оны «бар» (жекеше) немесе «мәні» (көпше) деп атауға болады. Кері байланыс “example of” немесе «мысал» болып табылады.

2) **AKO жиыны мен ішкі жиыны арасындағы қатынас** («a kind of»), мысалы, «магистрлер – бұл студенттер жиынтығы». ISA қатынасынан айырмашылығы – жіктеу дегеніміз – «бір-бірімен» байланысы, Пошташы Ит Агент Кусать Объект Иесі Объект Иесі Объект Ұру Агент Тергеу Орын Аула Манера, ал ішкі жиын – «көптің көбіне». Орыс тілінде – «ішкі жиын».

3) **Бүтіннің және бөліктің қатынасы.** Меронимияның арақатынасы – бүтіннің бөлікке қатынасы («(“**has part**”»). *Мероним* – басқа объектінің құрамына кіретін объект. Холонимияның байланысы – бұл бөліктің бүтінге қатынасы (“is a part”). Қолденеңнің холонимі. Дене – бұл қолдың меронимі.

Қатынастардың иерархиялық түрлерін қолдана отырып, қандай объектілер кластар екенін және кластың даналары болып табылатындығын нақты ажырату қажет. Сонымен қатар, барлық пәндер бойынша бірдей тұжырымдаманың класс немесе данасы болуы міндетті емес. Сонымен, «адам» әрдайым «студенттер тобы» немесе «еңбек ұжымы» сияқты білім базаларында сынып болады, бірақ бұл биологиядағы білім базасындағы сүтқоректілер класының мысалы бола алады.

Семантикалық графиктің шыңдары объектілерді ғана емес, сонымен қатар қасиеттерді немесе қасиеттердің мәнін де көрсете алады. Графикте қасиеттерді көрсету оның көрінуін арттырады, бірақ өте бітелген болуы мүмкін.

Иерархиялық қатынастардан басқа, көбінесе семантикалық желілерде қатынастардың келесі түрлері қолданылады (екінші жақшада шыңдардың түрлері көрсетілген):

1) функционалдық қатынастар («жасайды», «әсер етеді», ...) (нысан – нысан);

2) сандық («көп», «аз», «тең», ...) (нысан – нысан немесе нысан – мүлік);

3) кеңістік («алыс», «жақын», «жақ», «жоғары», «астында», «жоғарыда», ...) (нысан – нысан);

4) уақытша («ертерек», «кейінірек», «бір уақытта», ...) (нысан – нысан);

5) атрибуттық («меншікке ие болу», «маңызды болуы мүмкін», ...) (нысан – бұл меншік немесе мүлік – құндылық);

6) логикалық («және», «немесе», «емес») (нысан – нысан немесе мүлік – мүлік);

7) лингвистикалық.

Қатынас түрлерінің саны өте үлкен болуы мүмкін. Бұл ретте негізгі мәселе білім базасына сұрау салудағы осы қатынастарды сәйкестендіру мүмкіндігі болып табылады. Осыған байланысты, шыңдардың санын көбейту арқылы қатынастардың түрлерін (және шыңдарын) азайтуға болады.

Мысалы, «семантикалық желі» – «арналған» – «деректер_ақпараттық» қатынасының орнына сіз «семантикалық желі» – «бар» – «мақсатты»; «мақсатты» – «бар» – «ұсыну»; «ұсыну» – «неге» – «деректер» қолдана аласыз. Орталықта етістігі бар желілер (Растье желілері) 6.1-кестеде көрсетілген төмендегі қосылыстармен жұмыс істейді.

Кесте 6.1. Растье графтарындағы қатынастар түрлері

Аты	**Түрі**	**Анықтау**	**Жеңілдетілген аты**
1	2	3	4
(ACC)	accusative	әсер ету объектісі	PATient
(ASS	assumptive	көзқарас	PERspectiv
(ATT)	attributive	қасиеттері сипаттамасы	CHARacteristic
(BEN)	benefactive	пайда алушы рөлінде болатын мән	BENeficiary
(CLASS)	classitive	сынып данасы	CLASsitiv
(COMP)	comparative	салыстыру арқылы біріктірілген элементтер	COMParison

Кестенің 6.1 жалғасы

(DAT)	dative	алушы	RECeiver
(ERG)	ergative	Эргативті, процессорлық немесе әрекеттік агент	AGEnt
(FIN)	final	Нәтиже немесе күтілетін мақсат	GOAL
(INST)	instrumental	пайдаланылған қаражат	MEAns
(LOC S)	spatial locative	кеңістіктегі ереже (ұстаным)	SPAce
(LOC T)	temporal locative	ереже (позиция) уақыт	TIME
(MAL)	malefactive	зардап шеккен тарап іс-әрекеті нәтижесінде	MALeficiary
(PART)	partitive	бүтін бөлігі	PARTitiv
(RES)	resultative	нәтижесі,әсері, салдары	EFFect (және CAUse)

6.4. Онтология және қатынастардың мұрагерлік ережелері

Білімді ұсыну – өте күрделі және шығармашылық процесс. Бұл жерде негізгі мәселе білім базасын құру әрдайым дерлік "нөлден" басталады; бұл ретте адам балалық шақтан бастап айналатын бастауыш деңгейдегі білім жоқ. Осындай тұрмыстық білім базаларын құру 25 жылдан астам уақыт бойы Сусогр компаниясымен жүргізіледі. (www.cyc.com) алайда, оның негізін қалаушы Даглас Ленаттың (Douglas Lenat) мойындауы бойынша, машина әлі де оның ата-анасы өз балаларынан үлкен екенін және адамдар қайтыс болған кезде газеттер жазудан бас тартатынын анық түрде хабарлауы қажет. Нақты пәндік саладағы білімді барлық даналарға ортақ және әр адамға жеке бөлуге болады. Егер объектілер класына арналған Білімді формCheat бір рет орындалса, содан кейін оны басқа авторлар жеке даналарды сипаттау кезінде пайдалана алады.

Осылайша, кез-келген пәндік саладағы білімді формалдау объектілер арасындағы негізгі қатынастарды және объектілердің жалпы қасиеттерін сипаттайтын жалпы деңгейдегі білімге негізделуі тиіс. Мұндай білім *онтология* деп аталады. Мысалы, "адам" класын сипаттау үшін онтология

"қол", "басы" және "қасиеттері бар", "туған күні бар" және т.б. объектілерімен "has part" қатынасын қамтуы мүмкін [28].

Мұндай білімді төменгі деңгейдегі объектілерге қолдану мұрагерлік ережелерінің көмегімен жүзеге асырылады:

Егер X AKO Y және Y AKO Z онда X AKO Z – егер X Y жиыны болса, ал сол, өз кезегінде, одан да көп z жиыны болса, онда x класы Z жиыны болып табылады.

Егер X ISA Y және Y AKO Z болса, онда X ISA Z – егер X Y класының данасы болса, және бұл өз кезегінде Z жиынының бөлігі болса, X - Z класының данасы.

Егер X has_part Y және Y has_part Z онда X has_part Z – егер X құрамында Y болса, ал ол өз кезегінде z құрамдас бөлігі болса, онда X құрамында Z болады.

Егер X ISA Y және Y has_part Z болса, онда X has_part Z – егер X Y класының данасы болса, ал ол өз кезегінде z құрамдас бөлігі болса, онда X данасының өз құрамында Z болады.

Егер X AKO Y және Y has_part Z болса, онда X has_part Z – егер X Y ішкі жиыны болса, ал ол өз кезегінде z құрамдас бөлігі болса, онда X класы Z өз құрамында болады.

Егер X ISA Y және Y has_a Z онда X has_a Z – егер X Y класының данасы болса, ал ол өз кезегінде Z қасиетіне ие болса, онда x данасының Z қасиеті болады.

Егер X AKO Y және Y has_a Z онда X has_a Z-егер X Y ішкі жиыны болса, ал ол өз кезегінде Z қасиеті болса, онда X класы Z қасиеті болады.

Мұрагерлік ережелерін қолдану әрбір дананың барлық қасиеттерін сипаттауға барлық қажетті қасиеттер сипатталған класқа жататынын ғана көрсетуге мүмкіндік береді.

Өлең шығармасын семантикалық желіге аудару мысалын қарастырайық және тек негізгі мағынаны көрсетеді. Өзгеріс үшін біз ағылшын тіліндегі нұсқаны аламыз (кімнен кім жазғанын түсінбейміз).

"The Cicada and the Ant"

Jean de La Fontaine (1988)

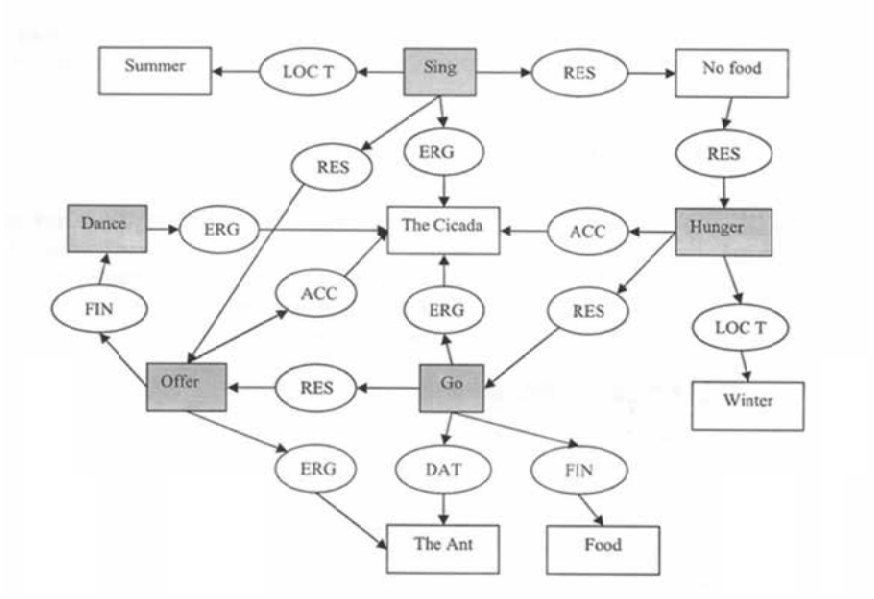
Cicada, having sung her song
All summer long,
Found herself without a crumb
" When winter winds did come.
Not a scrap was there to find
Of fly or earthworm, any kind.
hot?" Hungry she ran off to cry
To neighbor Ant, and specify:
She said, "I'll pay you everything
Before fall, my word as animal,
Interest and principal.
Well, no hasty lender is the Ant;
It's her finest virtue by a lot.
"And what did you do when it was
She then asked this mendicant.
"To all comers, night and day,

Asking for a loan of grist,
" A seed or two so she'd subsist
Just until the coming spring.

I sang. I hope you don't mind.
"You sang? Why, my joy is unconfined.
Now dance the winter away."

6.6-суреттегі кестеде шығарманың көркемдік жағы көрінбейді, тек процесстерді, оқиғаларды және олардың арасындағы себеп-салдарлық байланыстарды бейнелейді. Сонымен, ән айту процесі жүріп жатқанын көреміз («Sing»). Ән айтудың агенті (ергативті) – цикада, ал уақыт бойынша позиция – жаз. Жазда цикада жазудың нәтижесі қыста азық-түліктің болмауы (“No food”) және аштық ((“Hunger”) (уақыт – “Winter”). Аштықтың нысаны (“accusative”) – цикада.

Өкінішке орай, графикте жаз бен қыстың диапазоны бір-біріне қарама-қарсы орналасқан, сондықтан жаздан кейін қыстың басталу фактісі көрінбейді.

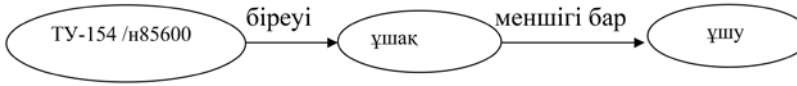


6.6-сурет. «The Cicada and the Ant » өлеңіне арналған Растье графы

Аштыққа байланысты, инелік құмыраға тамақ («Food») алу мақсатымен («Fin») келді (“Go”). Құмырсқа иненіктің биін («Dance») оған келгеніне және жазда ән салғанына байланысты ұсынады.

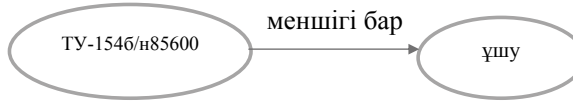
6.5. Семантикалық желілерді құру мәселелері

6.1-бөлімінде көрсетілгендей, семантикалық желі білімді математикалық тұрғыдан нақты түрде сақтауы керек. Осыған байланысты, оның құрылысы дәлдік пен пәндік аймақты және оған қатысты барлық ұғымдарды жақсы түсінуді талап етеді. Білімді ұсыну проблемалары Дрю МакДермоттың жоғарыда аталған жұмысында сипатталған [10]. Бір қарағанда, 6.3-бөлімде келтірілген мысалдарға ұқсас графиктерді салу оңай және табиғи түрде жүзеге асырылуы мүмкін. Алайда, бұл әрдайым бірдей емес. Келесі құрылысты қарастырайық:



Басқаша айтқанда, 85600 құйрығы бар Ту-154 ұшақтар санатына жатады, ал ұшақтар ұшуға қабілетті. Сыныптың жеке өкілі сыныптың сипаттамаларын иемденетіндіктен, 85600 құйрығымен Ту-154 ұша алады деп қорытынды жасаймыз.

Басқаша айтқанда, 85600 құйрығы бар Ту-154 ұшақтар санатына жатады, ал ұшақтар ұшуға қабілетті. Сыныптың жеке өкілі сыныптың ерекшеліктерін иемденетіндіктен, 85600 құйрығы бар Ту-154 ұша алады да, ол меншікке ие болады және біз дұрыс боламыз деп қорытынды жасаймыз.



Басқа мәселе ұғымдарға атаулар берумен байланысты. Жоғарыда келтірілген мысалда ұшақпен нақты ұшақ борт нөмірімен сәйкестендірілді. Әйтпесе, ұшақтың дәл қандай дана екенін түсініксіз болар еді.

«Ұшып кету» қасиетін анықтау үшін дананы дәл анықтау өте маңызды емес, оны ақаулықтарды жою туралы айту мүмкін емес.

Жоғарыда көрсетілген 85600 идентификациясы да толық емес. Біріншіден, бұл сандар комбинациясы кез-келген нәрсені, мысалы, телефон нөмірін көрсете алады. Екіншіден, ұшақ азаматтық авиацияға әскери авиациядан кіре алады, онда сәйкестендіру мүлде басқа, және біз ұшақтың өткен өмірі туралы, мысалы, алдыңғы жөндеу туралы ештеңе таба алмаймыз, және бұл мүлдем қолайсыз.

Түсініктерді сәйкестендіру мәселесі нақты өмірде де орын алады, бірақ мұнда біз "ұстара Оккаманы" қолданамыз: қажеттіліктен тыс мәндерді көбейту емес. Егер біз шағын компанияда сөйлессек, атаулар жеткілікті. Студенттік топта әр тегті анықтауға болады. Бұл ретте бір атқа ие, ал аттары – әкесінің аттарына да ие болады. Егер объектілердің көп санын қамтитын семантикалық желі құрылса, бір атау әр түрлі ұғымдарды көрсеткенде **синонимия** проблемасы үнсіз болады.

Егер ежелгі уақытта жеке адамды анықтау үшін «Назареттік Иса» деп айту жеткілікті болса, қазір паспортты рәсімдеуде қолданылатын «Аты – жөні» – «туған күні» – «туған жері» үштігі де қайталанатын сәйкестендіргіштердің болмауына кепілдік бермейді. Сонымен қатар, мұндай күрделі кілт (мәліметтер базасы тұрғысынан) көріну мен қабылдаудың ыңғайлылығына ықпал етпейді. Осыған байланысты тәжірибеде қабылданған жергілікті семантикалық желілер үшін тест кітапшасының нөмірлері, персонал нөмірлері, салық төлеушінің сәйкестендіру нөмірі(СТСН) және т.б. пайдаланылуы мүмкін. Тағы бір проблема – әр түрлі ұғымдарды білдіретін бір сөз қолданылатын **полисемия.** Синонимия мен полисемия үлкен желілерді құру проблемасын, атап айтқанда, әр түрлі авторлардың фрагменттерін біріктіруді қиындата түсуі мүмкін.

Шыңның атауы жай ғана символдық атау, оның мағынасы графиктің көрінуін арттырады. Оның қасиеттерінің жоғарғы жағы толығымен анықталған, мысалы, адам үшін – тегі, аты, әкесінің аты, туған күні және т.б.

6.6. Семантикалық желідегі фактілер мен ережелер

Субъект-предикат-нысан түрінде жазылған жоғарыда қарастырылған қатынастар тұрақты білім болып табылады, яғни. фактілер. Әрбір объект туралы барлық белгілі фактілерді енгізу негізсіз ұзақ уақытты қажет етуі мүмкін. Мысал ретінде отбасылық қатынастарды беру орынды. Кез-келген екі туыс үшін олардың арасындағы қарым-қатынастың атауы бар: ағасы, жиені, қайын енесі және т.б. Осылайша, n = 10 адамнан тұратын отбасы үшін қатынастар саны n * (n-1) = 90 болады. Бұл жағдайда қарым-қатынастың бір бөлігі бастапқы (жұбай және ата-ана-бала) болып

табылады, ал басқа қатынастар бастауыштан басталады. Егер бастапқы қатынастар негізінде қайталама қатынастардың қалай анықталатындығы туралы ақпарат ережелер түрінде жазылса, онда білім объектісіне әр объект үшін тек бастапқы фактілерді ғана енгізуге болады. Отбасылық қатынастар үшін бұл отбасының әр мүшесі – біреудің баласы және ата-анасы, сондай-ақ біреудің жұбайы және басқа ештеңе емес деп болжайтын болсақ, бұл кем дегенде 3 есе азайтылуды білдіреді.

Ережелерді ұсыну тілінің стандарттарының бірі SWRL – Semantical Web Rule Language (http://www.w3.org/Submission/SWRL/). Бұл тіл XML кеңейтімі болып табылады және оның үстіндегі құрылыстар тек машинаны түсіндіру үшін арналған. Ереже редакторлары, әдетте, «human readable» ережелердің нұсқасын оларды құру және күйіне келтіру үшін Пролог ережелеріне ұқсас форматта ұсынады. Төменде адамның оқуы үшін «ағай» қатынасын анықтауға арналған ереженің үзіндісі және осы ереженің бастапқы коды SWRL-де берілген:

$hasParent(?x1,?x2) \wedge hasBrother(?x2,?x3) \Rightarrow hasUncle(?x1,?x3)$ …

```
<swrl:Imp rdf:ID="Bros">
<swrl:body>
<swrl:AtomList>
<rdf:first>
<rdf:Description>
<rdf:type rdf:resource="&swrl;ClassAtom"/>
<swrl:argument1>
<rdf:Description rdf:about="#x3"/>
</swrl:argument1>
<swrl:classPredicate rdf:resource="#Person"/>
</rdf:Description>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<rdf:Description>
<rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
<swrl:argument2>
<rdf:Description rdf:about="#x3"/>
…
<rdf:Description rdf:about="#x1"/>
</swrl:argument1>
<swrl:propertyPredicate rdf:resource="#hasBrother"/>
</rdf:Description>
</rdf:first>
```

<rdf:rest rdf:resource="&rdf;nil"/>
</swrl:AtomList>
</swrl:head>
</swrl:Imp>

Семантикалық желі объектілері үшін ережелерді орнату арқылы біз ашық немесе жабық әлем проблемасына тап болуымыз мүмкін (Open or Closed World Assumption). Ашық әлем туралы болжам ешкімнің әлем туралы толық ақпараты жоқ дегенді білдіреді, сондықтан тұжырымдар тек белгілі нәрсеге сүйене отырып жасалуы керек. Жабық әлем жорамалы барлық ақпарат бақылаушыға белгілі деп болжайды. Мысал ретінде «өгей ана» түріндегі туыстық қатынасты айтуға болады. Келесі мәліметтер білім базасында болсын:

Андрей – Егордың ата-анасы.
Юлия – Андрейдің әйелі.

Жабық әлемнің болжамына сәйкес, Юлия – Егордың өгей шешесі, өйткені дерекқорда оның анасы екендігі туралы ақпарат жоқ. Ашық әлемде Юлияны Егордың өгей шешесі деп санауға болады, егер оның анасы Юлия емес, басқа әйел екендігі белгілі болса.

Ережелерді қолдану білім базасында толық емес ақпарат болған жағдайда өте пайдалы. Мысалы, алдыңғы мысалда Андрейдің тұлға екендігі айтылған делік, бірақ Юлия деген тұлға туралы ақпарат жоқ. Сонда барлық ережелер ұқсас

Егер X адам болса, онда X-те тегі бар

Юлияға қолдануға болмайды. Егер сіз ереже жасасаңыз

Егер X адам және X жұбайы болса, онда Y - Y адам

сонда Юлияның да ер екені анықталуы мүмкін. Ережелерді қолданудың жағымды әсерінен басқа, бір кемшілігі де бар: білім базасының көлемі өскен сайын космостық пропорцияларға дейін өсетін комбинаторлық күрделілік. Мысалы, егер білім қоры өте аз болса, әрқайсысы үш фактінің 100 фактісі мен 10 ережесі болса, ережелерді фактілерге қолдануға тырысудың жалпы саны 10 * 100 * 100 * 100 = 107-ге жетуі мүмкін, өйткені әрбір ереже барлық мүмкін фактілер дәйекті түрде қойылады. Семантикалық желіде іздеудің мұндай «қарапайым» жүзеге асырылуы мүмкін емес екені анық. Кез-келген іздеу тапсырмасындағыдай, мұнда комбинаторлық күрделілікті төмендету мәселесін шешу қажет. Ережелерді өңдеуді жеделдетудің мысалы ретінде **Rete** (Рити) алгоритмін қолдануға болады [24], оның негізгі мағынасы ағаш салу, оның әрбір түйіні ережелер шарттарының бөлігіне сәйкес келеді және осы шарттарды қанағаттандыратын фактілер тізімін сақтайды. Ережелерді қолдану барысында үнемі жаңа фактілер пайда болатындықтан, олар желі арқылы қозғалады және шыңдардағы фактілер тізімі жаңартылады. **Rete**

алгоритмінің қиыншылығы – бұл жадтың үлкен көлемі, өйткені дәл сол фактілер граф шеттерінде тізімдерде қайталанатын болады.

Баламалы шешім жолдарының бірі ретінде әрбір құжат үшін барлық мүмкін ережелерді бір рет іске қосуға және нәтижелерді фактілер түрінде сақтауға болады. Туыстық байланыстар базасы үшін бұл құжатқа алдымен тек бастапқы байланыстар (ата-аналар-балалар мен ерлі-зайыптылар) енгізілгенін білдіреді, содан кейін олардың ішінен барлық жанама қатынастар (немере қарындастар және т.б.) есептеледі, олар тең құқықта білім базасын толықтырады. Осыдан кейін барлық фактілер бірдей тез алынады. Мұндай тәсіл прецеденттер (Case based reasoning) негізіндегі қорытынды түрі болып табылады және оны адам интеллектіндегі дағдылардың аналогы деп санауға болады. Шын мәнінде, біз әрдайым ұқсас әрекет етеміз, мысалы, ауызша сөйлеу. Егер біз әр сөз тіркесін құрғанда тіл ережелерін қолдансақ, сөйлеу жылдамдығы сағатына бірнеше сөйлемден аспауы керек. Бұл әсіресе, орыс мәтінін шет тіліне аударғанда байқалады. Егер тілдік дизайн бізге таныс болса (бұрын бірнеше рет қолданылған), онда аударма жылдам қарқынмен жүреді. Біз бірінші рет ұсыныс жасайық, онда аударма процесі ондаған және жүздеген рет баяулайды.

Интеллектуалды семантикалық желі агенті

Семантикалық желі құру оңай шаруа емес. Бірақ біз онымен айналысқаннан кейін сұрақ туындайды, бірақ семантикалық желіден білімді қалай алу керек? Бұл үшін пайдаланушының сұранысы бойынша қажетті білімді іздеп, нәтиже беретін арнайы бағдарлама әзірленуі керек.

Қазіргі уақытта RDF форматындағы семантикалық желілер түрінде білім негіздеріне арналған бірнеше сұраныс тілдері бар, атап айтқанда DQL, R-DEVICE, RDFQ, RDQ, RDQL, SeRQL. Ең стандартталған тіл SPARQL болып табылады, ол World Wide Web (W3C) [29] деректерге қол жеткізу жөніндегі Data Access Working Group (DAWG) стандарттауды өтті.

Әр түрлі бағдарламалық платформаларға арналған SPARQL тілінің бірнеше нұсқалары бар. Автор олардың бірнешеуін сынап көрді және SPARQL тілдік процестегі сұраулар тек фактілерді (үштіктер тақырыбы-предикат-объект), бірақ ережелерді түсінбейтіні белгілі болды. Осылайша, онтологияны құру бойынша барлық жұмыс мағынасыз болып қалады.

Бұл кемшілікті жою үшін автор семантикалық құжаттарды ұсынудың жеңілдетілген тілін және білімнің визуализациясын және қарапайым сұраныстарды орындауды қолдайтын бағдарлама жасады. Осы пәннің бір бөлігі ретінде семантикалық желілерді құруға және зерттеуге арналған SEMANTIC бағдарламасы, осындай ақылды агенттің қасиеттері туралы

түсініктерді қамтиды. Атап айтқанда, бағдарлама мұрагерлік ережелерін білім қорында жазылған барлық фактілерге қолданады, сонымен қатар қолданушыға өз ережелерін жасауға мүмкіндік береді.

6.7. Контексті басқару

Семантикалық желінің барлық нысандарын бірегейлендіру қажет-тілігі фактілерді қосу процедурасын күрделендіріп қана қоймай, сонымен қатар білім алу өте қиын болатынына алып келеді. Сіз бұл мәселені түсінуді қарапайым мысалмен жеңілдете аласыз. Көршіміздің жасанды интеллект туралы дәріс жазбаларын сұрап алуды бір күн сұрап көрейік, ол өз кезегінде өзінің құрбысынан қарыз алған. Сонда диалог шамамен былайша болады:

"Ресей Федерациясының азаматы Сидоров Владимир Иванович, 1985 жылы Саратов қаласында туған, Санкт-Петербург қ .20.05.2003 51 ЖМ берілген № 60 04 123456 паспорты бар, маған, Ресей Федерациясының азаматы Петров Иван Викторовичке бер, 22.04.1986 жылы Псков қаласында туған, паспорты бар, № 6606 654321, 24 сағат 00 минутта" жасанды интеллект" пәні бойынша дәріс конспектісін есептеу техникасы кафедрасының доценті, т.ғ.к., Бессмертный Игорь Александрович оқиды...".

Күнделікті жағдайда қол жетімді емес фраза, бірақ полицияның есебінде өте жақсы. Әлбетте, семантикалық желіні құрған кезде сіз барлық нысандарды біркелкі анықтап, анықтай аласыз, бірақ бұл жұмысты айтарлықтай қиындатады. Бірақ білімге қол жеткізу үшін нақты өмірде болатын диалог сияқты қарапайым диалог жүргізуге мүмкіндік беру керек. Мұндай функцияны білімге қол жеткізуді қамтамасыз ететін зияткерлік агентке жүктеуге болады.

Контекстік база екі компоненттен тұруы керек: тұрақты және уақытша. Тұрақты контекст – бұл диалог процесінде өзгермейтін білім. Мысалы, біз оның қай уақытта екенін білгіміз келеді. Бұл сұраққа ешкімге қиындық туғызбайды, тақырыптың орналасқан жері туралы ақпаратсыз жауап беру мүмкін емес. Сондықтан, мәтінмәндік мәліметтер базасында тақырып қай жерде, сонымен қатар осы уақыттың белдеуі туралы ақпарат болуы керек. Басқаша айтқанда, мәтінмән дерекқорға жүктелуі керек.

Уақытша контекст – бұл диалог процесінде құрылған немесе жойылған (ұмытылған) фактілер, сонымен қатар диалогты жеңілдету үшін құрылған уақытша бірлестіктер. Уақытша фактілер, мысалы, бұрын қойылған сұрақтарға жауаптар, яғни сырттан әкелінген және білім базасында сақтауды қажет етпейтін білім. Мұндай фактілердің мысалы

диагноз қоюға тырысатын дәрігердің сұрақтарына пациенттің жауаптары болуы мүмкін. Мұндай жадтың болмауы диалогты склеротика туралы көптеген әзілге айналдырады. Уақытша қауымдастықтар объектілерге немесе фактілерге қысқаша атауларды осы диалогта ғана қолдануға мүмкіндік береді. Уақытша қауымдастықтар күнделікті өмірде де, құжаттарда да кеңінен қолданылады.

Мысалы, шарттардың мәтіндерінде әдетте "ООО РОГА және КОПЫТА, директор А.А. Фунт тұлғасында, жарғы негізінде әрекет ететін, бұдан әрі Сатып алушы деп аталатын айналым пайдаланылады...".

Осылайша, контекстік база қолданушыға қарапайым тәсілмен диалог құруға мүмкіндік беретін семантикалық желінің қарапайым моделін (моделін) құруға мүмкіндік береді.

6.8. Семантикалық желі және семантикалық паутина

6.2-бөлімде көрсетілгендей, семантикалық желі ұзақ тарихы бар, және осы уақытқа дейін бұл түсінік ешқандай екіұштылық тудырған жоқ.

Алайда, Бүкіләлемдік паутина өнертапқышы (WWW) Тим Бернерс Ли 2001 жылы "Semantic Web" тұжырымдамасын жариялағаннан кейін, орыс тіліндегі әдебиетте бұл екі ұғымдар араласа бастады

Түсініспеушіліктерді жою үшін "Semantic Web" ұғымын "Семантикалық Паутина" ретінде немесе осы мәтінде "Паутина" деп аудару ұсынылады. Тим Бернерс Ли идеясының мәні интернет ресурстарын компьютерлік өңдеу үшін қол жетімді және қазіргі уақытта қолданылатын құжаттардың мәтіндік талдауының орнына дүниежүзілік паутина ресурстарының қасиеттері мен мазмұнын бір мағыналы сипаттайтын арнайы метадеректермен қамтамасыз ету болып табылады. Атап айтқанда, Интернеттің барлық ресурстарын әрбір құжаттың авторын орнатуға мүмкіндік беретін тегтермен жабдықтау көзделіп отыр. Айта кету керек, бұл шешім WWW тұжырымдамасын орындау үшін, Интернет білім қорынан ақпараттық «қоқыс үйіндісіне» айналғаннан кейін, ресурстардың рұқсат етілмеген белгісіздігіне байланысты қабылданды.

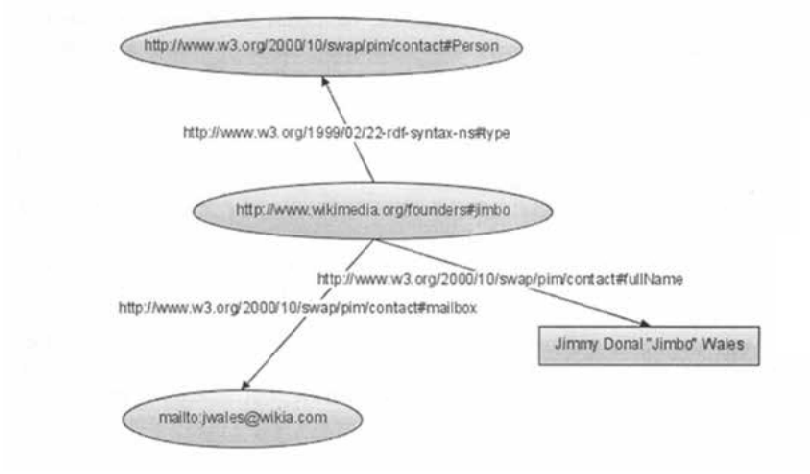
Алайда, семантикалық тордың осы қасиеті бізді соңғы кезекте қызықтырады. Біздің қызығушылығымыздың ортасында-семантикалық Паутина қасиеттері пайдаланушыға ол алғысы келетін ақпаратты жеткізу.

WWW-ден семантикалық Паутинаның негізгі айырмашылығы – WWW ресурстары арасындағы байланыс құжаттардың орналасуын, ал Паутина байланысы – мазмұнды бейнелейді. Рас, WWW-да, сондай-ақ, өрмекші сілтемелерде де ресурстар адрестері бар, бірақ WWW-да сілтемелерсіз; олардың мағынасын пайдаланушы түсінуі тиіс. Интернетте

сілтемелер машинада өңдеуге болатын нақты көрсетілген. Осылайша, семантикалық веб интернетті қолданушы құжаттан құжатқа сілтемелерді басқанда, қазіргі веб-серфингтің орнына ақпарат іздеуді білдіреді [25].

6.9. Семантикалық Паутина: принциптері мен ағымдағы жағдайы

Семантикалық Паутина негізінде сол триплет тақырыбы -предикат-нысан жатыр. Айырмашылық триплет элементтерінің әрқайсысын көрсету. Торда тақырыбы, нысан және предикат ресурстардың әмбебап идентификаторлары немесе URL (Uniform Resource Identifier) көрсетілген. Мысал ретінде 6.7-суретте көрсетілген бағандар болуы мүмкін.

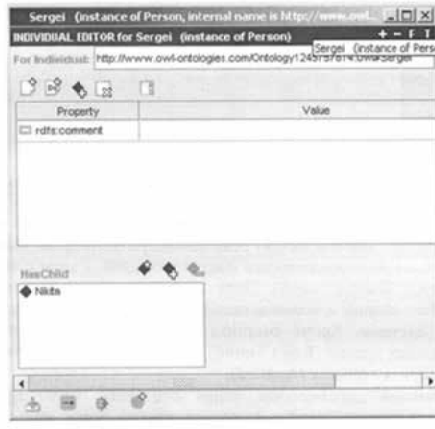


6.7-сурет. Википедия негізін салушы визит картасының графы

Интернеттегі білімді ұсынудың ең танымал тілі – бұл RDF. Төменде Сергейдің Никита атты баласы бар, ал Никитаның ата-анасы Сергеймен қарым-қатынасын сипаттайтын RDF құжатының үзіндісі келтірілген.

```
<Person rdf:ID="Nikita">
<HasParent rdf:resource="#Sergei"/>
</Person>
<owl:Class rdf:ID="Person"/>
<Person rdf:ID="Sergei">
<HasChild rdf:resource="#Nikita"/>
</Person>
```

Бұл пішім адам оқуға арналмаған. Семантикалық құжаттардың редакторлары, мысалы, protégé (http://protege.stanford.edu/), деректерді экрандық формада (фреймдерде) енгізуге және өңдеуге мүмкіндік береді.



6.8-сурет

Қазіргі уақытта RDF / OWL құжаттарын жеке энтузиастар жасайды. Интернеттегі құжаттар саны аз, оларды арнайы іздеу сервері, мысалы SWOOGLE көмегімен табуға болады. Жобаны әзірлеу көптеген себептерге байланысты тоқтатылып отыр. Бұл негізінен семантикалық ресурстарды құрудың тез және тікелей пайдасының жоқтығы, білімді қалыптастырудың күрделілігі және білімді шығаратын әмбебап агенттердің болмауы. Интеллектуалды агенттердің сатылымы студенттік әзірлемелермен, мысалы, Саутгемптон сыра нұсқаулығы (http://www.twine.com/item/11by40gxk-p1/southampton-pub-guide-in-rdf) және қандай шарапты ұсынатын шарап агентімен шектеледі. әр ыдысты қолданған дұрыс.

ҚОРЫТЫНДЫ

Жасанды интеллект – бұл адамның дәстүрлі құзіреті болып саналатын шығармашылық функцияларды орындау үшін интеллектуалды жүйелердің меншігі, бұл құралдар жүйесі, тіпті өте тар және күрделі тапсырманы шешетін математикалық алгоритмдердің жиынтығын айтуға болады. Пилотсыз көлік А нүктесінен В нүктесіне дейін жүре алады, бірақ 6 санын 7-ге көбейту туралы сұраққа жауап бере алмайды. Біз бұл сұраққа жауап беретін басқа құралды орналастыра аламыз, бірақ олар бір-бірімен байланысты болмайды. Сондықтан сіз барып, батырмаға шахтаны қазып алатын роботты жасай аласыз, ұшқыштың орнына ұшақта ұшатын робот жасай аласыз. Бірақ бұл екі түрлі робот болады, және оларды басқаратын ортақ робот жоқ - адам оны әлдеқайда тиімді етеді. Қойылған мақсаттарды қою керек, не істеу керектігін түсіну керек күрделі міндеттерде жасанды интеллект адамға шексіз жоғалады. Ия, бәлкім, робот миналарды саперден гөрі жақсы қазып шығар, мүмкін, егер ол қаза алмаса, өкінбеймін. Ұшақтарда олар ұшып кетуі мүмкін, дәлірек айтатын болсақ.

Жасанды интеллект (artificial intelligence) – компьютерлік жүйелердің адам ақыл-ойының жеке функцияларын қабылдау қабілеті, мысалы, алдыңғы тәжірибе мен сыртқы әсерлерді ұтымды тCheat негізінде алынған оңтайлы шешімдерді таңдау және қабылдау.

Бұл анықтамада білім термині тек миға сезім мүшелері арқылы енетін ақпаратты ғана білдірмейді. Бұл білім түрі өте маңызды, бірақ зияткерлік белсенділік үшін жеткіліксіз. Біздің қоршаған орта объектілері сезімге әсер етіп қана қоймай, сонымен бірге бір-бірімен белгілі бір қатынастарда болатын қасиетке ие. Қоршаған ортада интеллектуалды белсенділікті жүзеге асыру үшін (немесе тіпті бар болса да) білім жүйесінде осы әлемнің моделі болу керек екені түсінікті. Қоршаған ортаның осы ақпараттық моделінде нақты объектілер, олардың қасиеттері және олардың арасындағы қатынастар тек бейнеленбейді және есте сақталады, сонымен қатар, осы анықтамада айтылғандай, ақыл-ойды мақсатты түрде өзгертуге болады.

Осылайша, Жасанды интеллект – бұл компьютерлік жүйенің өздігінен оқу кезінде белгілі бір күрделілік класының мәселелерін шешуге және осы мәселелерді біздің өміріміздің барлық салаларында шешуге арналған бағдарламалар (ең алдымен эвристикалық) құру мүмкіндігі.

Іс жүзінде ЖИ мүмкіндіктерінің ауқымы шексіз: ғарыштық зерттеулер, әскери ғылым, робототехника, өнеркәсіп, ауыл шаруашылығы, көлік, медицина, білім және т.б.

«Цифрлық Қазақстан» мемлекеттік бағдарламасы аясында ұлттық экономиканың барлық салаларында өндіріс процестерін цифрландыруға

және роботтандыруға көп көңіл бөлінеді. Бұл бағдарламаны іске асыруда ЖИ әдістері мен құралдарын жасау маңызды рөл атқарады. Тәжірибеде ЖИ қолдану осы оқулықтың екінші томында талқыланады.

Өзін өзі бақылауға арналған сұрақтар

1. Семантикалық желілерге анықтама беріңіз.
2. Бұл семантикалық желінің элементі.
3. Не үшін пайдаланылады семантикалық желі жүйелерінде жасанды интеллект?
4. Талдау және мәтіндік ақпаратты іздеу есептерінде семантикалық желілер аппаратын қолдану саласындағы ең озық зерттеме дегеніміз не?
5. Семантикалық желілердің типтерін атаңыз.
6. Реляциялық графтың мысалын келтіріңіз.
7. Қандай желілер пропозициялық желілер деп аталады?
8. Оқиғаларды көрсету үшін қандай бағандар қолайлы?
9. "Пошташы"семантикалық желісін салыңыз.
10. Өсімдік графтарындағы байланыс түрлерін атаңыз.
11. Семантикалық желілерді құру мәселелерін көрсетіңіз.
12. Семантикалық желідегі фактілер мен ережелерді түсіндіріңіз.
13. PDF форматындағы семантикалық желілер түріндегі білім базаларына сұраныс тілдерін атаңыз.
14. Семантикалық желі контекст базасы қандай екі компоненттен тұрады?
15. Семантикалық паутина түсінігін беріңіз.

ГЛОССАРИЙ

А

Абдукция – ереже мен нәтижеден алғышарттар шығаратын синтетикалық тұжырым нысаны.

Абстрагирование – эмпирикалық объектінің басқа қасиеттерінен немесе қатынастарынан алаңдату арқылы кейбір қасиеттерді немесе қарым-қатынасты ойдан бөліп алу (түсіну).

Ағаш – бір шыңы тамыр, қалған шыңдары тек бір ғана әкесі бар және барлық шыңдары тамыр шыңының ұрпақтары болып табылады.

Ағаштың биіктігі – тамырдан жапыраққа дейінгі ең ұзақ жол.

Ағаштың тәжі – бұл барлық жапырақтардың жиынтығы.

Амфиболос (грек amphibolos – екілік, қосарлылық) – логикалық қате, оның негізінде тілдік өрнектердің екіұштылығы жатыр.

Ақылды қолдауы бар жүйе дегеніміз – өз бетінше шешім қабылдауға қабілетті жүйе.

Атом – бұл предикат формасы немесе кейбір теңдік, яғни өрнек (s = t), мұндағы s және t терминдер

Б

Бейресми тапсырмалар – келесі сипаттамалардың бірі немесе бірнешеуі бар міндеттер:

1) олар сандық түрде берілуі мүмкін емес, яғни сапалы түрде немесе тақ жиындар теориясының терминдерінде беріледі;

2) мақсаттар нақты белгілі бір мақсатты функцияның терминдерінде көрсетілуі мүмкін емес;

3) тапсырмаларды алгоритмдік шешу жоқ;

4) алгоритмдік шешім бар, бірақ оны ресурстардың шектелуінен (уақыт, жады) пайдалануға болмайды.

Білім – оның міндеттерін шешуге мүмкіндік беретін пәндік саланың анықталған заңдылықтары (принциптер, байланыстар, заңдар).

Білім қорын басқару жүйесі (БҚБЖ) – біліммен жұмысты қамтамасыз ететін құралдар жиынтығы.

Білім жөніндегі Инженер (когнитолог, инженер-интерпретатор) – сарапшы мен білім базасы арасындағы аралық буфер рөлінде өнер көрсететін ЖИ жөніндегі маман.

Білімдер базасы (ББ) – бұл ақпаратты ұсыну тілінде (әдетте табиғиға жақын) компьютерлік ортада жазылатын домендік білім жиынтығы, InS ядросы.

Д

Дедукция дегеніміз – белгілі бір істерге жалпы ережелерді қолдануға негізделген, нәтиже шығатын аналитикалық процесс.

Декларативтік білім – құрылымдалған деректер жиынтығы түріндегі фактілер.

Деректер – 1) белгілі жағдайлар; 2) «шындық» мағынасы белгілі қатынастар немесе қасиеттер.

Е

Ереже (Пролог тілінде) – бұл қорытынды немесе басқа тұжырымдар немесе фактілер шын болса, белгілі болатын қорытынды.

Ж

Жалпы мақсаттағы ЖИ жүйелері – бұл тек белгіленген процедураларды орындап қана қоймай, мета-іздеу процедуралары негізінде жаңа нақты мәселелерді шешуге арналған процедураларды құратын және пайдаланатын жүйелер.

Жағдайлар – нысандар арасындағы өзара әрекеттесудің барлық түрлері.

Жалпылау – кез-келген басқа ұғымдарды салыстыру арқылы тұжырымдаманы ақылмен таңдау.

Жасанды интеллект (ИИ) – бұл информатиканың бір бағыты, оның мақсаты қолданушыға-бағдарламашы өзінің міндеттерін қоюға және шешуге мүмкіндік беретін, дәстүрлі түрде интеллектуалды болып саналатын, табиғи тілдің шектеулі Ішкі жиынында ЭЕМ-мен қарым-қатынас жасай отырып, аппараттық-бағдарламалық құралдарды әзірлеу.

Жеке константтар мен жеке айнымалылар (логика бойынша) тұрақты және ауыспалыға ұқсас, математикалық анализде олардың айырмашылығы нақты сандар емес, жеке адамдар болып табылады.

И

Индукция – алғышарттар мен нәтижеге сүйене отырып, ережені шығаратын синтетикалық пайымдаулар.

Интеллект – 1) тәжірибені үйрену және әртүрлі жағдайларға бейімделу барысында білімді алу, есте сақтау және мақсатты түрде өзгерту арқылы мидың проблемаларды (интеллектуалдық) шеше алу қабілеті;

2) дербес, тиімді (дұрыс, ресурстардың мүмкіндігінше аз шығындарымен) сапалы (дұрыс, қарапайым, ресурстардың мүмкіндігінше аз шығындалуын талап ететін) шешімдерді (оның ішінде жаңа, бұрын

белгісіз) әр түрлі күрделі "міндеттерді", оның ішінде жаңа, бұрын белгісіз (ең дұрысында – кез келген мүмкін болатын "міндеттерді") табу қабілеті.

Интеллектуалданған жүйе – бұл шешім қабылдайтын оператор-тұлғалардың қатысуымен(ШҚҚ) міндеттерді шешу кезінде зияткерлік қолдаумен ақпараттық-есептеу жүйелері (АЕЖ).

Интеллектуалды жүйе дегеніміз – оператордың (шешім қабылдаушы – ШҚҚ) қатысуынсыз мәселелерді шешуде зияткерлік қолдау көрсететін ақпараттық-есептеу жүйесі (АЕЖ).

Интеллектуалды анықтамалық – бұл ақпараттық жүйелер, оның ядросы пәндік білімнің генераторы болып табылады және мақсаты тар тақырып аясында анықтамалық мәліметтерді құру болып табылады.

Интеллектуалды агент – интеллектуалды қабілеттері бар жүйе (адам, бағдарлама).

Интенсивті – нақты қасиеттерді көрсете отырып, абстракцияның неғұрлым жоғары деңгейі туралы түсінік.

К

Кадр дегеніміз – объектілердің, құбылыстардың, жағдайлардың, процестердің және т.б. стереотиптік сыныптары туралы білімді ұсыну үшін қажет ең аз ақпараттық құрылым.

Қарапайым тұжырымдама – бұл ұғымның атауынан, оның кеңеюінен және интенсивтіліктен тұратын үштік.

Қиғаштау – бұл іздеуді басқару әдісі.

«Күшті ЖИ» – адамның мінез-құлқын (роботтарды) көшіретін машиналарды жасау.

Кешенді тұжырымдама – белгілі бір ережелерді қолдану арқылы бұрын анықталған тұжырымдама.

Құрама тізімдер дегеніміз – элементтердің бірнеше түрін қолданатын тізімдер.

Л

Логика – 1) дұрыс ойлаудың нысандары мен әдістері туралы ғылым; 2) ұғымдардың, пікірлердің, ақыл-ойдың және басқа да абстрактілі объектілердің арасындағы әмбебап (жалпы мәнді) өзара байланыс туралы ғылым.

Логикалық байланыстыру (предикаттарды есептеу) – формулаларды құру үшін қызмет ететін логикалық операция.

Логикалық бағдарламалау – бұл жоғары деңгейдегі тіл ретінде Хорн фраз түріндегі бірінші ретті предикат логикасы қолданылатын информатикаға арналған тәсілдердің бірі.

Логомахия – пікірталас барысында қатысушылар бастапқы ұғымдарды нақтыламағандықтан, ортақ көзқарасқа келе алмайтын сөздер туралы дау.

М

Мәлімдеме – дәлелсіз тұрақты предикат немесе нөлдік емес предикат формасы.

Міндет – мыналарды жүзеге асыру қажет болған кезде проблемалық жағдайлардың сыныбы: 1) ақпарат жинау; 2) жағдайды бағалау; 3) шешімдер қабылдау; 4) іс-әрекеттерді жүзеге асыру.

Метазнания – білім туралы: Жеке объект туралы білімнің көлемі мен шығу тегі туралы, нақты ақпараттың сенімділігі туралы немесе жеке фактілердің салыстырмалы маңыздылығы туралы білім.

Н

Нейрокомпьютер – бұл табиғи нейрондық желінің кейбір формальды моделін іске асыратын бағдарламалық-техникалық жүйе.

Нейроподобты желі – бір-бірімен және сыртқы ортамен белгілі бір түрде қосылған нейроподобты элементтердің жиынтығы.

П

Пайдаланушы – жүйе арналған тұлға.

Перцептрон – бұл үлгіні тануға арналған құрылғы.

Предикат – 1) бұл тиісті терминдермен бірге предикаттық атау; 2) (Пролог тілінде) меншіктің атауы немесе объектілер арасындағы қатынастар дәлелдер дәйектілігі.

Предикаттық тұрақты дегеніміз – предикатты сипаттайтын қатынас белгісі. **Предикативті константшындықтың мәнін өзгертпейді.**

Предикативті форма дегеніміз – тиісті терминдермен үйлесетін предикаттық тұрақты.

Предикативті ұғым дегеніміз – белгілі бір эмпирикалық объектінің меншігі ретінде қарастырылатын түсінік.

Предикация қатынасы – кейбір эмпирикалық объектінің жеке және предикат алдындағы концептілерін тұтас абстрактілі объектіге байланыстыратын қатынас.

Процедуралық білім – фактілерді өңдеу процедуралары түріндегі алгоритмдер.

Р

Рекурсия базисі – бұл бастапқы жағдайды немесе тоқтату кезіндегі жағдайды анықтайтын ұсыныс.

Рекурсивті процедура – бұл рекурсияны тоқтататын шарт орындалғанға дейін өзін шақыратын процедура.

Рекурсия кезеңі – ереже, оның құрамында міндетті түрде субгогал ретінде белгілі предикат шақыруы болады.

Ресми жүйе – бұл таза абстрактілі әдістердің жиынтығы, онда таза синтаксистік интерпретациядағы әр түрлі таңбалармен жұмыс істеу ережелері семантикалық мазмұнды ескерусіз берілген.

С

Салыстыру – объектілердің ұқсастықтары мен айырмашылықтарын белгілеу.

Сарапшы – жоғары білікті маман, қарастырылып отырған тақырып бойынша тәжірибе алмасуға келісті. Пайдаланушы интерфейсі – InS жұмысының барлық кезеңдерінде InS пайдаланушыларымен диалогты жүзеге асыратын бағдарламалар жиынтығы.

Сараптама жүйесі (білімге негізделген жүйе – БНЖ) – нақты бағдарламалық бағыттағы мамандардың білімін формальды түрде жинақтайтын бағдарламалық кешен.

Сараптамалық жүйелер (СЖ) (немесе білімнің инжинирингі) – жасанды интеллект бағыты, оның міндеті білім мен шығару процедураларын қолданатын және адам сарапшылары үшін қиын мәселелерді шешетін бағдарламаларды (құрылғыларды) зерттеу және әзірлеу болып табылады.

Сараптамалық жүйенің синтез технологиясы – бұл мамандардың біліміне негізделген, нашар құрылымдалған пәндік салаларда бейресми есептерді шешетін жүйелер құру технологиясы.

Семантикалық желі – бұл шыңдары ұғымдар, ал доғалар олардың арасындағы қатынастар болып табылатын бағытталған граф.

Синтез – бұл әртүрлі нысандардың тұтас бір объектіге айналуы.

Сұрау (Пологта) – бұл орындауға арналған бағдарламаға берілетін мақсаттар.

Т

Тақырып аймағы – мәселенің шешімімен байланысты шындықтың бөлігі.

Терең білім – тақырыптың құрылымын және жеке ұғымдардың өзара байланысын көрсететін абстракциялар, суреттер, аналогиялар.

Тікелей сатып алу – бұл компьютерлік жүйе ақпарат көзі мен ақпарат көзі арасында делдал болып табылатын тәсіл.

Тізімнің басы – тізімнің бірінші элементі.

Түсініктеменің ішкі жүйесі – пайдаланушыға сұрақтарға жауап алуға мүмкіндік беретін бағдарлама: бұл немесе басқа ұсыныс қалай қабылданды және жүйе неге осындай шешім қабылдады?

Тұжырымдамалар эмпирикалық объектілердің қарапайым және күрделі қасиеттері (атрибуттары) ретінде адам түсінуіне қол жетімді дерексіз нысандардың мәні болып табылады.

Тізім – бұл басқа нысандардың ақырғы санын қамтитын мәліметтер нысаны.

Тізімнің құрылымы (Lisp) – бұл тізім, оның элементтері атомдар да, тізімнің басқа құрылымдары да, қарапайым тізімдер де бола алады

Термин – бұл әр ауыспалы және әрбір функционалды нысанда.

Тізімнің құйрығы – барлық кейінгі элементтерді қоскқанда тізімнің бөлігі.

Ф

Функционалды тұрақты – бұл терминдердің сәйкес санымен үйлескенде, функционалдық форманы құрайтын, семантикалық домені жеке тұрақтылар жиынтығы болатын қисынды предикат. Нөлдік функционалды тұрақты – жеке тұрақты

Функционалды форма дегеніміз – тиісті терминдермен үйлесетін функционалды тұрақты.

Функционалды тұтастық – қажетті нәтижені, нәтижені алу уақыты мен тәсілдерін, нәтиженің жеткіліктілігін тCheat құралдарын таңдау мүмкіндігі.

Э

Эвристика – мамандардың тәжірибесінен алынған білім.

Экстенциалды – иерархияның төменгі деңгейінің тұжырымдамаларын немесе анықталғанға қатысты фактілерді тізімдеу арқылы анықтау.

Эквокуация – бір сөзді әртүрлі мағынада қолдануға негізделген логикалық қате

ARTIFICIAL INTELLIGENCE: MODERN THEORY AND PRACTICE

Decoding of abbreviations

AI – Artificial intelligence
AAI – Association of artificial intelligence
ANN – Artificial neural network
ANFIS – Adaptive neuro fuzzy inference system
CS – computer science
CWA – Closed world assumption
DAWG – Data access working group
DBMS – Data base management system
DK – Declarative knowledge
ES – Expert system
FNN – feedback neural network
EPAMP – Elementary Perceiving and Memorizing Program
FRL – Frame representation language
IP – Internal interpretability
IS – Information system
IJAIC – International Joint AI Conference
HM – Hopfield model
KB – knowledge bases
KBMS – knowledge base management system
KS – Knowledge sources
LBMI – Leningrad Branch of the Mathematical Institute
MNN – multi-layer neural networks
NSS – Newell, Shaw, Simon.
PDC – Park-Distance-Control
PK – Procedural knowledge
SDM – Sample-driven module
SQL – Structured query language
SWI – Switch in
SWRL – Semantic web rule language
RAAI – Russian Association for AI
RCO – Russian context optimizer
RDF – Resource description framework
XML – eXtensible Markup Language
USA – United states of America
WM – working memory
WWW – World wide web

Published to the 75-th anniversary of professor Baimukhamedov M.F.

FOREWORD

Recently, there has been an increase in interest in artificial intelligence, caused by increased requirements for information systems. Humanity is steadily moving towards a new information revolution, comparable in scale to the development of the Internet.

Artificial intelligence is an area of computer science, the purpose of which is to develop hardware and software tools that allow a non-programmer user to set and solve his traditionally considered intellectual tasks, communicating with computers in a limited subset of the natural language.

The history of artificial intelligence as a new scientific direction begins in the middle of the 20th century. By this time, many prerequisites for its inception had already been formed: among philosophers there was a long debate about the nature of man and the process of understanding the world, neurophysiologists and psychologists developed a number of theories regarding the work of the human brain and thinking, economists and mathematicians asked questions of optimal calculations and representations of knowledge about the world in formalized form; finally, the foundation of the mathematical theory of computation – the theory of algorithms – was born and the first computers were created.

The purpose of this manual is to outline the main directions and methods used in artificial intelligence, as well as to determine the possibility of their use in various fields of human activity. This tutorial has six chapters. The first provides a brief introduction to artificial intelligence, examines the history of its development as a scientific direction, highlights the main areas of artificial intelligence, examines the history of research and the main directions of research in the field of artificial intelligence.

The second chapter is devoted to the basics of programming in the Prolog language, the description of Prolog as a declarative language, the description of the structure and main elements of programs in the Prolog language, the concept of a predicate is given, the facts and rules, recursion and clipping in the Prolog are considered. The third chapter considers probabilistic reasoning, gives the basics of the theory of fuzzy logic, describes Bayesian networks, the Monty Hall paradox. The fourth chapter gives the concept of a neural network, describes the principle of constructing neural networks, considers the

features of using neural networks, describes neural networks in recognition problems, as well as design, training and adaptation of neural networks. The fifth chapter discusses the theoretical and practical issues of developing expert systems, describes algorithmic models, logical models for representing knowledge, production models and modules driven by samples. The sixth chapter is devoted to semantic networks. The types of semantic networks, the types of relationships in semantic networks are considered, the facts and rules in the semantic network are described, the problems of constructing semantic networks are given, the concept of the semantic web is given, the principles and the current state of the semantic web are considered.

The manual contains a large number of illustrations and tables. A glossary is provided for the convenience of studying the material.

INTRODUCTION

Ignacy Belda, in his work "Mind, Machines and Mathematics" wrote: "Artificial intelligence has gradually entered our lives. Sooner or later, the day will come when there will be machines with the same level of creativity, sensations and emotional intelligence as a person. On the day that this happens, we will realize that we are not alone."

One of the most prominent features of our time is computers that have penetrated all spheres of human activity. The steady increase in their performance observed over the entire period of the existence of computers has made it possible to endow the intellectual functions of even mobile devices. We are already used to the fact that the mobile phone's camera recognizes the scene and adjusts focus and exposure accordingly. Many drivers can not imagine moving around the city without a navigation system that helps to avoid traffic jams. If 20-25 years ago, when the Internet was just beginning to conquer the information space, in order to access the necessary resources it was necessary to know the URL and even guides to Internet resources ("yellow pages") were issued, now it's enough to type a phrase in the search system, it will give out not only a list of links to relevant resources, but – in some cases – will immediately give an answer to a question of interest to a user. These are all examples of intelligent systems in action.

Currently, research in artificial intelligence has identified six main areas [1]:

1. Submission of knowledge. Within the framework of this direction, problems associated with the formalization and presentation of knowledge in the memory of the AI system are solved. For this, special knowledge representation models and knowledge description languages are developed, various types of knowledge are introduced. The problem of knowledge representation is one of the main problems for the AI system, since the functioning of such a system is based on knowledge about the problem area that is stored in its memory.

2. Knowledge manipulating. To use knowledge in solving the problem, you should teach the AI system to operate with it. In the framework of this direction, methods are developed for replenishing knowledge based on their incomplete descriptions, methods for reliable and plausible inference based on existing knowledge are created, and models of reasoning based on knowledge and imitating the features of human reasoning are proposed. The manipulation of knowledge is very closely connected with the representation of knowledge, and these two areas can be divided only conditionally.

3. Communication. The range of tasks in this area includes: the problem of understanding and synthesis of coherent texts in a natural language, understanding and synthesis of speech, the theory of communication models

between a person and the AI system. Based on research in this direction, methods are being developed for constructing linguistic processes, question-answer systems, dialogue systems and other AI systems, the purpose of which is to provide comfortable conditions for human interaction with the AI system.

4. Perception. This direction includes the development of methods for presenting information about visual images in the knowledge base, the creation of methods for transition from visual scenes to their textual description and methods of reverse transition, the creation of tools for generating visual scenes based on internal representations in AI systems.

5. Training. To develop the ability of AI systems to solve problems that they have not encountered before, methods are being developed to formulate the conditions of tasks to describe a problem situation or to observe it, methods of transition from a known solution of particular problems (examples) to solving a general problem, creating methods for decomposing the original tasks for smaller and already known for AI systems. In this direction, AI has done very little.

6. Behavior. Since AI systems must operate in some environment, it is necessary to develop some behavioral procedures that would allow them to interact adequately with the environment, other AI systems, and people. This direction in AI is very poorly developed.

In practice, the range of capabilities of AI is almost endless: space research, military science, robotics, industry, agriculture, transport, medicine, education, etc.

For example, modern artificial intelligence systems are able to effectively control robotic devices, thanks to a significantly larger number of diverse information sensors and devices.

Robots are electrical devices designed to automate human labor.

AI methods allow you to create algorithms and hardware solutions for robotic systems operating in extreme conditions. A distinctive feature of these complexes will be an independent intelligent control system based on neural networks and fuzzy logic, the creation of an intelligent control system for a robotic system with the possibility of topographic and hardware adaptation depending on the extreme situation. An example of the practical result of the application of AI methods is the creation of a guard robot for working in domestic and industrial premises with a wide range of input systems from video surveillance to radiation monitoring and designed to ensure the safety of these rooms. The intelligent control system of the robot automatically adapts to the layout of the premises, provides automatic control of the movement of the robot taking into account the layout of the protected room, conducts intelligent data processing providing online monitoring and independently makes a control

decision, including sending messages via the telecommunication system in extreme situations.

Recent advances in AI can be represented by the following commercial projects [2]:

• The Remote Agent program developed at NASA is used to comprehensively control the operation of spacecraft far beyond the limits of near-Earth orbit, including Diagnostics and troubleshooting as they arise.

• Diagnostics. Medical diagnostic programs have managed to reach the level of an experienced doctor in several areas of medicine.

• Supply Planning. During the crisis in the Persian Gulf in 1991. The DART (Dynamic Analysis and Re-planning) system was deployed in the US Army, which provided automated delivery planning and scheduling of transportation, covering simultaneously up to 50,000 cars, people and cargo. The developers of this system said that this application alone paid off for their 30-year investment in artificial intelligence.

Modern advances in AI over the past decade have been represented by the following developments:

1. IBM's Deep Blue program was the first program to defeat the world champion in Kasparov's chess match.

2. Alvinn's computer vision system has been trained to drive in a lane. For 2,850 miles, the system has been driving 98% of the time.

3. In China, artificial intelligence has become a popular host. He reads the news in English, and looks and sounds like a real person – Zhang Zhao, an employee of "Cinhua" News Agency. The whole simulation was generated on a computer: the texts of the announcers, facial expressions and the movement of the lips of real people were combined. A video with a real person is loaded into the program, and the AI, using machine learning, independently analyzes gestures, the manner of conversation and other details in order to further reproduce it on the air.

4. At the University of North Carolina School of Pharmacy, scientists created AI from two neural networks. One downloads data on the structure and properties of molecules, as well as the desired effect. The second neural network learns from the first: assimilates this data and selects possible solutions. AI now works with over 1.7 million molecules. This will help to significantly speed up the process of developing new drugs, and successful results can become the basis for creating, for example, new antibiotics.

5. Experts from the Microsoft Application and Services Group in East Asia have created an artificial program that can "experience" emotions and talk to people "humanly". An AI named Xiaoice answers questions like a 17 year old girl. If she does not know the topic, she can lie. If she is caught lying, they will get angry or embarrassed. Xiaoice can be sarcastic, suspicious and impatient –

these qualities are well known to us all. The unpredictability of Xiaoice makes communicating with her very similar to communicating with a person. So far, such an AI is a something new, and the Chinese communicate with it when they want to have fun or when it is boring. But its creator is working on improving Xiaoice. Who knows, maybe Xiaoice will become Skynet's grandmother.

6. A group of scientists from Moscow State University, together with the technology startup HautAI OU, created the PhotoAgeClock artificial intelligence, which can determine a person's chronological age in the eyes. analysis of this particular zone. The neural network studied 8,500 photographs of the area around the eyes and learned to determine age with an accuracy of two years.

If the XX century was distinguished by the development of communication tools and the popularization of computing technologies, then the XXI century, apparently, will be the century of development of artificial intelligence and robotics. Artificial intelligence is already able to replace a whole team of the best lawyers, wins chess at world champions, performs an in-depth analysis of terabytes of video materials in minutes and even learned to create even more advanced artificial intelligence almost independently. As for robotics and human substitution, banking institutions have long been massively reducing cashiers, replacing them with terminals, and in the near future it is also expected to replace guides, bartenders, waiters, cleaners and cashiers in supermarkets.

In conclusion, it should be noted that the field of AI is increasingly gaining momentum. It is possible that in the coming years they will launch projects that are currently working in test mode: unmanned taxis, submarines or fighters controlled by AI, or virtual border guards who conduct searches at airports.

CHAPTER 1. INTRODUCTION TO ARTIFICIAL INTELLIGENCE

1.1. The concept of artificial intelligence

Artificial intelligence (AI) is a new area of computer science, the subject of study of which is any intellectual human activity that obeys pre-known laws. Figuratively, this direction is called the "eldest son of computer science," since many of the tasks that it has not solved are gradually being solved within the framework of artificial intelligence. It is known that the subject of computer science is information processing. The field of AI includes such cases (tasks) from this processing that cannot be performed using simple and accurate algorithmic methods, and of which there are a great many.

AI relies on knowledge of the process of human thinking. At the same time, it is not known exactly how the human brain works, however, for the development of effectively working programs with AI elements, the knowledge about the features of human intelligence that science has today is enough. At the same time, the AI is not trying to copy exactly the work of the human brain, but is trying to simulate its functions using computer technology.

From the very moment of its birth, AI has been developing as an interdisciplinary direction, interacting with computer science and cybernetics, cognitive sciences, logic and mathematics, linguistics and psychology, biology and medicine (Fig. 1).

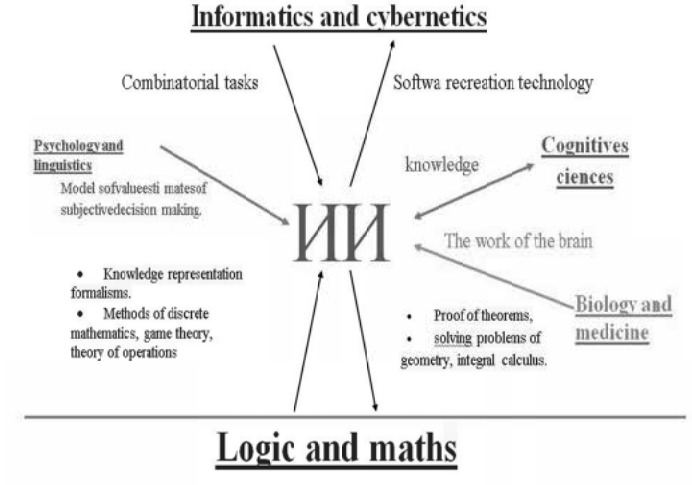


Fig. 1.1 – Interaction of AI with other disciplines

Computer science and cybernetics. Many experts came to AI from computer science and cybernetics. Also, many combinatorial problems that cannot be solved by traditional methods in computer science migrated to the field of AI. In addition, the results obtained in AI are borrowed when creating software and become part of Computer Science.

Cognitive science. Cognitive sciences are knowledge sciences. AI also deals with knowledge. But cognitive sciences use not only information and neurobiological approaches, but also consider the social and psycholinguistic aspects of the use of knowledge.

Logic and mathematics. Logic is the basis of all known formalisms of knowledge representation, as well as programming languages such as Lisp and Prolog. To solve AI problems, methods of discrete mathematics, game theory, and theory of operations are used. In turn, AI can be used to prove theorems, solve problems in various fields of mathematics: geometry, integral calculus.

Psychology and Linguistics. Recently, AI experts have become interested in the psychological aspects of human behavior in order to model it. Psychology helps to build models of value assessments, making subjective decisions. The psychology of communication is of interest "Human-computer", psycholinguistics. Computer linguistics is a part of AI, which is based on mathematical methods of processing natural and artificial languages, on the one hand, and on the phenomenology of language on the other.

Biology and medicine. Biology and medicine allow you to better study and understand the work of the brain, vision systems, hearing and other natural sensors and give a new impetus to the modeling of their work.

There is no single definition of AI, just as there is no single definition of natural intelligence. Among many points of view on this scientific field, three dominate now [3]:

1. Research in the field of AI is fundamental research, within the framework of which models and methods for solving problems that are traditionally considered intellectual and previously unsupported by formalization and automation are developed.

2. AI is a new field of computer science associated with new ideas for solving computer problems, with the development of a fundamentally different programming technology, with the transition to a computer architecture that rejects the classical architecture, which dates back to the first computers.

3. As a result of work in the field of AI, many application systems are born that solve problems for which previously created systems were not suitable.

To illustrate the first approach, we can give an example with a calculator. At the beginning of the century, arithmetic calculations with multi-valued numbers were the destiny of a few gifted individuals, and the ability to perform such arithmetic operations in the mind was rightfully considered a unique gift of

nature and was the object of scientific research. Nowadays, the invention of the calculator has made this ability available even to a third grader. The same is in AI: it enhances the intellectual capabilities of a person, taking on the solution of previously unformalized tasks.

To illustrate the second approach, we can consider the story with an attempt to create a fifth-generation computer. In the mid-80s, Japan announced the launch of an ambitious fifth-generation computer project. The project was based on the idea of hardware implementation of the PROLOG language. However, the project ended in failure, although it had a strong influence on the development and distribution of the language PROLOG as a programming language. The reason for the failure was the hasty conclusion that one language (let it be quite universal) can provide a unique solution for all problems. The practice has shown that while a universal programming paradigm for solving all problems has not been invented and is unlikely to appear. This is due to the fact that each task is a part of the subject area, requiring careful study and a specific approach. Attempts to create new computer architectures continue and are associated with parallel and distributed computing, neurocomputers, probabilistic and fuzzy processors.

The works in the field of creating expert systems (ES) can be attributed to the third, most pragmatic direction in AI. Expert systems are software systems that replace a human specialist in narrow areas of intellectual activity that require the use of special knowledge. The creation of ES in the field of medicine (such as MYCIN) allows you to spread knowledge to the most remote areas. Thus, in combination with telecommunication access, any rural doctor can get advice from such a system, replacing him communication with a narrow request specialist.

In the USSR, AI found its supporters almost from the moment of its appearance. However, this discipline did not immediately receive official recognition. AI has been criticized as a sub-branch of cybernetics, considered "pseudoscience." Until some point in time, the shocking name of “artificial intelligence” also played a negative role. So, in the Presidium of the Academy of Sciences, there was a joke that “those who lack natural” are engaged in “artificial intelligence”. However, today AI is an officially recognized scientific direction in Russia, the magazines "Control Systems and Machines" and "AI News" are published, scientific conferences and seminars are held.

There is a Russian AI Association with about 200 members, the president of which is Doctor of Technical Sciences D.A. Pospelov, and the Honorary President is Academician G.S. Pospelov. There is the Russian Institute of Artificial Intelligence under the Council of the President of the Russian Federation on computer science and computer engineering. Within the RAS there is a Scientific Council on the problem of “Artificial Intelligence”. With

the participation of this Council, many books on the subject of AI, translations were published. The works of D.A. Pospelov, Litvintseva and Kandrashina are well known – in the field of knowledge representation and processing, E.V. Popov and Khoroshevsky – in the field of natural language processing and expert systems, Averkin and Melikhov – in the field of fuzzy logic and fuzzy sets, Stefanyuk – in the field of learning systems, Kuznetsov, Finn and Vagin – in the field of logic and knowledge representation.

1.2. The history of the development of artificial intelligence as a scientific direction

The idea of creating an artificial likeness of the human mind to solve complex problems and simulate mental ability has been in the air since ancient times.

In ancient Egypt, a “living” mechanical statue of the god Amun was created. In Homer, in the Iliad, the god Hephaestus forged humanoid creatures-automatons. In literature, this idea was played out many times: from Galatea Pygmalion to Pinocchio Papa Carlo. However, the founder of artificial intelligence is considered the medieval Spanish philosopher, mathematician and poet R. Lullius (c. 1235-c. 1315), who in the XIV century tried to create a machine for solving various problems on the basis of a universal classification of concepts.

In the XVIII century. G. Leibniz (1646-1716) and R. Descartes (1596-1650) independently developed this idea by proposing universal classification languages for all sciences. These ideas formed the basis of theoretical developments in the field of creating artificial intelligence (Fig. 2).

The development of artificial intelligence as a scientific direction became possible only after the creation of computers. This happened in the 40s. XX century At the same time, N. Wiener (1894-1964) created his fundamental work on a new science – cybernetics.

The term artificial intelligence was proposed in 1956 at a seminar with the same name at Stanford University (USA). The seminar was devoted to the development of logical rather than computational problems. Soon after the recognition of artificial intelligence as an independent branch of science, there was a division into two main areas: neurocybernetics and cybernetics of “black box”. And only in the present time, tendencies toward the unification of these parts again into a single whole became noticeable.

In the USSR in 1954, at the Moscow State University, under the guidance of Professor A.A.Lyapunov (1911-1973), the seminar "Automata and Thinking" began its work. This seminar was attended by major physiologists, linguists,

psychologists, and mathematicians. It is believed that it was at this time that artificial intelligence was born in Russia. As abroad, the directions of neurocybernetics and cybernetics of the "black box" stood out.

In 1956-1963 Intensive searches were conducted for models and algorithms of human thinking and the development of the first programs. It turned out that not one of the existing sciences – philosophy, psychology, linguistics – can offer such an algorithm. Then cybernetics suggested creating its own models. Various approaches have been created and tested.

The first research in AI related to the creation of a program for playing chess, since it was believed that the ability to play chess is an indicator of high intelligence. In 1954, the American scientist Newell decided to create such a program. Shannon suggested, and Turing clarified the method of creating such a program. The Americans Shaw and Simon, in collaboration with a group of Dutch psychologists from Amsterdam, led by de Groot, created such a program. Along the way, a special language IPL1 (1956) was created, intended for manipulating information in symbolic form, which was the predecessor of the Lisp language (MacCarthy, 1960).

However, the first program of artificial intelligence was the Logic-Theorist program, designed to prove theorems in the propositional calculus (August 9, 1956). The program for playing chess was created in 1957 (NSS – Newell, Shaw, Simon). Its structure and the structure of the Logic-Theorist program formed the basis for creating the GPS-General Problem Solving program. This program, analyzing the differences between situations and constructing goals, solves puzzles such as the Tower of Hanoi or calculates indefinite integrals. The EPAM program (Elementary Perceiving and Memorizing Program) is an elementary program for perception and memorization, conceived by Feigenbaum. In 1957, an article appeared by Chomsky, one of the founders of computer linguistics, on transformational grammars.

At the end of the 50-s a labyrinth search model was born. This approach presents the task as a certain graph reflecting the state space, and in this graph a search is made for the optimal path from the input to the resultant. A lot of work was done to develop this model, but the idea was not widely adopted in solving practical problems.

The beginning of the 60-s is the era of heuristic programming. Heuristics – a rule that is theoretically unreasonable, but allows you to reduce the number of searches in the search space. Heuristic programming - development of an action strategy based on well-known, predefined heuristics.

In the 60s, the first programs were created that work with queries in a natural language. The BASEBALL program (Green et al., 1961) answered queries about the results of past baseball matches; the STUDENT program (Bobrow, 1964) was available to solve algebraic problems formulated in English.

Great hopes were pinned on work in the field of machine translation, the beginning of which is associated with the name of the national linguist Belskaya. However, it took researchers many years to understand that automatic translation is not an isolated problem and requires the successful implementation of such a necessary stage as understanding.

Among the most significant results obtained by domestic scientists in the 60s, it should be noted the algorithm of M. Bongard – "Kora", modeling the activity of the human brain in pattern recognition.

In 1963-1970 methods of mathematical logic began to be connected to the solution of problems. A new approach to formal logic, based on reducing reasoning to contradiction, appeared in 1965. (J. Robinson). On the basis of the method of resolutions, which made it possible to automatically prove theorems in the presence of a set of initial axioms, the language Prolog was created in 1973.

In the USSR in 1954-1964 separate programs are created and the search for solutions to logical problems is investigated. In Leningrad (LBMI - Leningrad Branch of the Mathematical Institute named after V.A. Steklov) a program is being created that automatically proves the theorems (Aliyev, LBMI). It is based on the original reverse conclusion of S.Yu. Maslov, similar to the method of resolutions of Robinson.

In the years 1965-1980 a new science is developing – situational management (corresponds to the representation of knowledge in Western terminology). The founder of this scientific school is professor D.A. Pospelov developed special models for representing situations – representations of knowledge.

Abroad, research in the field of AI is accompanied by the development of a new generation of programming languages and the creation of increasingly sophisticated programming systems (Lisp, Prolog, Plannar, QA4, Macsyma, Refal, ATNL, TMS).

The results obtained begin to be used in robotics, when controlling robots, fixed or mobile, operating in real three-dimensional space. This raises the problem of creating artificial organs of perception.

Until 1968, researchers worked mainly with individual "microspaces", they created systems suitable for such specific and limited application fields as games, Euclidean geometry, integral calculus, the "world of cubes", processing simple and short phrases with a small vocabulary . Almost all of these systems used the same approach – the simplification of combinatorics, based on reducing the necessary enumeration of alternatives based on common sense, using numerical estimation functions and various heuristics. In the early 70s there was a quantum leap in research on artificial intelligence. There are two reasons for this [3].

■ Firstly. All researchers gradually realized that all previously created programs lack the most important – in-depth knowledge in the relevant field. The difference between an expert and an ordinary person is that the expert has experience in this field, i.e. accumulated knowledge over the years.

■ Secondly. A specific problem arises: how to transfer this knowledge to a program if its immediate creator does not possess this knowledge. The answer is clear: the program itself should isolate them from the data received from the expert.

Research to solve problems and understand the natural language has one common problem – the representation of knowledge. By 1970 it was

created many programs based on these ideas. The first one is the DENDRAL program. It is designed to generate structural formulas of chemical compounds based on information from a mass spectrometer. The program was developed at Stanford with the participation of Nobel laureate D. Lederberg. This program gained experience in the process of its own functioning. An expert in it laid many thousands of elementary facts presented in the form of separate rules. The system in question was one of the first expert systems and the results of its work are amazing. The system is currently being delivered to consumers along with a spectrometer.

In 1971, Terry Vinograd developed the SHRDLU system, which models a robot manipulating cubes. You can speak in English with the robot. The system is interested not only in the syntax of phrases, but also correctly understands their meaning thanks to semantic and pragmatic knowledge about its "world of cubes".

Since the mid-80s, the commercialization of artificial intelligence has been taking place abroad. Annual investments have been grown, industrial expert systems are being created. Interest in self-learning systems has been grown.

In our country, in 1980-1990 active research is being carried out in the field of knowledge representation, languages of knowledge representation, expert systems (over 300) are being developed. At the Moscow State University, the language REFAL is created. In 1988, the AAI - Association of Artificial Intelligence was created. Its members are more than 300 researchers. Association President - D.A. Pospelov. The largest centers are in Moscow, St. Petersburg, Novosibirsk.

1.3. History of Artificial intelligence research and basic concepts in this field

The history of artificial intelligence (AI) begins long before our era. Aristotle was the first to try to determine the laws of "right thinking" or the processes of irrefutable reasoning. Attempts to create mechanical calculating

devices in the Middle Ages greatly impressed contemporaries. The most famous machine, built in 1642 by Blaise Pascal. Pascal wrote that "an arithmetic machine produces an effect that seems closer to thinking than any animal action". The possibilities of the practical implementation of AI have appeared since the creation of electronic computers. At this time, a philosophical discussion began on the topic “Can a machine think?” The result of this discussion was a test proposed by Alan Turing in the 50s. XX century [4]. The test is as follows: There are two teletypes (at that time there were no other terminal devices). One of the teletypes is connected to the machine, the other to the apparatus, behind which the person is sitting. Several experts alternately engage in dialogue on each of the teletypes. If most experts cannot recognize the machine in one of the interlocutors within five minutes, the Turing test is considered to have passed successfully. The Turing test played a role in the development of artificial intelligence, including criticism of the test itself. Here you can draw an analogy with aviation. Good aircraft, according to the logic of the Turing test, should be considered those that are indistinguishable from birds to such an extent that even birds take them for their own.

The development of aviation began when the designers stopped copying the birds, and started aerodynamics, materials science and the theory of strength. Robotics became an industry after it stopped copying human anatomy. Similarly, the subjects of artificial intelligence gained the right to life after they stopped trying to build AI systems that think and act like people, and began to build systems that act and think rationally, i.e. achieving the best result.

The origin of research in the field of artificial intelligence (AI) took place in the following two directions: logical and neurocybernetic. Early research carried out in the 50-60s (N. Wiener, Turing, McCallock, Rosenblatt, Simon,

McCarthy, Slage, Samuel, Gelerner, N. Amosov): the appearance of the first developed LISP programming language for building AI systems; the appearance in the late 60s of integrated (intelligent) robots and first expert systems.

New boom in neurocybernetics is in the early 80s years (Hopfield model) [5]. The advent of logical programming and PROLOG language. 5th generation computer program. Strategic US computer initiative. AI research in the USSR and Russia.

From the very start of process modeling research thinking (late 40s) stood out two until recently almost independent directions:

- logical,
- neurocybernetic.

The first was based on the identification and application of intelligent systems of various logical and empirical techniques (heuristics) that a person

uses to solve any problems. Later with the advent of the concepts of “expert systems” (ES) in the beginning of 80s this direction resulted in a scientific and technological direction of computer science is “knowledge engineering”, which is engaged in the creation of so called "Knowledge Based Systems".

With this direction is usually associated the term **artificial intelligence.**

The second direction – neurocybernetic – was based on the construction of self-organizing systems consisting of many elements functionally similar to brain neurons. This direction began with the concept of the formal neuron McCallock-Pitts and Rosenblatt studies with various perceptron models - system learning pattern recognition. Due to relative successes in the logical direction of AI and low technological level in microelectronics, the neurocybernetic trend was almost forgotten from the late 60s to the early 80s, when new successful theoretical models (for example, “Hopfield model”) and super-large integrated circuits appeared.

The logical direction can be considered as modeling thinking at the level of consciousness or verbal or logical (focused) thinking [6]. Its advantages are:

- the possibility of a relatively easy understanding of the system;
- ease of displaying the process of reasoning of the system on its user interface in natural language or any formal language;
- attainability of the uniqueness of the system behavior in identical situations.

The disadvantages of the logical approach are:

- the difficulty and unnaturalness of the implementation of fuzzy characters (images);
- the difficulty (or even impossibility) of implementing an adequate behavior in the face of uncertainty (lack of knowledge, noisy data, inaccurate goals, etc.);
- the difficulty and inefficiency of parallelizing the decision process of tasks.

Neurocybernetic direction (or neuroinformatics) can be considered as modeling imaginative thinking and thinking on subconscious level (modeling of intuition, creative imagination, insight). Its advantages are the absence of flaws, inherent in the logical direction, and the disadvantages are the lack of it merits. In addition, in the neurocybernetic direction attracts the possibility (perhaps illusory), setting the basic very simple adaptation algorithms and structural features of artificial neural network, get a system that adjusts to behavior as you like complex and adequate to the task at hand. Moreover, its complexity depends only from quantitative factors of the neural network model. One more the advantage in the case of hardware implementation of a neural network is its survivability, i.e. ability to maintain acceptable solution efficiency tasks in case of failure of network elements. This is a property of neural networks.

Achieved through redundancy. In case of software implementation structural redundancy of neural networks allows them to work successfully in conditions of incomplete or noisy information [7].

What is the difference between the concept of "knowledge" from the concept of "data" or "information"? Recently, scientists have come to the conclusion that along with matter and energy information is objectively existing

an integral part of the material world that characterizes it orderliness (heterogeneity) or structure. Ability of the living creatures maintain their structure (orderliness) in a world where, probably, the desire to increase entropy (homogeneity) prevails, due to their ability to recognize the structure of the world and use recognition result (i.e. knowledge of the world) for purposes survival.

Thus, knowledge is perceived by a living being (subject) information from the outside world and, unlike "information knowledge is subjective. It depends on the characteristics of life experience. Subject, his history of relations with the external environment, i.e. from features of the process of learning or self-learning. At this level abstractions knowledge is unique and the exchange of knowledge between individuals is not can occur without loss unlike data in which encoded information (heterogeneity), and which can be transmitted from the transmitter lossless receiver (not considering distortion due to interference).

Knowledge is transmitted between subjects through any language representations of knowledge, the most typical representative of which is a natural language. By creating and using natural language, on the one hand, human strove to formalize and unify knowledge in order to transmit it in the same way to the huge number of people with different life experiences, and on the other hand, tried to enable the transfer of all the wealth of personal knowledge.

The first trend led to the emergence of various formalized special dialects of the language in various fields of knowledge (mathematics, physics, medicine, etc.).

The second led to the emergence of imaginative literature, at the heart of which lies the desire by means of language to cause associations (experiences) in the human brain, i.e. make him think and worry about the basis of knowledge gleaned from read, and their own knowledge.

By and large, all varieties of art are aimed at this – transfer knowledge through associations.

If we go from such a high level of abstraction (philosophical) to more down to earth, you can compare knowledge and data in their formalized form, which is usually done in the literature on artificial intelligence [8]. Then we can formulate the following differences of knowledge from data:

– knowledge is more structured;

- non-atomic elements of knowledge are not the most important in knowledge (as in the data), as the relationship between them;
- knowledge is more self-interpretive than data, i.e. in knowledge contains information on how to use them;
- knowledge is active in contrast to passive data, i.e. knowledge can generate actions of a system using them.

It should be noted that there is no sharp boundary between data and knowledge, because in the last twenty years, database management system developers increasingly makes them look like knowledge. An example is application of semantic networks (formalism for the representation of knowledge) for database design, the emergence of object-oriented databases, data stored procedures (this makes the data active to some extent), etc. Thus, the differences of knowledge from the data listed above, with the development of computer science tools are smoothed out.

In knowledge engineering, the following basic concepts about knowledge are distinguished, borrowed from semiotics – the science of sign systems:

- extension knowledge – superficial or specific knowledge,
- intensional knowledge – deep or abstract knowledge (knowledge of patterns),
- syntax – the structure of the sign system (data or knowledge),
- semantics – the meaning of the sign system (knowledge), i.e. equivalent its representation in another paradigm of knowledge representation (internal)
- pragmatics – goals associated with the sign system (for example, goals or purpose of a sentence in a natural language – team, question, explanation, etc.).

1.4. The main directions of research in the field of artificial intelligence

Currently, AI is a rapidly developing and highly branched scientific field. In computer linguistics alone, more than 40 conferences are held annually in the world. Almost every European country, as well as the United States, Canada, Japan, Russia, Southeast Asia, regularly holds national AI conferences. In Russia, this event is held every two years under the auspices of the Russian Association for AI (RAAI). In addition, the International Joint AI Conference (IJAIC) is held every two years. More than 3 thousand periodicals publish scientific results in this field. There is no complete and strict classification of all directions of AI; an attempt to classify the tasks that AI solves is presented in

Fig. 1.2. According to the classification of D.A. Pospelov in AI, there are two dominant approaches to research in the field of AI: neurobionic and information.

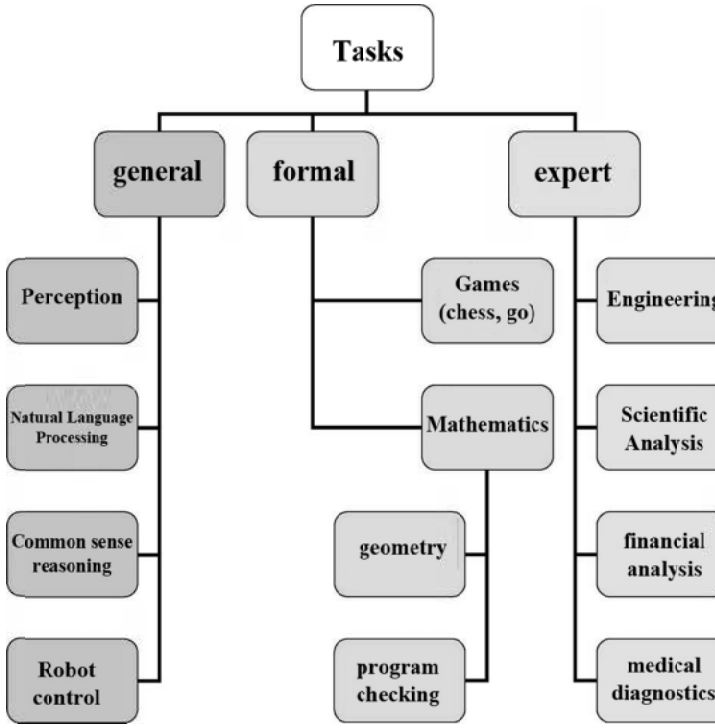


Fig. 1.2 – The tasks of AI

Supporters of the first approach set themselves the goal to artificially reproduce the processes that take place in the human brain. This direction is at the intersection of medicine, biology and cybernetics. At the same time, the human brain is studied, ways of its work are revealed, technical means are created for repeating biological structures and processes occurring in them.

Supporters of the second approach rely on studies of Intelligent Information Systems; intellectual problem solving programs; knowledge-based systems.

Questions for self-control

1. What is artificial intelligence?
2. What scientific areas does artificial intelligence interact with?
3. Describe the approaches to understanding the subject of artificial intelligence as a scientific discipline.
4. Describe the current state of AI in Russia.
5. Describe the "pre-computer" stage of development of artificial intelligence.
6. Describe the development of artificial intelligence in the 70s. XX century
7. Describe the development of artificial intelligence in the 80s. XX century
8. Describe the main tasks of artificial intelligence.
9. What sections highlight in the field of artificial intelligence?
10. Give evidence of the possibility of modeling human thinking.
11. What is the reason for the transition to the problem of the influence of intellectual tools on society?
12. What is the reason and how can the problem of security of artificial intelligence systems be solved?

CHAPTER 2. PROLOG PROGRAMMING BASICS

2.1. Prolog as a declarative language

The development of the Prolog language began in 1970 by Alan Kulmeroe and Philippe Roussel [9]. They wanted to create a language that could draw logical conclusions based on a given text. The name Prolog is short for "Programming in logic". This language was developed in Marseille in 1972. Prolog is a programming language that is based not on an algorithm, but on predicate logic.

If a program in an algorithmic (procedural) language is a sequence of instructions that are executed in a given order, then the program in Prolog contains only a description of the task, and the Prolog machine searches for a solution, guided by this description. For example, there is the logical task of covering a chessboard with a knight's move. In any algorithmic language, the solution of this problem requires the construction of a rather complex algorithm.

At the Prolog, it is enough to describe the rules by which the horse walks, after which the Prolog will independently find the solution. The flip side of such simplicity is the resource consumption of programs. For example, in another popular task of placing eight queens on a chessboard that do not beat each other, the complete decision tree has 648 vertices. Obviously, finding a solution in such a tree will take an unacceptably long time. Programming in Prolog consists of the following steps:

- **announcements of some facts about objects and relations between them;**
- **definition of some rules about objects and relations between them;**
- **wording questions about objects and the relationship between them.**

2.2. Concept of predicate

The main element of the Prolog program is the predicate. From a mathematical point of view, a predicate is a function that returns a binary value (true or false). In Prolog, the predicate denotes the relationship between objects, which can also be true. Consider the notion of a predicate in Prolog by the example of the celebrity family of Pugacheva – Kirkorov, although now the former [10]. First, we write the parent-child relationship. In the Prolog syntax, the expression “Boris is Alla’s parent” is as follows: parent (boris, alla). Here parent is the name of the predicate, and boris and alla are the arguments. The arguments boris and alla are constants, so they are written in lowercase letters.

With a capital letter in the Prolog, variables begin. The dot means the end of the predicate, as well as the end of the sentence in natural language. We will also record parental relationships for other family members:

parent(bedros, filipp).

parent(kristina, denis).

parent(edmuntas, kristina).

parent(vladimir, denis).

parent(alla, kristina).

Now we give the concept of "spouse":
spouse (filipp, alla).
spouse (vladimir, kristina).

The resulting set of predicates forms a knowledge base about the star family. Compare with how the same data will be presented in a relational database.

Parents table

Parent	***Child***
Boris	*Alla*
Bedros	*Filipp*
Alla	*Kristina*
Edmuntas	*Kristina*
Kristina	*Deni*

Spouse table

S1	***S2***
Alla	*Filipp*
Kristina	*Vladimir*

As you can see, there is a similarity at the level of data representation. But this is where it ends. But at the level of data extraction there is a big difference. In order to extract knowledge, a relational database needs to create a query to

select data, for example, in SQL. Suppose we want to know who Christina's parents are. We must write a query of the following form:

SELECT Parent FROM Parents WHERE Child = "Kristina"

In the Prologue, the request for knowledge extraction is described by the same predicates as this knowledge is presented. If we substitute the following predicate to the Prolog (target predicate):

parent (alla, kristina).

This target can be read as follows: Is Alla the parent of Christina? By comparing this goal with the contents of the knowledge base, Prolog will establish that this statement is true and report it. The above SQL query (Who is Christina's parent?) In the Prolog is as follows:

parent (X, kristina).

Here X is a variable to which the desired values should be assigned. A variable in Prolog is an analog of a pronoun or a question word. From the above knowledge base, Prolog will extract two answers:

X = alla

X = edmuntas

We can formulate the question as follows: Does Christina have parents?

parent (_, kristina).

The Prologue will give the answer: Yes.

A variable starting with an underscore is called an anonymous variable and can take on any value (similar to someone). For more complex queries in databases, you need to create views or create subqueries in SQL. In Prolog, everything is much simpler. Let's find whose granddaughter is Christina. The request will look like this:

parent (X, kristina), parent (Y, X).

This entry means: Find Y, the parent of X, who, in turn, is the parent of Christina. The comma in the prologue is identical to the AND union or conjunction. In response to such a request, Prolog will give the following answer:

X = alla

Y = edmuntas

You can issue the following request:

parent (alla, _).

Now an anonymous variable is used as the second argument. This request can be read as follows: Does Alla have children? The prologue will give the answer: Yes.

Since a predicate is a binary function that can return true or false (a statement that can also be true or false), the result of the program on Prolog is the determination of whether the target is true or false. Assigning values to variables, outputting results, etc. – these are just side effects.

Thus, the program on the Prolog consists of predicates. The Prolog program and knowledge base are synonyms. The goal is also formulated in the form of predicates. Running a program on Prolog is a resolution of purpose.

2.3. How does the Prolog interpreter work?

The process of finding a solution in the Prolog is to compare the target predicate with the knowledge base predicates [10]. This process is called unification.

Let the Prolog system be presented with the goal from the previous subsection (we want to find whose granddaughter is Christina):

parent (X, kristina), parent (Y, X).

The prologue extracts from it the first subgoal of parent (X, kristina) and begins to compare it with the knowledge base (to carry out unification). The knowledge base is repeated below:

parent(boris, alla).

parent(bedros, filipp).

parent(edmuntas, kristina).

parent(alla, kristina).

parent(kristina, denis).

First, the first predicate and subgoal are compared:
parent (*boris, alla*) and parent (*X, kristina*)
The first argument *bori*s maps to the variable *X.*

It should be borne in mind that the Prolog distinguishes between the states of the variables *free* and *bound*. ***If both variables are connected, then during unification they are compared. If one of them is free, then assignment occurs. Reassignment of values to variables is not allowed.*** This significantly distinguishes Prolog from other languages.

Thus, the variable *X*, which is still free, is assigned the value *boris*. After that, the second arguments, *alla* and *kristina*, are unified. Since these are constants, and *alla* is not equal to *kristina*, the unification of the predicate *parent (boris, alla*) and the subgoal of parent (*X, kristina*) ends in failure.

Since there are several instances of the *parent* predicate in the knowledge base, such a predicate is called *non-derministic*. If there is one predicate, then it is called *deterministic*. In the case of an ambiguous predicate, after a failure, a

rollback is performed – transition to the next instance of the predicate. At the same time, the assignment of a value to variables is also canceled, if any. Then the predicate unification is performed.

parent (bedros, filipp) and *parent (X, kristina)*

Obviously, the result of unification will be the same, *failure*. When you roll back to the next predicate *parent (edmuntas, kristina*), the picture will be different: *X* will be assigned the value *edmuntas*, and matching the second arguments will also be successful, since *kristina = kristina*. Thus, the first subgoal will be completed. The prolog remembers which instance of the predicate worked and sets the rollback pointer to the next predicate:

parent(boris, alla).

parent(bedros, filipp).

parent(edmuntas, kristina).

> parent(alla, kristina).

parent(kristina, denis).

after which it will go to the second subgoal of *parent (Y, X*), where *X = edmuntas*, i.e. The prologue sets itself the following subgoal:

parent (Y, edmuntas).

In search of Edmuntas' parent, Prolog again begins to unify this predicate from the beginning of the knowledge base, starting with *parent (boris, alla*). It is easy to see that this time the search of all predicates will fail, i.e. subgoal *parent (Y, edmuntas*) did not give a positive decision. In this case, the Prolog rolls back to the previous subgoal (moves backward through the list of subgoals) and tries to find an alternative solution for *parent (X, kristina*). In this case, *X* again becomes a free variable, and Prolog returns to the rollback point:

parent(boris, alla).

parent(bedros, filipp).

parent(edmuntas, kristina).

> parent(alla, kristina).

parent(kristina, denis).

that is, to the predicate *parent (alla, kristina*), matching it with the subgoal of *parent (X, kristina*).

Now *X* is assigned the value *alla*, the rollback pointer is set to the predicate *parent (kristina, denis*), and again the next subgoal of *parent (Y, X*) is executed, where *X = alla*. The prologue again begins to unify the subgoal of *parent (Y,*

alla) with the knowledge base, starting with the first predicate. In the first predicate, the constant *boris* and the free variable *Y* are unified. Assignment *Y = boris* occurs, then the second arguments are compared. Since *alla = alla*, matching succeeds. Thus, a solution was found: Kristina is Boris's granddaughter. Note that the failure that befell us in the search for paternal grandfather Christina is connected only with the incompleteness of the knowledge base.

So, the Prolog interpreter automatically searches for a solution. The search engine is implemented using rollback after failure. Rollback occurs to the next instance of an ambiguous predicate. The implementation of the program on the Prolog (goal resolution) is to unify the goal with the knowledge base [7].

2.4. Facts and Rules in Prolog

The above described request establishing a grandparent-grandson relationship may be required more than once in the future. In this regard, it is advisable to remember it for future use in other queries. In the Prolog knowledge base you can store not only facts, but also rules, i.e. conditional relationship. A grandparent-grandson relationship can be written as follows:

grandparent (X, Y) if parent (X, Z), parent (Z, Y).

Read this as follows: *X* is the progenitor of *Y*, if *X* is the parent of *Z* and *Z* is the parent of *Y*. The predicate *grandparent (X, Y)* is called the heading of the rule, and the expression to the right of *if* is the body of the rule.

Note: The synonym for the "if" link in the rule is ": -".

Thus, as in databases, in the Prolog knowledge base in the form of facts we store primary knowledge, and the derivatives of them are written in the form of rules, which we refer to in the same way as facts.

Fact is what is known.

A rule is a way of generating new facts based on existing ones.

For family relationships, we can establish many rules, eliminating the need to introduce additional facts, for example, who is a brother, nephew, etc. The rule determining the relation of brother (sister):

sibling (X, Y): - parent (Z, X), parent (Z, Y), X <> Y.

Comparison predicate *X <> Y* is needed to resolve collisions like "my father's son, but I'm not a brother." The rule defining an uncle type relationship is as follows:

uncle (X, Y): - parent (Z, Y), sibling (X, Z).

When during the resolution of a goal the Prolog encounters not a fact, but a rule, then at first it unifies the heading of the rule, i.e. compares related

variables and assigns values to free variables. If the arguments are successfully unified, the Prolog substitutes the values of the arguments from the header into the first predicate in the body of the rule and sets this predicate as a subgoal, which it starts to unify with the knowledge base. In case of successful resolution of this subgoal, the Prolog proceeds to the next condition of the rule. If unification of this condition predicate results in failure, then Prolog rolls back to the previous condition of the rule.

This rollback occurs only if this previous predicate is ambiguous. Let us illustrate this with an example. We set ourselves the goal of finding out who is the progenitor of Christina:

grandparent (Who, kristina).

Having received such a goal, Prolog begins to unify it with the rule:

grandparent (X, Y): - parent (X, Z), parent (Z, Y).

The variable *Who* in the target predicate is a free variable and its unification with the *X* variable in the rule header will always be successful. It should be noted that in Prolog all variables are local, i.e. exists only inside the rule. We could use *X* instead of *Who*, and these would be different variables that would be unified in exactly the same way. If you need to create global variables, use dynamic facts that are created by the assert predicate and destroyed by the *retract* or *retractall* predicate.

Then the constant *kristina* is unified with the variable *Y*. Since the variables in the rule header are always free first, the assignment is made: *Y = kristina*. Since the unification of the rule heading was successful, the Prolog delves into the body of the rule and sets the first predicate of the rule body as a sub-goal, substituting the variables if they are related:

parent (X, Z).

The variables *X* and *Z* are free, so the unification of this subgoal with the first predicate parent from the knowledge base will be successful:

X = boris, Z = alla

After that, the Prolog proceeds to the second predicate in the rule, substituting the values of *X* and *Y*:

parent (alla, kristina).

The resolution of this subgoal gives the truth, and the values of the variables assigned during unification are returned by the Prolog

Who = X = boris.

Thus, during the resolution of the main goal, the Prolog independently sets its own goals, guided by the rules in the knowledge base.

Consider another example:

grandparent (Who, denis).

Similar to the previous example, Prolog unifies the *grandparent (X, Y*) rule header and assigns the value *Y = denis*. Going deeper into the body of the rule,

the Prolog forms the subgoal of *parent (X, Z)*. This subgoal returns, as in the previous example,

X = boris, Z = alla

The prolog proceeds to the second predicate of the rule, substituting the values of the variables in *parent (Z, Y*):

parent (alla, denis).

Trying to unify this subgoal, Prolog maps the variables (*alla, denis*) to the first instance of the parent predicate, fails, rolls back to the next instance, and so on. Since the fact of *parent (alla, denis)* is not in the knowledge base, the resolution of this subgoal is unsuccessful. Therefore, the unification of the first predicate of the rule with the values *X = boris, Z = alla* is incorrect. ***Therefore, the Prolog performs a rollback to the previous condition of the rule and tries to find another solution for the subgoal parent (X, Z*).** In this case, the assignment of variables is canceled (*X = boris, Z = alla*). Variables *X* and *Z* become free again. Note that rollback here is possible only on an ambiguous predicate. If in the chain of predicates inside the rule there are both unambiguous and ambiguous predicates, then rollback after failure is performed to the nearest ambiguous predicate. At the first unification of this predicate, Prolog set a rollback pointer to the following instance of parent:

parent (boris, alla).
> parent(bedros, filipp).

parent(edmuntas, kristina).

parent(alla, kristina).

parent(kristina, denis).

When rolling back, the Prolog proceeds to unify this fact and sets the rollback pointer to the third instance:

parent (boris, alla).
parent (bedros, filipp).
> parent (edmuntas, kristina).
parent (alla, kristina).
parent (kristina, denis).

After unifying the second predicate parent with the subgoal *parent (X, Z),* the Prolog will assign values to the variables:

X = bedros, Z = filipp

and again goes to the second predicate *parent (Z, Y*):

parent (bedros, denis).

Obviously, the resolution of this subgoal also fails. The prolog again rolls back to the previous predicate of the rule and to the third instance of the fact *parent*. Only the unification of the fourth fact will be successful:

parent (alla, kristina),

whereby we get

Who = X = alla.

This is how the Prolog interpreter works if there are rules in the knowledge base.

Thus, ***Fact is knowledge based on constants (immutable knowledge). Rules are knowledge that is derived from facts.***

The set of facts and rules does not contain an algorithm.

Rules and facts exist independently of each other.

The combination of rules for the conclusion of the result occurs during the resolution of the goal. Variables in the rule header exist only within this rule.

When rolling back inside a rule, it goes to the previous ambiguous predicate in the rule.

2.5. Recursions in Prolog

If we are interested in whether *X* is the ancestor of *Y*, then we should set goals in sequence:

parent (X, Y).

grandparent (X, Y).

grandgrandparent (X, Y).

grandgrandgrandparent (X, Y).

and so on, where:

grandgrandparent (X, Y): - parent (X, Z), grandparent (Z, Y).

grandgrandgrandparent (X, Y): - parent (X, Z), grandgrandparent (Z, Y).

Instead, the Prolog allows you to write this rule as follows:

predecessor (X, Y): - parent (X, Y).

predecessor (X, Y): - parent (X, Z), predecessor (Z, Y).

This is where the recursive call of the predicate *predecessor* takes place. Recursion in Prolog is a powerful tool for building very compact and efficient programs.

Consider using recursion as an example of calculating factorial.

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$$

Recursive definition of factorial: $0! = 1;\ n! = n \cdot (n-1)!$ A prolog program that implements factorial calculation will look like this:

f(0,1).

f(N,F) :- N1=N-1, f(N1,F1), F=F1·N.

Please, note that the program, in essence, consists of nothing more than a recursive mathematical description of the factorial function given above. Run the program in trace mode and set the target *f (3, X*). The prologue begins to match the target predicate with the knowledge base (CALL means calling the predicate, RETURN – shutting down the predicate, FAIL – failure, REDO – rollback):

CALL	*f (3, X)*	the target maps to *f (0,1*)
FAIL		failure because 3 ≠ 0
REDO	*f (3, X)*	Rolls back to the next instance of *f ()*
	N = 3,	We enter the body of the rule. Assign *N = 3*
	N1 = 2	Find *N1 = 2*
	f (2, X)	The prologue sets a subgoal for itself
CALL	*f (2, X)*	which maps to *f (0,1)*
FAIL		failure because *2 ≠ 0*
REDO	*f (2, X),*	Rolls back to the next instance of *f ()*
	N = 2,	Again we enter the body of the rule. Assign *N = 2*
	N1 = 1	Find *N1 = 1*. This is another *N1*
	f (1, X)	The prologue sets a subgoal for itself *f (1, X)*
CALL	*f (1, X)*	which maps to *f (0,1)*
FAIL		failure because 1 ≠ 0
REDO	*f (1, X)*	Rolls back to the next instance of *f ()*
	N = 1	Again we enter the body of the rule. Assign *N = 1*
	N1 = 0	Find *N1 = 0.*
	f (0, X)	The prologue sets a subgoal for itself *f (0, X)*
CALL	*f (0, X)*	which maps to *f (0,1)*
RETURN	*X = 1*	Successful. Return from lower recursion level
	F = 1	Multiplication 1 by 1 (factorial from 1)
RETURN	*X = 1*	Return from the next recursion level
	F = 2	Multiplication 2 by 1 (factorial from 2)
RETURN	*X = 2*	Return from the next recursion level
	F = 6	Multiplication 3 by 2 (factorial of 3)
RETURN	*X = 6*	Return from program

You may notice that the logic of the factorial calculation program depends on the location in the text of the predicate that determines the way out of

recursion. Unification is performed in the order of the predicates in the program text, and if the predicate *f (0,1*) is put at the end, exit from the recursion will be impossible. Thus, the declarative nature of the Prolog is not absolute for the convenience of its use. A variant of the program in which predicates can be arranged in any order is presented below:

f (N, F): - N> 0, N1 = N-1, f (N1, F1), F = F1 · N.
f(0,1).

We also note that in this recursive predicate there are actions in the body of the rule after the recursive call. This is called tail recursion.

The violation of tail recursion, also called non-tail recursion, requires remembering the entire environment of the recursive call (and not just the result), therefore, it leads to large memory costs. There are techniques for eliminating non-tail recursions [10].

The following is an example of calculating factorial without non-tailing recursion.

```
f (N1, N, F1, F): -          % if N1! = F1, then N! = F
  N2 = N1 + 1,
  F2 = F1 * N2,              % (N1 + 1)! = F2
  f (N2, N, F2, F),          % if N2! = F2, then N! = F
  f (N, N, F, F).            % Condition for exiting recursion
```

The goal to calculate 3! is as follows*: f (0,3,1, F).*

Recursion in the Prolog is not always used to perform multiple repetitive actions. Let us recall the knowledge base of the "star" family, in which there is a predicate describing marital relations, in particular, *spouse (filipp, alla*). Marital relations, unlike parental relations, are symmetrical. Philip is husband of Alla, just as Alla is the wife of Philip. Moreover, in the predicate, the position of the arguments is fixed. In other words, if the fact in the knowledge base is written as follows: *spouse (filipp, alla*). and the target predicate is: *spouse (alla, filipp*). then the result will be negative. In order to show Prolog that this predicate is symmetric with respect to arguments, we can apply the rule:

spouse(X,Y) :- spouse(Y,X).

As an example, consider a known collision. Two families lived in the village: a mother with a daughter and a father with a son. Let's give them names. Let the mother and daughter be called Maria and Dasha, and father and son Oleg and Sergey, respectively. The first letters of the names will tell us who is who, otherwise we will get confused. In the knowledge base, these facts will be reflected in the following form:

parent (oleg, sergei).
parent (maria, dasha).

Due to the vicissitudes of fate, Oleg married Dasha, and Maria married Sergei:

spouse (oleg, dasha).

spouse (sergei, maria).

To symmetry marital relations, we introduce the rule:

spouse (X, Y): - spouse (Y, X).

The manners in the village are simple, therefore, the father's wife should be called mom, and mother's husband - father. The rules for this will be as follows:

parent (X, Y): - spouse (X, Z), parent (Z, Y).

Let's try to find a grandson, Sergei. We set a goal

grandparent (sergei, Who).

The prolog finds the grandparent (*X, Y*) rule: - *parent (X, Z), parent (Z, Y)* and sets a subgoal from the first predicate in the body of this rule:

parent (sergei, Z).

Sergei has no children, so Prolog refers to the *rule parent (X, Y): - spouse (X, Z), parent (Z, Y)* and sets a goal

spouse (sergei, Z).

The prologue gives the result *Z = maria.* Second parent rule predicate

parent (maria, Y).

The value *Y = dasha* is returned. That is, Sergei as the husband of Mary is considered the father of Dasha. Now the Prolog proceeds to the second predicate in the grandfather rule:

parent (dasha, Y).

Dasha also has no children, so Prolog refers to the rule *parent (X, Y): - spouse (X, Z), parent (Z, Y)* and first tries to find a wife for Dasha:

spouse (dasha, Z).

Result: *Z = oleg.* The second predicate of this rule parent (*oleg, Y*) will give *Y = sergei.* That is, Sergei has a grandson to himself! The knowledge base we created faithfully reflects this collision.

2.6. Clipping in Prolog

From all of the above, we can make a fair conclusion that the Prolog interpreter, fulfilling the goal resolution, always makes a complete walk around the decision tree. The descent down the tree corresponds to a deepening in the body of the rule, returning up and moving to the next branch – rollback after failure. Consider the concept of clipping on a specific example [10].

Let us invited to the village. At the same time, they usually give detailed instructions on how to drive and find a specific house. For example, the instruction we received is as follows:

1. Enter the village of Vasino (and maybe Vanino, it is written illegibly).
2. Turn right.
3. Get to the well.
4. The desired house of red brick – the first from the well on the right side.

We write the rule for finding a house ("dacha") with the Prolog predicates:

```
dacha1 (X): -
                enter_village (X),
                _a_house.
find_a_house: - turn_right,
                meet_mine
                see_a_red_brick_house.
enter_village (vasino).
enter_village (vanino).
```

We begin the search (Fig. 2.1).

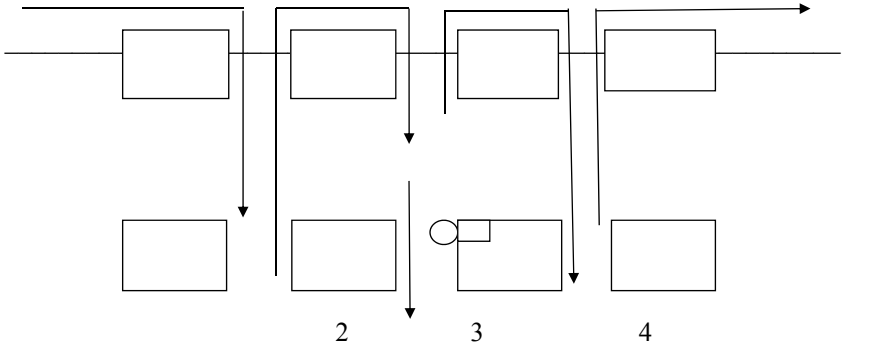


Fig. 2.1 – Illustration of clipping in Prolog

We drive into the village and turn right (arrow 1). We drive to the end of the village and do not find a well. We carry out the rollback, that is, we return to the highway and go to the next turn to the right (arrow 2), we reach the well and see that the nearest house is made of white brick. We go to the end of the street and see that there are no more wells (arrow 3). We roll back onto the highway and go to the next turn (arrow 4), drive through the entire street and roll back, since there are no wells here (arrow 5). From this we conclude that the name of the village we read incorrectly, and the house should be in the next village. In the village of Vanino, then we repeat all over again.

Clipping reduces the number of run branches of the decision tree. If the owner of the house ("dacha") gives us important additional information that there is only one well in each village (and if the post office or the police are set as a guide, we can guess for ourselves that they can be in the village in only one instance), we won't perform the actions indicated by arrows 3 and 4, and we immediately understand that we've got into the wrong village.

Clipping is a predicate that is indicated by an exclamation mark and is inserted into the rule. Clipping occurs after the Prolog passes through it. Clipping destroys the rollback indicators established during the unification of this rule, i.e., makes the previous predicates of this rule unambiguous.

If we want to indicate in the house search program that the well is the only one in the village, we must cut off after the well found:

```
dacha2(X) :-
        enter_village(X),
        find_house.
        find_a_house :- turn_right,
          meet_mine, !,
          see_a_red_brick_house.
enter_village(vasino).
enter_village(vanino).
```

As soon as the Prolog reaches clipping, the entire predicate *find_house* becomes unambiguous, therefore, further failure in the predicate *see_a_red_brick_house* will result in the predicate *find_house* also failing. If at the same time the predicate *enter_village* is ambiguous (as shown in the program), then we have a chance to find a house in another village, since clipping in the rule *find_house* does not affect the properties of the predicate *dacha* and the rollback pointers set in it.

If the predicate *enter_village* is unique, then we will have to return home. We did not find a house. If the house near the well is still made of red brick, then the presence of clipping will not affect the search result. Thus, clipping should always be included in the program on Prolog whenever we know for sure that the solution is either unique or does not exist.

Note. If the house search program looks like this:

```
dacha3(X) :-
        enter_village(X),
```

```
        turn_right,
        meet_mine, !,
        see_a_red_brick_house.
enter_village(vasino).
enter_village(vanino).
```

then clipping inserted at the specified location will result in all previous predicates in the rule *dacha* becoming unambiguous, including *enter_village*. In other words, we will be prohibited from looking for a house in the next village.

2.7. Prolog Lists

All the variables that we worked with earlier were scalars. Now consider the data aggregates in the Prolog. One of them is a list. A list is an ordered sequence of elements of the same type. List items are enclosed in square brackets and separated by commas:

```
[ 1, 2, 3, 4, 6, 7, 8] –                    integer
[mon,tue, wed, thu, fri, sat, sun] –          symbol
["Ivanov "," Petrov "] –                     string
[1.5, 2.22, 0.001, 0] –                      real
[] –                                         empty list
```

Lists are declared as follows:

```
domains
    sym = symbol *                  % list of symbol values
    intlist = integer *             % integer list
    realist = real *                % list of real numbers
```

The only operation allowed on the list is clipping the head from the tail, and this operation can only be "placed" in the predicate arguments. Let's write some useful predicates for working with lists. In particular, how to determine whether a variable is a list item? We will break the list into the head and tail. The desired value is either in the head of the list or in the tail:

```
member (H, [H | _] ).
member (X, [H | T] ) :- member (X, T).
```

Using the above technique, you can search not only for one element, but also for more (for example, a pair of consecutive values in the list)

memb2(H1, H2, [H1, H2 | _]).

memb2(H1, H2, [H | T]) :- memb2(H1, H2, T)

To include a value in the list (it's easiest to think of), you can make the following rule: *incl (H, T, [H | T*]).

To exclude a value from the list is somewhat more difficult:

excl (H, [H | T], T). % exclude from the head
excl (X, [H, T], [H | TT]): - excl (X, T, TT). % exclude from the tail

The following are other examples of list predicates:

1. The program for displaying list items one at a time:

print_list ([]). % output from recursion
print_list ([H | T]): - write (H, ""), % list head and space output
print_list (T). % tail output

If you set a goal

printlist (["I," "remember," "wonderful," "moment"]).

then the result will be as follows:

I remember wonderful moment

2. The same program, but listing in the reverse order:

print_inverse ([]). % exit from recursion
print_inverse ([H | T]): - print_inverse (T), % tail output
write (H, ""). % output of list head and space

Goal:

write_inverse (["I," "remember," "wonderful," "moment"]).

Result:

moment wonderful remember I

3. A program that calculates the sum of the elements of a list:

sum ([], 0). % exit from recursion,
sum ([H | T], S): - sum (T, S1), % S1 - the sum of the elements in the tail the list
S = S1 + H.% S - it remains to add only the head to the sum

Goal:

sum ([1,2,3,4,5,6], S), write ("Sum =", S).

Result: *Sum = 21*

4. A program that counts the number of list items:

count ([], 0) . % exit from recursion,
count ([H | T], S): - count (T, S1), % S1 - element counter

at the tail of the list

S = S1 + 1. % S - it remains to add one to the sum

Goal:

count ([1,2,3,4,5,6], S), write ("Number =", S).

Result: *Number = 6*

2.8. Example: Solving the logical problem of a wolf, goat and cabbage

Consider the well-known logical problem of the wolf, goat and cabbage [10]. The farmer must transport the wolf, goat and cabbage to the other side of the river. The carrying capacity of the boat is such that at one time you can take something on board: either a wolf, or a goat, or cabbage. In the presence of an old man, nobody eats anyone. If he goes away, the wolf will eat the goat, and the goat will eat cabbage.

To solve this problem, we organize two lists. One list will reflect the contents of the left bank, the second - the right. Initially, all are on the left bank. The list of the left bank: [*wolf, goat, cabbage*], the list of the right bank is empty: []. Define predicates describing the problem.

```
stuff (wolf).                                   % transfer of cargo
    stuff (goat).
    stuff (cabbage).
    / * conflict conditions * /
    conflict (X): - member (wolf, X), member (goat, X).
     conflict (X); - member (goat, X), member (cabbage, X).
    / * Predicates describing boat movements:
    From the left bank to the right * /
    go_right ([], _).          % termination condition (left bank list is empty)
    go_right (L, R): -
    stuff (X),                                  % select the cargo,
    member (X, L),                              % which is on the left bank
    excl (X, L, LL),                % exclude from the list of the left bank
    not (conflict (LL)),            % on the left bank there is no conflict
    incl (X, R, RR),                % include in the list of right bank
    write (LL, "-", X, "->", R),                % display a message
    go_left (LL, RR).                           % row back
    / * Movement to the left is possible in two versions. If no conflict arises on
the right bank, then the farmer travels alone * /
        go_left (L, R): - not (conflict (R)),
```

```
        write (L, "<-------", R),                    % display a message
        go_right (L, R).                             % call the predicate move right
        / * If a conflict arises on the right bank, someone needs to be taken back.
This is the only clue we tell the program. The rest of the Prolog will look for a
solution on its own * /
    go_left (L, R): -
        stuff (X),                                   % select the cargo,
        member (X, R),                               % which is on the right bank
        excl (X, R, RR),                             % exclude from the list of right bank
        not (conflict (RR)),                         % on the right bank there is no conflict
        incl (X, L, LL),                             % include in the list of the left bank
        write (L, "<-", X, "-", RR),                 % display a message
        go_right (LL, RR).                           % row back
        The goal challenge should be as follows:
          go_right ([wolf, goat, cabbage], []).
```

If you run this program, you will quickly find that on the first trip the old man will take a goat to the right bank. The second trip - the wolf. You can't leave a wolf with a goat on the right bank, so he will take the first cargo that comes in back, which will be the wolf. And so it will carry him endlessly.

The disadvantage of this program is that the farmer does not remember who he just brought. To strengthen his memory, add the name of the last transported cargo to the predicates *go_left* and *go_right*. The final version of the program is as follows (the changes are shown in bold):

```
    / * The task of the wolf, goat and cabbage * /
      domains          % section is required for the PDC Prolog. Is not needed in
SWI-Prolog
      stuff = wolf; goat; cabbage; nil          % create your data type (nil – empty)
      list = stuff *                                          % data type list
     predicates                          % section is required for the PDC Prolog.
    Is not needed in SWI-Prolog
        member (stuff, list)
        incl (stuff, list, list)
        excl (stuff, list, list)
        conflict (list)
        go_right (list, list, stuff)
        go_left (list, list, stuff)
    clauses

        stuff(wolf).

        stuff(goat).
```

```
stuff(cabbage).
member (H, [H | _] ).
member (X, [H | T] ) :- member (X, T).
incl (H, T, [H | T]).
excl (H, [H | T], T ).
excl (X, [H,T], [H | TT] ) :- excl (X, T, TT).
conflict(X): - member(wolf, X), member(goat, X).
conflict(X); - member(goat, X), member(cabbage, X).
go_right( L, R,Last) :-

stuff (X),                          % select the cargo,
   X <> Last,                       % but not the one that was just transported
    member (X, L),                  % which is on the left bank
    excl (X, L, LL),                % exclude from the list of the left bank
   not (conflict (LL)),             % there is no conflict on the left bank
   incl (X, R, RR),                 % include in the list of right bank
    write (LL, "-", X, "->", R),    % display a message
   go_left (LL, RR, X).             % row back
    / * If there is no conflict on the right bank, then one goes * /
   go_left (L, R, Last): - not (conflict (R)),
    write (L, "<-------", R),       % display a message
   go_right (L, R, nil).            call the predicate move right
% nil means they didn't carry anything
   / * If there is a conflict on the right bank, you need to take someone back,
but not the one you just brought * /
    go_left (L, R, Last): -
    stuff (X),                      % select the cargo,
   X <> Last,                       % is not the one that was just transported
   member (X, R),                   % which is on the right bank
    excl (X, R, RR),                % exclude from the list of right bank
   not (conflict (RR)),             % on the right bank no conflict
```

incl (X, L, LL), % include in the list of the left bank

write (L, "<-", X, "-", RR), % display a message

go_right (LL, RR, X). % row back and remember that we drove *X*

goal

*go_right (([wolf, goat, cabbage], [], **nil**).*

/ * End of program * /

Questions for self-control

1. By whom and when did the development of the Prolog language begin?
2. What does the programming language Prolog mean?
3. What is knowledge? What types of knowledge distinguish?
4. Give a classification of knowledge.
5. What is the main element of the Prolog program?
6. Explain the concept of a predicate, give examples.
7. What is logic as a science? Give an object definition of logic?
8. What is a concept, scope of a concept and addition to the scope of a concept?
9. What distinguish types of concepts?
10. Describe the essence of the basic techniques for understanding concepts.
11. What is the purpose of dividing the concept? What are the rules of logical division?
12. Give examples of complex judgments.
13. What is the conclusion?
14. List the basic laws of logic.
15. What is a logical conclusion? Give an example.

CHAPTER 3. PROBABILISTIC REASONING

3.1. Fundamentals of the theory of fuzzy logic

Fuzzy logic is a branch of mathematics that is a generalization of classical logic and set theory, based on the concept of a fuzzy set, first introduced by Lotfie Zadeh in 1965 as an object with a function of membership of an element in a set, taking any values in the interval [0, 1], and not just 0 or 1 [12]. Based on this concept, various logical operations on fuzzy sets are introduced and the concept of a linguistic variable is formulated, the values of which are fuzzy sets.

The subject of fuzzy logic is the study of reasoning under conditions of fuzziness, similar to reasoning in the usual sense, and their application in computer systems.

Currently, there are at least two main areas of scientific research in the field of fuzzy logic:

- fuzzy logic in the broad sense (theory of approximate calculations);
- fuzzy logic in the narrow sense (symbolic fuzzy logic).

Symbolic Fuzzy Logic

Symbolic fuzzy logic is based on the concept of t-norm. After choosing a certain t-norm (and it can be introduced in several different ways), it becomes possible to determine the basic operations on propositional variables: conjunction, disjunction, implication, negation, and others.

It is easy to prove the theorem that the distributivity present in classical logic is satisfied only when Gödel's t-norm is chosen as the **t-norm** [12].

In addition, for certain reasons, as an implication, the most commonly chosen operation is called residium (it, generally speaking, also depends on the choice of the t-norm).

The definition of the basic operations listed above leads to a formal definition of basic fuzzy logic, which has much in common with classical Boolean-valued logic (more precisely, with the calculus of propositions).

There are three main basic fuzzy logics: Lukasevich's logic, Gödel's logic and product logic. Interestingly, the union of any two of the three logics listed above leads to the classical Boolean valued logic.

Synthesis of continuous logic functions given in a table

Zade's fuzzy logic function always takes the value of one of its arguments or its negation. Thus, the function of fuzzy logic can be set by a selection table, which lists all the options for ordering arguments and negatives, and for each option function value is indicated. For example, a row of a table of a function of two arguments may have the following form:

$$1)\ x_1 \le x_2 \le \bar{x}_2 \le \bar{x}_1 : x_2 \qquad 2)\ x_2 \bar{x}_2 \bar{x}_1 \qquad 3)\ y(x) = \sum_{i=1}^{N} \phi_i(x) \cdot \theta_i$$

However, an arbitrary selection table does not always specify a fuzzy logic function. In [12], a criterion was formulated that made it possible to establish whether the function specified by the selection table is a function of fuzzy logic and a simple synthesis algorithm was proposed based on the introduced concepts of the constituents of the minimum and maximum. The function of fuzzy logic is a disjunction of the constituent of the minimum, where the constituent of the maximum is the conjunction of the variables of the current region greater than or equal to the value of the function in this area (to the right of the value of the function in the inequality, including the value of the function). For example, for the specified row of the table, the constituent minimum has the form $x_2 \bar{x}_2 \bar{x}_1$.

Theory of approximate computing

The basic concept of fuzzy logic in the broad sense is a fuzzy set defined using the generalized concept of a characteristic function. Then the concepts of union, intersection, and complementation of sets are introduced (through the characteristic function; it can be defined in various ways), the concept of a fuzzy relationship, and also one of the most important concepts – the concept of a linguistic variable.

Generally speaking, even such a minimal set of definitions allows the use of fuzzy logic in some applications, but for the majority it is also necessary to specify an inference rule (and an implication operator).

Fuzzy logic and neural networks

Since fuzzy sets are described by membership functions, and t-norms and k-norms by ordinary mathematical operations, fuzzy logical reasoning can be imagined in the form of a neural network. For this, membership functions must be interpreted as functions of activation of neurons, signal transmission as connections, and logical t-norms and k-norms, as special types of neurons that perform mathematical corresponding operations. There is a wide variety of such neuro-fuzzy networks. For example, ANFIS (Adaptive Neuro fuzzy Inference System) is an adaptive neuro-fuzzy inference system [13].

It can be described in the universal form of approximators as:

$$y(x) = \sum_{i=1}^{N} \phi_i(x) \cdot \theta_i$$

in addition, some types of neural networks, such as radial basis networks (RBF), multilayer perceptrons (MLP), and also wavelets and splines can also be described by this formula.

Fuzzy logic in computer science

Fuzzy logic is a set of fuzzy rules in which radical ideas, intuitive guesses, and also the experience of specialists accumulated in the corresponding field can be used to achieve the set goal. Fuzzy logic is characterized by a lack of strict standards. Most often it is used in expert systems, neural networks and artificial intelligence systems. Instead of the traditional values of **True** and **False**, fuzzy logic uses a wider range of meanings, among which are **Truth, False, Maybe, Sometimes, I don't remember (Yes, Why, No, I haven't decided yet, I won't say ...).** Fuzzy logic is simply irreplaceable in those cases when there is no clear answer to the question (yes or no; "0" or "1") or all possible situations are unknown in advance. For example, in fuzzy logic, a statement of the form "X is a large number" is interpreted as having an inaccurate meaning, characterized by some fuzzy set. "Artificial intelligence and neural networks are an attempt to simulate human behavior on a computer. And since people rarely see the world around them in black and white, the need arises to use fuzzy logic."

The rules for calculating the reliability of complex statements T are as follows:

$$T(A \wedge B) = \min(T(A), T(B))$$

$$T(A \vee B) = \max(T(A), T(B))$$

$$T(\neg A) = 1 - T(A)$$

3.2. Bayesian Networks

Consider a simple example close to all students. To pass (Pass) the exam, you need to prepare for it (Study) or use the cheat sheet (Cheat). Thus, there are 3 Boolean variables. We would like to know the probability of passing the exam.

In those cases where the probabilities of elementary (atomic) events are known, one can use the probabilistic inference method based on the complete joint distribution, which is described by a 2x2x2 table [14].

	Study		¬Study	
	Cheat	¬Cheat	Cheat	¬Cheat
Pass	0,15	0,4	0,04	0,06
¬Pass	0,01	0,04	0,05	0,25

The sum of all the probabilities is equal to one. In each cell, the probability of an elementary event. This probability is the result, i.e. takes into account all

the factors. So, the probability of successful passing the exam 0.4 takes into account the probability of preparing for the exam and the likelihood that the student did not use the cheat sheet.

The probabilities of complex events are easily calculated by summing the corresponding rows or columns of the table. The probability of preparing for the exam is equal to the sum of the cells in the left half of the table and corresponds to the events (taught, did not use the cheat sheet and passed; taught, used and passed; taught, used and did not pass; studied, did not use and did not pass):

$$P(Study) = 0,15 + 0,4 + 0,01+ 0,04 = 0,6$$

The probability of using a cheat sheet is equal to the sum of the first and third columns:

$$P(Cheat) = 0,15 + 0,01 + 0,04 + 0,05 = 0,25$$

The probability of passing the exam is equal to the sum of the cells of the first row (taught, used and passed; taught, not used and passed; did not learn, used and passed; did not learn, did not use and passed):

$$P(Pass) = 0,15 + 0,4 + 0,04 + 0,06 = 0,65$$

The method of probabilistic inference based on complete joint distribution is rather a good illustration of the principle of probability formation than a practical tool, since the probabilities of elementary events are not always known. Often make assumptions about the equal probability of these events.

You can usually evaluate the probabilities of the truth of individual variables. Let P (Study) = 0.6 (in 40% of cases there will be more important matters than preparing for the exam), and

P (Cheat) = 0.25 (one out of four chance to use a cheat sheet).

These probabilities are called a priori or unconditional. They represent a degree of confidence in the truth of the statement in the absence of other data.

At first glance, the probability of passing the exam is 0.6 + 0.25 = 0.85. In fact, everything is more complicated. Study and Cheat events may overlap, i.e. occur together. You can learn the material and use a cheat sheet for insurance. You can also learn everything, but not pass (the professor found fault). You can pass without preparation and cheat sheets (just lucky).

To find the desired probability, you must have conditional probabilities, for example, P (Pass | Study) – the probability of passing the exam, provided that you are fully prepared.

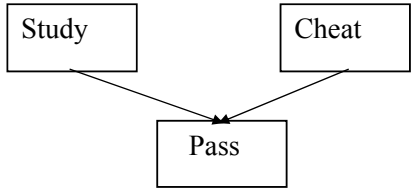
In general, the probability of event A is:

$$P(A) = \sum P(A|B_i)\, P(B_i) \qquad (3.1)$$

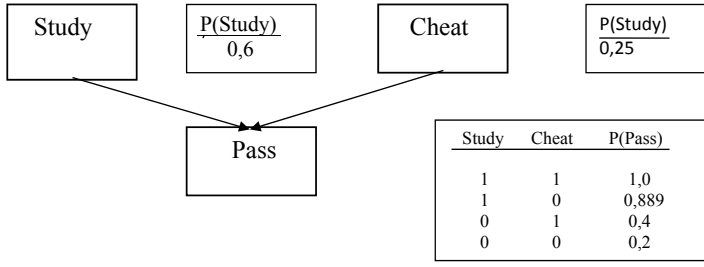
where P (B_i) is the a priori probability of the B_i event, P (A | B_i) is the conditional probability of event A provided that the Bi event is true.

Conditional probability links dependent events. If we introduce the fourth variable – sunny weather (Sunny), then we must specify conditional probabilities of type P (A | Sunny). In real situations, we may be faced with the fact that the dimension of the problem will be prohibitively large because of this.

To solve this problem, Bayesian networks are used, which make it possible to establish the dependence of variables and simplify the calculation of the total joint distribution. Below is a Bayesian network, considered as the example:



Note that in our model, the variables Study and Cheat are independent. Another model is possible when using the cheat sheet is due to the lack of preparation for the exam. It will be considered later.



Each vertex of the network corresponds to a random variable. The vertices are connected by directed ribs. If the arrow is directed from A to B, then A is called the parent vertex of B. Each vertex X_i is characterized by a conditional probability distribution P (X_i | Parents (X_i)), which quantifies the influence of its parent vertices on the vertex. In our example, we consider the following conditional probabilities known: the probability of passing the exam, on condition to the preparation and using cheat sheets, is equal to one; when preparing and not using the cheat sheet – 0.889; on condition to using the cheat sheet without preparing for the exam – 0.4; and the probability of passing the exam without preparation and a cheat sheet is 0.2 (just lucky).The main gain when using Bayesian networks is that the probability of any state is only based

on the parent (nearest) vertices, and not all the vertices on which this vertex depends:

$$P(X_i \mid Xi\text{-}1, X_{i-2}, ..., X_1) = P(X_i \mid Parents(X_i)) \quad (3.2)$$

In our example, we have the vertex Study, which in fact can be the end result of related events: the student attended lectures, had a compendium, he had access to a computer, he had time, etc. As a result of these events, we have the fact of preparing for the exam. The event of using the cheat sheet is also the result of a number of events, the probabilities of which must be disposed of: the availability of time, technical equipment, appropriate clothing for the secret use of the cheat sheet. To calculate the total joint distribution, you need to know all the conditional probabilities, for example, the probability of having a radio transmitter if you attend lectures. To calculate the probability of passing the exam, we only need to know the probabilities P (Study) and P (Cheat).

The rule above is called **a chain rule.** Following this rule, it is simple enough to calculate the probabilities of events, progressively moving in the direction of the arrows. In this example, we can calculate the probability of passing the exam:

$$P(Pass) = P(Pass \mid Study, Cheat) * P(Study) * P(Cheat) +$$
$$+ P(Pass \mid Study, \neg Cheat) * P(Study) * P(\neg Cheat) +$$
$$+ P(Pass \mid \neg Study, Cheat) * P(\neg Study) * P(Cheat) +$$
$$+ P(Pass \mid \neg Study, \neg Cheat) * P(\neg Study) * P(\neg Cheat) =$$
$$= 1{,}0 * 0{,}6 * 0{,}25 + 0{,}889 * 0{,}6 * 0{,}75 + 0{,}4 * 0{,}4 * 0{,}25 +$$
$$+ 0{,}2 * 0{,}4 * 0{,}75 = 0{,}65$$

In our highly simplified example, this gain in computational complexity is not noticeable, since the Bayesian network chain is not so long. The second factor is the independence of the variables Study and Cheat. Experienced students may notice some implausible probabilities: when preparing for the exam, the cheat sheet only adds the risk of being caught and kicked out of the exam. Change the logic as follows. We assume that the student will try to use the cheat sheet only if she does not prepare for the exam. Bayesian network will change as shown below.

The conditional probability of using the cheat sheet is zero in the case of preparation for the exam and 0.75 in the absence of preparation.

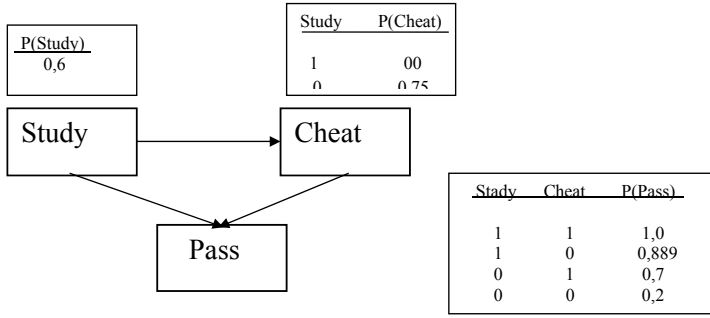
Probability

P (Pass | Study, Cheat) = 0.

Calculate P (Cheat) and P (¬Cheat):

P (Cheat) = P (Cheat | ¬Study) * P (¬Study) = 0.75 * (1-0.6) = 0.3

P (¬Cheat) = 1- P (Cheat) = 0.7



Now, using the chain rule, we can calculate the probability of passing the exam:

P (Pass) = P (Pass | Study, ¬Cheat) * P (Study) *
P (¬Cheat) + P (Pass | ¬Study, Cheat) * P (¬Study) *
P (Cheat) + P (Pass | ¬Study, ¬Cheat)
* P (¬Study) * P (¬Cheat) =
0.889 * 0.6 * 0.7 + 0.7 * 0.4 * 0.3 + 0.2 * 0.4 * 0.7 = 0.513

Bayesian networks allow solving inverse problems. For example, it is known that a student passed the exam successfully. It is required to find the probability that he was prepared for the exam. To do this, use the Bayesian rule [13]:

$$P(A \mid B) = P(B \mid A) * P(A) / P(B) \quad (3.3)$$

In our case, the probability of passing the exam, which was successfully passed, P (Pass) = 0.513; the probability of passing the exam in preparation for it P (Pass | Study, ¬Cheat) = 0.889; the probability of preparing and not using the cheat sheet P (Study, ¬Cheat) = 0.6 * 0.7 = 0.42, therefore,

P(Study, ¬Cheat | Pass) = P(Pass | Study, ¬Cheat) * P(Study, ¬Cheat) / P(Pass) = 0,889* 0,6 * 0,7 / 0,513 = 0,727.

We now find the probability that the exam was passed using the cheat sheet:

P (¬Study, Cheat | Pass) = P (Pass | ¬Study, Cheat) * P (¬Study, Cheat) / P (Pass) = = 0.7 * 0.4 * 0.3 * / 0.513 = 0.164

And finally, the likelihood that the reason for passing the exam was pure luck:

P (¬Study, ¬Cheat | Pass) = P (Pass | ¬Study, ¬Cheat) * P (¬Study, ¬Cheat) / P (Pass) = 0.2 * 0.4 * 0.7 / 0.513 = 0.109

Thus, Bayesian networks provide the decomposition of complex problems and at the same time eliminate the need to specify many conditional probabilities.

3.3. Illustration: The Monty Hall Paradox

To illustrate the use of the Bayesian rule, we consider the following problem, which is called the Monty Hall paradox in the literature [15]. Let you participate in a television game, during which you need to make a choice of one of three doors. Behind one of them is a car, behind the other two are goats. Suppose you pointed to door number 1. The host opens another door, let it be door number 3, behind which a goat is discovered, and invites you, before it's too late, to change your mind and open door number 2. Question: Does it make sense to listen to the opinion of the presenter and open door number two? At first glance, the probabilities that the car is located behind one of the remaining doors are P (C1) = P (C2) = 1/2, and it makes no sense to change the decision. However, with a careful approach, we can extract useful information from the fact that the host opened the door number 3. Let the a priori probabilities that the car is outside door No. 1, No. 2 and No. 3 P (C1) = P (C2) = P (C3) = 1/3.

We will find the probabilities that the presenter will open door No. 2 and No. 3 (he cannot open door No. 1, since we have already pointed it out). If the car is located outside door number 1, then the conditional probabilities that it will open doors number 2 and number 3:

P (D = 2 | C1) = 0.5; P (D = 3 | C1) = 0.5

That is, the presenter does not care which of the remaining doors to open. If the car is located outside door number 2, then the likelihood that it will open the doors number 2 and number 3:

$$P (D = 2 | C2) = 0; P (D = 3 | C2) = 1$$

Since the car is behind the second door, the leader cannot open this door, but with a probability of 1 it will open the third door. And vice versa, if the car is behind the third door, then the corresponding probabilities:

$$P (D = 2 | C3) = 1; P (D = 3 | C3) = 0$$

So, we have the probabilities of which door the host will open, depending on where the car is. To find the inverse conditional probability, we use the Bayesian rule. Since it is known that D = 3, then

$$P (Ci | D = 3) = P (D = 3 | Ci) * P (Ci) / P (D = 3)$$

Substituting C1 and C2 into this formula, we obtain:

P (C1 | D = 3) = P (D = 3 | C1) * P (C1) / P (D = 3) = ½ * 1/3 / 1/2 = 1/3.

P (C2 | D = 3) = P (D = 3 | C2) * P (C2) / P (D = 3) = 1 * 1/3 / 1/2 = 2/3.

Thus, the likelihood that the car is behind the second door is twice as high! The conclusion can be supported by logical reasoning: If the car is behind the first door that we have already chosen, then the TV host does not care which of the remaining doors to choose. If the car is behind the second door, then the TV host has no choice but the door number 3. Having changed the choice, in the worst case we remain with the same chances, and in the best we get a car.

3.4. Observational based learning

The examples discussed in the previous section may raise a reasonable question: how to use these wonderful tools in real life? Where to get all these probabilities? Mathematicians bypass this question quite simply. All reasoning begins with the words: let the probability of event A be equal to X. Thus, mathematicians do what they can, as they need. The engineer, on the other hand, has to do what is needed in the way possible. If you do not take ideal examples, such as throwing dice or dealing a deck of cards, then you can only find probabilities based on observations. In our exam example, only exit poll can be the source of information, i.e. an anonymous exit survey. The results of this survey may look as follows:

№ i/o	Study	Cheat	Pass
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	No	Yes
4	Yes	No	No
5	No	Yes	No
6	No	No	No
7	Yes	No	Yes
8	No	Yes	No
9	Yes	No	Yes
10	No	No	Yes

From this sample, we can calculate the probability of passing the exam, which is equal to the ratio of the number of successfully passed to the total number of those who passed P (Pass) = 6/10 = 0.6. Similarly, we get the

probability of preparing P (Study) = 5/10 = 0.5 and using the cheat sheet P (Cheat) = 3/10 = 0.3. In the same way, we can get conditional probabilities P (Pass | Study) = 3/5 = 0.6 (the ratio of the number of those who passed the exam for those preparing for it to the total number of those preparing for the exam) and P (Pass | Cheat) = 1 / 3. It is convenient to display the distribution in the form of a tree (Fig. 3.1):

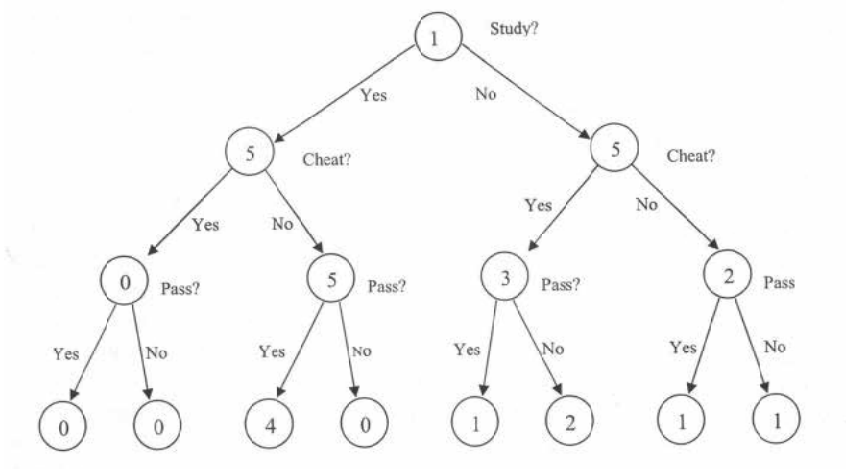


Fig. 3.1 – Observation tree

Here, as in the search problems considered earlier, it is desirable to form a tree of minimal depth. In this regard, it is necessary first to involve the most important, i.e. The most informative search attributes. One suitable criterion for selecting an attribute is the amount of information contained in the response.

The amount of information is calculated by the Shannon formula. If the possible answers vi have probabilities p (vi), then the information content of the actual answer I, measured in bits, is

$$I (p (v_1), p (v_2), ..., p (v_n) = -\sum p (v_i) \log2 p (v_i) \qquad (3.4)$$

We calculate the amount of information in the answer about the preparation for the exam:

I (p (Study), p (¬Study)) = - 5/10 * log2 5/10 - 5/10 * log2 5/10 = 1 bit

Amount of information in the cheat sheet answer:

I (p (Cheat), p (¬Cheat)) = - 3/10 * log2 3/10 -7/10 * log2 7/10 = 1.533 bits

Thus, the answer about the cheat sheet is one and a half times more informative, and you should put it earlier in the search tree.

Another problem is the sufficiency of information in the training set. The above example gives a very rough estimate. Actual probabilities may vary significantly. The other extreme is the large sample size commensurate with the general population [16]. If we arrange an exit poll for all students without exception, we will get an exhaustive answer to all the questions and we will no longer need models. The usefulness of using models is achieved when, on a small sample, we can conclude about the entire population. For example, if you learn that 10% of students pass the exam using cheat sheets, you can change the format of the exam, for example, allow you to use any sources, but complicate the questions.

One way to measure sample sufficiency is to gradually add data to the original survey table by comparing the probability changes in each new iteration. As soon as the change in the results becomes less than the permissible error, the training can be considered completed.

Questions for self-control

1. What does fuzzy logic mean?
2. What is the subject of fuzzy logic.
3. What is the basis of symbolic fuzzy logic.
4. What are the three main basic fuzzy logics.
5. List the basic laws of logic.
6. What is a logical conclusion? Give an example.
7. Give the concept of a linguistic variable.
8. Give a definition of a fuzzy set.
9. List the shape of the curves to define membership functions.
10. What does ANFIS (Adaptive Neuro fuzzy Inference System) mean?
11. What is judgment? Give an example.
12. What structural elements of judgment do you know? Give a definition and give an example.
13. List the shapes of the curves to define membership functions.
14. To solve what problems are Bayesian networks used?
15. What does the Monty Hall paradox mean?

CHAPTER 4. NEURAL NETWORKS

4.1 The concept of a neural network.

Neural networks are based on an attempt to recreate a primitive model of nervous systems in biological organisms. In living things, a neuron is an electrically excitable cell that processes, stores, and transmits information through electrical and chemical signals through synaptic connections. The neuron has a complex structure and narrow specialization. Connecting with each other to transmit signals using synapses, neurons create biological neural networks. In the human brain, there are an average of about 65 billion neurons and 100 trillion synapses. In fact, this is the basic mechanism of learning and brain activity of all living beings, i.e. – their intelligence.

For example, in Pavlov's classic experiment, every time just before feeding the dog, the bell rang, and the dog quickly learned the connection between the bell and food. From a physiological point of view, the result of an experiment in dog's brain was the establishment of synaptic connections between the parts of the cerebral cortex that are responsible for hearing and the areas that are responsible for controlling the salivary glands. As a result, by the excitement of bell's sound, the dog began to salivate. So the dog learned to react to signals (data) coming from the outside world and make a "correct" conclusion.

It is the ability of biological nervous systems to learn and correct their mistakes that formed the basis of research in the field of artificial intelligence. Their initial task was to try to artificially reproduce the low-level structure of the brain – i.e. create a computer "artificial brain." As a result, the concept of an "artificial neuron" was proposed – a mathematical function that transforms several input facts into one output, assigning influence weights to them [17].

Each artificial neuron can take a weighted sum of input signals and, if the total input exceeds a certain threshold level, transmit the binary signal further.

Artificial neurons are connected in networks – connecting the outputs of some neurons with the inputs of others. Artificial neurons connected and interacting with each other create an artificial neural network – a specific mathematical model that can be implemented on software or hardware. In simple terms, a neural network is just a black box program that receives data and gives answers. Being built from a very large number of simple elements, a neural network is capable of solving extremely complex problems.

The mathematical model of a single neuron (perceptron) was first proposed in 1943 by American neurophysiologists and mathematicians Warren McCallock, Walter Pitts, and they also proposed the definition of an artificial neural network. Physically, a model using a computer was modeled in 1957 by Frank Rosenblatt. We can say that neural networks are one of the oldest ideas for the practical implementation of AI.

Currently, there are many models for implementing neural networks. There are “classic” single-layer neural networks, they are used to solve simple problems. A single-layer neural network is mathematically identical to the usual polynomial, weight function, traditionally used in expert models. The number of variables in the polynomial is equal to the number of network inputs, and the coefficients before the variables are equal to the weighting coefficients of the synapses.

There are mathematical models in which the output of one neural network is directed to the input of another, and cascades of connections are created, the so-called multi-layer neural networks (MNN) and one of its most powerful options is convolutional neural networks (CNN).

MNN have great computing capabilities, but also require huge computing resources. Given the placement of IT systems in the cloud infrastructure, multilayer neural networks have become available to a larger number of users and now they have become the foundation of modern AI solutions. In 2016, Digital Reasoning from the United States, a cognitive computing company, created and trained a neural network of 160 billion digital neurons. This is significantly more powerful than the neural networks available to Google (11.2 billion neurons) and the US National Laboratory in Livermore (15 billion neurons).

Another interesting type of neural network is recurrent neural network (RNN), when the output from the network layer is fed back to one of the inputs. Such platforms have a “memory effect" and they are able to track the dynamics of changes in input factors. A simple example is a smile. A person begins to smile with barely noticeable movements of the facial muscles of his eyes and face, before he clearly shows his emotions. RNN allows you to detect such movement in the early phases, which is useful for predicting the behavior of a living object in time by analyzing a series of images or constructing a consistent flow of speech in a natural language.

4.2. The principle of neural networks construction

The standard way to solve any problem using a computer is that the task is investigated, an algorithm is compiled, which is implemented in a programming language in the form of a program. After debugging, the program is ready for operation.

Let us be entrusted with creating a robot playing basketball, namely, that part of it that is responsible for throwing the ball into the basket. The first thing you need to start with is to compose differential equations for the flight of the ball from different positions, make adjustments for air resistance, parallax of

sensors with respect to the actuator, etc. After that, you can start programming. At the same time, hardly anyone would suggest that Shaquille O'Neill knows at least elementary mathematics, not to mention higher education. The human brain, obviously, solves this problem in a fundamentally different way. Everyone knows what this method is - repeated repetition.

In the previous section, we examined training, the result of which is the probability of the occurrence of certain events. Is it possible to create such a computational structure that without any programming, based on repeated repetitions, could learn to solve problems. Such a structure is called a neural network. The idea is borrowed from nature. A neural network is a collection of neurons – computational elements (sometimes called perceptrons), each of which has several synapse inputs and one axon output [17].

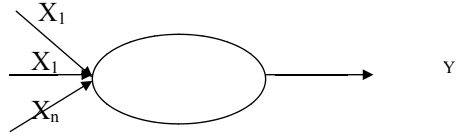


Fig. 4.1 – Single neuron

The intelligence of a single neuron (Fig. 4.1) is low. We can assume that it implements a simple regression model for N independent variables in the set. If we combine a lot of neurons into network structures, then the implemented function can be arbitrarily complex. The network shown in Figure 4.2 has distinct layers, i.e. rows of neurons equidistant from the entrance (exit). Other structures can also be created, including those with feedback (recurrent).

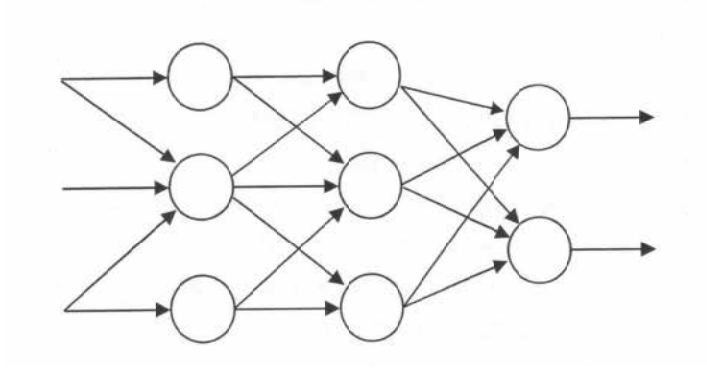


Fig. 4.2 – Neural network

The implementation of such a network can be hardware when each neuron is run on a separate microprocessor, or software if the neurons are emulated in special programs like spreadsheets. The structure of an individual neuron can be arbitrary, but the following is most often used (Fig. 4.3):

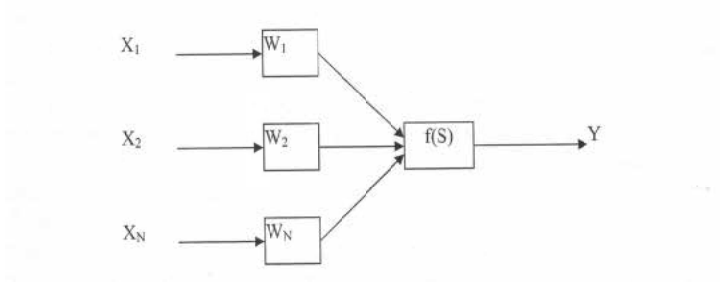


Fig. 4.3 – Neuron structure

The input signals (variables) Xi are weighted (multiplied by Wi coefficients called synaptic weights), then summed, and the resulting weighted sum

$$S = W_1X_1 + W_2X_2 + ... + W_NX_N \quad (4.1)$$

undergoes a change in the function f (S), called the activation function. The output signal Y may also be subject to weighting (scaling). The sigmoid function $Y = 1 / (1 + \exp(-\lambda S))$, as well as the hyperbolic tangent, logarithmic function, linear and others are most often used as the activation function. The main requirement for such functions is monotony.

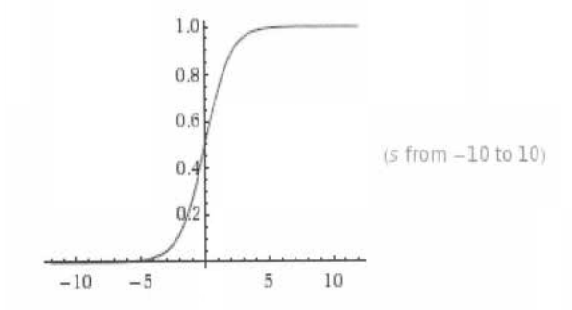


Fig. 4.4 – Logistic Sigmoid Function

Let us now consider the computational capabilities of a single neuron. Let the number of synapses be three, and the synaptic weights W_1, W_2, W_3 equal

1.0, and the activation function has the following form, represented by a graph (this is a logistic sigmoid function shifted along the X axis to the right by 0.5 (Fig. 4.5):

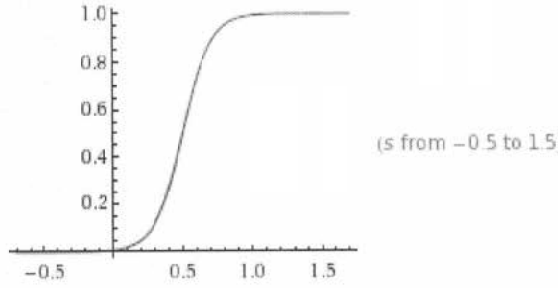


Fig. 4.5 – Type of activation function for implementing Boolean functions

Let the input signals take the values 0 or 1. In this case, the output will be approximately 1 if at least one of the input signals is 1, then we have a disjunction function. If the synaptic weights W_1, W_2, W_3 are set to 0.2, then the same neuron will implement the conjunction function.

4.3. Neural network training

The brain analogy does not end with the structure of a neuron and a network of neurons. The idea of training neural networks was also borrowed from nature. It is known that the human brain is capable of self-learning, and often succeeds, not knowing the nature of the processes that underlie the actions performed. For example, in order to get a ball into a basketball hoop, a robot basketball player must measure the distance to the hoop and direction, calculate a parabolic trajectory, and make a throw taking into account the mass of the ball and air resistance. A man can do without this only through training.

Repeatedly making throws and observing the results, he adjusts his actions, gradually improving his technique. At the same time, the corresponding structures of neurons responsible for the casting technique are formed in his brain. Thus, an indispensable attribute of training is repeated repetition and the ability to immediately evaluate the result. For neural networks, this process can be represented by the following algorithm (Fig. 4.6) [18].

The choice of the structure of the neural network is a separate task and consists in choosing the network topology and activation functions of each neuron. Initially, the parameters of neurons are set arbitrarily.

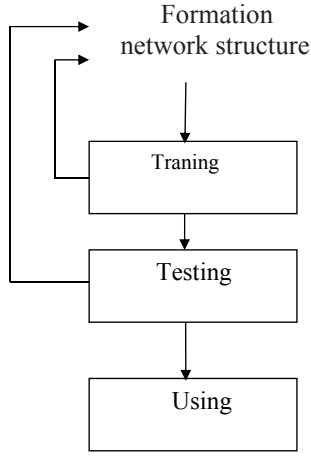


Fig. 4.6 – The process of creating and using a neural network

The training consists in the fact that special training data is supplied to the network input, that is, such input data for which the output is known. The resulting data is generated at the output, the results are compared with the expected ones, and the error value is calculated [18,19].

After that, in a certain sequence, the parameters of the neural network are corrected in order to minimize the error function. If satisfactory accuracy cannot be achieved, you should change the network structure and repeat the training on a set of training data. After the network is trained, testing is performed, that is, accuracy control on special test data. This means that all data should be divided into two subsets: on the first, network training is performed, and on the second, testing. This partition can be random or regular, for example, every second record of the original data array can be used for testing.

Testing from training differs in that the test data only checks the accuracy, and since this data is not used to select network parameters, they can serve as a criterion for the quality of training. By analogy with human training, testing can be likened to an exam. As an example, consider the task of optical text recognition. Let the matrix of points of the selected image fragment corresponding to the recognizable symbol be fed to the network input as input signals (Fig. 4.7). Let the matrix of points of a recognizable sign be fed to the network input. At the output, signals corresponding to the recognized sign are formed. Network training consists in repeatedly “presenting” the network of different options for writing characters along with ready-made “answers”.

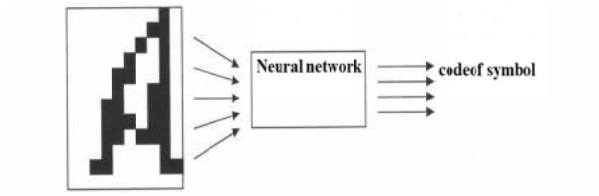


Fig. 4.7 – Neural network in the problem of character recognition

Note that the network does not try to remember all possible options for the drawing of each character, but only forms an output signal

$$Y_j = f(X_1, X_2, \ldots X_n) \tag{4.2}$$

as a function of input variables. Note that this approach to text recognition has a significant advantage over others in speed. So is man: to read the text, he does not ask for the name of the font, and does not search in memory for all possible character styles. Associations in the brain are established instantly. The task of training the network has a huge dimension. So, for training a network consisting of only 10 neurons, each of which has 3 synapses, it is necessary to select values of at least 40 parameters (30 values of Wi – synaptic weights, and 10 parameters of λi activation functions). If each parameter is selected with a resolution of 1/100, then the total number of network runs on the set of training data will be 100^{40}. Obviously, even supercomputers cannot do such a task.

This problem is satisfactorily solved using the back propagation algorithm, which is as follows. Initially, all network parameters are set arbitrarily.

1. Training data is run through the network and the total error function E = sum (E_i^2) is calculated, where $E_i = Y_i - y_i$, Y_i is the calculated value of the output quantity, y_i is the expected value.

2. The value of the derivatives of the error function for each parameter is calculated, and on their basis – the calculation of corrections to the parameters of the neural network.

3. The network parameters are adjusted by the amount of corrections, after which steps 2 and 3 are repeated from the beginning until the error function drops to a predetermined level.

Despite its simplicity, this algorithm is very time-consuming, and its acceleration is an urgent task.

If as a result of training a satisfactory result was not obtained, then it is necessary to change the network structure. This can be done manually, or the structure can be selected from a predefined set (library of structures). Software products that support such solutions exist. But the most successful approach

should be recognized in which the network structure is automatically generated. An example is the NGO neural network of BioComp Systems Inc.

This approach consists in applying genetic algorithms to this task [26-28]. The fact is that in the process of training the network, “strong” neurons and connections between them (sensitive to changes in parameters) and “weak” (parameters of which can be changed arbitrarily without significantly affecting the final result) are detected. Using this data, you can control the population of neurons. Weak neurons and synapses must die, and for the development of the structure, as well as to prevent universal "extinction", the network undergoes a "mutation": it is added randomly or in another way, for example, to strengthen the "strong" neurons, new neurons and synaptic connections are added [20].

Thus, through many generations, the number of which can reach tens of thousands, the network will have an optimal structure.

In this regard, the question may arise: why spend such an amount of time on optimizing the network structure if training a network of maximum dimension with a full set of connections takes a deliberately less time? In addition, it was previously said that the performance of a trained neural network is quite large.

There are two reasons for this. The first is that often the trained network is subsequently implemented on another platform, in particular, on the hardware level. The second reason – optimization often leads to a significant reduction in the number of input variables due to the exclusion of redundant ones that do not affect the final result. This fact gives neural networks a qualitatively new property: you don’t have to worry about which input data is important and which is not – during the training process, the excess will be discarded. In statistics, a similar function is performed by stepwise linear regression, analysis of variance and factor analysis.

For example, we are trying to use a neural network to predict the dollar rate. As initial data, we can substitute the behavior of this course for several years (by the way, this method of predicting values based on only their behavior in the past is called "technical analysis"), economic indicators and indices (Dow Jones, NASDAQ, consumer prices), as well as any other, up to meteorological observations. If a neural network that gives good forecasts is obtained on all these data, then using it will be quite tiring, and sometimes expensive. Moreover, the identification of significant parameters in itself is a valuable result. If it turns out that the air temperature affects the dollar, you can explore the nature of this phenomenon. They may object to me that such a connection can be detected by comparing graphs of air temperature and the dollar exchange rate.

However, direct comparison allows you to detect mainly linear dependence, moreover, human intelligence is included in the work here.

4.4. Features of the use of neural networks

The full automation of the choice of the topology and parameters of the neural network, provided by software packages such as NGOs, can create the illusion that you can completely not control the modeling process using a neural network. However, the use of neural networks, as well as statistical methods of analysis, is a creative process that requires an understanding of the principles of operation of this tool. The consequences of improper construction and training of the neural network are mainly generalization or retraining [20].

A generalization is called excessive simplification of the network, in which it does not reproduce small dependencies. For example, summarizing a network for weather forecasting will cause it to give an average temperature for a given season. The extreme degree of generalization in this problem is the average annual temperature. Naturally, there will be little benefit from such a forecast.

Retraining is another extreme, in which the neural network has an unnecessarily complex structure and adjusts the results too carefully to the desired ones in the learning process. As a result, on data not participating in the training, the network shows the worst results. A sign of retraining is a significant difference between the errors at the training stage and at the testing stage. Another danger of learning too long in combination with a genetic modification of the network topology is the depreciation of test results. If, in a single training procedure, the test data obviously does not participate in the learning process, then with network modifications, the test data is used to reject unsuccessful topologies and, thus, affect the final network.

A significant drawback of neural networks is the inability to extrapolate. In other words, a neural network trained in a certain range of values of input variables is not able to predict outside this range. In practice, this is usually expressed in that the output variable does not change outside the field of study, as shown in Fig. 4.8. If the network is trained in the range of -0.5 <X <0.5, then outside this range the graph of the output function usually lies horizontally.

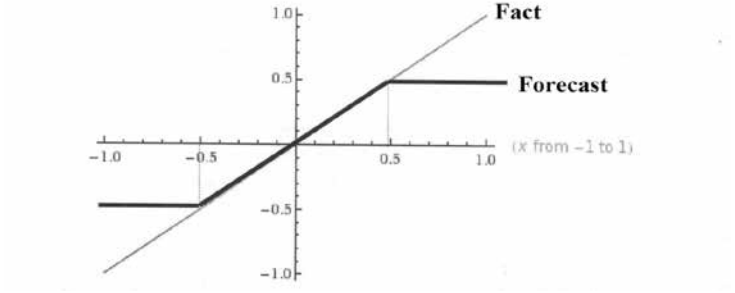


Fig. 4.8 – Neural network behavior outside the field of study

Despite the fact that the neural network takes care of any, even the most complex, conversion of input variables to output, data preprocessing is usually not without meaning. So, in the case of forecasting in the stock market, it is probably advisable to switch from absolute quotes to relative values, i.e. increments from one observation to another. In this case, it doesn't matter for the neural network which fluctuations around which levels to use for training, if we are only interested in the direction of the quotes, up or down 19]. In absolute terms, a network trained on quotation fluctuations around the values of 1000 or 5000 is absolutely useless for predicting the 3000 in the vicinity.

4.5. Neural networks in recognition tasks

An image, a class is a classification grouping in a classification system that unites a certain group of objects according to some criterion. The figurative perception of the world is one of the properties of the living brain, which makes it possible to understand the endless stream of perceived information and maintain orientation in disparate data about the external world. Perceiving the outside world, we always classify information, that is, divide them into groups of similar, but not identical, phenomena. For example, despite the significant difference, the same group includes all the letters "A" written in different handwritings, or all sounds corresponding to the same note taken in any octave and on any instrument. To make up the concept of a group of perceptions, it is enough to familiarize yourself with a small number of its representatives. This property of the brain allows us to formulate such a concept as an image [21].

Images have a characteristic property, which is manifested in the fact that familiarization with a finite number of phenomena from the same set makes it possible to recognize an arbitrarily large number of its representatives. Images have characteristic objective properties in the sense that different people studying on different observation material, for the most part, classify the same objects equally and independently. It is this objectivity of images that allows people around the world to understand each other.

In general, the problem of pattern recognition consists of two parts: training and recognition. Training is carried out by showing individual objects with an indication of their belonging to a particular image. As a result of training, the recognition system should acquire the ability to respond with the same reactions to all objects of one image and with other reactions to all objects of distinguishable images. It is very important that the learning process should be completed only by showing a finite number of objects. As objects of training can be either pictures (Figure 4.9), or other visual images (letters, numbers) [21].

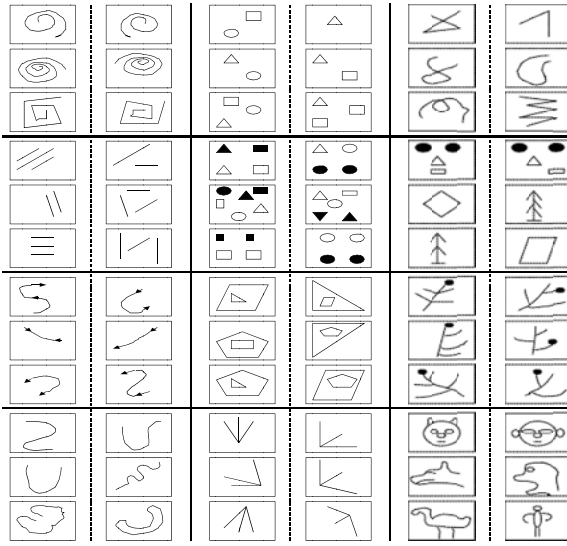


Fig. 4.9 – Example of learning objects

It is important that in the learning process only the objects themselves and their belonging to the image are indicated. Learning is followed by the process of recognizing new objects, which characterizes the actions of an already trained system. Automation of these procedures is the problem of training in pattern recognition. In the case when a person himself decides or invents, and then imposes a classification rule on the machine, the recognition problem is partially solved, since the person takes over the main and primary part of the problem (training).

The choice of the initial description of objects is one of the central tasks of the problem of pattern recognition. With a successful choice of the initial description (space of attributes), the recognition problem may turn out to be trivial, and, conversely, an incorrectly selected initial description can lead either to a very difficult further processing of the information or to the absence of a solution at all.

4.6. Design, training and adaptation of neural networks

Training is a process, as a result of which the system gradually acquires the ability to respond with the necessary responses to certain sets of external influences, and adaptation is the adjustment of the parameters and structure of

the system in order to achieve the required control quality under continuous changes in external conditions. All the pictures presented in Figure 2 characterize the learning task. In each of these tasks several examples are given (training sequence) of correctly solved problems. If it were possible to notice a certain universal property that does not depend either on the nature of the forms or on their images, but determines only their ability to separability, then along with the usual task of learning recognition using information about the belonging of each object from the training sequence to a particular image, one could pose another classification problem – the so-called task of learning without a teacher. This kind of task at a descriptive level can be formulated as follows: objects are presented simultaneously or sequentially to the system without any indication of their belonging to the images.

The input device of the system maps many objects to many images and, using some property of separability of images embedded in it in advance, makes an independent classification of these objects. After such a process of self-learning, the system should acquire the ability to recognize not only already familiar objects (objects from the training sequence), but also those that have not been presented before. The process of self-learning of a system is such a process as a result of which this system without prompting a teacher acquires the ability to generate the same reactions to images of objects of the same image and different reactions to images of different figures [21]. The role of the teacher in this case is only in prompting the system of some objective property, the same for all images and determining the ability to divide many objects into images.

Such an objective property is the compactness of images. The relative position of the points in the selected space already contains information on how to divide the set of points. This information determines the property of image separability, which turns out to be sufficient for self-learning of pattern recognition system.

In order to enter an image into a machine, you need to translate it into machine language, i.e. encode, represent in the form of some combination of characters that the machine can operate. Coding of flat figures can be done in a variety of ways. Better to strive for the most “natural” coding of images. We will draw the figures on a certain field, divided by vertical and horizontal lines into identical elements - squares. Elements on which the image has fallen will be completely blackened, the rest will be left white. Let us agree to designate black elements as one, white elements as zero. We introduce the sequential numbering of all elements of the field, for example, in each row from left to right and along the lines from top to bottom. Then each figure drawn on such a field will be unambiguously displayed by a code consisting of as many digits (ones and zeros) as there are elements in the field.

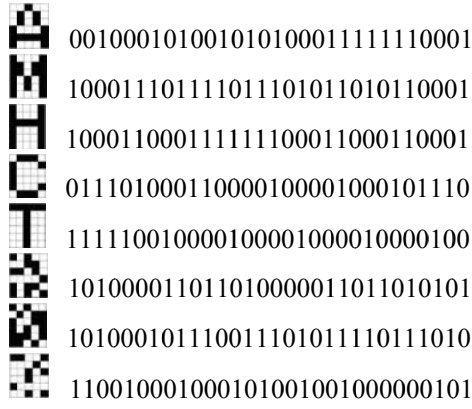


Figure 4.10 – Examples of projection and coding of images

Such coding (Figure 4.10) is considered "natural" because dividing the image into elements underlies the operation of our visual apparatus. Sensitive elements of the retina transmit signals through their nerve fibers to the brain, the intensity of which depends on the illumination of this element. Thus, the image, designed by the optical system of the eye on the retina, is divided by rods and cones into separate sections, and transmitted to the brain by elements in some code. Individual elements of the field are called receptors, and the field itself is called the receptor field.

The set of all planar figures that can be depicted on the field of receptors is a certain set. Each concrete figure from this totality is an object of this set. Any of these objects corresponds to a specific code. Similarly, any code corresponds to a specific image on the field of receptors. A one-to-one correspondence between codes and images will allow you to operate only with codes, bearing in mind that an image can always be reproduced by its code.

The capacity of an artificial neural network (ANN) is the number of images presented to the inputs of an ANN for recognition. To separate multiple input images, for example, in two classes, just one output is enough. Moreover, each logical level – "1" and "0" – will denote a separate class [20]. At two outputs you can already encode 4 classes and so on. To increase the reliability of the classification, it is desirable to introduce redundancy by isolating each class with one neuron in the output layer or, even better, several, each of which is trained to determine whether the image belongs to the class with its degree of reliability, for example: high, medium and low. Such ANN allow one to classify input images combined into fuzzy (blurry or intersecting) sets. This property brings similar ANN to real-life conditions.

The process of training ANNs in a new class of tasks includes the following stages:

1. The statement of the problem is formulated and a set of key parameters characterizing the subject area is highlighted.

2. A neural network paradigm is selected (a model that includes the form of input data, a threshold function, network structure, and learning algorithms) that is most suitable for solving this class of problems. As a rule, modern neuropackages, neuroboards and neurocomputers allow us to implement not only one, but several basic paradigms.

3. A possibly wider set of training examples is being prepared, organized as sets of input data associated with known output values. Input values for training may be incomplete and partially contradictory.

4. The input data are presented in turn by the ANN, and the resulting output value is compared with the standard. Then, the weighting coefficients of interneuronal connections are adjusted to minimize the error between the real and the desired output of the network.

5. The training is repeated until the total error in the entire set of input values reaches an acceptable level, or the ANN comes to a stationary state. The considered method for training a neural network is called "error backpropagation" and is among the classical algorithms of neuromathematics.

A customized and trained ANN can be used on real input data, not only prompting the user with the correct solution, but also evaluating the degree of its reliability.

4.7. Development of a neural network for the recognition of acoustic signals

Among the various configurations of artificial neural networks, there are those whose classification according to the principle of learning, strictly speaking, does not suit learning with a teacher, or learning without a teacher. In such networks, synapse weights are calculated only once before the network starts functioning based on information about the data being processed, and all network training comes down to this calculation. On the one hand, the presentation of a priori information can be regarded as a teacher's help, but on the other hand, the network actually just remembers the samples before real data arrives at its input, and cannot change its behavior, so talk about the feedback link with the "world" (teacher) is not necessary. Of the networks with similar operation logic, the most famous are the Hopfield network and the Hamming network, which are usually used to organize associative memory [22].

The block diagram of the Hopfield network is shown in Figure 4.11. It consists of a single layer of neurons, the number of which is simultaneously the number of inputs and outputs of the network. Each neuron is connected by synapses to all other neurons, and also has one input synapse through which the signal is input. Output signals, as usual, are formed on axons [5].

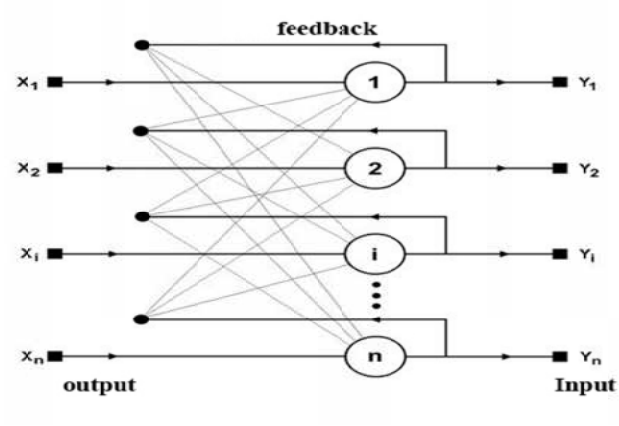


Fig. 4.11 – Block diagram of the Hopfield network

The problem solved by this network as associative memory, as a rule, is formulated as follows. A certain set of binary signals is known (images, sound digitizations, other data describing certain objects or process characteristics), which are considered exemplary. The network should be able to select ("recall" from partial information) the corresponding sample (if any) from an arbitrary non-ideal signal fed to its input or "give a conclusion" that the input data do not correspond to any of the samples. In the general case, any signal can be described by the vector X = {xi: i = 0 ... n-1}, *n* is the number of neurons in the network and the dimension of the input and output vectors. Each xi element is either +1 or -1.

We denote the vector describing the k-th sample by Xk, and its components, respectively, are xik, k = 0 ... m-1, m is the number of samples. When the network recognizes (or "remembers") any pattern based on the data presented to it, its outputs will contain it, that is, Y = Xk, where Y is the vector of the network output values: Y = {yi: i = 0 ,. ..n-1}. Otherwise, the output vector does not match any exemplary.

If, for example, the signals are some images, then, having graphically displayed the data from the network output, it will be possible to see a picture

that completely coincides with one of the model ones (in case of success) or "free improvisation" of the network (in case of failure).

At the network initialization stage, synapse weights are set as follows [5]:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \ i = j \end{cases} \quad (4.1)$$

Here i and j are the indices, respectively, of the presynaptic and postsynaptic neurons; x_{ik}, x_{jk} – the i-th and j-th elements of the vector of the k-th pattern.

The network functioning algorithm is as follows (*n* is the iteration number):

1. An unknown signal is supplied to the network inputs. In fact, its input is arried out by directly setting the axon values:

$$y_i(0) = x_i, \ i = 0...n-1 \quad (4.2)$$

therefore, the designation on the network diagram of the input synapses in explicit form is purely conditional. A zero in the bracket to the right of yi means zero iteration in the network cycle.

2. The new state of neurons is calculated,

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p)$$

$$j = 0 \ ... \ n-1 \quad (4.3)$$

and new axon values

$$y_j(p+1) = f\left[s_j(p+1)\right] \quad (4.4)$$

where f is the activation function in the form of a jump, shown in Figure 4.12

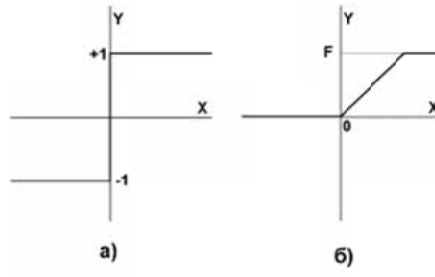


Fig. 4.12 – Activation functions

3. Check if the output axon values have changed during the last iteration. If yes, go to step 2, otherwise (if the outputs have stabilized) – end. In this case, the output vector is a sample that is best combined with the input data.

As mentioned above, sometimes the network cannot perform recognition and produces a non-existent image at the output. This is due to the problem of network limitations. For a Hopfield network, the number of memorized images m should not exceed a value approximately equal to 0.15 • n. In addition, if the two images A and B are very similar, they will probably cause cross-association in the network, that is, presenting vector A at the network inputs will cause vector B to appear on its outputs and vice versa.

Obviously, all the weighting coefficients of the synapses of one layer of neurons can be reduced to a matrix W, in which each element wij defines the value of the i-th synaptic connection of the j-th neuron. Thus, the process taking place in the NN (neural network) can be written in matrix form:

$$Y = F(XW) \tag{4.5}$$

where X and Y are the input and output signal vectors, respectively, F (V) is the activation function applied elementwise to the components of the vector V.

Theoretically, the number of layers and the number of neurons in each layer can be arbitrary, but in fact it is limited by the resources of a computer or a specialized microcircuit, on which NN is usually implemented. The more complex the NN, the larger the tasks that are subject to it.

Questions for self-control

1. Give the concept of a neural network.
2. What does multilayer neural networks mean?
3. What is the principle of construction of neural networks.
4. How is the hardware implementation of a neural network implemented?
5. Describe the structure of the neuron.
6. What are the computing capabilities of a single neuron.
7. What are the features of using neural networks.
8. Create a neural network in the task of character recognition.
9. Describe the learning process of an artificial neural network.
10. What stages does the learning process of an artificial neural network include?
11. Give the examples of projection and coding of images
12. Development of a neural network for the recognition of acoustic signals.
13. Give a block diagram of the Hopfield network.
14. How is the task formulated by the Hopfield network formulated.
15. By what the number of layers and the number of neurons in each layer of NN can be limited?

CHAPTER 5. EXPERT SYSTEMS

5.1. The concept of expert system (ES).

An expert system is a program that is able to replace a human expert in his professional activities. Structurally, the expert system consists of a knowledge base, an output engine, and a user interface [23].

The knowledge base has the same meaning as in the Prolog, i.e. consists of facts and rules. The inference engine accesses the knowledge base and converts the user's request into a response, asking him questions if necessary, using the user interface. This structure allows you to develop expert systems, adding new knowledge to it, and you do not need to rewrite the program. An empty expert system (without a knowledge base) is called an expert system shell and can be used for many subject areas.

The creation of an expert system consists in formalization, i.e. transforming expert knowledge into the form that is required for the shell of the expert system. In other words, an expert person is required who is a carrier of knowledge and is able to formulate this knowledge for inclusion in the knowledge base. This fact is decisive for choosing an expert system as a tool for solving the problem. A human expert is far from always able to present his knowledge in the form required by the format of the knowledge base. In such cases, the knowledge engineer, who is the "translator" between the expert and the knowledge base, comes into play.

Consider the operation of ES on a simple example. Let our knowledge base be used to classify animals. Suppose we saw an animal and want to identify it. Let the knowledge base contain only two animals, a zebra and a leopard. The rules for describing these animals (not in the syntax of a specific knowledge base, but in a free translation into Russian) are as follows:

"IF the animal belongs to the class of mammals AND the animal belongs to the species of predators AND the animal has black spots, THEN the animal is a leopard."

"IF an animal belongs to the class of mammals AND the animal belongs to the species of herbivores AND the animal has black and white transverse stripes, THEN the animal is a zebra."

ES begins to test hypotheses in the order they are located in the knowledge base, in this case, starting with a leopard. To establish the truth of this hypothesis, the expert system must first establish whether the animal belongs to the class of mammals. To do this, she must find the rule in the knowledge base:

"IF a female animal has mammary glands, THEN the animal belongs to the class of mammals."

Now, to establish the animal's mammals, it remains to be seen whether it has mammary glands. If there is no corresponding rule in the knowledge base, then the user himself must give an answer to this question. In this case, the expert system should ask the question: **"IS IT TRUE THAT the female animal has mammary glands?"**

If the user answers in the affirmative, then the fact "the animal belongs to the class of mammals" is considered established (for simplicity we will not consider the situation when we met a male). After this, the rule for determining the leopard requires to establish whether the animal is a predator. It is possible to establish this fact reliably only when being eaten, therefore, for humane reasons, the rule is placed in the knowledge base: "**IF the animal has claws OR the animal has fangs, THEN the animal belongs to the species of predators".**

The user did not see any fangs or claws, but, on the contrary, noticed hooves. Consequently, the leopard hypothesis is rejected, and ES proceeds to test the next zebra hypothesis. The fact "mammal" has already been established, and now we need to establish whether the animal belongs to the species of herbivores. The corresponding rule is as follows: **"IF the animal has horns OR the animal has hooves, THEN the animal belongs to the species of herbivores."**

ES will first ask the question: "IS IT TRUE THAT the animal has horns?"

The user answers negatively, and since the conditions in this rule are related by the OR operation, the following question is asked: "IS IT TRUE that the animal has hooves?"

The user answers in the affirmative, therefore, the fact "the animal belongs to the species of herbivores" is considered established.

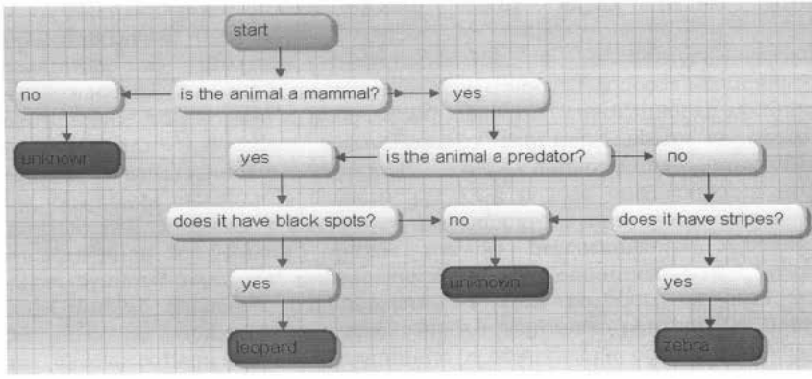


Fig. 5.1 – Example of rules submission

In order to test the zebra hypothesis, it remains to verify the last condition: “IS IT TRUE that the animal has black and white transverse stripes?” If there are stripes, then the zebra hypothesis is confirmed, and the task assigned to the expert system is considered completed. Figure 5.1 contains a fragment of a simple expert system implemented in the VISIRULE environment developed by LPA (http://lpa.co.uk).

The main requirement for the design of ES is that the user should be asked such questions that he is able to answer. In other words, an expert system converts user knowledge into expert knowledge. In the above example of animal identification, the issue of mammary glands is an obviously unfortunate question (especially if we are talking about a leopard that we accidentally met). Instead, you should use the rule: **"IF the animal has hair, THEN the animal belongs to the class of mammals."**

Another important property of the expert system is the ability to explain how one or another conclusion was obtained, which is the best confirmation of the truth of the conclusion. If we recall the stories of A. Conan-Doyle, then, like Dr. Watson, at first any conclusion of Sherlock Holmes seems paradoxical and invented, but after explaining how this conclusion was made, we are surprised that we did not thought about it.

ES are used in various fields of knowledge, including medicine (diagnostics and treatment), exploration (determining the prospects of deposits), chemistry (predicting the properties of organic compounds). Recently, expert systems have been actively used in Internet commerce to help customers choose products. The most famous is the GURU expert system on Yandex (http://market.yandex.ru/guru-categories.xml), which replaces a qualified sales assistant with the widest list of goods.

5.2. Models of knowledge representation in ES

5.2.1 The concept of data and knowledge

The information that computers deal with is divided into procedural and declarative. Procedural information is embodied in programs that are executed in the process of solving problems, declarative information is in the data with which these programs work [23].

Procedural knowledge (PK) describes the sequence of actions that can be used in solving problems (for example, computer programs, descriptions of algorithms, assembly instructions for a certain product).

Procedural knowledge can be described using an algorithmic knowledge representation model.

Declarative knowledge (DK) is all knowledge that is not procedural, namely: articles in encyclopedias, wording of laws in physics, chemistry, other sciences, etc.

When studying ES, the question traditionally arises - what is knowledge and how do they differ from ordinary data.

Data are individual facts that characterize objects, processes and phenomena of the subject area, as well as their properties.

Knowledge is based on empirical data. They are the result of a person's mental activity aimed at summarizing his or her experience obtained as a result of practical activity.

Knowledge is the laws of the subject area (principles, relationships, laws) obtained as a result of practical activities and professional experience, allowing specialists to set and solve problems in this area.

For storing data, databases are used (they are characterized by a large volume and relatively small unit cost of information), for storing knowledge - a knowledge base (a small amount, but extremely expensive information arrays). The knowledge base is the basis of any intellectual system.

Knowledge can be classified into the following categories.

Superficial – knowledge of the visible interactions between individual events and facts in the subject area.

Deep – abstractions, analogies, schemes, reflecting the structure and nature of the processes taking place in the subject area. This knowledge explains the phenomena and can be used to predict the behavior of objects.

For example.

Superficial: press the call button, a bell rings.

Deep: circuit diagram of the call and wiring.

Modern ES work mainly with superficial knowledge. This is due to the fact that at the moment there are no universal methods to identify the underlying structures of knowledge and work with them.

Features of knowledge.

1. *Internal interpretability.* Each information unit must have a unique name by which the IS finds it, and also respond to requests in which this name is mentioned. When the data stored in the memory were devoid of names, there was no possibility of their identification by the system. Only a program could extract the data, retrieving it from memory at the direction of the programmer who wrote the program. What was hidden behind this or that binary code of a machine word was unknown to the system.

If, for example, it was necessary to record information about the employees of the institution in the memory of a computer, presented in Table 1, then without an internal interpretation, a set of four machine words corresponding to the lines of this table would be entered into the memory of a computer.

Moreover, the system does not have information about which groups of binary bits in these machine words encoded information about specialists. They are known only to the programmer who uses the data in table 5.1 to solve the problems that arise. The system is not able to answer questions like "What do you know about Petrov?" or "Are there any plumbers among the specialists?".

Table 5.1

Surname	**Year of birth**	**Specialty**	**Experience, number of years**
Popov	1965	Locksmith	5
Sidorov	1946	Turner	20
Ivanov	1925	Turner	30
Petrov	1937	Plumber	25

During the transition to knowledge, information about some protostructure of information units is entered into the computer memory. In this example, it is a special machine word, which indicates in which categories the information about surnames, years of birth, specialties and length of service is stored. In this case, special dictionaries must be specified that list the surnames, years of birth, specialty, and length of service that are available in the system memory. All these *attributes* can play the role of names for those machine words that correspond to the rows of the table. On them you can search for the necessary information. Each row of the table will be an instance of the protostructure.

Currently, DBMSs provide the implementation of the internal interpretability of all information units stored in the database.

2. *Structuring*. Information units must have a flexible structure. For them, the "matryoshka principle" should be fulfilled, that is, the recursive embeddability of some information units into others. Each information unit can be included in any other, and from each information unit it is possible to distinguish some of its information units. In other words, there should be the possibility of an arbitrary establishment between separate information units of relationships such as "part-whole", "gender-type" or "element-class".

3. *Connectivity*. In the information base between information units the possibility of establishing relations of various types should be provided. First of all, these relationships can characterize the relationship between information units. Relations semantics can be declarative or procedural in nature. For example, two or more information units can be connected by the relation "simultaneously", two information units by the relation "cause – effect" or the relation "be near". The above relations characterize declarative knowledge. If the relation "argument – function" is established between two information units, then it characterizes the procedural knowledge associated with the

calculation of certain functions. Next, we distinguish between structuring relationships, functional relationships, causal relationships, and semantic relationships. With the help of the former, hierarchies of information units are defined, the latter carry procedural information that allows one to find (calculate) information units through others, the third ones specify causal relationships, the fourth ones correspond to all other relations.

Other links can be established between information units, for example, determining the order of choosing information units from memory or indicating that two information units are incompatible with each other in the same description.

The above three features of knowledge allow us to introduce a general model for the representation of knowledge, which can be called a semantic network, which is a hierarchical network at the vertices of which there are information units. These units are provided with individual names. The arcs of the semantic network correspond to various relationships between information units.

Moreover, hierarchical relations are determined by structuring relations, and non-hierarchical relations are determined by relations of other types.

4. *The semantic metric.* On a set of information units, in some cases it is useful to specify a relation characterizing the situational proximity of information units, i.e., the strength of the associative connection between information units. It could be called the relevance relation for information units. Such an attitude makes it possible to single out some typical situations in the information base (for example, “purchase”, “traffic control at the intersection”). The relevance ratio when working with information units allows you to find knowledge close to already found.

5. *Activity.* Since the advent of the computer and the separation of the information units used in it into data and commands, a situation has developed in which the data is passive and the teams are active. All processes in a computer are initiated by commands, and data is used by these commands only if necessary. For IS, this situation is not acceptable. Like a person, in the IS the actualization of certain actions is facilitated by the knowledge available in the system. Thus, the execution of programs in the IS should be initiated by the current state of the information base. The appearance in the database of facts or descriptions of events, the establishment of relationships can become a source of system activity.

The five features of information units listed above determine the line beyond which data turns into knowledge, and databases grow into knowledge bases (KB).

The combination of tools for working with knowledge forms a knowledge base management system (KBMS).

Currently, there are no knowledge bases in which internal interpretability, structuring, connectivity are fully implemented, a semantic measure is introduced, and knowledge activity is ensured.

Any subject area is characterized by its own set of concepts and relations between them, specific methods for solving problems. Knowledge of the subject area and how to solve problems in it is very diverse.

The difference between declarative and procedural knowledge can be expressed as the difference between "KNOW WHAT" and "KNOW HOW". Procedural knowledge is based on the premise that intellectual activity is knowledge of the problem environment embedded in programs, i.e. knowledge of how to use certain entities. Declarative knowledge is based on the premise that knowledge of certain entities ("KNOW WHAT") does not have deep connections with the procedures used to process these entities. When using declarative knowledge, it is believed that intelligence is based on a universal set of procedures that process facts of any type, and on many specific facts that describe a particular area of knowledge. The main advantage of declarative knowledge compared to procedural knowledge is that declarative knowledge does not need to indicate the way in which specific fragments of knowledge are used. Simple statements can be used in several ways, and it will be inconvenient to fix these methods in advance. This property provides flexibility and cost-effectiveness of declarative knowledge, as it allows different uses of the same facts.

In declarative knowledge, knowledge is considered as a set of independent or weakly dependent facts, which allows the modification of knowledge and training by simply adding or removing statements. For procedural knowledge, the problem of modification is much more complicated, since it is necessary to consider how this statement is used. However, it is known that there are a significant number of entities that are convenient to represent in the form of procedures and very difficult in a purely declarative representation. The desire to use the advantages of declarative and procedural knowledge led to the development of formalisms that use a mixed representation: declarative with attached procedures (for example, a frame representation or networks with attached procedures) or procedural in the form of modules with declarative samples. In its most advanced form, this problem is implemented in an object-oriented approach.

There are four main models for representing knowledge: a production model, a semantic network, frames, and a logical model.

The production model of knowledge representation is one of the most common in intelligent systems. The model is based on production systems.

The semantic network is a more visual way of representing knowledge. The basis of such a model is the idea of representing any knowledge in the form of a set of objects (concepts) and the relationships between them.

The frame model is also widely used in artificial intelligence systems (for example, in expert systems (ES)). A frame is the smallest possible description of the essence of a phenomenon, event, situation, process or object.

The basis of *logical models* of knowledge representation is the concept of a formal system (theory).

Examples of formal theories are predicate calculus and any particular production system. In logical models, as a rule, first-order predicate calculus is used, supplemented by a number of heuristic strategies. These methods are deductive type systems, i.e. they use a model for obtaining output from a given system of premises using a fixed system of inference rules.

Further development of predicate systems are inductive type systems in which the inference rules are generated by the system based on processing a finite number of training examples.

In logical models of knowledge representation, the relations existing between separate units of knowledge are expressed using the rather poor means provided by the syntactic rules of the formal system used.

Each of the considered models of knowledge representation can serve as the basis for creating a programming language oriented to work with knowledge. Such languages are the FRL (Frame Representation Language), which is based on frame representations, and the Prolog language, which is based on the production representation model. However, different models of knowledge representation have their advantages and disadvantages. Therefore, in the late 80's. there has been a tendency to create combined knowledge representation languages. Most often, frame and production models are combined.

5.2.2. Algorithmic Models

Consider a preliminary simple life situation: what should be done if you need to involve a person in solving the problem who is not familiar with its solution.

1. Choose a method (method, order) for solving the problem and study it in detail.
2. Inform the performer of the selected method in an absolutely understandable form for him.
3. The contractor solves the problem strictly in accordance with the method.

Going deeper into the essence of this process, let us take a closer look at each of the stages.

The first stage of this process usually does not cause difficulties, since for most of the problems encountered the solution method is either known from practice, or suggested by common sense, or described in the literature. Often

the main difficulty is to select from several methods one that would best meet some requirements, for example: minimum labor input, maximum efficiency, etc.

The second stage is much more complicated. If the method (method) for solving the problem is described arbitrarily, then there is no guarantee that it will be correctly understood by the executor. Therefore, the description of the method should be performed in accordance with certain rules, namely:

- highlight the quantities that are the source for the task;
- divide the process of solving the problem into stages that are known to the performer and can be performed without any explanation;
- indicate the order of the stages;
- indicate the sign of the end of the process of solving the problem;
- indicate in all cases what is the result of solving the problem.

The description of the method performed in accordance with these rules is called an algorithm for solving the problem. Writing such a description is usually not easy, but following it, mechanically performing all the steps indicated in it in the required order, the performer can always correctly solve the problem.

An *algorithm* is a method for solving a problem, written according to certain rules, providing uniqueness of its understanding and mechanical execution for all values of the source data (from a certain set of values), and in accordance with the sequence of the indicated actions of the algorithm for a finite number of steps, the problem is solved.

An example of an algorithm is a culinary recipe for preparing a dish.

Consider the simplest algorithm – the tea brewing algorithm:

1. Prepare the initial values – tea, water, teapot, glass, spoon.
2. Pour water into the kettle.
3. Bring water to a boil and remove from heat.
4. Pour tea into the kettle.
5. Bring water to a boil (but not boil), remove from heat.
6. Tea is ready. Process terminate.

5.2.3. Logical models of knowledge representation

The main idea of the logical approach is to consider the entire system of knowledge necessary for solving applied problems and organizing the interaction of computers with the user as a set of facts (statements).

Facts are presented as formulas in some logic (first or higher orders, multi-valued, modal, fuzzy, or some other). The knowledge system is displayed by a combination of such formulas. Being represented in a computer, it forms a

knowledge base. The formulas are indivisible and when modifying the knowledge base can only be added and deleted.

Logical methods provide a developed apparatus for deriving new facts from those that are explicitly presented in the knowledge base. The primary primitive of knowledge manipulation is the inference operation. This determines the intensive use of logical methods in creating expert systems and problem solvers. Another application of this apparatus, important for any intelligent systems, is the ability to control the logical integrity of the knowledge base, i.e., its consistency and compliance with certain predefined rules (integrity restrictions).

Logical methods for representing knowledge provide a simple and clear notation for recording facts that has well-defined semantics (at least for methods based on traditional first-order logic). Each fact is presented in the knowledge base only once, regardless of how it will be used in the future. The knowledge base developed using logical methods is usually quite simple to understand [23].

In the representation of knowledge, formal logical models are distinguished, based on the classical calculus of predicates of the first order, when the subject area and the task are described as a set of axioms.

The basis of logical models is the concept of a formal theory, given by the four:

$$S = <B, F, A, R> \tag{5.1}$$

where

B – is a countable set of basic symbols (alphabet) of the theory S;

F – a subset of expressions of the theory S, called the formulas of the theory (expressions are understood as finite sequences of basic symbols of the theory S);

A – is the distinguished set of formulas called axioms of the theory S, i.e. many a priori true formulas;

R – is a finite set of relations $\{r_1, r_2, ..., r_n\}$ between formulas, called inference rules.

Usually there is an effective procedure (many syntax rules) that allows you to build syntactically correct expressions from B – formulas.

For each r_j, there exists a positive integer j such that for each set consisting of j formulas and for each formula f, the question of whether the data of j formulas with respect to r_j with formula f is effectively solved. If the relation r_j is performed, then f is called a *direct consequence* of the given j formulas according to the rule r_j.

Inference rules allow you to expand many formulas that are considered true in the framework of this theory.

A formal theory is called decidable if there is a single effective procedure that allows one to find out for any given formula whether its conclusion exists

in *S*. *A* formal system *S* is called consistent if there is no formula *A* such that *A* and $\neg A$ are deducible in *S*.

The most common formal system used to represent knowledge is first-order predicate calculus. The predicate calculus alphabet consists of the following character set:

- punctuation marks «{», « (», «,», «)», «.», «;», «}»;
- propositional connectives $\{\neg, \vee, \wedge, \supset\}$;
- quantifier signs $\{\forall, \exists\}$;
- symbols of variables x_k, k = 1,2, ...;
- n-local functional letters: f_k^n, $k \geq 1$, $n \geq 0$ (f_k^0 called constant letters);
- n-local predicate letters (characters): p_k^n, $k \geq 1$, $n \geq 1$.

In the future, for simplicity, instead of x_k we will use u, v, x, y, z, ...; instead of f_k^0 – a, b, c, d, ...; instead of f_k^n $(n \neq 0)$ – f, g, h, ...; and instead of p_k^n – P, Q, R, S, T, V, W ...

From the symbols of the alphabet, you can build various expressions. Terms, elementary formulas (atoms) and correctly constructed formulas (or just formulas) are distinguished. Every symbol of a variable or constant letter is a term. If $t_1,...,t_n$ $(n \geq 1)$ – are terms, then $f_k^n(t_1,...,t_n)$ is a term. If p_k^n is a predicate letter, and $t_1, ..., t_n$ are terms, then $p_k^n(t_1,...,t_n)$ is an elementary formula (atom).

Atom is the simplest indivisible element.

Atom is a correctly constructed formula. If *A* and *B* are correctly constructed formulas, then $\neg A$, $A \vee B$, $A \wedge B$, $A \supset B$ are correctly constructed formulas. If *A* is a correctly constructed formula and *x* is a variable in *A*, then the constructions $(\forall x)A$ and $(\exists x)A$ are correctly constructed formulas. An expression is a well-constructed formula if it is obtained in compliance with the above rules.

In order to give the formula content, it is interpreted as a statement regarding the subject area under consideration. By interpretation we mean any system consisting of a nonempty set *D*, called the *domain of interpretation*, and some correspondence relating to each predicate letter p_k^n some n-place relation in *D*; to each functional letter f_k^n – some n – local function representing $D^n \to D$, and to each constant letter f_k^0 – some element from *D*.

For a given interpretation, the variables are thought to be "running through" all the values from the domain *D* of this interpretation, and the value "true" (T) or "false" (F) is assigned to any elementary formula. The value is assigned to the elementary formula $p_k^n(t_1,...,t_n)$ according to the following rule: if the terms of the predicate letter correspond to elements from *D* satisfying the relation

defined by this interpretation, then the value of the elementary formula will be true, otherwise it will be false. The value of a non-elementary formula is calculated recursively, based on the values of its constituent formulas. Obviously, the values of the formulas can be true or false depending on the chosen interpretation.

The main task to be solved within the framework of predicate calculus is to find out the truth or falsity of a given formula in a certain area of interpretation. Moreover, a special role is given to universally valid formulas, i.e. formulas true in any interpretation, and impracticable formulas, i.e. formulas false in any interpretation. The following fundamental deduction theorem holds: let formulas $B_1, ..., B_n$ and formula A be given. Formula A is a logical consequence of $B_1, ..., B_n$ if and only if the formula $B_1 \wedge ... \wedge B_n \supset A$ is valid, i.e. $\models (B_1 \wedge ... \wedge B_n) \supset A$. Recall that a formula A logically follows from the formulas $B_1, ..., B_n$ if and only if every interpretation of J that satisfies $B_1 \wedge ... \wedge B_n$ also satisfies A. Formulas $B_1, ..., B_n$ are called premises, and A is the conclusion of the logical sequence and are denoted by $B_1, ..., B_n \mid = A$.

The task of proving the theorem is the clarification of the logical consequence of a certain formula A from a given set of formulas $B_1, ..., B_n$, which is equivalent to proving the validity of the formula $B_1 \wedge ... \wedge B_n \supset A$ or the impracticability of the formula $B_1 \wedge ... \wedge B_n \wedge |A$.

It is known that for calculating first-order predicates there is no general method for establishing the validity of any formulas, i.e. first-order predicate calculus is unsolvable. However, if some formula for calculating predicates is valid, then there is a procedure for checking its validity, i.e. predicate calculus can be called semisolvable.

Here is an example of some fact in the form of a predicate calculus formula:

GIVE (MICHAEL, VLADIMIR, BOOK);

$(\exists x)$ (ELEMENT (x, EVENT-GIVE) $\wedge$ SOURCE (x, MICHAEL) $\wedge$

ADDRESS (x, VLADIMIR) $\wedge$ OBJECT (x, BOOK).

Two ways of recording one fact are described here: "Michael gave the book to Vladimir."

The main advantage of using predicate calculus as a model for representing knowledge is the presence of a uniform formal procedure for proving theorems. However, a high degree of uniformity entails the main disadvantage of this approach - the difficulty of using heuristic proofs that reflect the specifics of a particular problem environment. This drawback is especially important when building expert systems, the computing power of which is mainly determined by knowledge characterizing the specifics of the problem environment. Other disadvantages of formal systems include their monotony, lack of tools for structuring the elements used and the inadmissibility of contradictions.

The desire to eliminate the shortcomings of formal systems when used as presentation models led to the emergence of semiotic systems. The semiotic system is formally defined by the eight:

$$S = <B, F, A, R, Q(B), Q(F), Q(A), Q(R)>. \quad (5.2)$$

Here, the first four components are the same as in the definition of a formal system, and the remaining components are the rules for changing the first four components under the influence of the experience accumulated in the knowledge base of the intellectual system about the structure and functioning of entities in this problematic environment.

The main drawback of logical methods is the lack of clear principles for organizing facts in the knowledge base. Without isolating and consistently applying such principles, a large model becomes a poorly visible conglomerate of independent facts that are difficult to analyze and process.

5.3. Production models and modules, driven by samples

In traditional programming, commands are set in a fixed sequence. By default, after the *i-th* command is executed, the $(i + 1)$-th command is executed if the *i-th* command is not a branch command. All branch points in traditional programming are explicitly specified. Such a programming method is convenient in cases where the processing sequence is little dependent on the data being processed, i.e. then when branching is an exception, not the norm. Otherwise, the program is best viewed as a collection of independent modules driven by samples. At each step of the work, such a program analyzes the current situation and determines from the analysis of the samples which module is suitable for processing this situation.

Each sample-driven module (SDM) consists of research and modification mechanisms for one or more data structures. The range of SDM can vary widely from a simple production rule to a procedure of an arbitrary degree of complexity caused by a sample. Each SDM at the next step of work analyzes the data of the working memory, checking for the presence of structures that are compared with its sample. Systems built on the basis of SDM are called sample-driven output systems. The control functions in these systems are provided by the interpreter.

From the point of view of knowledge representation, an approach using SDM can be characterized by the following features:

– separation of permanent knowledge stored in the knowledge base and temporary knowledge stored in working memory;

– structural independence of modules, facilitating the modification and improvement of the system, which is extremely important for ES, constantly modifying their knowledge. In addition, the independence of the modules simplifies the integration of programs written by different authors;

– separation of the control scheme from modules that carry knowledge of the problem area, which allows the use of various control schemes.

The systems controlled by the samples have different designs and are classified in accordance with the restrictions imposed on the modules (Figure 5.2). If such systems consist of modules localized at the tops of the network, then they are called network-based systems.

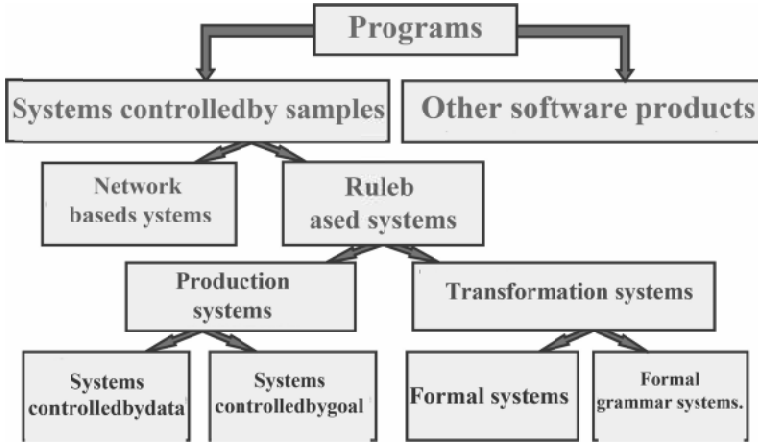


Fig. 5.2 – Classification of sample driven systems

Most sample-driven systems satisfy the following limitation: all studies of working memory data in each module are integrated and precede all data modification activities. Thus, the module is divided into two parts: a precondition that examines the data, and an action that modifies the data. Modules having this division are called rules, and systems using such rules are called *rule-based systems*.

Rule-based systems are divided by type of rule into *production* and *transformation*. Production systems are formed from rules in which comparison and planning (management) are the explicit functions of the system fixed in the interpreter. Transformation systems, unlike production ones, may not have explicit functions for matching and managing rules. Examples of transformational systems are formal and formal grammar systems.

Production systems can be divided into production systems controlled by data (preconditions of rules) and managed by goals (actions of rules). Traditionally, production systems are understood to mean only systems using data-driven output. Usually, a precondition (antecedent) is given in the form of a logical combination of statements about the working memory data, and the action (consequent) is some operation of memory modification. The complexity of the action varies significantly from a simple assignment operation to a function of an arbitrary degree of complexity.

In goal-driven production systems, preconditions and actions are data statements. Here the conclusion is carried out in the opposite direction from the statements that must be proved. It must be emphasized that samples can be given both declaratively and procedurally.

So, the representation of knowledge in the form of sample-driven modules and production rules has the following advantages:

– modularity of knowledge organization;

– independence of rules expressing independent fragments of knowledge;

– ease and naturalness of knowledge modification;

– separation of control knowledge from subject knowledge, which allows the use of various control strategies;

– the ability to create control mechanisms for a number of applications in order to automatically solve problems.

In general terms, production means an expression of the following form:

$$(i);\ Q;\ P;\ A \Rightarrow B;\ N. \tag{5.3}$$

Here i is the name of the product with which this product stands out from the whole set of products. The name may be a certain token reflecting the essence of the given product (for example, "buying a book" or "dialing a lock code"), or the serial number of the product in their set stored in the system's memory.

Element Q characterizes the scope of the product. Such spheres are easily distinguished in the cognitive structures of man. Our knowledge is sort of "laid out on the shelves". On one "shelf" knowledge is stored on how to cook food, on the other – how to get to work, etc. Dividing knowledge into separate areas saves time on finding the right knowledge. The same division into spheres in the IS knowledge base is also advisable when using production models to represent knowledge.

The main element of the product is its core: $A \Rightarrow B$. The interpretation of the core of the product can be different and depends on what is left and right of the sequence mark $\Rightarrow$. A typical reading of the product core looks like this: *IF A, THEN B,* more complex core designs allow an alternative choice on the right side, for example *IF A, THEN* B_1*, ELSE* B_2.

The sequence can be interpreted in the usual logical sense as a sign of the logical following of *B* from true *A* (if *A* is not a true expression, then nothing can be said about *B*). Other interpretations of the core of the product are possible, for example, *A* describes a certain condition necessary for action *B* to be performed.

The element *P* is the condition for the applicability of the core products. Usually *P* is a logical expression (usually a predicate). When P takes on the value "true", the core of the product is activated. If *P* is false, then the core of the product cannot be used. For example, if in the product "THE AVAILABILITY OF MONEY; IF YOU WANT TO PURCHASE A THING *X*, PAY ON THE CASH OFFICE AND GIVE A CHECK TO THE SELLER".

The condition for applicability of the product core is false, that is, there is no money, then it is impossible to use the product core.

Element *N* describes post-conditions of a product. They are updated only if the core product is implemented. Post-conditions of the product describe the actions and procedures that must be performed after the implementation of *B*. For example, after buying a certain item in a store, it is necessary to reduce the number of items of this type by one in the inventory of goods available in this store. *N* execution may not occur immediately after the implementation of the product's core.

If a certain set of products is stored in the system's memory, then they form a system of products. In the product system, special product management procedures must be defined, with the help of which the products are updated and the products selected from the list of actualized products are to be executed.

A number of IS use a combination of network and production knowledge representation models. In such models, declarative knowledge is described in the network component of the model, and procedural knowledge is described in production. In this case, they talk about the work of the production system on the semantic network.

Products along with frames are the most popular means of representing knowledge in IS. Products, on the one hand, are close to logical models, which makes it possible to organize effective output procedures on them, and, on the other hand, reflect knowledge more clearly than classical logical models. They do not have stringent restrictions characteristic of logical calculi, which makes it possible to change the interpretation of production elements.

The main disadvantage of this approach is its lower efficiency compared to traditional programming methods. Different authors classify production systems differently. Some relate them to declarative representation, others to procedural or declarative procedural. The discrepancies are explained by how broadly the concept of "production rule" is interpreted. Even in the simplest production rule (i.e., a rule that does not contain attached procedures) there is a procedural element, since it is assumed that the rule will be used to perform some action.

This is what distinguishes a procedural representation from a declarative one, since declarative knowledge does not carry any information about how it will be used. In more complex production rules, the degree of "proceduralism" is even higher. However, there is an element of declarativeness in the production rules and even in the modules controlled by the samples, since the way of using the rules and modules is not indicated in them. Thus, production rules combine the properties of both declarative and procedural representations, as well as representations in the form of frames and hierarchical networks.

The production model is most often used in industrial ES. It attracts developers with its visibility, high modularity, ease of additions and changes, and a simple inference mechanism.

Using Sample Driven Modules (SDM)

We will illustrate the use of sample-driven modules using the HEARSAY-III tool system intended for ES design and a modification of the HEARSAY-II speech understanding system.

All the methods for representing knowledge considered earlier used a special case of SDM. Indeed, each module was represented as a production rule. The complexity of the rules was very limited, which made it possible to express them in a form understandable to the expert. If the transformations performed by the module are very complex, then for their presentation it is necessary to resort to the procedural form. The desire to preserve the independence of the modules from each other led to the creation in HEARSAY-III of a circuit that ensures the interaction of the modules not directly, but through the working memory, called the "blackboard". Modules in HEARSAY-III are called Knowledge Sources (KS). Each KS consists of a condition program that determines the use of KS to the current state of the blackboard, and an action program that produces the results.

The blackboard is divided into several levels, at each of which data of a certain type is processed. The following levels are highlighted in the HEARSAY-II system: sentence, phrase, word, syllable, phoneme, etc. The search for a solution is considered by the system as an iterative process, consisting of putting forward hypotheses and verifying their credibility. The current state of the solution is presented as hypotheses on the blackboard. A hypothesis is an interpretation of some part of an oral utterance at a certain level. Hypotheses of various levels are combined into a directed graph (network), which allows to describe hypotheses of one level through hypotheses of a lower level.

So, in HEARSAY-III, working memory is represented in the form of a network, and knowledge of the problem environment is represented in the form of modules called according to the model. Using pattern-driven programs is a step towards procedural representation in an attempt to preserve the

independence of sources of knowledge. This approach (in contrast to the use of products and networks) allows us to solve much more complex problems, but reduces the ability to explain and acquire new knowledge. The use of sample-driven programs requires the development of a specific solver for each subject area, planning the decision process and using knowledge.

Use of the rules in the form of products, frames, networks allows you to create systems that focus on a specific class of tasks, while maintaining the ability to explain and acquire knowledge. However, the low power of such rules leads to a sharp decrease in efficiency in solving complex problems. For example, an experimental attempt to present a part of HEARSAY-II in the form of production rules led to a slowdown of about 1000 times. Common to all approaches considered is the use of patterns when invoking a module or rule.

Mixed representations (objects and rules)

As a rule, in ES, not one, but several representations are used. Executable statements are presented either in the form of production rules, or in the form of modules (procedures) called by the model. An object approach or network models (semantic networks and frames) are used to represent the model of subject area.

The main advantage of using object-oriented programming in the development of data processing systems is the support of methods that facilitate code reuse. However, as many researchers have noted, the effect of implementing an object-oriented programming technology begins to appear only after 5-8 years. This is due to the need to accumulate development experience and form a stable and fairly flexible class hierarchy. Obviously, such costs are unacceptable for knowledge engineering tools, where one of the defining requirements is the need to create a "quick prototype". Therefore, object-oriented tools for creating knowledge-based systems should include a library of standard, but fairly easily modifiable objects.

The implementation of the object-oriented approach in knowledge engineering systems brings to the fore another feature of it, namely: the possibility of natural decomposition of a task into a set of subtasks represented by fairly autonomous agents working with knowledge. Today this is the only practical opportunity to work in conditions of exponential growth of complexity (the number of interconnections), characteristic for systems using knowledge.

So, almost all the tools for creating dynamic ES support an object-oriented approach to system design, combined with the rules.

5.4. Work with an expert system

Let us dwell on the issue of working with an expert system. First of all, it is important to identify the circle of persons (roles) who, one way or another, are

involved in this process. It should be noted that the process of working with an expert system means both the actions necessary for its creation and direct interaction to obtain the expert knowledge stored in it. Thus, interaction with ES is carried out:

- expert in the subject area whose tasks will be solved by the ES;
- knowledge engineer – specialist in the development of ES;
- programmer – a specialist in the development of tools (IS);
- end user – a person or group of people using the system to obtain solutions.

The expert determines the knowledge (data and rules characterizing the subject area), ensures the completeness and correctness of the knowledge displayed in the ES.

The knowledge engineer helps the expert identify and structure the knowledge necessary for the operation of the ES, selects the tool (IS) that is most suitable for the given subject area, determines the way of representing knowledge in this IP, selects and programs (by traditional means) standard functions (typical for this subject area) to be used in the rules introduced by the expert.

It should be noted that the absence among the participants of the development of a knowledge engineer (i.e., his replacement by a programmer) either leads to failure in the process of creating an ES or significantly lengthens it.

The programmer develops IS containing all the main components of the ES, carries out the pairing of the IS with the environment in which it will be used.

ES works in two modes: the acquisition of knowledge and problem solving (consultation or use of ES).

In the mode of acquiring knowledge, an expert communicates with ES through the mediation of a knowledge engineer. The expert describes the subject area as a set of data and rules. Data defines objects, their characteristics and values that exist in the field of expertise. The rules determine the methods of data manipulation characteristic of the subject area under consideration. The expert, using the knowledge acquisition component, fills the system with knowledge that allows the ES in the solution mode to independently (without an expert) solve problems from the subject area. The explanatory component plays an important role in the mode of acquiring knowledge.

It is because of explanatory component that the expert at the testing stage localizes the reasons for the unsuccessful operation of the ES, which allows the expert to purposefully modify old or introduce new knowledge. Typically, the explanatory component reports the following: how rules use user information; why data or rules were used or not used; what conclusions were made, etc. All explanations are usually made in a limited natural language or graphic language.

The mode of knowledge acquisition in the traditional approach to program development corresponds to the stages of algorithmization, programming, and debugging performed by the programmer. Thus, unlike the traditional approach, the development of programs is carried out by an expert (using ES) who does not own programming, and not a programmer.

In the consultation mode, communication with the ES is carried out by the end user who is interested in the result and (or) the method of obtaining the solution.

Depending on the purpose of the ES, the user may not be a specialist in this subject area, in this case he turns to the ES for advice, not knowing how to get an answer himself, or be a specialist, in this case he turns to the ES to either speed up the process of obtaining the result, either assign routine work to ES. The term “user” is ambiguous, since in addition to the end user, an expert, a knowledge engineer, and a programmer can use ES. Therefore, when they want to emphasize that we are talking about who the ES was made for, the term "end user" is used.

In the consultation mode, data about the user's task is processed by a dialog component that performs the following actions:

1) distributes the roles of participants (user and ES) and organizes their interaction in the process of cooperative problem solving;

2) converts user data about the task, presented in the language familiar to the user, into the internal language of the system;

3) converts the system messages presented in the internal language into messages in the language familiar to the user (usually it is a limited natural language or graphic language).

After processing, the data enters the working memory (WM). Based on the input data from the WM , general data on the subject area and the rules from the KB, the solver (interpreter) generates a solution to the problem.

In contrast to traditional ES programs, in the task solution mode, it not only performs the prescribed sequence of operations, but also pre-forms it. If the response of the ES is not clear to the user, then he may require an explanation of how the answer is received.

Questions for self-control

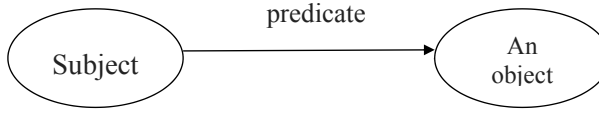
1. Give the concept of an expert system
2. What are the main features of expert systems?
3. What are the reasons that contribute to the spread of ES?
4. What is the purpose of ES?
5. By what criteria can any ES be characterized?

6. How to determine the search space and the number of active agents of the problem being solved? How does this affect the characteristics of ES?
7. Describe the ES according to the class of tasks being solved, using the following aspects: problems of expansion, extension, transformation.
8. In what areas of science and technology have ES been most widely used? Give examples.
9. Describe the range of tasks solved with the help of ES in medicine?
10. What determines the complexity of the development of ES?
11. What stages of the development of an expert system do you know?
12. What are the main components of ES? Indicate the purpose of each of them.
13. Explain the role of the knowledge base and working memory in the work of ES.
14. Explain the algorithm of the expert system in the "consultation" mode.
15. Describe the work with the expert system.

CHAPTER 6. SEMANTIC NETWORKS

6.1. Definition, historical background

The semantic network is a structure for representing knowledge using a graph, in the form of nodes connected by arcs. Nodes correspond to concepts, and arcs correspond to relations between them. The element of the semantic network in the simplest case is a triple of the following form:



One and the same concept can be present in several triplets, which determines the network structure. The main property of semantic relations is arity, i.e. number of arguments. The above is an example of a binary relation, i.e. relations with arity 2. If all relations in the network are of the same type, then the network is called homogeneous. An example of a homogeneous network is the classification of species. In a heterogeneous network, the number of relationship types is more than one.

The main purpose of using semantic networks to represent knowledge is to ensure independence from the language, as well as to eliminate the inaccuracies and ambiguities inherent in natural languages. Natural language – as a living organism, develops and evolves towards the improvement of its main goal – to ensure understanding of the participants in the conversation. However, due to the natural laziness of native speakers, this optimization often boils down to the utmost simplification of constructions, as a result of which the meaning of a single phrase can be revealed only from the contextual environment.

A textbook example of not even ambiguity, but of "triple meaning" is the phrase "He met her in a meadow with flowers". It is completely incomprehensible where the flowers are: in his, in her, or in a meadow. Ambiguity affects all languages, including English. So, the question "Tell me, what has four wheels and flies" makes us remember which aircraft has four wheels.

The semantic network must contain knowledge in a mathematically accurate form.

Mathematics allows you to describe most of the phenomena in the world in the form of logical statements. Semantic networks arose as an attempt to visualize mathematical formulas. The ancestors of modern semantic networks

can be considered the existential graphs proposed by Charles Sanders Pierce in 1909. They were used in organic chemistry to represent logical statements in the form of special diagrams. Pierce called this method the "logic of the future."

An important undertaking in the study of networks was the work of the German psychologist Otto Zelts 1913 and 1922. In them, he used graphs and semantic relations to organize structures of concepts and associations, as well as to study methods of inheriting properties. Zelts's scientific research had a huge influence on the study of tactics in chess, which in turn influenced theorists like Simon and Newell. Researchers J. Anderson (1973), D. Norman (1975) and others used these works to model human memory and intellectual properties [6].

As for linguistics, Tesniere was the first scientist involved in the development of graphic descriptions. He used graphical notation for his dependency grammar. Tesniere significantly influenced the development of linguistics in Europe. Computer semantic networks were developed in detail by Richard Richens in 1956 as part of the Cambridge Language Learning Center project on machine translation. The machine translation process is divided into 2 parts: translation of the source text into an intermediate form of presentation, and then this intermediate form is translated into the desired language. Such an intermediate form was precisely semantic networks.

The first such system, which was created by Masterman, included 100 primitive concepts such as, for example, PEOPLE, THING, DO, BE. Using these concepts, she described a dictionary of 15,000 units, which also had a mechanism for transferring characteristics from a hypertype to a subtype. Some machine translation systems were based on the Ceccato correlation networks, which were a set of 56 different relationships, some of which are case relationships, relationships of subtype, member, unit, and whole. He used networks of concepts and relationships to guide the actions of a parsing program and resolving ambiguities. These studies were continued by Robert Simmons (1966), Wilkes (1972) and other scientists [25].

In artificial intelligence systems, semantic networks are used to answer various questions, to study the processes of learning, memorization, and reasoning. In the late 70s, networks became widespread. In the 80s, the boundaries between networks, frame structures, and linear forms of recording gradually erased.

Expressive power is no longer a decisive argument in favor of choosing networks or linear forms of recording, since ideas recorded using one recording form can easily be transferred to another. And vice versa, such important factors as readability, efficiency, artlessness and theoretical elegance began to play a particularly important role, as well as ease of introduction into a computer, editing and printing.

The understanding of the problems of knowledge representation using semantic networks was greatly influenced by Drew McDermott's essay "Artificial Intelligence Encounters Natural Stupidity," which has been cited for 30 years.

The most advanced domestic development in the field of application of 70 semantic networks in the problems of analysis and search for textual information is a set of software products under the brand name RCO (Russian Context Optimizer) of "Garant Park Internet" company.

Finally, the concept of the Semantic Web should be mentioned, which will be discussed in more detail in subsections 6.7 and 6.8. The semantic Web is a further development of the World Wide Web, which consists in the fact that the documents posted on its resources contain semantic markup that allows you to extract the meaning of the information contained in these documents. Such semantic documents are created mainly in the RDF format (Resource Description Framework) - an extension of the XML language, supplemented by means of representing triplets subject-predicate-object.

6.2. Types of Semantic Networks

A semantic network in which all relationships are binary forms a relational graph (Fig. 6.1).

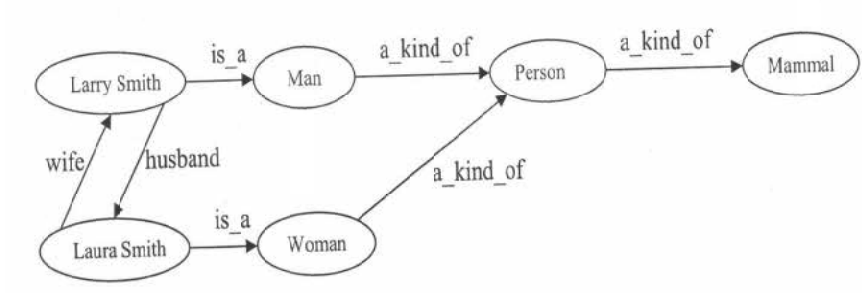


Fig. 6.1 – Example 0f Relational Graph

Despite the fact that the concepts "Larry Smith" and "Man" are denoted by the same ellipses on the graph, their semantic quality is different.

"Larry Smith" is an instance of the class of men, and "Man" is a lot of men. In turn, "Man" and "Woman" are subsets of a more powerful set of people ("Person"). Thus, as in relational databases, there are one-to-one, one-to-many, and many-to-many relationships.

If arity is different from two, it is already more difficult to depict the semantic network in the form of a graph. But, as in databases, you can normalize and bring everything to a binary relationship. For example, a unary relation like “motor – running” can be reduced to “motor” -> “state” -> “work”. The relationship with arity of type 3 “A plane flies from Petersburg to Moscow” operates with three concepts: the subject is a plane, and the objects are Moscow and Petersburg. Predicate - flight execution (“flies”). Introducing the additional concept of “flight”, this relation can be represented with the following fragment of the semantic network (Fig. 6.2):

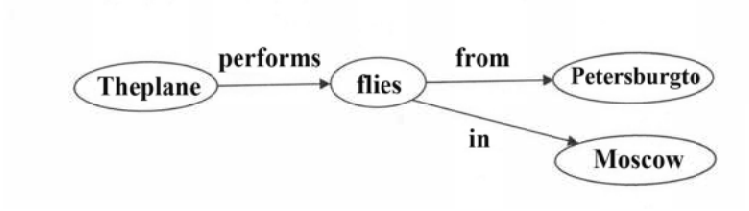


Fig. 6.2 – To the definition of a conceptual graph

In many cases, as a concept in relation, it is not a simple object or subject that is involved, but another relation or a whole fragment of the semantic network. For example, a boy sees a plane in the sky and thinks that this plane is flying from Petersburg to Moscow.

Such networks are called propositional networks, and the graph of such network is called a conceptual graph [6] (Fig. 6.3).

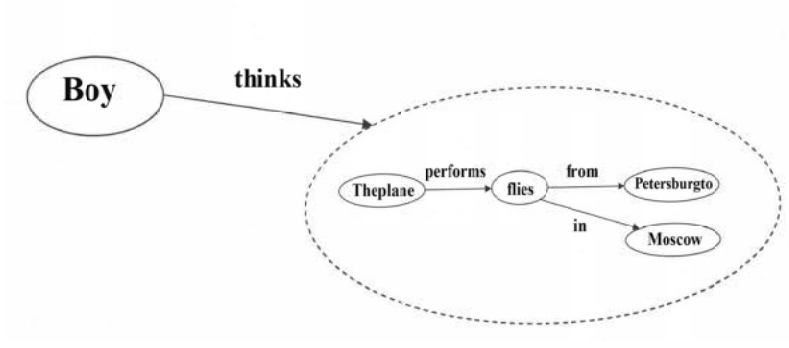


Fig. 6.3 – Conceptual graph

Nesting relationships in propositional networks can be arbitrarily large. For example, the boy's parents may believe that he is mistaken if he thinks the plane is flying from St. Petersburg to Moscow, and their grandfather and grandmother, in turn, may believe that the parents underestimate the grandson's deductive abilities, etc.

For presenting events, a graph with a verb in the center or the graph Rastier is more suitable. Example: A dog bites a postman (Fig. 6.4).

Fig. 6.4 – Graph Rastier

The basis is not a concept, but an action or event, in this case biting. The subject or biting agent is the dog, and the object is the postman. A graph centered in a verb allows you to do without nesting graphs. We can simply expand the knowledge base about this event (Fig. 6.5):

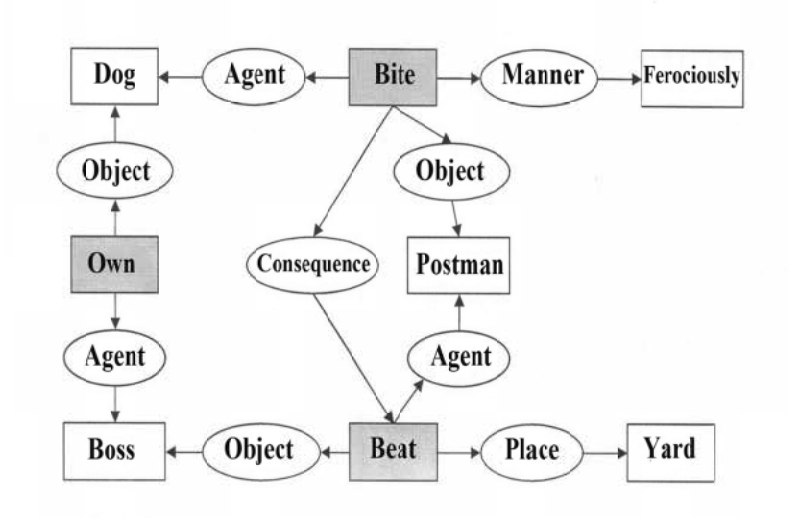


Fig. 6.5 – Semantic Network "Postman"

The consequence of the dog biting the postman fiercely was that the postman was beaten in the courtyard of the owner of the dog.

6.3. Types of Relationships in Semantic Networks

The most common type of relationship in semantic networks is the hierarchical type, which describes the relationship between elements, sets, and parts of objects. These include:

1) ISA classification relation (from English “is a”). It is said that a multitude (class) classifies its specimens (for example, “Socrates is a man”). This relationship is sometimes referred to as “member of”. In Russian, it can be called "is" (singular) or "essence" (plural). The inverse relationship is “example of” or “example”.

2) The relationship between a plurality and a subset of AKO (“a kind of”), for example, “Masters is a subset of students”. The difference from the ISA relationship is that classification is a one-to-one relationship, Postman Dog Agent Bite Object Own Object Agent Owner Object Beat Agent Consequence Place Yard Manner is fierce for many ", and a subset is" many to many ". In Russian - “subset”

3) The ratio of the whole and the part. The ratio of meronomy is the relation of the whole to the part (“has part”). Meronym - an object that is part of another object. The relation of holonym is the relation of the part to the whole (“is a part”). The hand is a holonym for the body. The body is a meronym for the hand.

Applying hierarchical types of relations, it is necessary to clearly distinguish which objects are classes and which are instances of classes. Moreover, it is not necessary that the same concept be a class or an instance in all subject areas. So, “man” will always be a class in knowledge bases such as “student group” or “labor collective”, but it can be an instance of the class of mammals in the knowledge base in biology.

The vertices of a semantic graph can denote not only objects, but also properties or property values. Displaying properties on a graph increases its visibility, but can be very cluttered.

In addition to hierarchical relations, the following types of relations are often used in semantic networks (the types of vertices are indicated in the second brackets):

1) functional relationships (“produces”, “influences”, ...) (object - object);

2) quantitative (“more”, “less”, “equal”, ...) (object - object or object - property);

3) spatial (“far from”, “close to”, “for”, “above”, “under”, “above”, ...) (object - object);

4) temporary (“earlier”, “later”, “simultaneously with”, ...) (object - object);
5) attributive (“have a property”, “have a value”, ...) (an object is a property or property is a value);

6) logical (“and”, “or”, “not”) (object - object or property - property);

7) linguistic.

The number of relationship types can be very large. The main problem with this is the possibility of identifying these relationships in queries to the knowledge base. In this regard, it is preferable to reduce the number of types of relationships (and vertices) by increasing the number of vertices.

Networks with a verb in the center (Rastier networks) operate with the following types of connections shown in Table 6.1.

Table 6.1. Types of relationships in Rastier graphs

Name	**Type**	**Definition,**	**Simplified Name**
(ACC)	accusative	Object of influence	PATient
(ASS)	Assumptive	Point of view	PERspective
(ATT)	Attributive	Property, characteristic	CHARacteristic
(BEN)	benefactive	BeneficiaryEntity	BENeficiary
(CLAS)	classitive	Class instance	CLASsitive
(COMP)	comparative	Items combined by comparison	COMParison
(DAT)	dative	Recipient	RECeiver
(ERG)	ergative	Ergative, agent of process or action	AGEnt
(FIN)	final	ResultorE expectedgoa	GOAL
(INST)	instrumental	Toolsused.	MEAns
(LOC S)	Spatial locative	Positioninspace	SPAce
(LOC T)	Temporal locative	Positionintime	TIME
(MAL)	malefactive	Sideinjuredas a result of action	MALeficiary
(PART)	partitive	Part of the whole	PARTitive
(RES)	Resultative	result, effect, consequence.	EFFect (or CAUse)

6.4. Ontologies and rules of inheritance of relations

Presenting knowledge is a very complex and creative process. The main problem here is that creating a knowledge base almost always starts "from scratch"; while there are no so-called entry-level knowledge that a person acquires, starting in early childhood. The creation of such household knowledge bases has been underway for more than 25 years by Cycorp. (www.cyc.com), however, according to its founder, Douglas Lenat, the machine still needs to be explicitly informed that parents are older than their children and that people stop writing newspapers when they die . Knowledge of a specific subject area can be divided into common for all copies and individual for each. Obviously, if the formalization of knowledge for a class of objects is performed once, then it can be used by other authors in the description of individual instances.

Thus, the formalization of knowledge of any subject area should be based on knowledge of a more general level that describes the basic relations between objects and general properties of objects. Such knowledge is called ontologies. For example, an ontology for describing the class "man" may contain the relationship "has part" with objects "hand", "head" and properties "has a name", "has a date of birth", etc. [6].

The application of such knowledge to lower-level objects is carried out using inheritance rules:

If X AKO Y and Y AKO Z then X AKO Z - if X is a subset of Y, and that, in turn, is part of an even larger set Z, then the class X is a subset of Z.

If X ISA Y and Y AKO Z then X ISA Z - if X is an instance of class Y, and that, in turn, is part of the set Z, then X is an instance of class Z.

If X has_part of Y and Y has_part of Z then X has_of part Z - if X has Y in its composition, and that, in turn, has an integral part of Z, then X has Z in its composition.

If X ISA Y and Y has_part of Z then X has_part of Z - if X is an instance of class Y, and that, in turn, has an integral part of Z, then the instance of X includes Z.

If X is AKO Y and Y has_part of Z then X has_part of Z - if X is a subset of Y, and that, in turn, has an integral part of Z, then the class X has Z in its composition.

If X ISA Y and Y has_a Z then X has_a Z - if X is an instance of class Y, and that, in turn, has property Z, then instance X has property Z.

If X is AKO Y and Y has_a Z then X has_a Z - if X is a subset of Y, and that, in turn, has property Z, then the class X has property Z.

Using inheritance rules allows the description of all properties of each instance to indicate only belonging to a class for which all necessary properties are already described.

Consider the example of translating a poetic work into a semantic network more schematically and only reflects the main meaning. For a change we will take an English-language option (we will not understand who wrote off from whom).

"The Cicada and the Ant"

Jean de La Fontaine (1988)

Cicada, having sung her song
All summer long,
Found herself without a crumb
" When winter winds did come.
Not a scrap was there to find
Of fly or earthworm, any kind.
Hungry she ran off to cry
To neighbor Ant, and specify:
Asking for a loan of grist,
"A seed or two so she'd subsist
Just until the coming spring.

She said, "I'll pay you everything
Before fall, my word as animal,
Interest and principal.
Well, no hasty lender is the Ant;
It's her finest virtue by a lot.
"And what did you do when it was hot?"
She then asked this mendicant.
"To all comers, night and day,
I sang. I hope you don't mind.
"You sang? Why, my joy is unconfined.
Now dance the winter away."

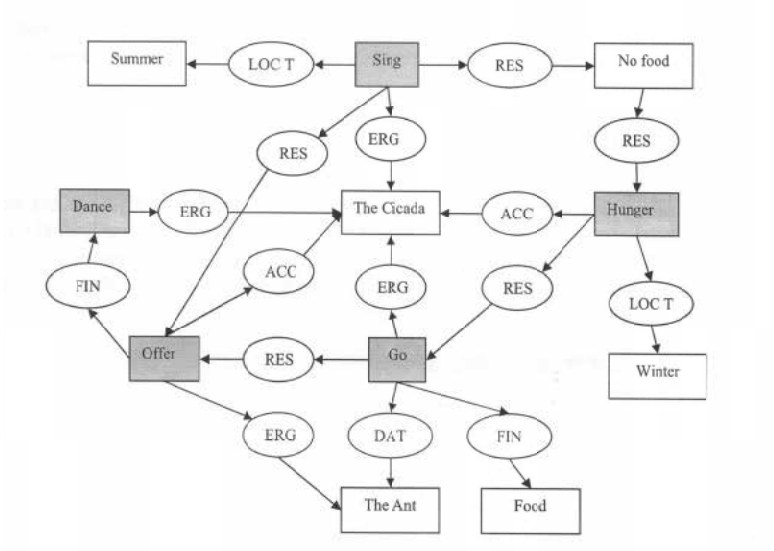


Fig. 6.6 – Graph Rastier for the poem "The Cicada and the Ant"

The graph in Fig.6.6 does not reproduce the artistic side of the work, but merely reflects the processes, events, and cause-effect relationships between them. So, we see that there is a singing process ("Sing"). The agent (ergative) of singing is cicada, and the position in time is summer. The result of singing cicada in the summer was the lack of food ("No food") and starvation ("Hunger") in the winter (position in time – "Winter"). The object of fasting ("accusative") is the cicada.

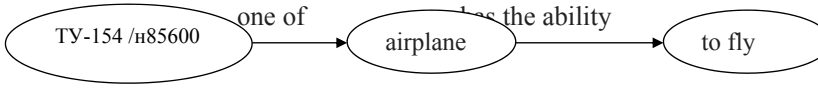
Unfortunately, on the graph, summer and winter are at diametrically opposite points, and the fact of the onset of winter after summer could not be reflected.

Due to hunger, the dragonfly came ("Go") to the ant with the goal ("Fin") to get food ("Food"). The ant suggested that the dragonfly dance ("Dance") due to the fact that she came to him and the fact that she sang in the summer.

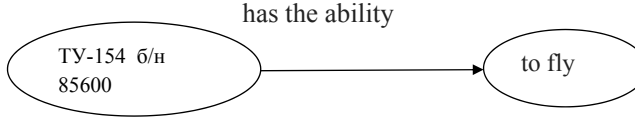
6.5. Problems of building semantic networks

As shown in Section 6.1, a semantic network must store knowledge in a mathematically accurate form. In this regard, its construction requires accuracy and a good understanding of the subject area and all related concepts. The problems of knowledge representation were described in the already mentioned work of Drew McDermott [10].

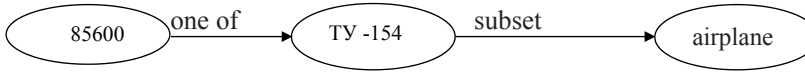
At first glance, the construction of graphs similar to those given in subsection 6.3. examples can pass easily and naturally. However, this is far from always the case. Consider the following construction:



In other words, the Tu-154 with tail number 85600 is an instance of the class of aircraft, and aircraft have the ability to fly. Since an individual representative of the class inherits the characteristics of the class, we conclude that the Tu-154 with tail number 85600 can also fly.



Another problem is the naming of concepts. In the above example with an airplane, it was not in vain that a particular airplane was identified by its tail number. Otherwise, it would not be clear which instance of the aircraft is meant.



To identify the "fly" property, the exact identification of an instance is not critical, which cannot be said when it comes to troubleshooting. The above identification of 85600 is also not exhaustive. Firstly, this combination of numbers can refer to anything, for example, a phone number. Secondly, an airplane can get into civil aviation from military aviation, where the identification is completely different, and we will not be able to find out anything about the past life of the aircraft, for example, about previous repairs, and this is completely unacceptable.

The problem of identifying concepts also occurs in real life, but here we wisely use the Occam's razor: Do not multiply entities beyond necessity. If we communicate in a small company, enough names. In the student group, each surname can be identified. At the same time namesakes are awarded a name, and namesake – also patronymic. If a semantic network is created that encompasses a large number of objects, the problem of **synonymy** inevitably arises, when one name indicates different concepts.

And if in antiquity it was enough to say "Jesus is from Nazareth" to identify a person, now even the triad "Name" – "date of birth" – "place of birth" used in passport registration does not guarantee the absence of duplicate identifiers. In addition, such a cumbersome key (in terms of databases) does not contribute to the visibility and ease of perception. In this regard, for the local semantic networks, accepted in practice test book numbers, personnel numbers, TIN, etc. can be used. Another problem is **polysemy**, when one word is used to denote various concepts.

Synonymy and polysemy can catastrophically complicate the problem of building large networks and, in particular, combining fragments written by different authors.

The name of the vertex is just a symbolic name, its meaningfulness only increases the visibility of the graph. The top of its properties is fully identified, for example, for a person – last name, first name, middle name, date of birth, etc.

6.6. Facts and Rules in Semantic Network

The relations considered above, written in the form of subject-predicate-object, are constant knowledge, i.e. facts. Entering all known facts about each object may require unreasonably long time. As an example, it is appropriate to give family relationships. For any two relatives, there is a name for the relationship between them: uncle, nephew, mother-in-law, etc. Thus, for a family of n = 10 people, the number of relationships will be n * (n-1) = 90. In this case, part of the relationship is primary (spouse and parent-child), other relationships stem from primary. If information on how secondary relations are determined on the basis of primary relations is written in the form of rules, then for each object only primary facts can be entered into the knowledge base. For family relations, this means a reduction of no less than n / 3 times, if we assume that each family member is someone's child and parent, as well as someone else's spouse, and nothing more.

One of the standards for the rule submission language is the SWRL – Semantic Web Rule Language (http://www.w3.org/Submission/SWRL/). This language is an extension of XML, and constructions on it are intended solely for machine interpretation. The rule editors usually provide a variant of the "human readable" rules in a format similar to the rules on the Prolog for creating and debugging them. The following is a fragment of the rule for determining the relationship "uncle" for human reading and the source code of this rule in SWRL:

$hasParent(?x1,?x2) \wedge hasBrother(?x2,?x3) \Rightarrow hasUncle(?x1,?x3)$ …

```
<swrl:Imp rdf:ID="Bros">
<swrl:body>
<swrl:AtomList>
<rdf:first>
<rdf:Description>
<rdf:type rdf:resource="&swrl;ClassAtom"/>
<swrl:argument1>
<rdf:Description rdf:about="#x3"/>
</swrl:argument1>
<swrl:classPredicate rdf:resource="#Person"/>
</rdf:Description>
</rdf:first>
<rdf:rest>
```

```
<swrl:AtomList>
<rdf:first>
<rdf:Description>
<rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
<swrl:argument2>
<rdf:Description rdf:about="#x3"/>
…
<rdf:Description rdf:about="#x1"/>
</swrl:argument1>
<swrl:propertyPredicate rdf:resource="#hasBrother"/>
</rdf:Description>
</rdf:first>
<rdf:rest rdf:resource="&rdf;nil"/>
</swrl:AtomList>
</swrl:head>
</swrl:Imp>
```

By setting rules for objects of the semantic network, we may encounter the problem of an open or closed world (Open or Closed World Assumption). The assumption of an open world implies that no one has complete information about the world, therefore, conclusions should be made solely on the basis of what is known. The closed world assumption assumes that all information is known to the observer. An example is the kinship relationship of the "stepmother" type. Let the following facts be in the knowledge base:

Andrey is a parent of Yegor.

Julia is the wife of Andrey.

In accordance with the assumption of a closed world, Julia is Yegor's stepmother, since there is no information in the database that she is his mother. In the open world, Julia can be considered Yegor's stepmother only if it is known that his mother is not Julia, but another woman.

Application of the rules is extremely useful in cases where the knowledge base contains incomplete information. Suppose, for example, in the previous example that Andrei is a person, but there is no information that Julia is a person. Then all the rules are like:

If X is a person, then X has a surname that cannot be applied to Julia. If you create a rule *If X is person and X spouse Y then Y is person* ,it will be possible to establish the fact that Julia is also a man. In addition to the positive effect of using the rules, there is also a drawback: combinatorial complexity, which, as the volume of the knowledge base grows, quickly grows to cosmic proportions.

So, for example, if in a fairly small knowledge base there are 100 facts and 10 rules of three facts each, the total number of attempts to apply the rules to the facts can reach 10 * 100 * 100 * 100 = 10^7, since each rule will be sequentially substituted all possible facts. Obviously, such a "naive implementation" of search in the semantic network is not viable. As in any search task, here it is necessary to solve the problem of reducing combinatorial complexity.

As an example of accelerating the processing of rules, we can use the Rete algorithm [24], the main meaning of which is to construct a tree, each node of which corresponds to a part of the conditions of the rules and stores a list of facts that satisfy these conditions. Since new facts constantly arise during the application of the rules, they are driven through the network and fact lists at the tops are updated. The bottleneck of the Rete algorithm is the large amount of memory required, since the same facts are repeatedly duplicated in lists at the vertices of the graph.

As one of the alternative solutions, you can run all the possible rules for each document once and save the results as facts. For the base of kinship relations, this will mean that at first only primary relationships (parents-children and spouses) are entered in the document, then all indirect relationships (grandchildren, etc.) are calculated from them, which then fill up the knowledge base on equal terms. After that, all facts will be retrieved equally quickly. This approach is a type of case based reasoning, and it can be considered an analogue of skills in human intelligence.

In fact, we almost always act by analogy, for example, in oral speech. If we applied the rules of the language when constructing each phrase, then the speech speed would not exceed several sentences per hour. This is especially noticeable when we translate the Russian text into a foreign language. If the linguistic construction is familiar to us (reused earlier), then the translation is proceeding at a fast pace. If we create a proposal for the first time, then the translation process slows down tens and hundreds of times.

Intelligent semantic network agent

Building a semantic network is not an easy task. But after we deal with it, the question arises, but how to extract knowledge from the semantic network? Obviously, a special program should be developed for this purpose, which, based on a user's request, will search for the required knowledge and produce a result. Currently, there are several languages for querying knowledge bases in the form of semantic networks in the RDF format, in particular, DQL, R-DEVICE, RDFQ, RDQ, RDQL, SeRQL. The most standardized language is SPARQL, which has passed standardization in the Data Access Working Group (DAWG) of the World Wide Web (W3C) [29]. There are several

implementations of the SPARQL language for various software platforms. The author tested some of them, and it turned out that queries in the SPARQL language only process facts (subject-predicate-object triplets), but do not understand the rules. Thus, all the work on creating ontologies becomes meaningless.

To eliminate this drawback, the author has developed a simplified language for the presentation of semantic documents and a program that supports the visualization of knowledge and the fulfillment of simple queries. The SEMANTIC program, offered as part of this discipline as a shell for creating and researching semantic networks, contains the rudiments of the properties of such an intelligent agent. In particular, the program applies the inheritance rules to all facts recorded in the knowledge base, and also allows the user to create their own rules.

6.7. Context management

The need to unambiguously identify all the objects of the semantic network leads not only to the complication of the procedure for adding facts, but also to the fact that the extraction of knowledge becomes very cumbersome. To simplify the understanding of this problem is not a simple example. Let us want for a day to ask our neighbor for a lecture notes on artificial intelligence, which he, in turn, borrowed from his girlfriend. Then the dialogue will be approximately as follows:

“A citizen of the Russian Federation, Vladimir Sidorov, who was born in 1985 in Saratov, having a passport No. 60 04 123456, issued on 05/05/2003 to the 51st OM of St. Petersburg, give me, citizen of the Russian Federation Petrov Ivan Viktorovich, who was born in 04/22/1986 in Pskov, having a passport No. 6606 654321, for 24 hours 00 minutes lecture notes on the discipline "Artificial Intelligence", which is read by Ph.D., assistant professor of computer engineering Bessmertny Igor Aleksandrovich, ... ".

A phrase inconceivable in everyday situations, but perfectly normal in a police report. Obviously, when creating a semantic network once you can try and identify all the objects unambiguously, although this will significantly complicate the work. But for access to knowledge it is necessary to give the opportunity to conduct a simplified dialogue, similar to the one that takes place in real life. Such a function can be assigned to an intelligent agent that provides access to knowledge.

The context base should consist of two components: permanent and temporary. A constant context is knowledge that does not change in the process of dialogue. For example, we want to know what time it is. This question,

which does not cause any difficulties for anyone, cannot be answered without information about the location of the subject. Therefore, the context database should contain information about where the subject is, as well as the time zone of this place. In other words, the context should be loaded into the database.

Temporary context – these are facts that are established or destroyed (forgotten) in the process of dialogue, as well as temporary associations established to simplify the dialogue. Temporary facts are, for example, answers to questions that were asked earlier, i.e. knowledge brought from outside and not requiring preservation in the knowledge base. An example of such facts can be the patient's answers to the questions of a doctor who is trying to make a diagnosis. The absence of such a memory will make the dialogue look like numerous jokes about sclerotic people. Temporary associations make it possible to assign short names to objects or facts for use only in this dialog.

Temporary associations are widely used both in everyday life and in documents. For example, in the texts of contracts, a turnover of the type "LLC HORNS AND HOOPS represented by Director Funt A.A., acting on the basis of the charter, hereinafter referred to as the BUYER ..." is usually used.

Thus, the context base will allow creating for the user a simplified representation (model) of the semantic network, which will allow dialogue in the usual way.

6.8. Semantic Network and Semantic Web

As shown in Subsection 6.2, the semantic network has a long history, and until recently, this concept did not cause any ambiguities. However, after the inventor of the World Wide Web (WWW) Tim Berners Lee in 2001 proclaimed the concept of "Semantic Web," in Russian-language literature, these two concepts began to mix. To eliminate misunderstandings, it is proposed to translate the concept of "Semantic Web" as "Semantic Web" or, in this text, simply "Web".

The essence of the idea of Tim Berners Lee is to provide Internet resources with special metadata available for computer processing and uniquely characterizing the properties and content of the World Wide Web resources, instead of the currently used text analysis of documents. In particular, it is proposed to provide all Internet resources with tags that enable the author of each document to be identified. It should be noted that this decision was made in pursuit of the WWW concept after the Internet turned from a knowledge repository into an informational "garbage dump" due to the permissible anonymity of resources.

However, this property of the semantic web is of last interest to us. Our focus is on the properties of the Semantic Web to deliver to the user the information that he wants to receive.The main difference between the Semantic Web and WWW is that the links between WWW resources reflect the placement of documents, and the links of the Web reflect content. True, in both the WWW and the Web, links contain resource addresses, but in WWW the links are anonymous; their meaning should be understood by the user.

On the Web, links have an explicitly indicated meaning, which is available for machine processing. Thus, the Semantic Web implies a robotic search for information instead of the current Web surfing, when the Internet user clicks on links from document to document [25].

6.9. Semantic Web: principles and current status

The Semantic Web is based on the same triplet subject-predicate-object. The difference is in the presentation of each of the elements of the triplet. On the Web, the subject, object, and predicate are represented by universal resource.

An example is the graph shown in Fig. 6.7.

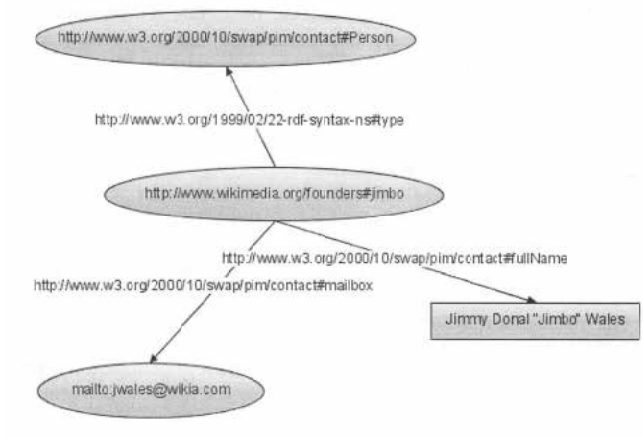


Fig. 6.7 – Wikipedia founder business card graph

The most popular knowledge representation language on the Web is RDF. A fragment of the RDF document describing the relationship Sergei has a child, Nikita, and Nikita has a parent, Sergei, is given below.

<Person rdf:ID="Nikita">

<HasParent rdf:resource="#Sergei"/>

</Person>

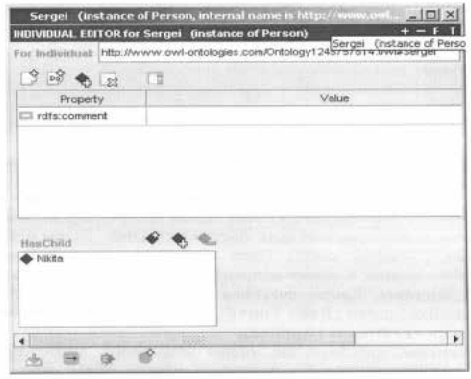
<owl:Class rdf:ID="Person"/>

<Person rdf:ID="Sergei">

<HasChild rdf:resource="#Nikita"/>

</Person>

This format is not intended for human reading. Semantic document editors, for example, the most common Protégé ontology editor (http://protege.stanford.edu/), provide the ability to enter and edit facts in screen forms (frames), as shown below.



Currently, the creation of RDF / OWL documents is carried out by individual enthusiasts. The number of documents on the Internet is small, and they can be found using a special search server, for example, SWOOGLE. Project development is being held back for many reasons. This is mainly the lack of quick and direct benefits from the creation of semantic resources, the complexity of formalizing knowledge and the lack of universal agents for extracting knowledge. The sales of intelligent agents on the Internet are limited to student developments like a beer guide in Southampton (http://www.twine.com/item/11by40gxk-p1/southampton-pub-guide-in-rdf) and a wine agent advising which wine better to use with each of the dishes.

CONCLUSION

Artificial intelligence is the property of intelligent systems to perform creative functions that are traditionally considered the prerogative of a person, it is a system of tools, one might even say a set of mathematical algorithms that solves a very narrow and complex task. An unmanned vehicle can travel from point A to point B, but it is not able to answer the question of how much 6 will be multiplied by 7. We can put another tool that will answer this question, but they will not be related to friend. Therefore, you can make some kind of robot that will go and dig a mine on the button, you can make a robot that will fly in an airplane instead of a pilot. But these will be two different robots, and there is no common robot that controls them – a person does this much more efficiently. In complex tasks where you need to set goals, understand what you need to do or not do, artificial intelligence infinitely loses to a person. Yes, probably, the robot will dig mines better than the sapper, at least it really will not be so sorry if it fails to dig. In airplanes, maybe they will fly, things will go that shoot more accurately.

Artificial intelligence – the ability of computer systems to take on individual functions of human intelligence, for example, to choose and make optimal decisions based on knowledge gained from previous experience and a rational analysis of external influences.

In this definition, the term knowledge means not only the information that enters the brain through the senses. This type of knowledge is extremely important, but insufficient for intellectual activity. The fact is that the objects of our environment have the property not only to affect the senses, but also to be in certain relationships with each other.

It is clear that in order to carry out intellectual activity in the environment (or even just exist), it is necessary to have a model of this world in the knowledge system. In this information model of the environment, real objects, their properties and the relationships between them are not only displayed and remembered, but also, as noted in this definition of intelligence, can be mentally transformed purposefully.

Thus, Artificial Intelligence is the ability of a computer system to create programs (primarily heuristic) during self-learning to solve problems of a certain class of complexity and solve these problems in all areas of our lives.

In practice, the range of capabilities of AI is almost endless: space research, military science, robotics, industry, agriculture, transport, medicine, education, etc. Within the framework of the state program Digital Kazakhstan, much attention will be paid to digitalization and robotization of production processes in all sectors of the national economy. An important role in the implementation of this program is played by the development of AI methods and tools. The use of AI in practice is discussed in the second volume of this tutorial.

Questions for self-control

1. Give the concept of an expert system
2. What are the main features of expert systems?
3. What are the reasons that contribute to the spread of ES?
4. What is the purpose of ES?
5. By what criteria can any ES be characterized?
6. How to determine the search space and the number of active agents of the problem being solved? How does this affect the characteristics of ES?
7. Describe the ES according to the class of tasks being solved, using the following aspects: problems of expansion, extension, transformation.
8. In what areas of science and technology have ES been most widely used? Give examples.
9. Describe the range of tasks solved with the help of ES in medicine?
10. What determines the complexity of the development of ES?
11. What stages of the development of an expert system do you know?
12. What are the main components of ES? Indicate the purpose of each of them.
13. Explain the role of the knowledge base and working memory in the work of ES.
14. Explain the algorithm of the expert system in the "consultation" mode.
15. Describe the work with the expert system.

GLOSSARY

A

Abduction – is a form of synthetic inference that deduces the premises from the rule and result.

Abstraction – is the mental selection (understanding) of a property or relationship by distracting from other properties or relationships of an empirical object.

The **acquisition of knowledge** – the identification of knowledge from sources and their transformation into the desired form, as well as the transfer to the IIS knowledge base.

Amphibole (from Greek amphibolos – ambiguity, duality) – a logical error, based on the ambiguity of language expressions.

Analysis – is the mental division of an empirical or abstract object into its structural components (parts, properties, relationships).

An atom – is a predicate form or some equality, i.e. expression of type (s = t), where s and t are terms.

Artificial intelligence (AI) is one of the areas of computer science, the purpose of which is the development of hardware and software tools that allow a non-programmer to set and solve their tasks, which are traditionally considered intellectual, communicating with computers in a limited subset of the natural language.

C

Clipping is a way to control your search.

Comparison – the establishment of similarities or differences between objects.

A **complex concept** is a concept formed from previously defined by the application of certain rules.

Compound lists are lists that use more than one type of item.

D

Deep knowledge – abstractions, images, analogies, which reflect the understanding of the structure of the subject area and the relationship of individual concepts.

Deduction is an analytical process based on the application of general rules to particular cases, with the conclusion of the result.

Declarative knowledge – facts in the form of sets of structured data.

E

An **entity** is an object of an arbitrary nature belonging to the real or imaginary (virtual) world.

An **expert** is a highly qualified specialist who agreed to share experience in the subject area under consideration.

Expert systems (ES) (or **knowledge engineering**) – the direction of artificial intelligence, the task of which is to research and develop programs (devices) that use knowledge and output procedures and solve problems that are difficult for human experts.

Explanation subsystem – a program that allows the user to receive answers to questions: how was this or that recommendation received and why did the system make such a decision?

Extensional – definition by listing the concepts of a lower level of the hierarchy or facts related to the defined.

F

Facts – 1) well-known circumstances; 2) relationships or properties that are known to have the meaning of "truth."

A **formal system** is a set of purely abstract methods, in which the rules for operating with a variety of characters in a purely syntactic interpretation are presented without taking into account the semantic content.

A **frame** is a minimal information structure necessary to represent knowledge about stereotyped classes of objects, phenomena, situations, processes, etc.

A **functional constant** is a logical predicate that, when combined with a suitable number of terms, forms a functional form, the semantic region of which is a set of individual constants.

Functional integrity – the ability to select the desired result, time and means of obtaining the result, means of analyzing the sufficiency of the result.

G

Generalization – the mental selection of a concept by comparing any other concepts.

Generalization of knowledge is the process of obtaining knowledge that explains the facts and is able to explain, classify or predict new ones.

H

Heuristics – knowledge from the experience of experts.

Hopfield networks are a subset of feedback networks that are guaranteed to reach a steady state.

I

Individual constants and **individual variables** (in logic) are similar to constants and variables from mathematical analysis, with the only difference being that the area of their change is individuals, not real numbers.

Induction – synthetic reasoning that derives a rule based on the premises and the result.

Inference is the derivation of a formula based on many other logical formulas by applying inference rules.

Inference is a complex abstract object in which, using certain relationships, one or more propositions are combined into a single whole.

Intelligence – 1) the ability of the brain to solve (intellectual) problems by acquiring, remembering, and purposefully transforming knowledge in the process of learning from experience and adapting to various circumstances;

2) the ability to independently, efficiently (true, with the lowest possible cost of resources) find high-quality (true, simple, requiring the lowest possible cost of resources) solutions (including new, previously unknown) of various complex "tasks", including new ones previously unknown (ideally, any possible "tasks").

An **intelligent system** is an information and computing system (ICS) with intellectual support in solving problems without the participation of an operator (decision maker – decision maker).

Intelligent reference books are information systems, the core of which is a generator of subject knowledge, and the purpose is to generate reference data in a narrow subject area.

Intelligent agent – a system (person, program) with intellectual abilities.

J

Judgment is a structurally complex object that reflects the objective connection between the object and its property.

K

Knowledge base (KB) is the core of IP, the body of knowledge of a subject area recorded on a computer medium in the language of knowledge representation (usually close to natural).

Knowledge is the identified laws of the subject area (principles, relationships, laws), allowing to solve its problems.

Knowledge Engineer (Cognitologist, Interpreter) – AI Specialist, acting as an intermediate buffer between the expert and the knowledge base.

Knowledge Base Management System (KBMS) – a set of tools that provide work with knowledge.

L

A **list** is a data object containing a finite number of other objects.

List structure (Lisp) is a list, the elements of which can be both atoms and other list structures, including ordinary lists.

Logic – 1) the science of the forms and methods of proper thinking; 2) the science of universal (universally valid) relationships between concepts, judgments, conclusions and other abstract objects.

A **logical connective** (in the predicate calculus) is a logical operation that serves to formulas.

Logical programming is one of the approaches to computer science, in which the logic of first-order predicates in the form of Horn phrases is used as a high-level language.

M

Meta-knowledge is a knowledge about knowledge: about the volume and origin of knowledge about a particular object, about the reliability of specific information, or about the relative importance of individual facts.

N

Neurocomputer is a software and hardware system that implements some formal model of a natural neural network.

A **neural-like network** is a collection of neural-like elements connected in a certain way to each other and to the external environment.

P

Parameter Stream (Visual Prolog) – a list of input and output arguments for this predicate.

Perceptron is a device for pattern recognition.

Predicate – 1) it is a predicate name along with a suitable number of terms; 2) (in Prolog) the name of a property or relationship between objects with sequence of arguments.

A **predicate constant** is a relation designation that describes a predicate. The predicate constant does not change its value of truth.

A **predicate form** is a predicate constant combined with a suitable number of terms.

Procedural knowledge – algorithms in the form of fact processing procedures.

R

A **recursion basis** is a sentence that defines a certain initial situation or situation at the time of termination.

The **request** (in the Prolog) is the goals that are given to the program for execution.

A **recursive procedure** is a procedure that calls itself until a condition is met that stops the recursion.

The **recursion step** is a rule, the body of which necessarily contains, as a subgoal, the call of the defined predicate

A **rule** (in Prolog) is a conclusion that is known to be true if one or more of the other conclusions or facts are true.

S

A **semantic network** is a directed graph whose vertices are concepts, and arcs are the relationships between them.

Situations – all kinds of interactions between objects.

Solver (logical inference machine) – a program that simulates the expert's reasoning based on the knowledge available in the knowledge base.

The **subject area** is the part of reality associated with the solution of a problem.

A **statement** is a predicate constant with no arguments, or a zero-place predicate form.

Superficial knowledge is a combination of empirical associations and causal relationships between concepts of the subject area.

Synthesis is the mental union of various objects into a holistic object.

A **system with intelligent support** is a system capable of making decisions independently.

The **synthesis technology** of an expert system is the technology of creating, based on the knowledge of experts, systems that solve informal problems in poorly structured subject areas.

T

Task is a class of problem situations when it is necessary to carry out: 1) collection of information; 2) an assessment of the situation; 3) decision making; 4) the implementation of actions.

The **tail** of the list is part of the list, including all subsequent elements.

A **term** is every variable and every functional form.

A **tree** is a graph with one root vertex, the remaining vertices have only one father, and all vertices are descendants of the root vertex.

U

Unification – coincidence of a goal with a rule head or fact.

Unformalized tasks are tasks that have one or more of the following characteristics:

1) they cannot be specified in numerical form, i.e. are given in a qualitative form or in terms of the theory of fuzzy sets;
2) goals cannot be expressed in terms of a precisely defined objective function;

3) there is no algorithmic solution to problems;
4) an algorithmic solution exists, but it cannot be used due to limited resources (time, memory).

User – the person for whom the system is intended.

ӘДЕБИЕТТЕР ТІЗІМІ
LITERATURE

1. Стюарт Рассел, Питер Норвиг. Искусственный интеллект. Современный подход. // Учебник, второе издание, Санкт-Петербург, 2016. – 258 с.
2. www.swi-prolog.org. Официальный сайт разработчиков транслятора SWIProlog.
3. О.Е.Масленникова, И.В.Попова. Основы искусственного интеллекта: учеб. пособие / О.Е.Масленникова, И.В.Попова. – Магнитогорск: МаГУ, 2008. – 282 с.
4. Turing Alan «Computing Machinery and Intelligence». //Mind LIX (236): 433-460, doi: 10.1093/mind/LIX.236.433, ISSN 0026-4423, retrieved 2008-08-4.
5. Нейронная сеть Хопфилда. //Материал из Википедии – свободной энциклопедии. http://en.wikipedia.org/wiki_algorithm.
6. О.Е.Масленникова, И.В.Гаврилова. Основы искусственного интел-лекта // Учеб. пособие, издательство «Флинта».2013. – 265 с.
7. Люгер Дж.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем = Artificial Intelligence: Structures and Strategies for Complex Problem Solving / Под ред. Н.Н.Куссуль. – 4-е изд. – М.: Вильямс, 2005. – 864 с.
8. Стюарт Рассел, Питер Норвиг. Искусственный интеллект: Современный подход. 2-е изд.: пер. с англ. // – М.: Изд.дом «Вильямс», 2006. – 1408 с.
9. Алан Кулмероэ и Филипп Русселом. ЭОР SWI-Prolog.pdf. https: // do.kpfu.ru › pluginfile.php › course › overviewfiles.
10. И.А.Бессмертный. Искусственный интеллект. // – СПб: СПбГУ ИТМО, 2010. – 132 с.
11. Клоксин У., Меллиш К. Программирование на языке Пролог //[PDF] https://www.twirpx.com
12. Батыршин И.З. Основные операции нечеткой логики и их обобщения. // Казань: Отечество, 2001. – с.100.
13. Белоусов Р.Л., Дрожжин Н.А., Костенчук М.И. Построение нечетких лингвистических переменных с использованием методов кластерного анализа данных. // Журнал «Прикладная информатика», №1(55), 2015. – с. 67-74.
14. Jang, J.-S. R., «ANFIS: Adaptive-Network-based Fuzzy Inference Systems, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, pp. 665-685, May 1993.

15. Парадокс Монти Холла – Википедия. https://ru.wikipedia.org › wiki.
16. Зайцев Д.А.; Сарбей В.Г.; Слепцов А.И. Синтез функций непре-рывной логики заданных таблично (рус.) // Кибернетика и системный анализ : журнал, 1998. № 2. – С. 47-56.
17. Роберт Каллан. Основные концепции нейронных сетей. Журнал «Информатика и вычислительная техника», №55, 2003. с. 45-52.
18. Д.Рутковская, М.Пилиньский, Л.Рутковский. Нейронные сети, генетические алгоритмы и нечеткие системы. Издательство: Горячая Линия – Телеком, 2006. – 243 с.
19. Баймухамедов М.Ф., Тажиева Ш.Ж. Применение нейронной сети для моделирования сложных технологических процессов. // Журнал «Актуальные научные исследования в современном мире» ISCIENCE.IN.UA, Выпуск 6(50), Переяслав-Хмельницкий. – с. 67-74.
20. В.В.Круглов, М.И.Дли, Р.Ю.Голунов. Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2000. – 224 с.
21. А.Б.Барский. Нейронные сети: распознавание, управление, принятие решений. Издательство: Финансы и статистика, 2004. – 221 с.
22. McDermott, Drew. “Artificial Intelligence Meets Natural Stupidity”, SIGART Newsletter, No.57 (April, 1976), pp. 4-9.
23. Баймухамедов М.Ф., Герауф И.И. Экспертные системы.//Учебник, Изд-во «Костанайский печатный двор», Костанай, 2007 г. – 262 с.
24. http://en.wikipedia.org/wiki/Rete_algorithm
25. https://newtonew.com › book › artificial-intelligence-books.
26. Боранбаев С.Н. Теория информационных систем. //Астана: Елорда, 2006. – 212 с.
27. Боранбаев С.Н. Математические модели распределения ресурсов. // Астана: Елорда, 2006. – 216 с.
28. Боранбаев С.Н., Бигаринов Р.А. Информационные системы поддержки принятия решений. // – Астана: ЕНУ имени Л.Н. Гумилева, 2010. – 220 с.

Баймухамедов М.Ф.

ЖАСАНДЫ ИНТЕЛЛЕКТ: ҚАЗІРГІ ЗАМАНҒЫ ТЕОРИЯ ЖӘНЕ ТӘЖІРИБЕ

1 Бөлім

ARTIFICIAL INTELLIGENCE: MODERN THEORY AND PRACTICE

Volume 1

ISBN 978-601-7991-32-6

Компьютерде беттеген және мұқаба дизайнін жасаған – **Любовицкая Ольга**

Басуға 2020 жылы қол қойылды.
Форматы 60x84 1/16. Көлемі 15,5 баспа табақ.
Times гарнитурасы. Офсеттік басылым.
Тапсырыс № ___. Тиражы – 500 дана.

«Бастау» баспасы
Мемлекеттік лицензия – № 0000036
ҚР Білім және ғылым министрлігі.
ҚР Ұлттық мемлекеттік кітап палатасының
халықаралық код беру туралы №155 –
978-601-281 сертификаты.
Қазақстан Республикасы Ұлттық бизнес-рейтингінің
«Лидер отрасли – 2018» ұлттық сертификаты.
Алматы қаласы, Сейфуллин даңғылы, 458/460-95.
Тел.: 279 49 53, 279 97 32.

«Полиграфсервис» баспаханасында басылды (тел.: 233 32 53).
Алматы қаласы, 050050, Зеленая көшесі, 13-а.