**Teма 12. Решаване на задачи чрез дедуктивен извод върху правила**

Както беше посочено в Тема 3, в БИС намират приложение различни видове интелигентни технологии. Те се използват главно за Data Mining, прогнозиране, логически (дедуктивен) извод върху явни знания, предимно във форма на правила.

В архитектурата на БИС (Тема 2) участват модули база от знания (БЗ) и машина (механизъм) за извод. Те не са задължителни, но подпомагат вземането на решения при неструктурирани задачи. Например, при диагностиране на възникващи проблеми във фирмата, трябва да се определи какъв вид са и да се дадат препоръки за решаването им.

Дедуктивният (логическият) извод се изпълнява върху база от знания, състояща се от правила, които представляват истинни зависимости. Той позволява да се докаже или отхвърли истинността на свързана с тях цел, с което да се установи, че тя е следствие на зададената база от правила. Дедуктивният извод ще бъде разгледан на основата на езика Prolog.

Prolog е декларативен (описателен) език, много мощен и лек за изучване. Главната разлика от класическите алгоритмични езици е в **начина на описване на проблемите**. Описват се структурата на обектите и отношенията между тях във вид на факти и правила. Зададените факти и правила образуват **базата от знания** на програмите. Базата се интерпретира дедуктивно (т.е. чрез логически изчисления) от вградените изчислителни механизми на езика. Самата програма **представлява база от знания + дедуктивен механизъм**. Не се определя алгоритмична последователност от елементарни действия за обработка, както това е присъщо на традиционните езици.

За изразяване на връзки и отношения между обектите се използват **предикати**. Те имат следния вид:

отношение(аргумент1, аргумент2,.... аргументn).

Например, предикатът ima(student,kniga) изразява изречението „Студент има книга“. Най-често като отношение се записва сказуемото в изречението.

Изразните средства на езика, основните „думи“, се наричат терми. Те могат да бъдат прости- символни и числови и съставни- структури, списъци и формули. От друга страна термите могат да бъдат константи и променливи. **Променливите** се използват за записване на свойства, отнасящи се към групи обекти. В процеса на съпоставка, използван от метода на резолюцията, те може да се свързват с различни по тип и структура обекти. Записват се с букви или думи, започващи с главна буква или долно тире. Например, в предиката student(X), аргументът Х е променлива.

Програмата на Prolog е съвкупност от клаузи. Клаузите са основните логически единици в езика. В Prolog клаузите са в три разновидности- правила, факти, въпроси (цели). Всяка клауза завършва с точка.

**Правилата** в Prolog описват условно верни отношения, т.е. зададената релация е в сила при определени условия. Правилото има следния общ вид:

А:- B1, B2, …, Bm.

За да бъде целта А истина, е необходимо целите B1, B2, …, Bm също да са истинни. Ако е целите B1, B2, …, Bm са истинни, за целта А се казва, че успява, а в противен случай - че пропада.

Целта А се нарича **глава** на правилото, а целите B1, B2, …, Bm са **тяло**. Пример за правило:

uspeh(X):- obrazovan(X), trudolubiv(X).

Смисълът на записването е “Х има успех, ако Х е образован и Х е трудолюбив”.

Клауза без тяло се нарича **факт**, т.е. записването А:-. е факт. Съвкупността от факти в програмата представлява нейната база от данни.

Например: obrazovan(ivan).

Клауза от вида :- B1,B2,…,Bm се нарича **въпрос** или цел на програмата. Тя се записва така: ?- B1,B2,…,Bm.

Целите в програмата са свързани с предикатите, които се срещат в програмата. Например, ако се разгледа горното правило, може да се поставя като цел за доказване: ?- uspeh(X).

При логическия извод се използва методът на резолюцията[[1]](#footnote-1), заедно с предикатната логика от първи ред, които са в основата на езика Prolog.

Поставянето на въпрос (цел) стартира механизмът за удовлетворяване на цели. Търси се удовлетворимостта[[2]](#footnote-2) на записаната формула за някакви стойности на променливите в нея. Ако има такива стойности, Prolog отговаря с “yes” и при задаване на предикат за извеждане на тези стойности, те могат да се видят. Ако формулата в целта не може да се докаже с използване на текущата база данни, Prolog извежда “no”.

Въпросите може да съдържат една или няколко цели, свързани чрез конюнкция (връзка „и“, отбелязана със „ ,“) или дизюнкция (връзка „или“, отбелязана с „ ; “). При конюнктивна връзка въпросът успява, ако всяка от целите в него е успее, а при дизюнктивна връзка успява, ако поне една от целите успее.

Резолюцията в Prolog е свързана със съпоставянето на терми. Те са еднакви, ако представляват последователности от едни и същи символи или могат да се направят еднакви при подходящи замествания. В сила са следните правила за съпоставяне на два терма:

-Ако терм 1 и терм 2 са константи, те съвпадат само ако са записани еднакво;

-Ако терм 1 е променлива, а терм 2 е произволен терм, те винаги се унифицират като терм 1 се свързва с терм 2.

Два предиката са съпоставими, ако предикатните имена (отношението пред скобите) са еднакви и броят на аргументите им е еднакъв.

Нека разгледаме пример за програма на Prolog. В клаузите по- долу фактите 1- 3 описват кой студент в каква специалност учи. С целта, зададена във въпроса, се търси кой е този студент Х, който учи в специалност “Информатика”. Последният предикат в целта – write(X), е вграден в Prolog и служи за извеждане на екран на свързаната стойност за Х.

?- uchi(X,informatika),write(X).

uchi(krasen,informatika). /\* 1 \*/

факти

uchi(ana, marketing). /\* 2 \*/

uchi(milen,turizym). /\* 3 \*/

Фактите изразяват сведения за различни обекти. Те могат да се разглеждат като конкретизации на предикати със същите предикатни имена и с константи като аргументи.

За да отговори на въпроса, Prolog стартира претърсване на базата данни, започвайки от първата записана клауза. Ако се намери факт, съпоставим с целта, променливата се свързва със съответния аргумент, независимо от какъв тип е той. Тогава се казва, че целта успява. В разглеждания пример Х се свързва с “krasen”. Мястото на този факт в базата данни се маркира с указател за първата цел. Ако липсва подходящ факт, Prolog отговаря с “no” и целта пропада.

Съвпадението на целта с факт, незабавно удовлетворява целта, докато съвпадението й с правило, по-точно с лявата му страна, свежда задачата към нова последователност от цели- тези, зададени вдясно, т.е. за да успее едно правило, целите зададени в дясната му страна, трябва да успеят. Работата с конюнкция от цели както в правило, така и във въпрос започва отляво надясно. За всяка успяла цел се маркира мястото на съвпадение с факт от БД. При това, ако една променлива се свърже с определен обект, всяко нейно участие в клаузата представя същия обект. Така се осъществява предаване на параметри към предстоящите за удовлетворяване цели, съвместими с предишните цели.

Друг основен механизъм на изпълнителната система на Prolog е механизмът за възврат- backtracking. Идеята е следната: при опит за доказване на конюнкция от цели в правило или въпрос, Prolog започва с най - лявата и последователно разглежда следващите. При пропадане на една цел се прави опит за ново удовлетворяване на левия й съсед. Ако той пропадне, се преразглежда неговия ляв съсед и т.н. Ако при това пропадне първата цел, цялата конюнкция пропада. При ново удовлетворяване на дадена цел, Prolog се опитва да намери друга клауза, съвпадаща с целта. Търсенето в базата данни се изпълнява след маркера за място, поставен при предишното удовлетворяване на същата цел. Преди да започне търсенето се освобождават всички променливи, свързани при удовлетворяването на целта. Ако опитът за намиране на алтернативно решение успее, новата съвпаднала клауза се маркира и отново се опитва удовлетворяване на целта вдясно. Този механизъм се нарича **възврат – backtracking**. Работата му се илюстрира със следващия пример.

Пример: Какви общи интереси имат Иван и Николай? Дадени са факти, изразяващи кой от какво се интересува.

?- interes(ivan,X), interes(nikolaj,X),write(X).

interes(ivan,knigi). /\*факт 1\*/

interes (ivan,muzika). /\*факт 2\*/

interes(nikolaj,muzika). /\*факт 3\*/

interes(nikolaj,filmi). /\*факт 4\*/

При удовлетворяване на въпроса поведението на Prolog е следното:

1. Първата цел от въпроса се удовлетворява от факт 1 и Х се свързва с “knigi”. Поставя се маркер на факт 1, който е невидим за потребителя.

2. Втората цел се активира във вида interes(nikolaj,knigi). и не се удовлетворява, тъй като в базата няма такъв факт.

3. Дедуктивният механизъм прaви възврат и реактивира първата цел. Х се освобождава и претърсването на базата данни продължава след поставения маркер, където Х е свързана с “knigi”. Целта алтернативно се удовлетворява с факт 2 и Х се свързва с “muzika”.

4. Втората цел във въпроса се активира във вида interes(nikolaj,muzika). и се удовлетворява oт факт 3. Така целият въпрос успява при стойност на X “muzika”. Отговорът на Prolog е “muzika”.

Вграденият механизъм за възврат гарантира, че всички възможни решения на задачата ще бъдат намерени.

Важно е да се знае, че при възврат не всички последствия от удовлетворяването на цели могат да се анулират. Например последствията от цели, които имат странични ефекти, като входно - изходни операции и актуализиране на базата данни, не могат да се анулират.

Когато се разглежда дизюнкция от цели, тя успява ако поне една от целите успее и пропада, ако всички цели пропаднат. Преудовлетворяването на дизюнкция означава, че първоначално се търсят всички алтернативни решения за първата цел и при изчерпването им се разглежда следващата цел.

За управление на механизма за възврат са създадени вградените предикати ! (cut)- отсичане и fail. Първият от тях прекратява възврата, а вторият- fail го стартира. Поставен като цел, fail винаги пропада. Използва се за получаване на всички алтернативни решения. Cut служи за фиксиране избора на решение, а също за предизвикване пропадането на дадена цел.

В други случаи “!” се използва за указване на правило с най-голямо значение или че е намерено необходимото правило за удовлетворяване на целта. Употребата на “!” е подходяща и при еднократно удовлетворяване на целта.

Например: да се намери кой е подходящ за програмист, ако са дадени правилата: Х е подходящ за работа Y, ако Х е дипломиран като специалист Y и Х говори английски или Х е сертифициран като Y. С предимство е първото правило и ако то успее, второто няма да се проверява. Налична е база от данни.

?-podhodiast(X,programist),write(X),nl,fail.

podhodiast(X,Y):-diplomiran\_kato(X,Y),govori(X,еngl**), !** . /\*правило 1\*/

podhodiast(X,Y):-sertificiran(X,Y). /\*правило 2\*/

diplomiran\_kato(nikola,programist).

govori(nikola,еngl).

sertificiran(ivan,programist).

Отговор: nikola

Без използване на “!” ще се получат два отговора- nikola, ivan, защото във въпроса е използван fail. Но тъй като от основно значение по условие е да са изпълнени изискванията в първото правило, решението се фиксира с “!”. Ако бъдат намерени факти за удовлетворяване на първото правило, второто няма да се проверява. То ще се изследва, ако няма данни по първото правило.

За явно пропадане на дадена цел се използва комбинацията от cut и fail. Повече информация по темата може да се прочете в учебника от Атанасова Т.,„Логическо и функционално програмиране“, второ изд.,2012 г.

1. Атанасова Т., Логическо и функционално програмиране, второ изд., изд. „Наука и икономика“, Варна, 2012 [↑](#footnote-ref-1)
2. Когато съществува поне една комбинация от стойности на променливи, за която формулата е истина и поне една комбинация, за която е неистина, то формулата се нарича **удовлетворима.** [↑](#footnote-ref-2)