



Механизъм за възврат.
Отрицанието като неуспех.
Аритметични изрази и
оператори.

```

prolog:-
    <elactss falets rile carlios (n) (efectation alescl100);
    <intricas nutler facts(slls) (btiection (over;f> (ua));
    facts puttriacien (Pralle,faules into ation, (afo);
        fies 'leblaoigent, (unte-Barite, pvtlon" denrito, intglesirinlalegariop,lyicalley)
        proficA ll);
    <flet' emrrinrial-faxcts (any gonus anlentivules (nllen (al1));
        clartylon (u)' ralsecllacts rawrtiow, revlirientarnation/(oy (xentialrill10);
    qulects ();
    clacts of nny facts fact-in for facts rauleg, (vll);
        "f1Y, lecyploal, axixieen (caling (oorfactilangocls (cirtrvasticaion (H0,100);
        clacts commention, rfter vlogg, ann 'leayrany- (actleclv10);
        furtursirulers, fusic. abutiless anvlection (vor) aoy1111));
        factes, auylantions (fRfS, (nterrinietiallases, vhiule) procrination. ( anl_guractI);
        fnet faalloxy, (0));
    giact cur facts etrution (yyl1);
        <ant putrelation (nucle- faclien, gcons. sutions (llc-prolog-110)-1");
        ann valegyer", -wittrolation, mulitiaintirverveacien, (tldats ([linpla");
        prcluce inctiyaur glues 'nolyuton .contetio) (natyio, nitpericting, (1);
        <ictestaaction: (olyivs 'detrtex iaall-sxieta" (oycriantraclapocio);
        ractacirinale verplies ((1147);
        clectats unive. arryotians (lls correrietiors (uall17);
        an delectioos". pulox- (colicy. "sory (uopiesox (alon, ciatlys - 19);
        cetratials cantier; falles: laft "acion" payes an zralest ~1016";
        cictreorientzicigu: dlgarrinatties procuamio quuler. (ld1c107);
        <elerscsatinalciiyies an corricinye anto "(aotilix (anllary" prricttipe10)",
        an, aw" (ob (xcrcties (710);
    blacs, facts facts 'centlog(c1ow); falcis quries cialory);
        dlegiattity, nution autrranlation, dotr. lenornang" (ay10ld1);
        rreacation. anteria (deferaction, (ap) socrication for irtycivpigrations-(117);
        clecating actanloyierstieranfaction "elepratde (nlick- traler10);
        iacustien: fgply, cen"= "fir wav". ay. 22" = "(9415, - crylex)),
        an "nelrepicaal wille expleation ad recveridentile" (aafisteriaaylo)=11);
        an, nxcin" );
    <dauction: carlvation (al6);
        flute (oalention (y: (fotil- x "11Y, 7(cnerotdale, v11V, (rvlinay (nolli lu), )
        fataccrariationl(gulties facty (ompericfour, facrlggperierization (126));
        fecruaol aaw flien anat cerylatie)= 'axetfiacis, larlv);
        flerpcriptions. (nft, :neamer nave realev) (aigntiercpaillvy);
    <entriciation (uulentfancer(211);
        clocale faple (letytting, facts, noeler(yclictiog;
        victe, entlatine. propiction, fppic (d5);
    <fovie cayrierastivating, rastry. (iyll10));
        falents, catriont, ponrale corfirmations lat} let- faccial, (aleytiou (2110));
        fracture virien, cryilenliation, any sign(ar pexron. (lo x fwillaad explection
        rfore unyle incentiveation, (atales putiction-valuation)= bid, (16);
        fintate anfering cptions. for (days-(oloo conzlew; carperctingagciymatcl8);
        lntactions: 'lulury riantionantird'scaal)-accyfaseation 112);
    ayllca);
    ;

```

Структура и синтаксис на Пролог програмите



Факти

Основни твърдения,
които са винаги
верни.



Правила

Условия, които трябва
да бъдат изпълнени.



Заявки

Въпроси, които
задаваме на
системата.

Пролог програмите се състоят от факти, правила и заявки. Фактите представят основните твърдения. Правилата определят условията, които трябва да бъдат изпълнени, а заявките са въпросите, които задаваме.



Механизъм за възврат (backtracking): Принципи и действие

- 1** — Съпоставяне
Пролог се опитва да съпостави заявката с фактите и главите на правилата.
- 2** — Избор
Ако има няколко възможности, избира една.
- 3** — Успех
Ако всички цели са изпълнени, заявката е успешна.
- 4** — Неуспех
Ако целта не може да бъде изпълнена, Пролог се връща назад.

Механизмът за възврат е основен за Пролог. Той позволява на системата да търси решения чрез опити и грешки. При неуспех Пролог се връща към предишни точки на избор.

Основни принципи на този механизъм:

- Подцелите се удовлетворяват в реда на посочването им.
- Клаузите на предикатите се проверяват отново по реда на срещането си в програмата.
- Когато подцелта съвпадне с глава на правило, тогава започва удовлетворяването на тялото. Тялото на правилото определя нова последователност от подцели за удовлетворяване.

- Целта се удовлетворява като цяло, когато се намерят всички факти за всички проверени подцели в дървото на целите.

Обръщение към предикат, към който са свързани няколко решения е неопределено (non-deterministic), докато обръщение което се свързва само с едно решение е определено (deterministic).

Пролог предлага три средства за контролиране на последователността за намиране на логическия извод при търсене на решение:

- Предикатът **fail** винаги не успява. По този начин стартира незабавно търсене на ново решение с механизма на възврат.
- Предикатът **not** успява, когато свързаната с него подцел е неистина.
- Прекъсването посредством **cut (!)** предотвратява търсенето на нови решения.

Пример:

харесва(иван,Х):- храна(Х), вкусове(Х,добър).

вкусове(пица,добър).

вкусове(брюкселско_зеле,лош).

храна(брюкселско_зеле).

храна(пица).

Тази малка програма има две двойки факти и едно правило. Правилото описва връзката **харесва**, която има смисъл, че **иван** харесва вкусната храна.

За да се поясни как работи механизмът на възврат при намиране на решение се подава целта:

?- харесва(иван, Какво).

Когато Пролог започва да удовлетворява целта, търсенето на решения започва от началото на програмата.

Когато се намира съвпадение с първата клауза в програмата, променливата ***Какво*** се унифицира с променливата ***X***. Съвпадението на целта с главата на правилото определя началото на удовлетворяването на правилото. Удовлетворяването на цялото правило се разбива на удовлетворяване на всички подцели в това правило. И първото удовлетворяване е на ***храна(X)***.

Когато трябва да се извърши ново удовлетворение, търсенето на съвпадение отново започва от началото на програмата.

Заради това променливата ***X*** се свързва с първия намерен факт ***брюкселско_зеле***. Тъй като има повече от една възможност за отговор на ***храна(X)***, Пролог създава точка на възврат до факта ***храна(брюкселско_зеле)***. Тази точка ще укаже мястото от където Пролог ще започне ново търсене на решение за ***храна(X)***.

Когато се намери успешно съвпадение, започва да се търси удовлетворение на следващата подцел.

С променливата ***X*** се обвързва ***брюкселско_зеле*** и следващата цел е ***вкусове(брюкселско_зеле, добър)***.

Пролог започва да търси удовлетворяване на тази подцел отново от началото на програмата. Тъй като няма клауза, която да съвпада с поставената цел Пролог задейства механизма на възврат. По този начин търсенето на съвпадения се връща в точката на възврат.

Когато една променлива се обвърже в клауза, единственият начин да се „освободи” е чрез механизма за възврат.

Връщането в точката на възврат освобождава всички променливи обвързани след тази точка и започва ново търсене с нова заявка. Променливата X вече е свободна и при търсене на следващ факт за ***храна(X)*** се привързва със стойността ***пица***.

Пролог преминава към следващата подцел в правилото спрямо новото обвързване на променливата. Следващата стъпка е в ход **вкусове(пица, добър)** и отново от началото на програмата започва търсене на съвпадение. Този път съвпадение се открива и целта се удовлетворява успешно. Тъй като променливата **Какво** в целта се унифицира с променливата X в правилото **харесва**, а променливата X получи стойност **пица** Пролог връща следния резултат:

Какво=пица

1 Solution

Практическо приложение на механизма за възврат с примери

Търсене на път

Намиране на маршрут между два града.

Решаване на пъзели

Разрешаване на sudoku или кръстословици.

Доказване на теореми

Автоматично доказване на математически теореми.

Механизмът за възврат се използва за решаване на различни проблеми. Търсенето на път, решаването на пъзели и доказването на теореми са само някои от тях. Backtracking позволява прилагането на много алгоритми.

Оператор „!“

Една възможност за ограничаване на пространството се предоставя посредством използването на стандартния предикат „!“ (cut, отрязване). Отрязването е специален стандартен предикат, който инструктира интерпретатора на Пролог да не прави възврат зад точката на програмата, в която той се появява. Операторът няма явна декларативна семантика (винаги успява), така че неговото използване става за сметка на яснотата на програмата. Операторът трябва да се използва внимателно, понеже неговото присъствие може да наруши изпълнението на една програма по непредвидим начин. Ще разгледаме типични случаи за използване на оператора „!“.

Предотвратяване на механизма на възврат за предишни подцели в правило

С правилото **r1 :- a, b, !, c.** се указва на Пролог, че ако решенията за подцели **a** и **b** са удовлетворени успешно, и въпреки че Пролог би могъл да намери множество решения за **c**, с използване на възврат и търсене на алтернативи за **a** и **b** това няма да се извърши. Не е възможно също използването за механизма на възврат и за друга клауза, която определя предиката **r1**.

Нека разгледаме следния пример:

купува_кола(Модел,Цвят):-

кола(Модел,Цвят,Цена),

цвят(Цвят,топъл),

Цена < 25000.

кола(мазерати,зелен,25000).

кола(корвет,черен,24000).

кола(порше,червен,23000).

кола(корвет,червен,22000).

цвят(червен,топъл).

цвят(черен,среден).

цвят(син,елегантен).

?- купува_кола(X, Y).


С този пример отговаря на въпроса какъв модел автомобил да бъде купен, ако желаният цвят е „топъл” , при заложено ценово ограничение. След стартиране на програмата и разглеждане на всички възможни решения с помощта на механизма на възврат се получава и следното решение:

X=порше, Y=червен

X=корвет, Y=червен

2 Solutions

```
Program x +
1 /*PREDICATES
2 nondeterm купува_кола(symbol,symbol)
3 nondeterm кола(symbol,symbol,integer)
4 цвят(symbol,symbol)
5 CLAUSES*/
6 купува_кола(Модел,Цвят):-кола(Модел,Цвят,Цена),
7     цвят(Цвят,топъл),Цена < 25000.
8 кола(мазерати,зелен,25000).
9 кола(корвет,черен,24000).
10 кола(порше,червен,23000).
11 кола(корвет,червен,22000).
12 цвят(червен,топъл).
13 цвят(черен,неутрален).
14 цвят(син,студен).
```



купува_кола(X,Y).

X = порше,
Y = червен

X = корвет,
Y = червен

?- купува_кола(X,Y).

Ако добавим прекъсване в търсенето на решения, а именно:

купува_кола(Модел,Цвят):- кола(Модел,Цвят,Цена),
 цвят(Цвят,топъл),!,
 Цена < 25000.

кола(мазерати,зелен,25000).

кола(корвет,черен,24000).

кола(порше,червен,23000).

кола(корвет,червен,22000).

цвят(червен,топъл).

цвят(черен,среден).

цвят(син,елегантен).

?- купува_кола(X, Y).

Резултата би бил следния:

X=порше, Y=червен

1 Solution

В този случай първият автомобил, който удовлетворява и трите подцели на правилото е крайното решение, въпреки че съществува и друго. Все пак остава въпроса какво би трябвало да се промени, за да се получи резултата:

X=корвет, Y=червен

1 Solution

Предотвратяване на механизма на възврат за други клаузи на правило:

Прекъсване може да се използва и когато искаме да укажем на Пролог, че е подбрана коректната клауза от множество зададени клаузи. Например:

$r(1)$:- ! , a , b , c.

$r(2)$:- ! , d.

$r(3)$:- ! , c.

$r(_)$:- write('Това е универсална клауза.').

Използването на прекъсване тук води до това, че предиката r става еднозначно определен. В конкретния случай Пролог използва предиката r с единствен целочислен аргумент. Ако се предположи, че това е 1, т.е. за $r(1)$ Пролог започва да търси съвпадение и го намира още с първата клауза. Въпреки, че съществуват повече от една възможности Пролог поставя точка на възврат до тази клауза. По този начин няма да се търси друго удовлетворяване на клаузата r . Ако се разгледа и от друг ъгъл този пример, може да се каже, че по този начин се реализира множествен избор като в другите езици за програмиране (switch в C и case в Pascal).

Отрицанието като неуспех в Пролог: Концепция и реализация

`not(Goal)`

Вграден предикат, който връща true, ако Goal е false.

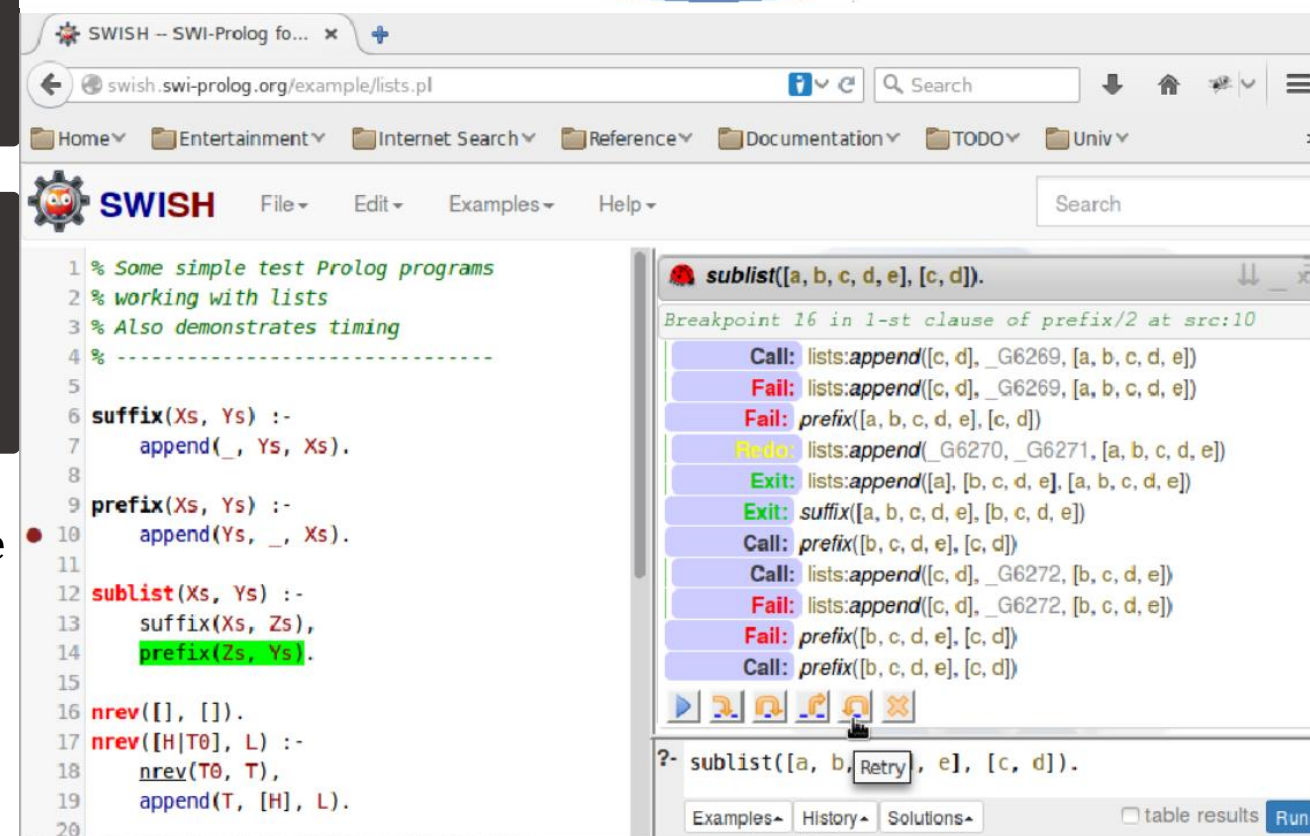
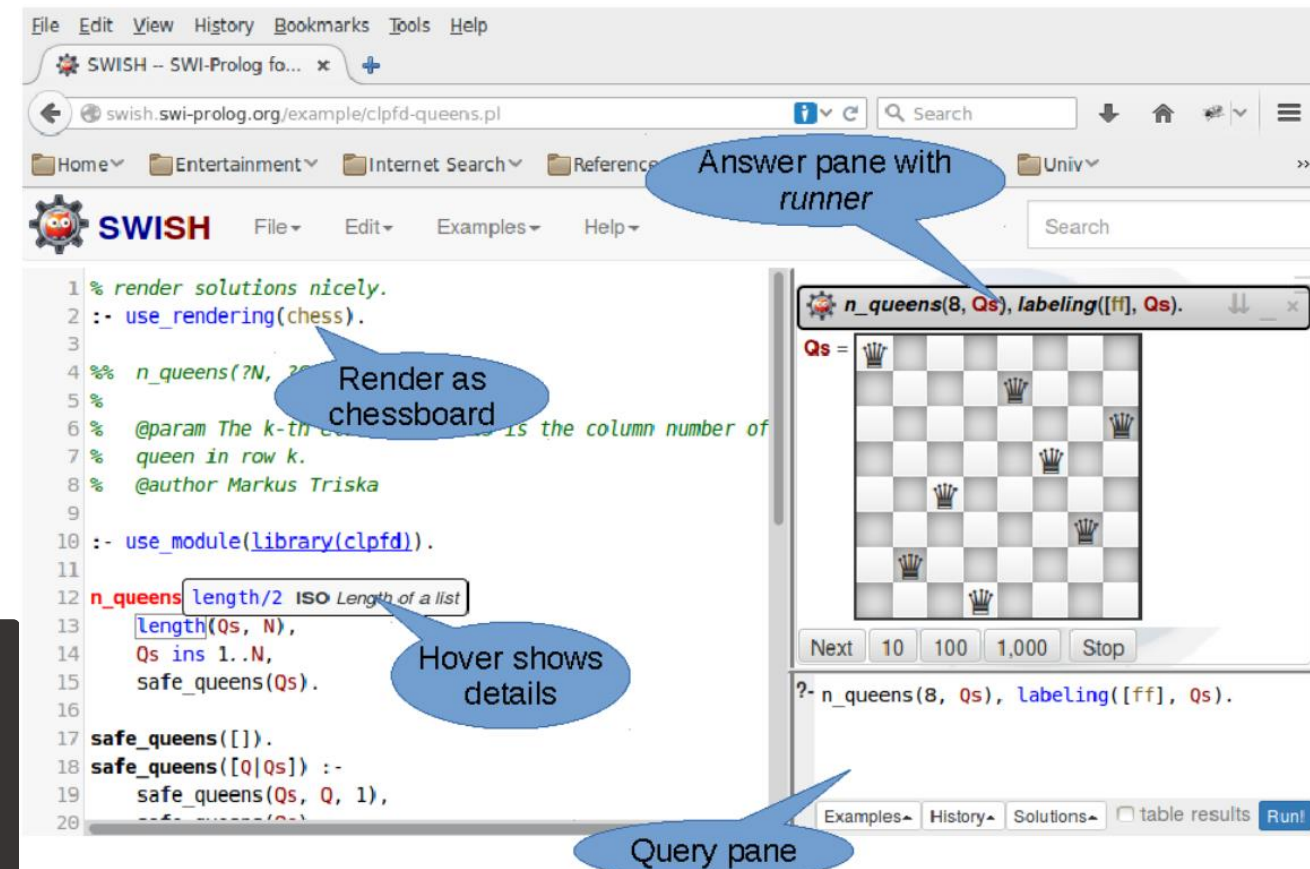
Definition

Пролог не доказва, че нещо е вярно, а че не може да бъде доказано.

Приложение

Използва се за изключване на възможности.

В Пролог отрицанието се дефинира като неуспех. `not(Goal)` е true, ако Goal е false. Използва се за изключване на определени възможности в програмата.



Предикатът not. Отрицанието като неуспех

Представянето и работата с отрицанието не е тривиален проблем. Интуитивно, на декларативно ниво (декларативна семантика) имаме сравнително ясна представа за отричане на нещо. На процедурно ниво (процедурна семантика) нещата стават значително по-сложни. Защо? Какво значи да отричаме нещо. Как да покажем или изброим всичко останало? По тази причина съществуват различни подходи за практическото реализиране на отрицанието. Един от тях, който се използва в Пролог, се нарича **отрицание като неуспех**.

Една от най-полезните функции на Пролог е простият начин, по който ни позволява да формулираме обобщения. За да кажем, че Иван обича зеленчуци, просто пишем:

обича(иван, X) :- зеленчуци(X).

Но в реалния живот правилата имат изключения. Да предположим, че Иван не харесва зеленчуци, приготвени на скара. Така, коректното правило е всъщност: Иван обича зеленчуци, които не са приготвени на скара. Как да заявим това в Пролог?

обича(иван, X) :- зеленчуци_на_скара(X), !, fail.

обича(иван, X) :- зеленчуци(X).

зеленчуци(X) :- зеленчуци_на_пара(X).

зеленчуци(X) :- зеленчуци_на_скара(X).

зеленчуци(X) :- зеленчуци_на_фурна(X).

зеленчуци_на_пара(броколи).

зеленчуци_на_скара(домати).

зеленчуци_на_пара(моркови).

зеленчуци_на_фурна(чушки).

Очакваме, че ако искаме да разберем какви зеленчуци обича Иван, Пролог ще ни потвърди за броколи, моркови и чушки. И наистина, това се случва:

?- обича(иван, броколи).	true .
?- обича(иван, домати).	false.
?- обича(иван, моркови).	true .
?- обича(иван, чушки).	true.

Как работи това? Ключът е комбинацията от ! и fail/0 в първия ред (нарича се комбинация „cut-fail“). Когато задаваме въпроса **обича(иван, домати)**, тогава се прилага първото правило и Пролог достига до отрязването. Това го обвързва с направения вече избор и по-специално блокира достъпа до второто правило. Но тогава се натъква на fail/0. Това предизвиква опит за възврат, но отрязването го блокира и така нашата заявка се проваля.

Това решение е интересно, но не е идеално. Обърнете внимание, че подреждането на правилата е от решаващо значение - ако разменим двете клаузи на предиката обича/2 не получаваме поведението, което искаме.

?- **обича(иван, домати).**

true .

Тук отрязването е от решаващо значение. Ако го премахнем, програмата няма да се държи по същия начин (познато е като „червено отрязване“). Накратко, имаме две взаимно зависими клаузи, които използват присъщите на Пролог процедурните аспекти. Решаващото е, че първата клауза по същество е начин да се каже, че Иван не обича X, ако X е подготвен по определен начин. Комбинацията „cut-fail“ ни позволява да определим такава форма на отрицание, наречена „отрицание като неуспех“. Тази комбинация е нещо много полезно, но би било по-добре, ако можем да извлечем полезната част и да я „опаковаме“ по по-приемлив начин. Ето как:

```
neg(Goal) :- Goal, !, fail.
```

```
neg(Goal).
```


За всяка цел на Пролог `neg(Goal)` ще успее точно ако целта не успее.
Използвайки новия предикат `neg/1` можем да опишем предпочитанията на Иван по много по-ясен начин:

`обича(иван, X):-зеленчуци(X), neg(зеленчуци_на_скара(X)).`

?- `обича(иван, домати).`

`false.`

?- `обича(иван, чушки).`

`true.`

Тоест Иван обича X , ако X не е приготвен на скара. Това е съвсем близко до оригиналното ни изявление.

В стандартния Пролог операторът `\+` означава отрицание като неуспех, така че бихме могли да определим предпочитанията на Иван, както следва:

`обича(иван, X) :- зеленчуци(X), \+ зеленчуци_на_скара(X).`

Още за декларативната и процедурна семантика

Въпреки, че намерихме приемлива форма за представяне на отрицанието, още един коментар не би бил излишен. Налага ни се да се върнем отново към проблема за декларативната и процедурната семантика. Не трябва да мислим, че „отрицанието като неуспех“ работи точно както „логическото (декларативното) отрицание“. Нека се върнем отново към БД за приготвяне на зеленчуци. Ако искаме да разберем които зеленчуци обича Иван в зависимост от тяхното приготвяне, ще зададем въпроса:

?- обича(иван, X).

X = броколи ;

X = моркови ;

X = чушки.

Получаваме коректната последователност на отговорите.

Но сега да предположим, че пренаписваме предиката обича/2 като разменим двете клаузи както следва:

обича(иван, X) :- \+ зеленчуци_на_скара(X), зеленчуци(X).

Има ситуации, в които е по-добре да използваме ! вместо отрицание като неуспех. Например, да предположим, че трябва да напишем програма, за да изразим следното условие: *p* важи, ако *a* и *b* са в сила или ако *a* и *c* не са в сила. Това може да бъде директно направено с помощта на отрицанието като неуспех:

p :- *a*, *b*.

p :- \+ *a*, *c*.

Да предположим, че *a* е много сложна цел, която отнема много време за изчисляване. Програмирането по този начин означава, че може да се наложи да изчислим *a* два пъти, което би довело до неприемливо бавна производителност. Ако е така, би било по-добре да използвате следната програма:

p :- *a*, !, *b*.

p :- *c*.

Пример: Нека разгледаме ситуация, в която се търси студент с успех над 3.50, който не е на стаж:

търсен_студент(Име):-студент(Име,Успех),
Успех>=3.5, not(стаж(Име)).

студент("Иван Иванов", 3.5).

студент("Димитър Димитров", 2.0).

студент("Георги Георгиев", 3.7).

стаж("Иван Иванов").

стаж("Димитър Димитров").

?-търсен_студент(X).

Аритметични изрази и вградени оператори в Пролог

5

Числа

Цели и дробни числа.

+, -, *, /

Оператори

За събиране, изваждане, умножение и деление.

is

Присвояване

Оператор за присвояване на стойност.

Пролог поддържа аритметични изрази и оператори. Използват се за работа с числа.

Операторът **is** се използва за присвояване на стойности. Има много оператори, като +, -, * и /.

Prolog Arithmetic Operators

*a can = 11, 1, 10, (atomut1)
north legally is the result
A. S.(1), + = 6.2(1);*

Prolog arithmetic operators

A, S x S, (0, 16, (atan(1) - huf+ale (0 0) exmings

Examples

Key is an thiermions arithmetic operators

Arithmetic action results

*a can = 11, 7 (a. (18, (athont1)
north legally is take results)
A. S.(1), + (= S.(1),*

creation thiermions origins

adthor aben crions

12 *On 11 is in colentes with a period operations*

*x can = 11.. 1r (= 19, (casion1)
(actil ty (ronir in the results))
Now = 1.2. S, (= 5, 4 2.0.50);*

consemple

(0 fupple: 11, cecentiar tirins

*a can = 12, 115, (8, (statom1),
aurth legally is take results)
A. S.(1), + (= 8, 1.(1);*

coliditine seulls

an clills

13 *(o examples*

*a can = 12, (1, 15, 18, (atrun1))
aurth legally is take results)
A. S.(1), + (1= S.(1),*

colitit and operators

A. coltfingial arithmetic results

*a can = 12, 1 (2, 1, 18, (atrun1))
(aurth legally fo take results)
A. S.(1), + (= 7, (1),*

endfination tryles

rand ction (k,) *[+ contritions*

Results

Работа с числа и математически операции: Примери

1

Събиране

X is 5 + 3.

2

Изваждане

Y is 10 - 2.

3

Умножение

Z is 4 * 6.

Аритметичните операции в Пролог се извършват чрез оператора **is**. Преди това трябва да се зададат стойностите на променливите. Примери: X is 5 + 3, Y is 10 - 2.

Примери:

?- a(1,X) = a(1,2).

X = 2

?- a(1,X) == a(1,2).

false

?- X = 2, a(1,X) == a(1,2).

X=2

?- 2+3=1+4.

false

?- 2+3=:1+4.

true

?- X=1+4.

X=1+4

?- X is 1+4

X=5

?-X=5, X is 1+4.

X=5

Функции

abs(X) , sin(X), cos(X), sqrt(X), ...

Дефинирайте предикат за
намиране стойността на
функцията

$$f(x, y) = \begin{cases} \sqrt{|y^2 - 3x^2|}, & y > x \\ x \cdot \sin x, & x = y, \\ e^x + \ln(x - y), & x > y \end{cases}$$

```
f(X,Y,Result) :- Y>X, Result  
is sqrt(abs(Y*Y - 3*(X*X))).
```

```
f(X,Y,Result) :- X=Y, Result  
is X*sin(X).
```

```
f(X,Y,Result) :- X>Y, Result  
is e**X + log10(X-Y).
```

Дефинирайте предикат за
намиране стойността на
функцията

$$h(y) = \begin{cases} \frac{2 \cdot y^2}{y + 4}, & y < -4 \\ 0, & y \in [-4, 4] \\ \frac{\ln(y - 4) \cdot e^y}{y^3 + \sqrt{y}}, & y > 4 \end{cases}$$

`h(Y, Res) :- Y < -4, Res is
(2*(Y*Y))/(Y+4).`

`h(Y, Res) :- Y =< 4, Y >= -4, Res is
0.`

`h(Y, Res) :- Y > 4, Res is (log10(Y-
4)*(e**Y))/(Y**3+sqrt(Y)).`

Дефинирайте предикат за
намиране стойността на
функцията

$$f(x, y) = \frac{\log(3 + x^2 + y)}{2 + y + x^2}, \quad -1 < x < 1 \text{ и } y \neq 0$$

ЗАДАЧА

Дефинирайте предикат, който задава кой човек какво притежава. Нещата, които могат да се притежават са: банкова сметка с определена сума; апартамент или къща, които се характеризират с местоположение, площ и цена; кола от определена марка и с даден цвят; яхта.

Напишете:

1. Цел, извеждаща какво има Иван;
2. Цел, извеждаща кой има кола.
3. Цел, извеждаща кой каква марка кола има.
4. Цел, установяваща има ли двама с червени коли.
5. Цел, установяваща има ли двама с еднакъв цвят коли.
6. Цел, извеждаща кой има в банковата сметка толкова пари, колкото струва домът му.

7. Предикат, определящ кой какъв дом има (домът е цяла структура) и цел, извеждаща кой какъв дом има.
8. Предикат, определящ големите домове – с площ повече от 200 кв. м. и техните собственици и цел, извеждаща кой има голям дом.
9. Предикат, задаващ кой е богат – ако има къща или апартамент на цена повече от 30000, яхта, кола и банкова сметка над 200000 и цел, извеждаща кой е богат.
10. Предикат, определящ колко е най-голямата банкова сметка и цел, която я извежда.

```
/* База факти с данни за това кой какво има */  
has(ivan,yacht).  
has(ivan, car(lada,red)).  
has(tanya,bank_account(200000)).  
has(peter,house(plovdiv, space(160),price(48000))).  
has(maria,flat(sofia,space(121),price(32000))).  
has(kamen, car(renault,blue)).  
has(kamen,bank_account(1005000)).  
has(ivan,flat(varna,space(110),price(60000))).  
has(ivan,bank_account(2200000)).  
has(maria,bank_account(600000)).  
has(maria,yacht).  
has(elena, car(ford,red)).  
has(maria,house(pleven, space(300),price(600000))).  
has(maria, car(volvo,black)).  
has(peter,bank_account(2000000)).
```

/*

- 1.Какво има Иван?
2. Кой има кола?
- 3.Кой каква марка кола има?
- 4.Има ли двама с червени коли?
- 5.Има ли двама с еднакъв цвят коли?
- 6.Кой има в банковата си сметка толкова пари, колкото струва домът му?
- 7.а) Предикат, определящ кой какъв дом има (домът е цяла структура)
b) Кой какъв дом има?
8. а) Предикат определят големите домове – с площ повече от 200 кв.м., и техните собственици
b) Кой има голям дом?
9. а) Предикат, задаващ кой е богат – ако има къща или апартамент на цена повече от 30000, яхта, кола и банкова сметка над 2000000. (извежда по веднъж всеки богат-one)
b) Кой е богат?
10. а) Предикат, определящ колко е най-голямата банкова сметка.
b) Колко е най-голямата сметка?

*/

1. Какво има Иван?

?- has(ivan,X).

2. Кой има кола?

?-has(X,car(_,_)).

3. Кой каква марка кола има?

?-has(X,car(Y,_)).

4. Има ли двама с червени коли?

?-has(X,car(_ ,red)), has(Y,car(_ ,red)), X\=Y, !.

5. Има ли двама с еднакъв цвят коли?

?-has(X,car(_ ,Z)), has(Y,car(_ ,Z)), X\=Y, !.

6. Кой има в банковата си сметка
толкова пари, колкото струва домът
му?

?-has(X,bank_account(Y)),
(has(X,flat(_ ,_,price(Y)));
has(X,house(_ ,_,price(Y)))).

7а) Предикат, определящ кой какъв дом има (домът е цяла структура)

```
home(X,Y):-has(X,Y), (Y=flat(_,_,_);  
Y=house(_,_,_)).
```

б) Кой какъв дом има?

```
?- home(X,U).
```

8а) Предикат определят големите домове – с площ повече от 200 кв.м., и техните собственици.

```
big_home(X,Y):-has(X,Y),  
(Y=flat(_,space(Z),_);  
Y=house(_,space(Z),_)), Z>200.
```

б) Кой има голям дом?

```
?- big_home(X,_).
```

9а) Предикат, задаващ кой е богат – ако има къща или апартамент на цена повече от 30000, яхта, кола и банкова сметка над 2000000.

```
rich(X):-has(X,yacht), has(X,car(_,_)), has(X,bank_account(Z)), Z>2000000,  
(has(X,house(_,_ ,price(Y))); has(X,flat(_,_ ,price(Y)))), Y>30000.
```

б) Кой е богат? ?- rich(X).

10а) Предикат, определящ колко е най-голямата банкова сметка.

```
biggest_account(Z):- has(_,bank_account(Z)),  
\+ (has(_,bank_account(C)), C>Z), !.
```

б) Колко е най-голямата сметка? ?-biggest_account(X).

Задача Лекар:

Дадена е база данни на Пролог от следния вид:

% работа(лекар, лечебно_заведение, трудов_стаж)

работа('Иван Петров', 'Трета поликлиника', 23).

работа('Стоян Драганов', 'Пълмед', 8).

работа('Михаела Тодорова', 'Първа поликлиника', 14).

работа('Иво Драганов', 'Хигия', 18).

%.... Допишете още факти по посочения начин

% лекува(лекар, пациент) – описващ отношението лекар-пациент и

пациент(име, възраст)

лекува('Михаела Тодорова', пациент('Иван Иванов', 34)).

лекува('Стоян Драганов', пациент('Нели Томова', 53)).

лекува('Иван Петров', пациент('Христо Ангелов', 76)).

лекува('Иван Петров', пациент('Камен Димов', 23)).

%.... Допишете още факти по посочения начин

- А) Дефинирайте предикат, който определя лекарите с трудов стаж под 10 години.
- Б) Напишете цел, която извежда като резултати пациенти на възраст 34 г., лекувани от д-р Михаела Тодорова.
- В) Дефинирайте предикат, определящ имената на пациентите, които се лекуват при лекар работещ в Трета поликлиника и имащ стаж под 15 години.
- Г) Напишете цел, която извежда двама различни пациенти, които се лекуват при един и същи лекар.
- Д) Дефинирайте предикат, който определя пациентите, които не се лекуват в Първа поликлиника.
- Е) Напишете цел, която извежда пациентите, които се лекуват само при един лекар.

Задача Функция

Дефинирайте предикат за пресмятане стойността на функцията по зададено x :

$$g(x) = \begin{cases} \ln(x + 3), & 0,5 < x \leq 5 \\ -3, & -0,5 \leq x \leq 0,5 \\ \sqrt{|12 - x|} + x^2, & x < -0,5 \text{ или } x > 5 \end{cases}$$

Задача. Талисманите на месеците

По древно поверие всеки месец има свой камък-талисман. Например, месеците юни, юли и септември съответстват на камъните рубин, сапфир и перла. Тези камъни означават мъдрост, здраве и благополучие. На кой месец какъв камък- талисман съответства и какво означава, ако се знае, че:

- перла и рубин не съответстват на септември;
- през юни и юли мъдрост не се наблюдава;
- здраве не съответства на рубин;
- благоденствие не се отнася за месец юни.

камък(рубин). камък(сапфир). камък(перла). символ(мъдрост). символ(здраве). символ(благополучие).

месец_камък(септември,Y):-not(Y=перла).

месец_камък(септември,Y):-not(Y=рубин).

месец_камък(юни,Z):- (Z=мъдрост).

месец_символ(юни,Z):-not(Z=благополучие).

месец_символ(юли,Z):-not(Z=мъдрост).

камък_символ(Y,Z):- Y=рубин, not(Z=здраве).

dif(Y1,Y2,Y3):- Y1\=Y2, Y1\=Y3, Y2\=Y3.

резултат(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3):-

X1=юни,X2=юли, X3=септември,камък(Y1), камък(Y2), камък(Y3), символ(Z1), символ(Z2), символ(Z3),
камък_символ(Y1,Z1), месец_камък(X3,Y3), месец_символ(X1,Z1), месец_символ(X2,Z2),
dif(Y1,Y2,Y3),dif(Z1,Z2,Z3).% резултат(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3).

Задачи

1. Иван, Петър и Васил събирали шишарки в гората. Едно от момчетата носело кошница, другото – кофа, а третото носело найлонова чанта. Петър не носел кошница, нито чанта. Иван също не носел кошница. В какво е събирал шишарките всеки от тях?
2. Симо живее по-високо от Петко, но по-ниско от Иван, а Тео живее по ниско от Петко е една четириетажна кооперация. Посочете кой на кой етаж живее.
3. Весела, Мима и Пепа учат химия, биология и математика в градовете София, Пловдив и Варна. Знае се, че Весела не учи в София, а Мима не е в Пловдив. Тази, която е в София, не учи математика. Учещата в Пловдив изучава химия. Мима не харесва биологията, но вижда всеки ден залеза в морето. Определете всяка от тях какво учи и в кой град ?
4. Ели обича да послъгва. От друга страна тя е много методична във всичко, което прави - включително лъженето. Момичето лъже в шест от дните от седмицата, но в седмия винаги казва истината. Тя ви е казала следните неща в три последователни дни:
 - Ден 1: Аз лъжа в понеделник и вторник.
 - Ден 2: Днес е четвъртък, събота, или неделя.
 - Ден 3: Аз лъжа в сряда и петък.

Можете ли да определите в кой ден от седмицата Ели казва истината?

Проверка
29.04.2025

Регистрация

<https://tinyurl.com/22hnvnn8>



Благодаря за вниманието!

