## 2. УНИВЕРСАЛНОСТ НА ПРОЛОГ

**Въведение.** Въпреки**,** че Пролог е предназначен за решаване на логически задачи и наподобява по-скоро експертна система, той е универсален език за програмиране. Ще обърнем внимание на онези механизми, които превръщат езика в универсален език. Това са съчетаването на фиксирания начин за търсене от началото към края и от ляво надясно с връщането (backtracking) към такова място за търсенето на нова възможност за удовлетворяване на пропадналата цел, което гарантира изпробването на всички възможни варианти.

Да разгледаме няколко примера.

**Пример 1.** Да се състави програма на Пролог, която намира абонатите на един и същ gsm оператор.

Упътване. Необходими са два предиката. Единият, чрез който се задават връзката между име на абоната, кодът на gsm оператора и номерът на gsm-а. Да наречем този предикат subscriber(name, gsm_code, telephone). Нека вторият предикат е same_gsm_operator(name1, name2) с два аргумента, които ще получават имената на абонатите на един и същи gsm оператор. GSM операторите, както знаем са три – Mtel, Globul и Vivatel.

Правилото за определяне на абонатите на един и същ gsm оператор може да се дефинира така: двама потребители са абонати на един и същ gsm оператор, ако кодът на gsm-ите им е един и същ, а имената им и самите номера на gsm-ите им са различни (*това трябва да се осигури чрез различие на имената на потребителите и на номерата на gsm-ите им, защото един потребител може да има няколко gsm-а*). Правилото на Пролог може да се запише така:

```
same_gsm_operator(X,Y):-subscriber (X,GSM_CODE,Tel1),
                        subscriber (Y,GSM_CODE,Tel2),
                        not(X=Y),
                        not(Tel1=Tel2).
```

Т.е. X и Y са абонати на един и същи gsm оператор, ако кодът на gsm-а на X е GSM_CODE и кодът на gsm-а на Y е също GSM_CODE и името X е различно от името Y, и номерът на телефона Tel1 на X е различен от номера на телефона Tel2 на Y.

Примерна програма.

```
subscriber ("Петър",  878, "222 333").
subscriber ("Стоян",  878, "344 344").
subscriber ("Милица", 895, "444 675").
subscriber ("Калин",  878, "236 457").
subscriber ("Надя",   886, "632 789").
subscriber ("Илия",   895, "465 788").
subscriber ("Таня",   886, "333 758").
subscriber ("Петър",  886, "333 758").
```

same_gsm_operator(X,Y):-subscriber(X,GSM_CODE,Tel1),
subscriber(Y,GSM_CODE,Tel2), not(X=Y),not(Tel1=Tel2).
Goal: same_gsm_operator(X,Y)

Работа на Пролог. При зададената цел X и Y са свободни променливи и машината започва от началото да проследява клаузите и достига до правилото. Първа подцел става предикатът subscriber (X,GSM_CODE,Tel1), който е пръв от дясната страна на разглежданата цел.

Търсенето на решение за тази подцел започва с ново проследяване от началото и X се свързва с ”Петър” , GSM_CODE с 878, а Tel1 - с “222 333”.

След това се преминава към търсене на решение за втората подцел subscriber (Y,GSM_CODE,Tel2). Сега Y е свободна променлива, но GSM_CODE е свързана с 878. Започва отново проследяване на всички клаузи отначало и Y се свързва с “Петър”.

Преминава се към удовлетворяване на третата подцел от правилото not(X=Y), според която X трябва да е различно от името, унифицирано с Y. Но в момента X и Y са еднакви и третата подцел е False, т.е. удовлетворяването ѝ пропада. При това положение сработва механизмът за автоматичен възврат (излизане от изхода Fail на третия предикат и влизане към предишната подцел от входа ѝ Redo). Това води до анулиране на свързването за Y и търсенето продължава към друго възможно решение за втората подцел. В нашия случай това е свързването Y=”Стоян”. Следва отново преминаване към третата подцел и този път тя е удовлетворена (True). Тогава се преминава към четвъртата подцел not(Tel1=Tel2). Тук Tel1 и Tel2 са свързани променливи съответно с номерата на телефоните на Петър и Стоян. Тези телефонни номера са различни и следователно тази подцел също е удовлетворена. Резултатът е намерено едно решение - Петър и Стоян са абонати на един и същи gsm оператор, защото кодовете на gsm-ите им са еднакви, имената им са различни и телефонните им номера са различни.

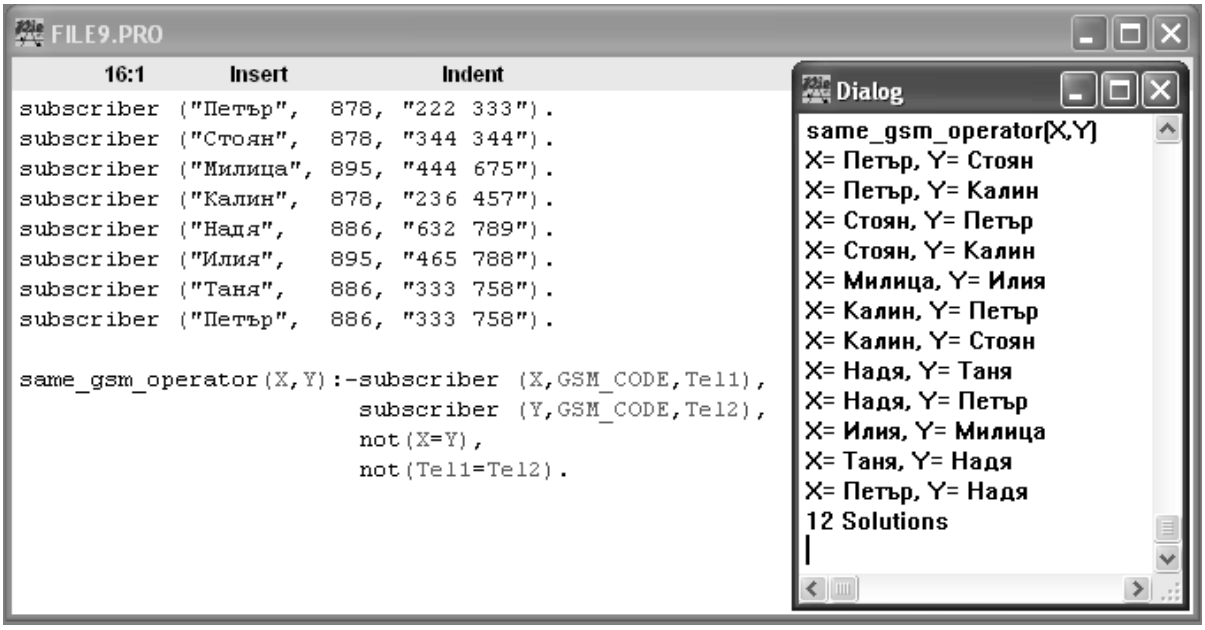
Машината продължава с търсене на друго възможно решение за втората подцел, докато се изчерпат всички възможности. След това се търси ново решение за първата подцел и отново се започва проследяване на всички възможни решения за втората, третата и четвъртата подцел и т.н. до изчерпване на всички възможни решения за всички подцели на зададената цел.

Отбележете: Търсенето е в строго фиксирана последователност – отгоре надолу и отляво надясно. Целите се удовлетворяват в същия фиксиран ред. При пропадане на цел изпълнението започва от позицията, която непосредствено следва позицията, довела до неуспеха. Така изпълнението се превръща в систематично проследяване за всички възможности и се управлява от съдържанието на програмата.

За всеки съставен предикат проследяването се управлява от толкова вложени цикъла, колкото са съставящите го предикати (за всеки по един). Това става ясно, ако опишем процедурно (т.е. от гледна точка на изпълнението)

предиката same_gsm_operator(X,Y) :- subscriber(X,GSM_CODE,Tel1), subscriber (Y,GSM_CODE,Tel2), not(X=Y), not(Tel1=Tel2).

За да се изпълни предикатът same_gsm_operator(X,Y), първо се задейства изпълнението на предиката subscriber(X,GSM_CODE,Tel1), след това – на предиката subscriber (Y,GSM_CODE,Tel2), след това на предиката not(X=Y) и последно - на предиката not(Tel1=Tel2).




**<u>Пример 2</u>**. Да се състави програма на Пролог, която да намира стойността на следната функция:

$$f(x) = \begin{cases} 0, & \text{за } x < 2, \\ 1, & \text{за } 2 <= x < 4, \\ 2, & \text{за } x >= 4. \end{cases}$$

<u>Примерна програма.</u>

```
f(X, 0) :- X < 2.
f(X, 1) :- X >= 2, X < 4.
f(X, 2) :- X >= 4.
```

В тази програмка клаузите са записани по начин много близък до общоприетия в математиката. Това е станало възможно, като в диалекта на езика са включени средства за представяне на предикати в по-обичаен за предметната област начин.

Програмата работи коректно независимо от реда на клаузите. Този пример илюстрира основната идея на декларативните езици - описва се задачата, без да се посочва пътят за нейното решаване.

Програмата обаче не работи ефективно. Ако се зададе цел f(-5, Y), решението ще бъде намерено още при първия ред, но програмата ще продължи да претърсва и останалите редове, вярна на твърдо заложения межанизъм за проследяване.

Тъй като за този пример е известно, че решението е винаги единствено и следователно след като е намерено решение по-нататъшното търсене не е

необходимо, нека използваме предиката cut (!), за да направим програмата по-ефективна. Този предикат действа като еднопосочен клапан и не позволява възобновяване на проследяването преди себе си.

Ето програмата с използване на предикат cut (!).

```
f(X, 0) :- X < 2,!.
f(X, 1) :- X < 4,!.
f(_, 2) .
```

При тази програма всеки предикат се изпълнява само по веднъж.

Отбележете: С помощта на предиката cut (!) може да се управлява процесът на изпълнение на програма в Пролог. Програми, в които се използва предикатът cut (!), изискват старателно подреждане на предикатите, за да не се изпускат възможни решения. Тази опасност е един от така наречените **странични ефекти** на Пролог (виж т.3 от заданието).

**Пример 3**. Целта е да се състави програма за разпечатване във вид на изречение името на абоната, номера на gsm-а му и името на gsm опператора на всички записани в базата знания индивиди**,** като всеки път се минава на нов ред. Например:

Петър има gsm: 878 55 34 55. Абонат е на Vivatel.

.....

Упътване. Необходима ни е база знания с предикат subscriber(subscriber_name, gsm_code, gsm_number), правило operator(operator_code, operator_name) за намиране на единственото съответствие на кода и името на gsm оператора и правило за разпечатване, което може да изглежда по следния начин:

```
gsms:-subscriber(SubscriberName,Code,GSM),operator(Code,OperatorName),
      write(SubscriberName, " има gsm: ", Code, " ", GSM, ". Абонат е на ",
      OperatorName), nl, fail.
```

Предикатът gsms претърсва базата, предикатът write отпечатва в зададения му ред стойностите на аргументите си, предикатът nl предизвиква преминаване на нов ред. Предикатът write има свойството да се изпълнява само веднъж. Когато той е включен в съставен оператор, след изчерпването на съставния оператор програмата спира своята работа. Петият предикат fail винаги дава като резултат false, т.е. пропада и заставя програмата да продължи изпълнението си от точката след тази, която е довела до неуспех.

Тъй като тялото на предиката gsms е съставна клауза, състояща се от свързани с логическо “И” (,) подцели, необходимо е всяка от съставящите я клаузи да е истина, за да бъде истина тялото (съставната клауза като цяло). Заради fail всяко намерено и разпечатано на екрана решение завършва с false и Пролог е принуден да търси ново решение до изчерпване на всички клаузи на предикатите subscriber(SubscriberName,Code,GSM) и operator(Code,OperatorName) в базата.

Примерна програма.

```
subscriber("Петър",   878, "55 34 55").
```

```
subscriber("Стоян",  878, "34 64 22").
subscriber("Милица", 895, "12 23 45").
subscriber("Калин",   878, "64 76 23").
subscriber("Надя",   886, "33 55 66").
subscriber("Илия",   895, "72 12 77").
subscriber("Таня",   886, "76 76 76").

operator(878,"Vivatel"):-!.
operator(886,"Mtel"):-!.
operator(895,"Globul"):-!.

gsms:-subscriber(SubscriberName,Code,GSM),operator(Code,OperatorName),
    write(SubscriberName, " има gsm: ", Code, " ", GSM, ". Абонат е на ",
OperatorName), nl, fail.
gsms.
Goal: gsms
```

Със задаването на тази цел ще получите всички имена на потребители и номерата на gsm-ите им, които са описани в клаузите на предиката subscriber (SubscriberName, Code, GSM). За всеки потребител ще получите и на кой gsm оператор е абонат според описаните съответствия на кода и името на gsm операторите в предиката operator(Code,OperatorName).

Работа на Пролог. При изпълнението на gsms се задейства последователната проверка на петте подцели. При първото проследяване subscriber(SubscriberName,Code,GSM) се удовлетворява от subscriber("Петър", 878, "55 34 55"), след това operator(Code,OperatorName) се удовлетворява от operator(878,"Vivatel"):-!.. Следва отпечатване и преминаване на нов ред. Последователността завършва с fail, чийто резултат false активира връщането назад и започване на ново проследяване от точката след тази, която е довела до провала. Операторът cut (!) винаги връща истина и забранява търсенето на алтернативни клаузи след намирането на първото решение за съответния предикат. В случая няма да се търси дали кодът 878 съответства на друго име на gsm оператор след, като е намерено едно съответствие на този код с името Vivatel. Този оператор се използва, когато сме сигурни, че решението е единствено.

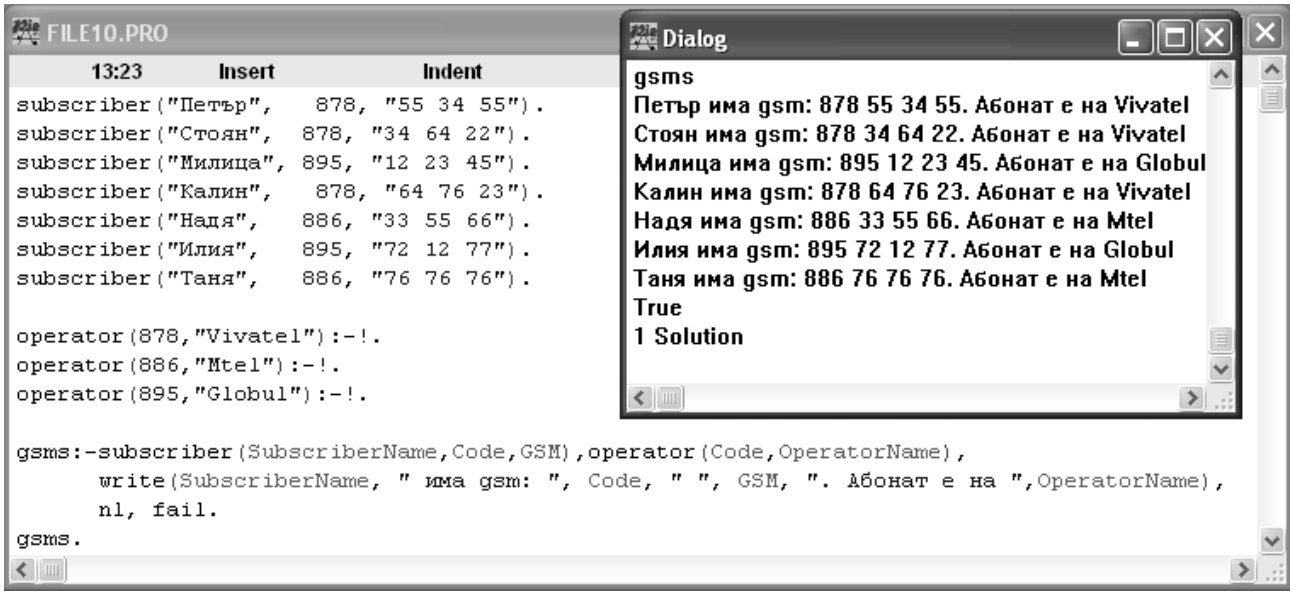
Ако се отстрани предикатът fail и програмата се стартира с цел phones, ще получите само едно решение – името, телефонния номер и компанията, записани на първо място в клаузите на предиката phone. (Поради свойствата на write)

Ако се отстрани втората клауза gsms след изчерпването на всички предикати subscriber(SubscriberName,Code,GSM), машината за извод може да увисне поради получаваната стойност false за петата по ред подцел в съставния оператор. Включването на втората клауза gsms отстранява тази възможност, като след изчерпването на базата възвратът е в точката на втората клауза gsms,

след което проверката на първата подцел на този съставен оператор пропада поради изчерпване на базата (завършва се правилно с изчерпване на най-външния цикъл).

Отбележете: Намесата в управлението на изпълнението в този пример бе наложено от естеството на самата задача. Когато се използват похвати за намеса в управлението на изпълнението, задължително трябва да се проследи за правилното приключване на проследяващия механизъм на Пролог.

Намесата в изпълнението понякога може да доведе до т.нар. старнични ефекти, изразяващи се например в получаването на несъществуващи решения. Без да се впускаме в подробности, ще приведем една препоръка. *Никога не смесвайте предикатите за извеждане с обработващите предикати на програмата.*



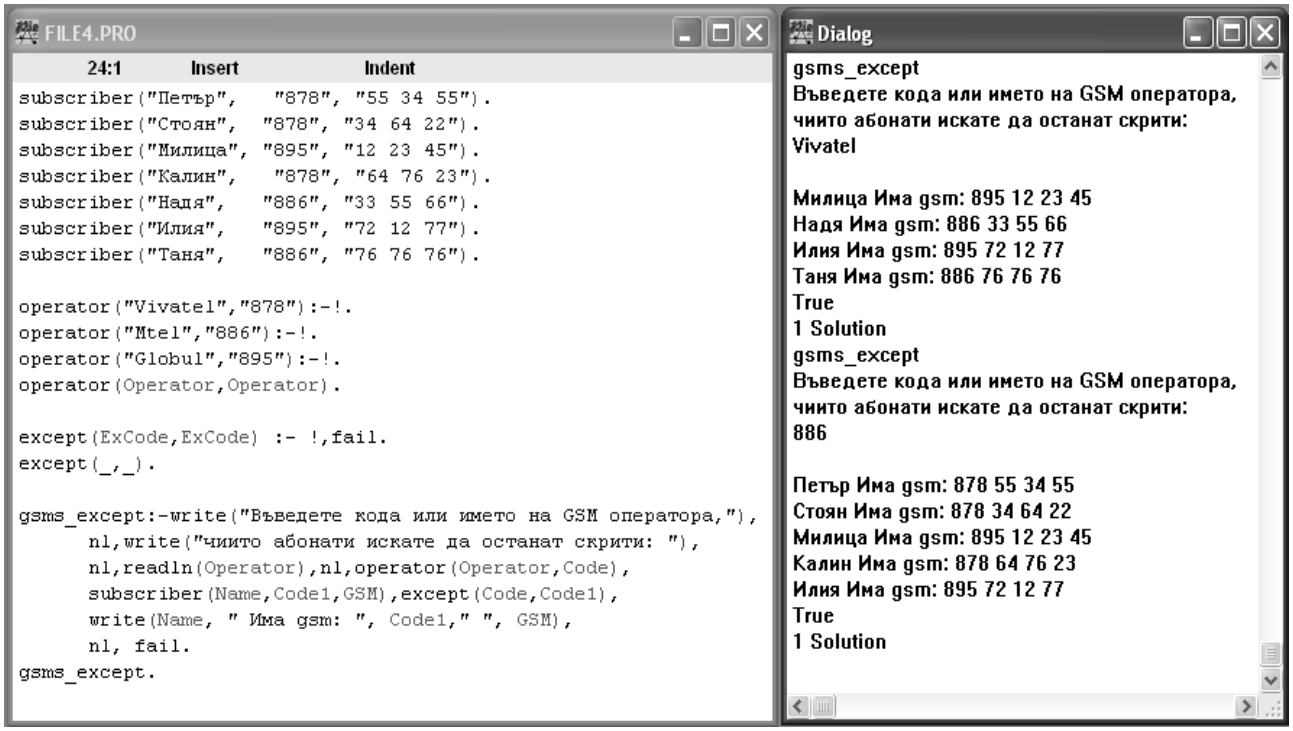

**Пример 4**. Да се разпечатат всички имена и телефонни номера, без тези на избран от потребителя gsm оператор.

С този пример ще покажем още една възможност за управление на изпълнението, която отдалечава съвременните диалекти на Пролог от декларативните езици и ги доближава до модерните универсални езици за програмиране. Ще бъдат използвани предикатите **fail**, **cut**(!) и **except.**

Упътване. Потребителят въвежда от клавиатурата името или кода на gsm оператора, чиито абонати желае да останат скрити. Ако потребителят въведе от клавиатурата името на gsm оператора, то унификацията на една от първите три клаузи на предиката operator(NameGSMOperator,CodeGSMOperator) ще се получи съответният му код във втория аргумент на този оператор. Ако потребителят въведе кода на gsm оператора, то от последната клауза на същия предикат вторият аргумент ще се унифицира с въведения код. Така се постига възможността вторият аргумент на предиката operator(NameGSMOperator, CodeGSMOperator) винаги да е унифициран с кода на gsm оператора, който по-нататък се използва за унифициране с кода на gsm оператора на абонатите.

Ако предикатът except(ExCode, ExCode) е true, се изпълнява неговото тяло. Първата клауза на except в програмата се изпълнява само ако въведеният от клавиатурата код или име на gsm оператор е същият като прочетения от базата данни код на gsm оператор на абоната. Второто except винаги се изпълнява поради анонимните променливи.

Когато първият except е true, изпълнението на тялото му едновременно забранява по-нататъшното проследяване (!) и задейства механизма за възврат (fail), като отправя изпълнението в точката след номера на абоната, който желаем да остане скрит.




Примерна програма.

```
subscriber("Петър",   "878", "55 34 55").
subscriber("Стоян",  "878", "34 64 22").
subscriber("Милица", "895", "12 23 45").
subscriber("Калин",   "878", "64 76 23").
subscriber("Надя",   "886", "33 55 66").
subscriber("Илия",   "895", "72 12 77").
subscriber("Таня",   "886", "76 76 76").

operator("Vivatel","878"):-!.
operator("Mtel","886"):-!.
operator("Globul","895"):-!.
operator(Operator,Operator).

except(ExCode,ExCode) :- !,fail.
except(_,_).
```

```
gsms_except:-write("Въведете кода или името на GSM оператора,"),
   nl,write("чиито абонати искате да останат скрити: "),
   nl,readln(Operator),nl,operator(Operator,Code),
   subscriber(Name,Code1,GSM),except(Code,Code1),
   write(Name, " Има gsm: ", Code1," ", GSM),
   nl, fail.
gsms_except.
```

**Задание**

1. Проследете изпълнението на програмата от Пример 1 по стъпки при търсенето на отговора на различни формулирани от вас въпроси. Формулирайте въпроси, чрез които да намерите броя на абонатите от различните gsm оператори, на абонатите от конкретен gsm оператор и др.

2. Преобразувайте програмата от Пример 1 така, че броят на възвратите да се намали.

Забележка. Забележете, че броят на gsm операторите е значително по-малък от броя на абонатите. Преобразувайте предиката subscriber(name, gsm_code, tel) в subscriber(gsm_code, name, tel).

3. Направете така, че преобразуваната в предното задание програма да намира всяка двойка абонати само по веднъж.

Забележка. Внимателно подредете клаузите и поставете на подходящи места предиката cut (!).

4. Съставете програма за изчисляването на следната функция: $Y=5x^2+3x-7$.

Забележка. Установете как в използвания от вас диалект на езика се записват математическите зависимости.

5. Съставете програма за изчисляването на следните функции:

$Y=5x^2+3x-7$,

$Z=(2x+1)x-Y$.

6. Пробвайте програмата от Пример 3, като задавате други цели и като премахвате fail и второто phones.

7. Пробвайте програмата от Пример 4, като задавате други цели.

Опитайте се да си отговорите защо е необходимо второто except.

Пробвайте програмата, като изключвате някои от управляващите предикати fail, cut(!) и except.

8. Съставете програма, която отпечатва имената и телефонните номера само на мъжете или само на жените.

Забележка. Необходим е предикат за свързване на името с пола на индивида.