Дәріс № 14

Артық кетуді басқару

Бірнеге пайдалы қосымша предикаттар ....

repeat – предикат, әрқашан шыншыл. Ол әрқашан дәлелдеудің бағытына тәуелсіз дәлелденеді. Себебі дәлелдеу солдан оңға қарай жүргізілгендіктен, солдан оңға қарай қозғалысты сәтті кері жөндеу деп қабылдаймыз:

1. repeat → (сәтті дәлелденеді)

1 repeat →←(repeat ке дейін жөндеу талпынысы)

2 repeat →←→ (сәтті дәлелденеді)

repeat предикаты Прологта келесі бағдарлама түрінде жазылуы мүмкін:

repeat.

repeat :– repeat.

Жолдарадың орнын ауыстырса не болады?

fail – предикат, әрқашан шыншыл емес. Ол ешқашан дәлелденбейді.

1. → fail (дәлелденбейді)
2. ← fail (дәлелденбейді)

Осыдан шығатыны: fail дан үтірден кейін жазылатындар ешқашан дәлелденбейді (біз онда жете алмаймыз).

fail предикаты Прологта келесі бағдарлама түрінде жазылуы мүмкін: fail :– 1 == 0.

repeat және fail дан келесі шексіз цикл алынуы мүмкін:

?– ..., repeat, ... , fail.

Бұл циклдан бағдарлама Out of memory бойынша шығады, себебі дәлелдеуде repeat дәлелдеудің жаңа бұтағы ретінде генерацияланып, жадыда орналасады.

Екі санның минимумын табу бағдарламасын қарастырайық:

1.min(X, Y, X) :– X<Y.

2.min(X, Y, Y).

егер X кіші Y, онда минимальды – X, әйтпесе – минимальды – Y. Бағдарламаны тексереміз:

?– min(2, 3, Min).

Min = 2

yes

бағдарламаның қатесі неде?

?– min(2, 3, Min), Min>2.

Min = 3

yes

Яғни минималды сан тең 2? Дұрыс шешім:

1 min(X, Y, X) :– X<Y.

2 min(X, Y, Y) :– X>=Y.

Дұрыс шешімнің екінші бір нұсқасы «кесумен»:

1 min(X, Y, X) :– X<Y, !.

2 min(X, Y, Y).

Кесу операторы Прологта «!» леп белгісімен белгіленеді. Кесу әрқашан тура дәлелдеуде дәлелденеді, ал жөндеу талпынысында қайта дәлелдеуді кесу көрсетілген ережеде рұқсат етпейді.

if-then-else конструкциясын Прологта қалай беруге болады:

A :- B, !, C.

A :- D.

Егер B дәлелдей алса, онда C дәлелденеді, әйтпесе D дәлелденеді. Егер кесуді алып тастаса, онда C дәлелдей алмай қалған жағдайда, B қайта дәлелденуі мүмкін, ал бұл if-then-else емес.

member предикаты (Item, List) элементтің тізімге кіріуінің бірнеше шешімін табуға мүмкіндік беретін. Оны тек бірінші кіруді іздейтіндей етіп жөндеуге болад.: member(Elem, [Elem | _]) :– !. member(Elem, [Head | Tail]) :– member(Elem, Tail).

Кесудің екі түрі бар: қызыл және жасыл.

Шартты бөлу.

Жасыл кесулер бағдарламаның орнидалу логикасына әсер етпей шешім болмайтын артық алу бұтақтарын кеседі. Қызылдар – шешімдерге әсер етіп оларлы кесім тастай алады.

Кесуді қолдануда **not** предикаты келтірілуі мүмкін

not(X) :– X, !, fail.

not(_).

Ол не істейді: егер X дәлелденсе, онда not – дұрыс емес, себебі кесу мен fail қосылуы not дәлелдеуінің сәтсіздігіне алып келеді. Егер X дәлелденбесе, онда X қандай екеніне қарамастан дұрыс болатын, бірінші ережеге альтернатива іздеуге тура келеді.

Мысал:

?– X = 2, not(X == 3).

X = 2

yes

?– X = 2, not(X == 2).

no

Бір тізімді басқа тізімнен алып тастау бағдарламасы **minus(L1, L2, Diff)**. Яғни Diff тек L1 де кездесіп, L2 де кездеспейтін элементтер орналасады.

```
minus([],_, []).
minus([X | L1], L2, L) :– member(X, L2), !, minus(L1, L2, L).
minus([X | L1], L2, [X | L]) :– minus(L1, L2, L).
```

Егер бастапқы тізім бос болса онда айырма нәтижесі де бос. Егер тізімнің басы нәтижелі тізімде бар болса, онда «басы» туралы ұмыта тұрамыз да алуды соңынан жасаймыз. Егер тізімде «басы» жоқ болса, онда қайта оралғанда «басын» нәтиже тізімге орналастыру керек.

Мысал:

```
?– minus([a, b], [b], L).
L = [a]
yes
```

read(X) – X ті ағымдағы кіру ағысынан оқу предикаты.

write(X) – X ті ағымдағы шығу ағысына шығару предикаты.

nl – каретканы ауыстыру белгісін шығу ағысына шығарады.

Қолдану мысалы:

?– write('Enter value: '), read(X), write('result = '), write(X).

Enter value: 12345.

result = 12345

X = 12345

Yes

12345 тен кейін нүкте тұрғанына назар аударыңыз!!! Осыдан кейін каретканы ауыстыру орындалған.

Файлдармен жұмыс істеу үшін:

see(ИмяФайла) – кіру ағысын файлдан берілгендерді оқу үшін бағыттайды.

seen – кіру ағысын жабады егер файлдан оқыған болса.

see(user) – кіру ағысын стандартты консолдан берілгендерді оқу үшін қайта бағыттайды. **tell(ИмяФайла)** – шығу ағысын файлдан берілгендерді шығару ұшін қайта бағыттайды.

told – шығу ағысын жабады, егер файлға шығарылған болса.

tell(user) – берілгендерді шығарудыңшығу ағысын стандартты консолға қайта бағыттайды.

Файлдар тек тізбектеле өңделе алады. Кіру файлынан әрбір оқу сұранысы ағымдағы кіріс ағысының ағымдағы

позициясынан оқиды. Оқылғаннан кейін ағымдағы пози-ция келесі әлі оқылмаған берілгендер элементіне ауысты-рылады. Келесі оқуға сұраныс жаңа ағымдағы позициядан оқуға алып келеді. Егер оқу сұранысы файлдың соңында орындалса, онда бұл сұранысқа **end_of_file** атомы беріледі (файл соңы). **Бір рет оқылған ақпаратты екінші оқу мүмкін емес.**

Файлда data.txt орналасады 12345 *нүктесіз және ауысты-ру кареткасынсыз.* Сұраныс нәтижесі:

?– see('data.txt'), write('Enter value: read(X),

write('result = '), write(X), seen.

Enter value:

uncaught exception: error(syntax_error('data.txt:1 (char:6) or operator expected after expression'), read/1)

Файлда data.txt орналасады 12345. *нүктемен және ауыс-тыру*

кареткасынсыз. Сұраныс нәтижесі:

?– see('data.txt'), write('Enter value: '), read(X),

write('result = '), write(X), seen.

Enter value: result = end_of_file

X = end_of_file

yes

В файле data.txt находится 12345. *с точкой и переводом каретки*.

Результат запроса:

?– see('data.txt'), write('Enter value: '), read(X), write('result35'), write(X), seen.

Enter value: result=12345

X = 12345

yes

Файлда берілгендер кіші әріптермен немесе бір таңбалы тырнақшаға алынуы керек!!!