# VI. Descrierea cerințelor individuale

a. Parsare:

- regulile vor avea formatul (ceea ce e scris cu gri e doar comentariu, nu face parte din format):

*reg^id (unde id este numărul regulii)*
*atribut_concluzie^(valoare && factor_de_certitudine^nr) (concluzia; nr este factorul de certitudine)*
*lista de premise { (condiţiile între acolade, premisele separate cu &&)*
*atr^valoare && (pentru atribute cu valori multiple)*
*atr && (pentru atribute booleene, valoare true)*
*^atr (pentru atribute booleene, valoare false)*
*}.*

- întrebările vor avea formatul:

*[atribut]*
*text-intrebare ^ 'conţinut întrebare'*
*optiuni ^ {val1 // val2 // val3 // ...} .*

- scopul se va defini:

*scop^atr.*

❖ **Modul de rezolvare:**

Se deschide fişierul cu reguli şi se citeşte câte o propoziţie cu predicatul **citeste_propozitie** până când nu se mai găseşte altă propoziţie. Fiecare propoziţie este procesată folosind predicatul **proceseaza**. Procesarea se face cu ajutorul regulilor DCG memorând în predicate dinamice informaţiile necesare.

b. Programul va citi dintr-un fişier de intrare datele pentru fiecare soluţie în parte. Acestea vor fi scrise în modul următor:

*nume_solutie /// descriere.*
*limbaje:l1,l2,....*
*----------*

Cuvântul *nume_solutie* va fi înlocuit cu numele soluţiei, iar cuvântul descriere cu textul corespunzător descrierii, iar separate prin virgule vor fi limbajele pe care ar fi recomandat ca un potenţial candidat să le cunoască.
După ce programul afişează soluţiile, va afişa şi un submeniu cu opţiunile: afis_detalii, meniu_anterior, iesire.
Pentru opţiunea *afis_detalii* se reafişeaza detaliat soluţiile (adică şi cu descrierea sub fiecare dintre ele). Fiecare două soluţii vor fi separate printr-o linie de #-uri.
Opţiunea *meniu_anterior* ne întoarce la meniu iniţial .
Opţiunea *iesire* termină programul.

- **Modul de rezolvare:**

Predicatul ***incarca_fis_desc*** citeşte fişierul cu descrieri. Acest predicat este apelat în predicatul ***submeniu***, care se ocupă cu afişarea şi procesarea opţiunilor. Predicatul ***submeniu*** este apelat imediat după afişarea soluţiilor. La selectarea opţiunii *afis_detalii* se apelează predicatul ***exec([afis_detalii])*** care face o afişare formatată a soluţiilor, descrierilor şi listei de limbaje.

c. Se va crea un predicat care generează o matrice în felul următor. Pentru o listă de posturi, se va calcula matricea în care prima linie conţine pe prima coloană un blank urmat pe coloanele următoare de numele posturilor. Următoarele linii vor corespunde fiecărui limbaj dintre cele precizate în fişier pentru posturile date. Astfel, pe fiecare astfel de linie, pe prima coloană avem limbajul, iar pe următoarele coloane un x dacă acel limbaj este necesar postului, şi un - dacă nu.
Trebuie să aveţi minim un set de răspunsuri din care rezultă cel puţin 3 soluţii (veţi specifica acest set de răspunsuri în documentaţie). Pentru aceste soluţii să aveţi şi cazuri de limbaje comune dar şi cazuri de limbaje care sunt necesare unui post dar altuia nu.
În meniul de după afişarea soluţiei, va apărea o opţiune afis_tabel, care, dacă e aleasă va afişa matricea creată la acest subpunct pentru ca utilizatorul să poată face o alegere mai informată (având în vedere că limbajele cerute nu sunt obligatorii mereu pentru un post; deci poate ajunge pe un post şi dacă nu ştie chiar tot, dar cu un factor de certitudine mai mic). Se va realiza o afişare formatată frumoasă a matricii, cu coloanele având drept dimensiune cel puţin numărul maxim de caracatere al cuvintelor de pe acea coloană.

- **Modul de rezolvare:**

Folosind predicatul ***get_joburi*** am obţinut lista de posturi pentru prima linie a matricei iar folosind predicatul ***get_limbaje*** obţinem lista de limbaje pentru posturile rezultate în soluţie.
Predicatul ***gen_matrice*** foloseşte ca predicat secundar, creând fiecare linie a matricei, apelând predicatul ***check_limbaje*** care verifică pentru fiecare post dacă un limbaj este necesar sau nu. Pentru a genera matricea în formatul specificat am creat predicatul ***generare*** pe care îl apelăm în opţiunea *afis_tabel* din meniul principal. De afişare se ocupă predicatul ***printLists*** care este apelat imediat după predicatul de generare.

d. Minim o întrebare va da posibilitatea de a răspunde cu mai multe variante (de exemplu întrebarea despre ce limbaje cunoaşte utilizatorul). Această întrebare va avea un format

special şi va fi memorată în baza de cunoştinţe cu un alt tip de predicat dinamic decât restul întrebărilor (pentru a putea fi făcută diferenţierea).

*rasp_multiplu[atribut]*
*text-intrebare ^ 'conţinut întrebare'*
*optiuni ^ {val1 // val2 // val3 // ...} .*

De asemenea în premisele regulilor, pentru atributele din tipul celor de mai sus se va permite specificarea unui subset de limbaje astfel: în loc de *atr^valoare* penru un atribut cu o singură valoare se va scrie:

*atr^(valoare1 & valoare2 & valoare3)*

şi va fi echivalent cu faptul că pentru atributul respectiv toate acele valori sunt valide.

- **Modul de rezolvare:**

Pentru acest tip de întrebare predicatul dinamic **interogabil** (folosit la parsare) are valoarea ultimului parametru, litera m semnificând multiplu. Procesarea întrebării se face repetându-se predicatul **citeste_opt** (citeşte opţiune de la utilizator) până când utilizatorul introduce opţiunea 'gata'. Repetarea predicatului are drept consecinţă asertarea unui fapt cu valoarea opţiunii alese până când se decide să se treacă la următoarea întrebare, astfel memorându-se un atribut cu mai multe valori în baza de cunoştinţe.

e. Se va crea un folder numit *output_joburi*. Se va adăuga o opţiune nouă în meniul principal, numită *consulta_detaliat*. În cazul în care utilizatorul selectează această opţiune, programul va folosi un director numit *log_joburi* (dacă există, îl va folosi pe acela, iar dacă nu există, îl va crea) aflat în interiorul folderului *output_joburi*. În acest folder va exista un fişier *log.txt* în care se va afişa pentru ultima consultare (deci fişierul este suprascris la fiecare consultare) un fel de log al luării deciziilor de către sistemul expert, în felul următor:

- Când va testa o regulă, va scrie în fişier: "Acum testez regula N." şi va afişa regula (afişarea regulii se va face în acelaşi format că şi în fişierul de demonstraţii). Numărul N reprezintă id-ul regulii.
- Când caută valoarea pentru un atribut (vezi predicatul realizează_scop), va afişa "Acum caut valoarea pentru atributul Atr". Se va înlocui Atr în propoziţie cu numele atributului.
- Când obţine valoarea pentru un atribut, fie din regulă fie din răspunsul utilizatorului, va scrie "Am obtinut valoarea pentru atributul Atr cu ajutorul utilizatorului/regulii N". Va fi scris "utilizatorului" dacă s-a obţinut valoarea din întrebare, respectiv "regulii N" dacă s-a obţinut valoarea din regula cu id-ul N.

- **Modul de rezolvare:**

Am creat directorul *output_joburi* cu ajutorul predicatului cu acelaşi nume, ***output_joburi***. Acest predicat îl apelăm la ***pornire***. Am adăugat opţiunea *consulta_detaliat* în meniul principal iar atunci când este selectată creează subdirectorul *log_joburi* în directorul *output_joburi*. Tot aici creăm şi fişierul *log.txt*. În continuare folosindu-ne de predicatul

dinamic ***cd*** pentru a scrie în fişier, îl asertam doar dacă a fost selectată opţiunea *consulta_detaliat*, altfel nu este assertat şi nu va scrie în fişier. În final apelăm predicatul ***scopuri_princ*** pentru a consulta sistemul. Am folosit predicatul ***realizeaza_scop*** pentru a scrie efectiv în fişier dacă predicatul dinamic este assertat. Pentru scrierea în fişier am folosit predicatul ***write*** şi am creat încă două predicate, ***premise1*** şi ***premise2***. ***Premise1*** ne ajută în parcurgerea şi scrierea atributelor iar ***premise2*** în parcurgerea şi scrierea premiselor atunci când scriem o regula.

f. În folderul *output_joburi* se va crea un folder *demonstratii* (prin program). În acest folder, în urma unei consultări a sistemului expert, se va crea câte un fişier numit *demonstratii_sv@solutie#fc.txt* pentru fiecare soluţie în parte. Cuvântul soluţie din numele fişierului va fi înlocuit cu valoarea soluţiei, iar fc cu factorul de certitudine. Fişierul va conţine, evident, demonstraţia pentru acea soluţie. În demonstraţii, afişarea regulilor se va face exact în formă în care au fost scrise în fişierul de intrare.:

*reg^id (unde id este numărul regulii)*
*atribut_concluzie^(valoare && factor_de_certitudine^nr)* *(concluzia; nr este factorul de certitudine)*
*lista de premise {* *(condiţiile între acolade, atributele separate cu virgulă)*
*atr^valoare &&* *(pentru atribute cu valori multiple)*
*atr &&* *(pentru atribute booleene, valoare true)*
*^atr* *(pentru atribute booleene, valoare false)*
*}.*

Forma de afişare a celorlalte tipuri de informaţii din demonstraţie se lasă la alegerea studenţilor.

- **Modul de rezolvare:**

Am creat subdirectorul *demonstratii* în directorul *output_joburi* cu ajutorul predicatului cu acelaşi nume, ***demonstratii***. Pe acesta îl apelăm în ***scopuri_princ***, adică la fiecare consultare, detaliată sau nu. Acest predicat verifică dacă directorul există deja altfel îl creează. Mai departe apelează predicatul ***fila***, care apelează la rândul sau predicatul ***numefis***, acesta se ocupă efectiv cu concatenarea tutoror elementelor pentru a crea numele fişierului. După ce fişierul a fost creat folosim predicaul existent ***cum*** pentru a scrie demonstraţiile în el. Pentru afişarea regulilor în formatul de mai sus am modificat predicatul ***afis_regula*** care este apelat de predicatul ***cum***.

# VII. Încheiere

În încheiere, în urma documentării şi realizării acestui proiect am conştientizat cât de importante sunt sistemele expert şi că sunt cea mai uşoară şi rapidă cale de a ajunge la un răspuns logic. Bineînţeles, acestea nu se comportă exact ca o persoană şi nu imită identic

creierul uman, însă prin simulare acestea pot simplifica răspunsurile anumitor întrebări şi a ne uşura viata de zi cu zi.

De asemenea, ni s-a părut interesant modul de luare a deciziilor, urmând nişte paşi concreţi, sistemul expert porneşte de la cunoaşterea scopului la verificarea fiecărei premise în parte până la calcularea unui factor de certitudine per total şi în final afişarea soluţiilor.

În concluzie, considerăm că sistemul nostru expert este util unei categorii de utilizatori deoarece mereu vor există studenţi/absolvenţi care au nevoie de îndrumare în carieră, şi anume în alegerea celui mai potrivit post. Chiar dacă nu este foarte precis, sistemul reuşeşte măcar să ghideze utilizatorul către unul sau mai multe posturi în care ar avea cele mai multe şanse de reuşită bazându-se pe cunoştinţele sale actuale.