Anca MIRON Daniela F. NISTE

SISTEME EXPERT ÎN ENERGETICĂ

Îndrumar de laborator

U.T.PRESS
Cluj- Napoca, 2025
ISBN 978-606-737-815-3

Anca MIRON *Daniela F. NISTE*

SISTEME EXPERT ÎN ENERGETICĂ

Îndrumar de laborator

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-815-3

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Sorin G. Pavel
Prof.dr.ing. Radu A. Tîrnovan

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-815-3

PREFAŢĂ

Prezentul material didactic este destinat să acopere activitatea practică din programa analitică a disciplinei *Sisteme Expert în Energetică,* pentru anul II, studii de master, specializarea energetică.

Îndrumarul de laborator a fost alcătuit ca un material didactic complementar orelor de curs, respectiv integrant a orelor de laborator, astfel încât noţiunile teoretice şi practice acumulate să ajute studenţii masteranzi să-și însușească un bagaj de cunoştinţe în domeniul sistemelor expert dedicate rezolvării problemelor caracteristice ingineriei energetice.

Concepute cu această abordare aplicativă, toate lucrările acestui material didactic sunt logic organizate și structurate. Într-adevăr, se începe cu lucrări de laborator simple, introductive din domeniul sistemelor expert, care treptat cresc în dificultate, ajungând, ca ultima lucrare de laborator să aibă cel mai mare nivel de complexitate în ceea ce privește utilizarea sistemelor expert în energetică.

Fiecare lucrare de laborator începe prin prezentarea clară a scopului lucrării, urmată de partea teoretică de care studenții au nevoie pentru înțelegerea problemelor și soluționarea exercițiile propuse. Toate lucrările de laborator vor fi realizate prin implicarea și utilizarea instrumentelor informatice dedicate implementării și construcției de sisteme expert specializate în soluționarea problemelor din energetică.

De asemenea, toate lucrările de laborator au ca material de studiu câte un sistem expert din domeniul energeticii. Trebuie specificat faptul că sistemele expert au fost dezvoltate ca material didactic și nu reflectă în totalitate realitatea ce se întâlnește în practică. Dar (dacă se omit informațiile care caracterizează problemele studiate), aspectele care fac referire la structura, caracteristicile, funcționarea, implementarea și testarea sistemelor expert sunt în totalitate conform

specificaților de specialitate.

Lucrările de laborator sunt împărțite în două mari categorii, și anume: (1) lucrări care folosesc sisteme expert cadru, respectiv (2) lucrări care se bazează pe utilizarea unor limbaje informatice sau librării dedicate dezvoltării de sisteme expert.

Competențele dobândite la finalul orelor aplicative sunt menite să le permită absolvenților abordarea și soluționarea cu succes a problemelor întâlnite în calitate de specialişti energeticieni folosind sisteme expert.

Cu toate eforturile depuse, autorii sunt convinşi că această formă a prezentului material didactic poate fi îmbunătăţită prin continuarea preocupărilor în domeniu, prin sugestiile studenţilor care îl vor utiliza precum şi ale altor specialişti.

Autorii

CUPRINS

1

STRUCTURA ȘI FUNCȚIONAREA SISTEMELOR EXPERT

1.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta structura de bază și modul de operare a sistemelor expert în vederea înțelegii acestor aspecte de bază ale respectivelor aplicații de IA, și anume a sistemelor expert dedicate rezolvării problemelor din energetică.

Pentru atingerea scopului se va folosi un sistem expert de clasificare dedicat clasificării problemelor de calitate a energiei electrice din sistemele electroenergetice (SECEE), care a fost dezvoltat prin implicarea unui sistem expert cadru (expert system shell), și anume CLIPS IDE.

1.2. Obiectivele problemei

Obiectivele lucrării sunt definirea sistemelor expert și a rolului lor precum și prezentarea componentelor de bază ale acestora. De asemenea se va demonstra modul de funcționarea unui sistem expert de clasificare particular. Pentru a aprofunda aspectele teoretice, se va testa sistemul expert studiat pentru a se scoate în evidență caracteristcile sistemelor expert de clasificare. La final se vor evidenția avantajele și limitările sistemul expert studiat.

1.3. Aspecte teoretice

Acest subcapitol cuprinde aspectele teoretice necesare înțelegerii părții practice ale acestei lucrări de laborator.

1.3.1. Sistemele expert

Sistemele expert sunt programe informatice din categoria aplicațiilor de inteligență artificială (IA), care copiază modul în care un expert uman rezolvă o problemă, care necesită cunoștințe de înaltă expertiză și experiență.

Sistemele expert se împart în trei mari categorii, dacă se ia în considerare scopul lor, și anume: sisteme expert de clasificare, sisteme expert de control și sisteme expert de prognoză.

Structura de bază a unui sistem expert cuprinde patru componente de bază, care au un rol bine definit. Aceste componente sunt baza de cunoștințe (BC), motorul de inferență (MI), modulul explicativ (ME) și modului de interfață cu utilizatorul (MIU). Aceste componente, precum și legătura dintre ele sunt prezentate grafic în figura 1.1.

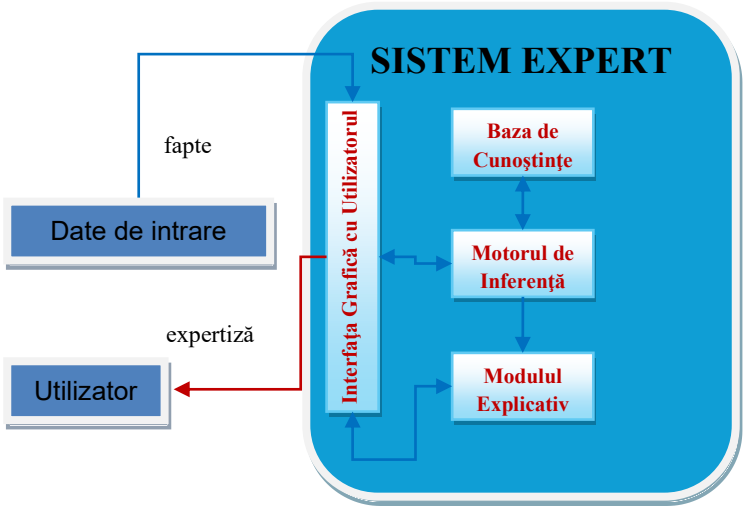


Figura 1.1. Structura unui sistem expert

Baza de cunoștințe reprezintă componenta care conține toate datele din domeniu de specialitate, adică cunoștințele expertului

uman. Baza de cunoștințe cuprinde datele sub formă de reguli și fapte. Regulile arată legătura dintre datele caracteristice problemei. Ele se schimbă foarte rar, ele fiind cunoștințele definitorii în rezolvarea problemei. Faptele sunt date care arată starea problemei de rezolvat. Ele pot fi date de intrare sau date intermediare.

Baza de cunoștințe comunică bidimensional cu MI, întrucât ia datele rezultate din acțiunile MI, dar și transmite acestuia din urmă datele necesare rezolvării problemei în funcție de datele de intrare. Baza de cunoștințe comunică cu celelalte componente ale unui SE indirect, prin intermediul MI.

Motorul de inferențe este un algoritm rezolutiv, care folosind datele de intrare, selectează regulile potrivite din BC pentru a determina soluția. Eficiența MI depinde de strategia de control folosită, algoritmii de căutare implicați și metodele de rezolvare a conflictelor utilizate. Figura 1.2. prezintă procesul de operare a MI, și în mare modul de operare a unui sistem expert (SE).

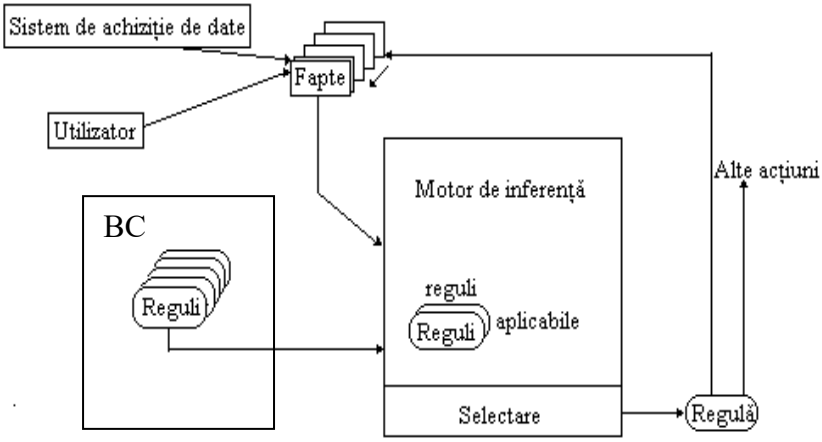


Figura 1.2. Modul de funcționare a MI a unui SE [1]

Motorul de inferență funcţionează prin realizarea repetitivă a patru etape principale:

1) Identificarea regulilor aplicabile, prin compararea faptelor cu datele care se regăsesc în reguli. Aceasta este etapa de

potrivire a datelor, în care MI examinează faptele (memoria de lucru) și caută regulile ale căror condiții (premisa, partea DACĂ) sunt îndeplinite de faptele actuale. Astfel, se compară antecedentele fiecărei reguli cu datele cunoscute pentru a detecta care reguli sunt aplicabile, adică eligibile pentru execuție. Acest pas determină regulile logice sunt candidate pentru raționament în contextul actual.

2) Selectarea regulilor care se aplică - Dacă se constată că se aplică mai multe reguli, MI folosește o metodă de rezolvare a conflictelor pentru a determina care regulă ar trebui declanșată prima.
3) Aplicarea regulilor - Odată ce o regulă este selectată, MI execută partea de acțiune a regulii (partea ATUNCI). Această acțiune duce de obicei la adăugarea de noi fapte BC, modificarea sau eliminarea faptelor existente și declanșarea de acțiuni, cum ar fi furnizarea unei recomandări, a unui diagnostic sau a unei decizii. Aceasta este faza în care sistemul își dezvoltă dinamic cunoștințele prin actualizarea stării sale interne.
4) Verificarea criteriului de oprire - După aplicarea unei reguli, MI verifică dacă procesul de raționament ar trebui să continue sau să se oprească. Criteriile de oprire pot include: atingerea unui obiectiv (de exemplu, sistemul a găsit o soluție), nu mai sunt aplicabile reguli, executarea unui număr predefinit de cicluri și atingerea unui prag de încredere. Acest pas asigură că procesul de inferență nu rulează la nesfârșit și se oprește atunci când raționamentul s-a finalizat cu succes sau a ajuns la un pas de stop logic.

Motorul de inferență comunică cu ME și MIU. Într-adevăr, MI trimite etapele rezolvării problemei ME pentru ca acesta să poată trimite explicațiile corespunzătoare utilizatorul. De asemenea, MI poate transmite sau cere informații prin intermediul MIU.

Modulul explicativ este în legătură bidirecțională cu MIU și MI. Această componentă conține un algoritm prin care se caută informația din SE, pentru ca utilizatorul să primească răspunsul la întrebarea pusă SE. Rolul ME este esențial în înțelegearea de către utilizator a problemei și a modului de rezolvarea a acesteia, asemenea expertului uman.

Modulul de interfață cu utilizatorul reprezintă elementul component al unui SE cu care utilizatorul intră în contact, deci comunică cu acesta. Astfel, importanţa MIU este justificată, având în vedere că de el depinde acceptarea inițială a SE. Acesta conține toate elementele grafice, text și complexe necesare utilizatorului și SE să comunice corespunzător, dar și să permită controlul asupra rezolvării soluției, respectiv accesul la date. Acest aspect este ilustrat în figura 1.3.

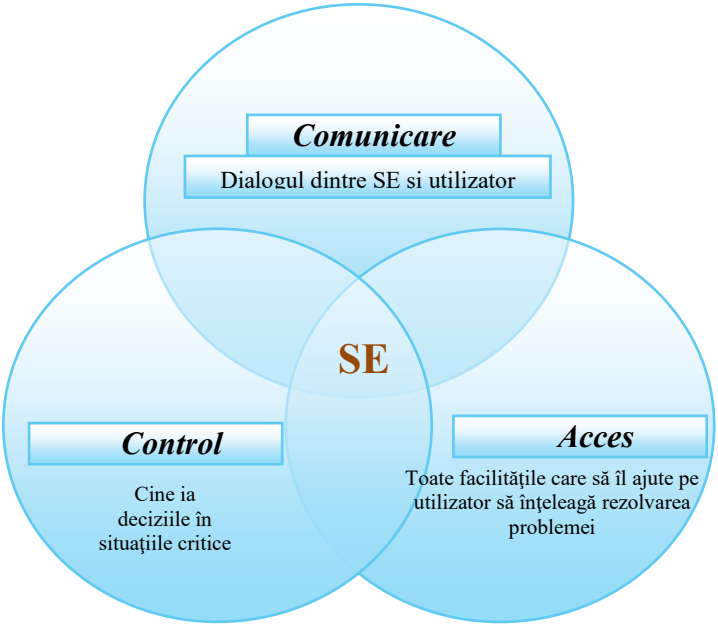


Figura 1.3. Caracteristicile MIU al unui SE

1.3.2. Sistemul de Producție Integrat în limbajul C (CLIPS)

CLIPS (C Language Integrated Production System - Sistemul de Producție Integrată în Limbajul C) este un sistem expert cadru, bazat pe reguli și dezvoltat de NASA între 1985 și 1996. Scris în C pentru portabilitate, CLIPS poate fi instalat și utilizat pe o gamă largă de platforme [2, 3]. Din 1996, CLIPS este disponibil ca software de domeniu public, și se poate accesa la adresa [4]. O captură a interfeței cu utilizatorul a CLIPS IDE versiunii 6.4.2 din 2025 este ilustrată în figura 3.4.

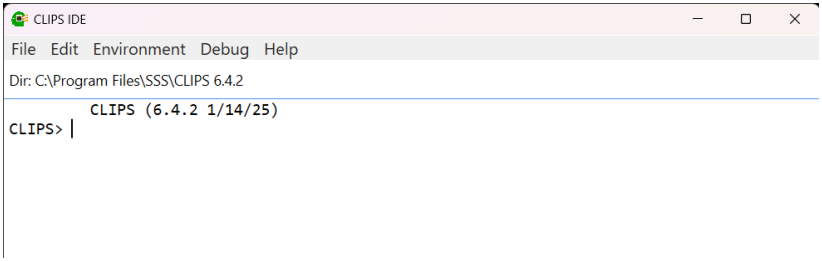


Figura 1.4. Captură CLIPS IDE 6.4.2

CLIPS este clasificat ca un instrument software dedicat dezvoltării de sisteme expert deoarece este un mediu complet, care include caracteristici precum un editor integrat și un instrument de depanare. Astfel CLIPS oferă elementele de bază ale unui sistem expert, anume:

1. listă de fapte și listă de instanțe: memorie globală pentru date;
2. bază de cunoștințe: conține toate regulile, baza de reguli;
3. motor de inferență: controlează execuția generală a regulilor.

Un program scris în CLIPS poate consta din reguli, fapte și obiecte. Motorul de inferență decide ce reguli ar trebui executate și când. Un sistem expert bazat pe reguli scris în CLIPS este un program bazat pe date, unde faptele și obiectele, dacă se dorește, sunt datele care stimulează execuția prin intermediul motorului de inferență.

CLIPS IDE oferă posibilitatea rulării unei aplicații C (*.clp) din câmpul ”Environment” a meniului principal și analiza în timp real a

tuturor pașilor realizați de aplicație, prin intermediul câmpului ”Debug” din meniul principal.

Ca și în cazul altor medii de programare, CLIPS recunoaște anumite cuvinte cheie. De exemplu pentru a introduce date în lista de fapte, se poate utiliza comanda:

assert,

linia de comandă va fi următoarea:

CLIPS> (assert (generator))

iar răspunsul va fi:

<Fact - 1>.

Celelalte cuvinte cheie care se vor folosi în cadrul orei de laborator, precum și rolul lor, sintaxa și exemple sunt prezentate în tabelul 1.1. Așa cum se poate observa din tabel, toate comenzile sunt introduse între paranteze rotunde, aspect preluat de la LISP (List Programming), chiar dacă la bază este C, ci nu acesta.

Tabelul 1.1. CLIPS – cuvinte cheie caracteristice

Element SE	**Cuvânt cheie**	**Rol / Sintaxa**	**Exemplu**
Fapte (facts)	Assert	Introduce o dată de tip fapt în lista CLIPS	(assert (LEA))
	Clear	Șterge lista de fapte	
	Retract	Eliminarea unui fapt din listă	(retract 3) Eliminarea faptului de pe poziția 3
	Facts	Permite afișarea listei de fapte. Afișare statică	(facts)
	Watch	Permite afișarea automată a faptelor fără a folosi comanda (facts). Afișare dinamică	(watch)
	Reset	Elimină toate activările din agendă, retrage toate faptele, șterge toate instanțele, atribuie variabilelor globale valorile	(reset)

		lor inițiale, afirmă toate faptele din construcțiile deffacts, creează toate instanțele din construcțiile definstances și setează modulul curent și focalizarea la modulul MAIN.	
	deffacts	Definește toate faptele în același timp.	(deffacts (LEA) (LEA1))
Reguli (rules)	Defrule	Introducerea unui reguli Dacă condiție, Atunci actiune. (defrule rule_name "optional_comment" (pattern_1) (pattern_2) (pattern_N) => (action_1) (action_2) (action_M))	(defrule R1 (LEA) => (assert (bine))
	ppdefrule	Afișarea unei reguli.	(ppdefrule R1)
	Rules	Afișarea tuturor regulilor în ordinea în care au fost introduse.	(rules)
	Agenda	Afișează lista de reguli care vor fi executate și ordinea lor.	(agenda)
Variabile	?<nume-variabila>	Modul de scriere a unei variabile.	?tens
	deftemplate	Definește un câmp de variabile denumite *slot*.	(deftemplate LEA (slot lungime) (slot curent))
	declare salience	Sintaxa folosită pentru a atribui o importanță	(declare (salience 100))

		regulilor. Se recomandă să se folosească rar pentru a nu afecta procesul rezolutiv.	
	type	Indică tipul variabilei.	type (tens float)
	run	Determină rularea programului.	(run)
	printout	Afișează un mesaj.	(printout t “text” crlf)
	do-for-all-facts	Realizează acțiunea pentru toate faptele.	
	fact-slot-value	Face referire la valoarea slotului unui fapt.	

Din tabelul 1.1 se poate conluziona, că o bază de cunoștințe în CLIPS IDE cuprinde definirea variabilelor și a altor date cu care lucrează sistemul, scrierea regulilor simple, precum și a unor reguli prioritare.

CLIPS IDE are implementat un motor de inferențe, care funcționează folosind strategia de control înainte, bazată pe date. Algoritmul de căutare implementat este algoritmul Rete, introdus prima dată în 1979 de Charles Forgy. Acest algoritm intră în categoria algoritmilor pattern-maching (potrivire a modelelor), și are capacitatea de a compara multe fapte cu reguli. Algoritmul Rete își bazează funcționarea pe construirea unei rețea de noduri, în care acestea reprezintă condițiile regulilor. Astfel, algoritmul evită parcurgerea regulilor executate, plus sunt memorate conexiuni parțiale. În ceea ce privește metodologia de rezolvare a conflictelor, CLIPS folosește trei metode astfel: (1) metoda priorității – se vor folosi prima dată regulile care au prioritate mai mare (salience), (2) dacă regulile aplicabile au aceeași prioritate, se va executa prima dată regula care folosește cele mai multe dintre faptele recente, și (3) metoda prima regulă aplicabilă (ordinea introducerii în baza de reguli).

CLIPS IDE nu are un ME implicit, dar are toate elementele componente prin care se poate construi unul.

În ceea ce privește interfața cu utilizatorul, CLIPS IDE are o interfață bazată pe text, iar pentru construirea unei interfețe grafice este nevoie de implicarea unei aplicații externe.

1.4. Sistemul expert de clasificare SECEE

Sistemul expert prototip SECEE a fost dezvoltat pentru a fi suportul de studiu al primei lucrări de laborator. În prima etapă, el a fost dezvoltat în varianta teoretică, iar apoi a fost implementat în CLIPS IDE. În acest capitol este prezentată varianta scrisă.

SECEE este un sistem expert de clasificare, care are ca date de intrare caracteristici ale energiei electrice, care se pot obține cu ajutorul unui analizor de calitate a energiei electrice. Datele de ieșire sunt reprezentate de un mesaj text prin care se face diagnoza de bază a calității energiei electrice în funcție de datele de intrare. În continuare se prezintă cele patru elemente componente de bază a SECEE.

1.4.1. Baza de cunoștințe

Baza de cunoștințe cuprinde definirea datelor de intrare, adică contextul, baza de reguli și faptele.

Datele de intrare, care vor reprezenta faptele introduse de utilizator sunt:

- Tensiunea [V] – valoarea efectivă a tensiunii. Această mărime se folosește pentru a identifica golurile de tensiune, supratensiunile și întreruperile în alimentare.
- Curent [A] – valoarea efectivă a curentului electric. Această mărime oferă informații despre mărimea sarcinii.
- THD [%] – valoarea factorului de distorsiune armonică a tensiunii. Această mărime este implicată în determinarea existenței unui regim deformant care depășește limita impusă în standard de 5 %.
- Defazaj [%] – este o mărime numerică procentuală care indică defazajul semnalului de tensiune real față de cel ideal de 2 %.
- Durata [s] – reprezintă intervalul de timp în care a avut perturbația (defectul de calitate a energiei electrice). Această mărime este folosită pentru a identifica golul de tensiune, supratensiunea normală de cea tranzitorie, respectiv întreruperile de scurtă durată de cele de lungă durată.

- Flicker frecvență [Hz] – este o valoarea numerică care reprezintă frecvența flickerului.

Aceste fapte sunt folosite pentru a definii 24 de reguli, care vor fi folosite pentru a oferi diagnoza finală. Regulile cuprind și informații despre cauza perturbațiilor, date care sunt folosite de către ME. Baza de reguli în varianta parțială, conținând primele 11 reguli este următoarea:

Gol de tensiune

R1. DACĂ tensiunea scade între 10% și 90% din valoarea nominală a tensiunii pentru mai puțin de 1 minut, ATUNCI tipul perturbației = gol de tensiune.

R2. DACĂ golul de tensiune apare simultan cu un curent ridicat, ATUNCI cauza probabilă = scurtcircuit.

R3. DACĂ golul de tensiune apare în timpul pornirii motorului, ATUNCI cauza = pornire motor dimensiune mare.

R4. DACĂ golul de tensiune apare doar pe o fază, ATUNCI cauza = defect sau dezechilibru monofazat.

Supratensiune

R5. DACĂ tensiunea crește peste 110% din valoarea nominală pentru mai puțin de 1 minut, ATUNCI tipul perturbației = supratensiune.

R6. DACĂ supratensiunea coincide cu comutarea condensatorului, ATUNCI cauza = conectarea bateriei de condensatoare.

R7. DACĂ supratensiunea apare după deconectarea unei sarcini mari, ATUNCI cauza = reducerea bruscă a sarcinii.

Armonice

R8. DACĂ THD > 5%, ATUNCI perturbație = distorsiune armonică.

R9. DACĂ ordinul armonic dominant = 3 sau 5, ATUNCI sursa probabilă = sarcini neliniare (redresoare, convertoare).

R10. DACĂ armonicele apar sunt de ordine impare (3, 5, 7), ATUNCI cauză = convertoare industriale sau iluminat fluorescent.

R11. DACĂ distorsiunea armonică crește atunci când se adaugă o sarcină neliniară, ATUNCI confirmare = sursă armonică prezentă.

1.4.2. Motorul de inferență

Motorul de inferență considerat are ca strategie de control, strategia de control înainte, care este bazată pe date, și deci ideală pentru sistemele expert de clasificare, care primesc multe date de intrare precum SECEE. Luându-se în considerare faptul că SECEE este implementat în CLIPS IDE, atunci rezultă că MI este implicit, folosind metodologia de operare prezentată anterior, adică strategia de control înainte, algoritmul Rete și rezolvarea conflictelor după indicatorul de prioritate, numărul de fapte și ordinea din baza de reguli.

1.4.3. Modulul explicativ

Modulul explicativ al SECEE oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, dar și despre cauza apariției unei anumite perturbații. Primul aspect se poate obține implicit prin evaluarea treptată a pașilor realizați de aplicație până la soluționarea problemei, iar explicațiile suplimentare sunt introduse ca reguli. Aceste reguli se observă în lista de reguli prezentată anterior.

1.4.4. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SECEE este tip text, având în vedere că CLIPS IDE nu oferă facilități grafice în această direcție, ci doar text. Și având în vedere scopul acestei lucrări de laborator, nu s-a insistat pe acest aspect, și deci comunicarea cu utilizatorul a SECEE este doar text.

1.4.5. Implementarea CLIPS IDE

SECEE a fost implementat în CLIPS IDE, așa cum ilustrează și figura 1.5., unde este prezentată partea de context a BC, adică definirea unor obiecte și variabile care vor fi folosite de către SECEE.

Liniile de comandă implementate în CLIPS IDE au fost salvate sub forma unui fișier text, care reprezintă materialul de lucru din cadrul orelor aplicative. În aceast material didactic sunt prezentate doar o parte din componentele SECEE implementate.

Baza de cunoștințe a SECEE a fost populată cu reguli de producție, care au fost grupate în funcție de perturbație. În continuare

sunt prezentate regulile pentru golul de tensiune, respectiv pentru regimul complex.

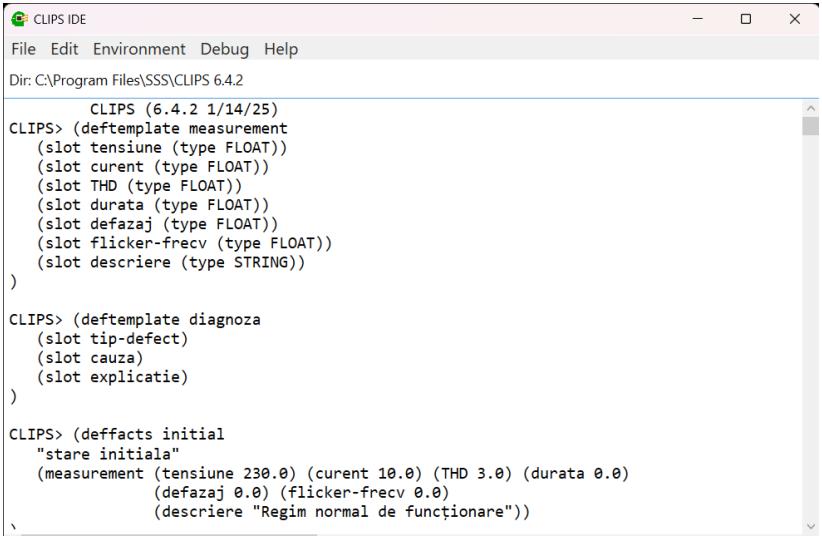


Figura 1.5. Captură SECEE – partea de context a BC

```
(defrule gol
  (date-intrare (tensiune ?v&:(< ?v 207)) (durata ?d&:(< ?d 60)))
  =>
  (assert (diagnoza (tip-defect "Gol de tensiune")
                    (cauza "Scurtcircuit sau Pornirea unui motor")
                    (explicatie "Tensiunea scade sub 90% din tensiunea
nominala pentru o durata < 60s")))
  (printout t "*** A fost detectat un gol de tensiune ***" crlf)
)

(defrule complex
  (diagnoza (tip-defect "Gol de tensiune"))
  (diagnoza (tip-defect "Distoriune armonica"))
  =>
  (assert (diagnoza (tip-defect "Suprapunere perturbatii")
                    (cauza " Retea de alimentare slaba")
```

```
                    (explicatie " Probleme de calitate care se suprapun:
    gol de tensiune + armonici")))
      (printout t "*** Perturbatie complexa detectata ***" crlf)
    )
```

MIU a fost de asemenea inclus ca o regulă cu prioritatea 100, prin care SECEE cere utilizatorului introducerea datelor:

```
    (defrule utilizator
      (declare (salience 100))
      =>
      (printout t crlf "====== SECEE ======" crlf)
      (printout t "Introduce valorile marimilor electrice masurate:"
crlf)
      (printout t "Tensiune (V): ") (bind ?v (read))
      (printout t "Curent (A): ") (bind ?i (read))
      (printout t "THD (%): ") (bind ?t (read))
      (printout t "Durata (s): ") (bind ?d (read))
      (printout t "Defazaj (%): ") (bind ?p (read))
      (printout t "Flicker frecventa (Hz): ") (bind ?f (read))
      (assert (date-intrare (tensiune ?v) (curent ?i) (THD ?t)
                    (durata ?d) (defazaj ?p)
                    (flicker-frecv ?f)
                    (descriere "date de intrare")))
```

Componenta ME a fost inclusă tot prin intermediul unei reguli ”de-ce”, care are prioritatea 90:

```
    (defrule de-ce
      (declare (salience 90))
      =>
      (printout t crlf "Vrei sa sti de ce a fost data aceasta solutie?
(yes/no): ")
      (bind ?ans (read))
      (if (eq ?ans yes)
        then
        (printout t crlf "------ EXPLICATIE ------" crlf)
        (do-for-all-facts ((?d diagnoza)) TRUE
          (printout t "Tip defect: " (fact-slot-value ?d tip-defect) crlf)
          (printout t "Motivul: " (fact-slot-value ?d explicatie) crlf)
```

```
        (printout t "Cauza problemei: " (fact-slot-value ?d cauza)
crlf crlf)
      )
    )
  )
```

1.4.6. Testarea sistemului expert

Testarea SE se realizează prin rularea lui, pentru a i se verifica eficacitatea. De asemenea, se poate urmări procesul de soluționare prin selectarea din ”Debug” a ”Agenda”, ”Facts” și ”Instances”, respectiv ”Watch” – All.

În figura 1.6. este ilustrată testarea SECEE realizată prin rularea aplicației cu linia de comandă (run), cu datele de intrare clar evidențiate.

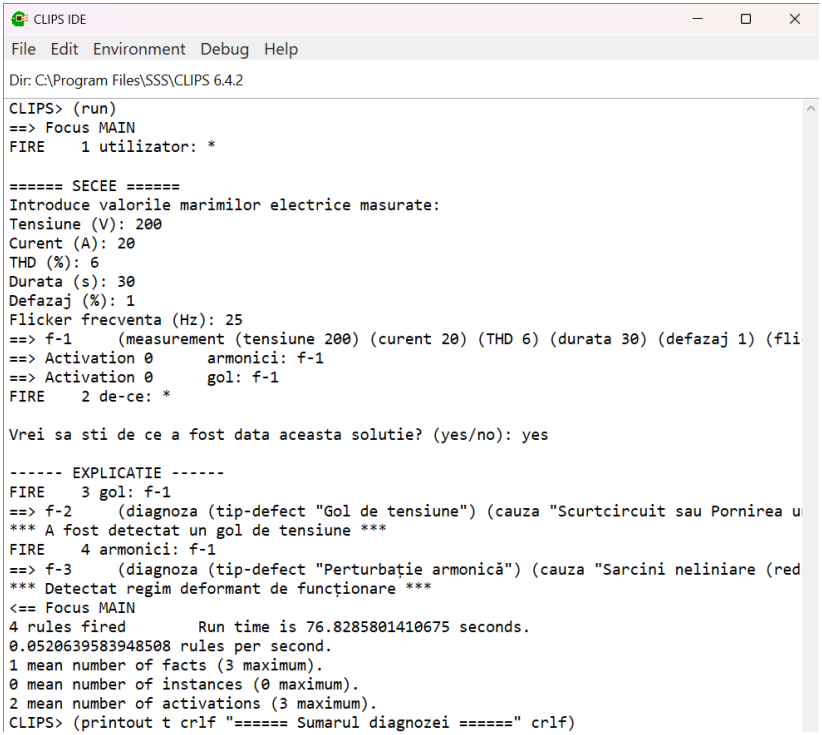


Figura 1.6. Captură SECEE – testarea sistemului expert

1.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea sistemului expert prezentat anterior prin folosirea fișierului text SEE-L1-SECEE.txt oferit în cadrul orei aplicative.
2. Activarea opțiunilor în CLIPS IDE necesare ilustrării etapelor realizate de aplicație în rezolvarea problemei, conform 1.4.6.
3. Analiza atentă a aplicației pentru identificarea celor trei componente ale SECEE, adică BC, ME și MIU.
4. Depanarea și testarea SECEE implementat prin introducerea setului de date de intrare din figura 1.6.
5. Testarea SECEE prin rularea sistemului folosind datele de intrare din tabelul 1.2.
6. Rezultatele se trec în tabelul 1.2.

Tabelul 1.2. Valorile testării sistemului expert SECEE

Date de intrare	**Rezultatele obținute**	**Observații**
Tensiune = 230 Curent = 10 THD = 2 Durata = 0 Defazaj = 0 Flicker = 0		
Tensiune = 200 Curent = 10 THD = 0 Durata = 30 Defazaj = 0 Flicker = 0		

Tensiune = 260 Curent = 10 THD = 0 Durata = 10 Defazaj = 0 Flicker = 0		
Tensiune = 230 Curent = 10 THD = 7 Durata = 0 Defazaj = 0 Flicker = 0		
Tensiune = 220 Curent = 20 THD = 0 Durata = 0 Defazaj = 4 Flicker = 0		
Tensiune = 235 Curent = 20 THD = 0 Durata = 0 Defazaj = 0 Flicker = 15		

Tensiune = 360 Curent = 20 THD = 0 Durata = 0 Defazaj = 0 Flicker = 0		
Tensiune = 0 Curent = 0 THD = 0 Durata = 30 Defazaj = 0 Flicker = 0		
Tensiune = 0 Curent = 0 THD = 0 Durata = 120 Defazaj = 0 Flicker = 0		
Tensiune = 190 Curent = 50 THD = 7 Durata = 30 Defazaj = 3 Flicker = 15		

1.6. Prelucrarea datelor și interpretarea rezultatelor

Datele din tabelul 1.2. se vor folosi pentru a determina următoarele informații, cu care se va completa coloana de ”Observații” a tabelului:

- Ordinea executării regulilor;
- Timpul de execuție;
- Numărul de reguli implicate.

1.7. Concluzii

Se vor formula concluzii asupra structurii și funcționării SECEE, precum și asupra avantajelor și limitărilor SECEE și a CLIPS IDE.

.....

.....

.....

.....

.....

.....

.....

2

CARACTERISTICILE SISTEMELOR EXPERT

2.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta caracteristicile de bază ale sistemelor expert în vederea înțelegii avantajelor și limitărilor sistemelor expert dedicate rezolvării problemelor din energetică.

Pentru atingerea scopului se va folosi un sistem expert de clasificare dedicat clasării problemelor apărute la mașini electrice care intră în componența sistemelor electroenergetice, mai exact realizează diagnoza defectelor la transformatoare de putere, generatoare, motoare electrice, dar și altele, care a fost dezvoltat prin implicarea unui sistem expert cadru (expert system shell), și anume SWI-Prolog. Sistemul expert dezvoltat a fost denumit SEDD.

2.2. Obiectivele lucării

Obiectivele lucrării sunt prezentarea atributelor sistemelor expert, precum și comparația cu programele informatice convenționale. De asemenea se demonstrează modul de funcționare al sistemului expert studiat pentru a se scoate în evidență avantajele și dezavantajele programului informatic. În ultima etapă se va testa sistemul expert SEDD pentru diferite valori ale datelor de intrare.

2.3. Noţiuni teoretice

Această secțiune prezintă aspectele teoretice studiate în această lucrare de laborator, și anume atributele unui SE, respectiv prezentarea aplicației informatice folosite, și anume SWI-Prolog.

2.3.1. Caracteristicile sistemelor expert

Un sistem expert se diferențiază de un program informatic convențional (PIC) din mai multe puncte de vedere, precum modul de soluționare a problemei, tipul datelor de intrare, dar și capacitatea de a oferi o explicație. Astfel, la un program convențional soluția este dată de algoritm, dar în cazul unui SE aceasta este dată de reguli și inferență. Datale de intrare la un PIC trebuie să fie corecte, contrar unui SE care poate lucra cu date incomplete și imprecise. Dacă se ia în considerare controlul în cadrul celor două, PIC și SE, la prima controlul este deținut de ordinea declarației, iar la SE de către motorul de inferență.

Atributele unui SE sunt acele caracteristici care le fac să se distingă de celelalte programe informatice convenționale, dar și de sistemele bazate pe cunoaștere. În paragraful precedent s-a prezentat o comparaţie între SE şi programele convenţionale, luând în considerare diferențele dintre acestea, se pot concluziona caracteristicile care definesc SE ca atare:

1. Nivel expert de competenţă – cunoștințele cu care lucrează un SE sunt dintr-un domeniu relativ restrâns şi sunt bine specificate, întrucât ele se bazează pe cunoștințele și experiența din domeniu a unui / a unor experți umani;. Cu observația că expertiza efectuată de SE depinde de cunoștințele introduse în baza de cunoștințe a SE.
2. Reprezentare declarativă a cunoștințelor – prin copierea modului de rezolvare a unei probleme de către un expert uman, SE împrumută o abordare cvasi-naturală de a introduce cunoștințele, de cele mai multe ori sub formă de reguli.

3. Mecanism de inferență – componenta care realizează deducțiile logice pentru a soluționa probleme este motorul de inferenţă.
4. Componență modulară - baza de cunoștințe este net separată de mecanismul de manipulare a cunoștințelor. Baza de cunoștințe poate fi folosită ulterior în alte aplicaţii, sau modificată în funcţie de necesităţile problemei. La fel, motorul de inferență poate fi implementat în alt SE de același tip (clasificare, control sau prognoză).
5. Comunicarea cu utilizatorul se face pe baza unor module specializate care realizează integrarea, comunicarea, explicarea și furnizarea de cunoștințe utilizatorului.

2.3.2. SWI – Prolog (PROgramming in LOGic)

SWI-Prolog este un instrument informatic open-source (gratuit) obținut prin implementarea limbajului de programare de nivel înalt Prolog. Astfel, acest instrument informatic respectă în mare măsură standardul ISO pentru Prolog, dar include și multe extensii practice care îl fac mai ușor de utilizat pentru aplicații din lumea reală.

SWI-Prolog oferă reguli de gestionare a constrângerilor, care permit utilizatorilor să definească o metodologie personalizată de rezolvare a constrângerilor. Sistemul acceptă programarea modulară, astfel încât se pot organiza și controla componentele în module separate. Mai mult, acest instrument include o colecție bogată de biblioteci încorporate pentru acțiuni precum manipularea listelor și a șirurilor de caractere, operațiuni de fișiere și de intrare/ieșire, comunicare HTTP, gestionarea JSON și RDF și conectivitate la baze de date.

De asemenea, SWI-Prolog are predefinit un depanator puternic care ajută la parcurgerea regulilor și la observarea procesului prin care motorul de inferență încearcă să demonstreze obiectivele, deci are o variantă rudimentară a modulului explicativ.

Interfața cu utilizatorul a SWI-Prolog este de tip text, ceea ce se observă în figura 2.1., în care este ilustrată captura acestuia. Aici se poate observa meniul principal al instrumentului informatic, care cuprinde File, Edit, Settings, Run, Debug și Help.

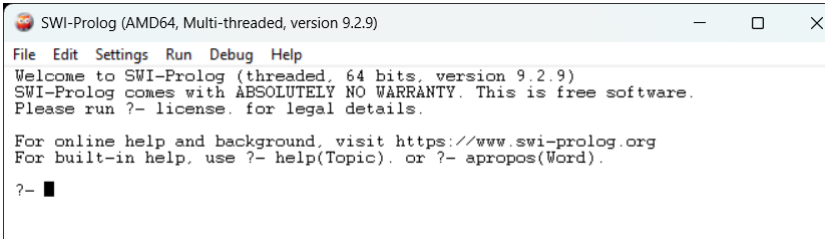


Figura 2.1. Captură SWI-Prolog, versiunea 9.2.9

File (fișier) permite deschiderea, salvarea și închiderea fișe=ierelor cu codul Prolog, având extensia ”pl”. Edit (editare) oferă funcționalități standard de editare a textului, precum și funcții specifice legate de codul Prolog. Debug (depanare) accesează funcții de depanare, cum ar fi oprirea execuției, examinarea variabilelor și urmărirea pașilor codului. Run (executare) permite rularea unei aplicații Prolog și compilarea fișierelor. Settings (setări) oferă accesul la diferite opțiuni de configurare a mediului de dezvoltare, inclusiv teme, gestionarea bibliotecilor și setări avansate.
, iar Help (ajutor) conține documentația online, manualul de utilizare și alte resurse de ajutor pentru a învăța SWI-Prolog [5].

SWI-Prolog poate include mai multe metode de reprezentare a cunoștințelor, care sunt sintetizate în tabelul 2.1.

Tabelul 2.1. Metode de reprezentare a cunoștintelor din SWI-Prolog

Reprezentarea cunoștințelor	**Sintaxa / Exemplu**
Declarativă Fapte și reguli	dispozitiv(transformator) defect(supraîncălzire) defect(supraîncălzire) :- temperatura(T), T > 90.
Procedurală	diagnoza(defect) :- verifica-conditii(defect), raport(defect).

Ierarhică	componenta(generator, [stator, rotor, rulmenti]). proprietate(rulment, [temperatura:85, vibratie:5.2]).
Relațională	conectat(trafo1, bara3). alimentare(generator1, trafo1).
Cantitativă	Sarcina-mare :- curent(I), I > 120. eficiența(E) :- input(Pin), output(Pout), E is (Pout / Pin) * 100.
Simbolurile colorate sunt predefinite în SWI-Prolog, resul sunt definite de utilizator.	
Meta	known(Fact) :- clause(Fact, _). Verificarea dacă o faptă este în baza de fapte. Esențial pentru modulul explicativ.
Ontologică	rdf(dispozitiv1, tip, transformator). rdf(dispozitiv1, temperatura, '95°C').

În ceea ce privește motorul de inferență, SWI-Prolog folosește strategia de control înapoi, căutarea în adâncime și metoda de prezolvare a conflictelor – aplicarea regulii din stânaga. Acest aspect se observă și prin modul de scriere a regulilor, care încep cu condiția și se finalizează cu premisa, astfel este mai ușor selectarea regulilor, a scopului principal și împărțirea lui în subscopuri.

2.4. Sistemul expert de clasificare SEDD

Sistemul expert prototip SEDD este un sistem expert prototip, din categoria sistemelor expert de clasificare, care a fost implementat folosind SWI-Prolog.

SEDD este baza de studiu în această lucrare de laborator, cu ajutorul căruia se scot în evidență atributele unui SE, adică modularitate, nivel înalt de expertiză, comunicarea cu utilizatorul,

respectiv caracteristicile bazei de cunoștințe și a motorului de inferență.

SEDD are capacitatea de a asista utilizatorul în procesul de identificare a defectelor la diferite tipuri de dispozitive electrice. Astfel, datele de intrare sunt reprezentate de factori care fac referire la elementele componente ale dispozitivelor, dar și factori externi. În funcție de valorile datelor de intrare, SEDD va oferi asistență și va da un diagnostic. În continuare este prezentată în detaliu varianta scrisă a SEDD, punându-se accent pe componentele SE și testarea acestuia.

2.4.1. Baza de cunoștințe

Baza de cunoștințe este formată din fapte și reguli. În prima etapă a funcționării SE, faptele sunt reprezentate de datele de intrare, care în cazul SEDD sunt următoarele:

- Temperatura (valoare numerică) – temperatura de funcționare a dispozitivului [°C].
- Vibrația (valoare numerică sau text) – nivelul vibrațiilor [mm/s] sau indice calitativ.
- Zgomotul (valoare numerică sau text) – nivelul zgomotului acustic (dB sau atribut calitativ).
- Nivel_ulei (valoare numerică sau text) – nivelul uleiului transformatorului/generatorului (% sau calitativ).
- Culoare_ulei (valoare text) – culoarea uleiului (de exemplu, clar, închis, lăptos).
- Rezistență_izolație (valoare numerică) – rezistența izolației [MΩ].
- Rezistență_înfășurării (valoare numerică) – rezistența măsurată a înfășurării [Ω].
- Curent_sarcină (valoare numerică) – curentul prin înfășurare sau circuit [A].
- Tensiunea (valoare numerică) – tensiunea de alimentare sau la borne [V].
- Temperatura_ambientală (valoare numerică) – temperatura ambiantă [°C].
- Umiditatea (valoare numerică) – umiditatea mediului [%].
- Descărcare_parțială (valoare text) – nivelul descărcării parțiale (pC sau calitativ).

- Miros (valoare text) – prezența mirosului (de exemplu, niciunul, ardere, ulei).
- Vârsta (valoare numerică) – vârsta echipamentului [ani].
- Ore_funcționare (valoare numerică) – numărul total de ore de funcționare de la instalare.

Aceste date de intrare se pot obține prin măsurători sau observații realizate de către utilizator.

Regulile din baza de cunoștințe folosesc aceste fapte, la care în procesul de implementare se adaugă și alte variabile care ajută la soluționarea problemei. Astfel, au fost definite 30 de reguli, care au fost clasificate în șase clase în funcție de tipul și localizarea defectului. În continuare sunt prezentate 10 dintre ele.

A. Protecție electrică și condiții de alimentare
   R1. DACĂ tensiunea de alimentare > 250 V → ATUNCI deconectați sarcina, alertați operatorul.
   R2. DACĂ tensiunea de alimentare < 210 V → ATUNCI reduceți sarcina, alertați operatorul.
   R3. DACĂ curent > 1,1 * curent nominal → ATUNCI declanșare protecție, înregistrați evenimentul.
   R4. DACĂ vârf de curent > 3 * curent nominal → ATUNCI declanșare întrerupător, înregistrați defecțiune gravă.
   R5. DACĂ orice curent de fază < 10 % din valoarea nominală → ATUNCI opriți motorul, alertați defecțiune pe fază.
   R6. DACĂ |IA – IB| > 15 % din valoarea nominală SAU |IB – IC| > 15 % → ATUNCI alertați dezechilibru, reduceți sarcina.
   R7. DACĂ tensiunea maximă – tensiunea minimă > 10 % din valoarea nominală → ATUNCI reduceți sarcina, alertați operatorul.
   R8. DACĂ curentul_la_masă > 30 mA → ATUNCI declanșare întrerupător, alertați întreținerea.
   R9. DACĂ curentul_h_5 > 10 % SAU curentul_h_7 > 5 % → ATUNCI programați analiza calitatea energiei electrice.
B. Defecțiuni și vibrații mecanice
   R10. DACĂ vibrații_rms > 5 mm/s → ATUNCI programați întreținere, reduceți viteza.

2.4.2. Motorul de inferențe

Motorul de inferență considerat are ca strategie de control, strategia de control înapoi, care este bazată pe scop, fiind strategia de control predefinită a SWI-Prolog. Luându-se în considerare faptul că SEDD este implementat printr-un sistem expert cadru, deci rezultă că motorul de inferență este implicit, folosind metodologia de operare menționată anterior, adică strategia de control înapoi, algoritmul de căutare în adâncime și rezolvarea conflictelor după metoda de avansare în arborele de căutare de la stânga la dreapta.

2.4.3. Modulul explicativ

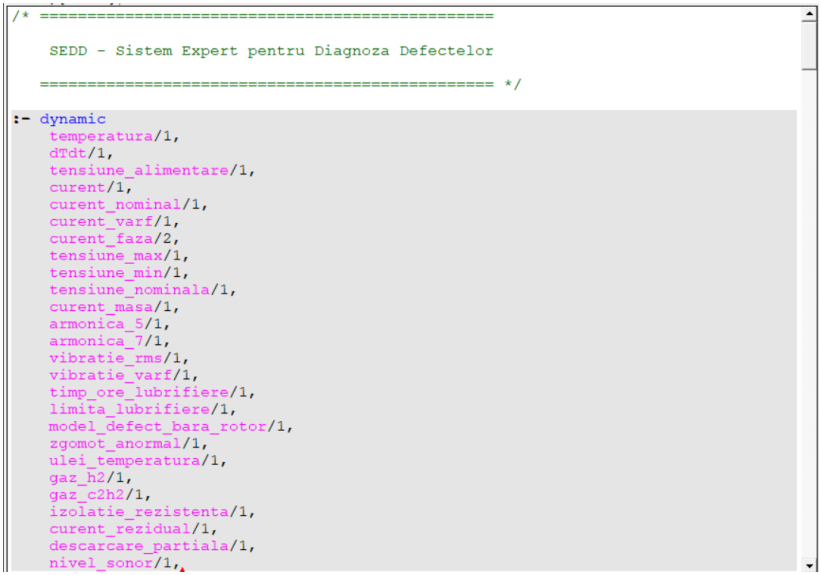
Modulul explicativ al SEDD oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, dar și despre motivul defectului și ce măsuri trebuie luate. Primul tip de informații se poate obține implicit prin analiza pașilor realizați de către SE pentru soluționarea problemei, iar explicațiile suplimentare sunt introduse ca reguli, care se observă în lista regulilor prezentate la punctul anterior.

2.4.4. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SEDD este tip text, având în vedere că SWI-Prolog nu oferă facilități grafice în această direcție, ci doar text. Luând în considerare scopul acestei lucrări de laborator, nu s-a insistat pe acest aspect, și în consecință interacțiunea cu utilizatorul a SEDD este doar text.

2.4.5. Implementarea în SWI-Prolog

SEDD a fost implementat în SWI-Prolog, textul programului informatic fiind salvat într-un fișier ”pl”, denumit ”SEE_L2_SEDD.pl”, care reprezintă materialul de lucru din cadrul orelor aplicative. Figura 2.2. ilustrează o captură din acest fișier, unde sunt definite variabilele cu care lucrează sistemul expert.



```
/* ================================================

    SEDD - Sistem Expert pentru Diagnoza Defectelor

   ================================================ */

:- dynamic
    temperatura/1,
    dTdt/1,
    tensiune_alimentare/1,
    curent/1,
    curent_nominal/1,
    curent_varf/1,
    curent_faza/2,
    tensiune_max/1,
    tensiune_min/1,
    tensiune_nominala/1,
    curent_masa/1,
    armonica_5/1,
    armonica_7/1,
    vibratie_rms/1,
    vibratie_varf/1,
    timp_ore_lubrifiere/1,
    limita_lubrifiere/1,
    model_defect_bara_rotor/1,
    zgomot_anormal/1,
    ulei_temperatura/1,
    gaz_h2/1,
    gaz_c2h2/1,
    izolatie_rezistenta/1,
    curent_rezidual/1,
    descarcare_partiala/1,
    nivel_sonor/1,
```

Figura 2.2. Captură SEDD – declararea variabilelor în BC

Baza de cunoștințe a SEDD cuprinde și baza de reguli cu 30 de reguli de diagnoza a defectelor a dispozitivelor electrice, așa cum se observă în figura 2.3, respectiv în următoarele paragrafe în care se specifică sintaxa pentru patru reguli.

```
verifica_supratensiune(Explicatie) :-
    tensiune_alimentare(V), V > 250,
    format(string(Explicatie), 'Tensiune_alimentare=~w > 250 → Supratensiune: deconecteaza sarcina', [V]).

verifica_subtensiune(Explicatie) :-
    tensiune_alimentare(V), V < 210,
    format(string(Explicatie), 'Tensiune_alimentare=~w < 210 → Subtensiune: reduce sarcina', [V]).

verifica_supracurent(Explicatie) :-
    curent(I), curent_nominal(R), I > 1.1 * R,
    format(string(Explicatie), 'Curent=~w > 1.1*nominal(~w) → Supracurent: declansare protectie', [I,R]).
```

verifica_dezechilibru(Explicatie) :-
(curent_faza(a,A); A = 0), (curent_faza(b,B); B = 0), (curent_faza(c,C); C = 0),
(Abs1 is abs(A-B), Abs2 is abs(B-C), (Abs1 > 0.15 ; Abs2 > 0.15)),
format(string(Explicatie), 'Curentii de faza difera (A=~w B=~w C=~w) → Dezechilibru', [A,B,C]).

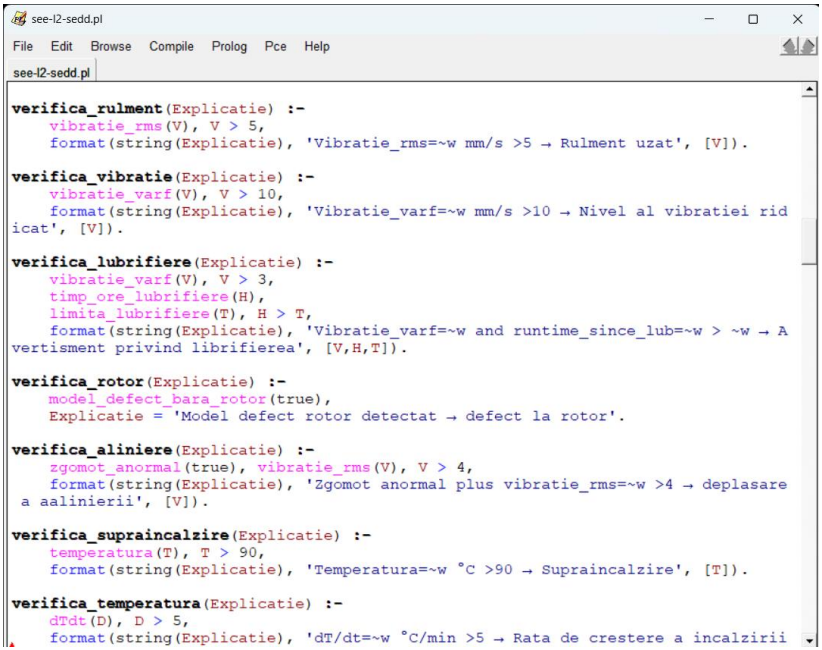


Figura 2.3. Captură SEDD – declararea regulilor în BC

Modulul de interfață cu uitilizatorul a SEDD este de tip text, el fiind introdus sub forma unui predicat (element caracteristic Prolog), cu sintaxa prezentată în continuare:

```
main_menu :-
  repeat,
  nl, writeln('=== Diagnoza defectelor a dispozitivelor ==='),
  writeln('1. Adauga fapt provenit de la senzor'),
```

```
    writeln('2. Stergerea faptelor de la senzori'),
    writeln('3. Realizeaza diagnoza (rezumat)'),
    writeln('4. Afiseaza explicatiile'),
    writeln('5. Exporta raportul intr-un fisier'),
    writeln('6. Afiseaza faptele actuale'),
    writeln('7. Iesire'),
    write('Selecteaza optiunea (1-7): '), flush_output(curent),
    read_line_to_string(user_input, ChoiceStr),
    handle_choice(ChoiceStr),
    ChoiceStr = "7", !.
```

Modulul explicativ oferă utilizatorului posibilitatea să obțină informații despre cauza defectelor, iar prin intermediul funcțiilor de depanare a SWI-Prolog modul în care s-a rezolvat problema.

2.4.6. Testarea SEDD

Testarea SE se realizează prin rularea lui, pentru a i se verifica eficacitatea. Linia de comandă necesară este:

```
?- [SE-L2-SEDD].
```

ceea ce va avea ca rezultat, încărcarea informațiilor din fișier în memoria instrumentului informatic, și deci recunoașterea tuturor regulilor din acesta.

Următorul pas este introducerea faptelor, folosind cuvântul predefinit assert, în felul următor:

```
?- assert(temperatura(96)).
```

Interogarea SE pentru a da o soluție se face prin linia de comandă:

```
?- diagnoza.
```

```
?- assert(temperatura(98)).
true.

?- diagnoza.
Sumar a defectelor detectate:
 - supraincalzire (sever): Opreste motor, notifica operator
true.

?-
```

Figura 2.4. Captură SEDD – testarea sistemului expert

2.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea sistemului expert prezentat anterior prin folosirea fișierului SEE-L2-SEDD.pl oferit în cadrul orei aplicative.
2. Analiza atentă a aplicației pentru identificarea celor trei componente ale SEDD, adică BC, ME și MIU, dar și atributele sistemului expert.
3. Controlul modului de rezolvare a problemei de către SEDD se face prin implicarea comenzii ”trace”, ”debug” sau ”spy”.
4. Depanarea și testarea SEDD implementat prin introducerea setului de date de intrare din figura 2.4.
5. Testarea SEDD prin rularea sistemului folosind datele de intrare din tabelul 2.2.
6. Rezultatele se trec în tabelul 2.2.
7. Se fac modificări asupra SEDD pentru a verifica modularitatea SE.

Tabelul 2.2. Valorile testării sistemului expert SEDD

Date de intrare	**Rezultatele obținute**	**Observații**
Temperatura=100 Tensiunea de alimentare = 190		

2.6. Prelucrarea datelor și interpretarea rezultatelor

Datele din tabelul 2.2. se vor folosi pentru a determina următoarele informații, cu care se va completa coloana de ”Observații” a tabelului:

- Ordinea executării regulilor;
- Numărul de reguli implicate;
- Explicațiile oferite.

2.7. Concluzii

Se vor formula concluzii asupra structurii, funcționării și atributelor SEDD, precum și asupra avantajelor și limitărilor SEDD și a SWI-Prolog.

.....

.....

.....

.....

.....

.....

.....

.....

.....

3

BAZA DE CUNOȘTINȚE A SISTEMELOR EXPERT

3.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta metodele de reprezentare a cunoștințelor folosite în construcția bazei de cunoștințe a sistemelor expert, în vederea înțelegii modalității de implementare a cunoștințelor de expertiză sub o formă accesibilă sistemelor expert dedicate rezolvării problemelor din energetică.

Pentru atingerea scopului va fi implicat ca subiect de studiu, un sistem expert de control dedicat controlului confortului, siguranței, securității și eficienței la un consumator casnic. Sistemul expert a fost denumit SECI (Sistem Expert Casa Inteligentă), care a fost dezvoltat prin folosirea unui instrument informatic dedicat, și anume IDE CLIPS și WebProtégé.

3.2. Obiectivele lucrării

Obiectivele lucrării de laborator sunt definirea metodelor de reprezentare a cunoștințelor în cadrul sistemelor expert, și anume bazate pe reguli de producție, cadre, clase și rețele semantice. Pentru fiecare metodă se vor evidenția avantajele, dezavantajele și caracteristicile de bază. În continuare, se va demonstra modul de funcționare a unui sistem expert particular considerând cele patru metode de reprezentare a cunoștințelor. Pentru a aprofunda aspectele teoretice, se va testa sistemul expert studiat în cele patru variante,

pentru a se scoate în evidență diferențele dintre diferitele variante ale sistemului expert. La final se vor evidenția avantajele și limitările sistemul expert studiat, în urma testării și analizei sistemului expert.

3.3. Noţiuni teoretice

3.3.1. Baza de cunoștințe

Baza de cunoștințe, BC cuprinde informaţiile sub formă de reguli de inferență și de fapte. Totalitatea regulilor compun baza de reguli (BR), iar totalitatea faptelor baza de fapte (BF) [6]. Regulile sunt componentele BC, care se referă la operaţiile care pot fi efectuate asupra datelor (mărimilor) conţinute în BR. În esenţă, regulile constituie un ansamblu complet şi necontradictoriu de cunoștințe necesare rezolvării unei probleme [1]. Faptele sunt componentele BC care conțin un adevăr, o stare a mărimilor din cadrul regulilor, deci ele caracterizează problema la fiecare pas.

Baza de cunoștințe este dezvoltată prin implicarea metodelor de reprezentarea cunoștințelor, prin care se utilizează simboluri recunoscute de calculator pentru a face corespondenţe cu datele caracteristice sistemelor energetice din lumea reală. Simbolurile folosite depind de limbajele de programare implicate, iar corespondenţele realizate au scopul de a oferi posibilitatea realizării unor raţionamente între simbolurile utilizate. Astfel, aceste corespondenţe sunt caracteristici ale mărimilor care sunt concludente pentru problema de rezolvat, şi care le diferenţiază de alte mărimi de acelaşi tip.

Metodele de reprezentare a cunoștințelor care sunt prezentat în această lucrare de laborator sunt regulile de producție, cadrele, clasele + obiectele și rețelele semantice. Astfel, în continuare se vor prezenta succint aceste patru abordări în dezvoltarea bazei de cunoștințe.

Regulile de producție sunt cele mai des folosite pentru reprezentarea cunoștințelor, întrucât s-a demonstrat că mulţi experţi memorează informaţiile și experienţele sub formă de reguli personalizate.

Structura unei reguli de producție este de forma:

Partea de condiţie → Partea de acţiune.

Această sintaxă poate fi interpretată prin expresia:

DACĂ *partea de condiţie* este îndeplinită**,**
ATUNCI *partea de acţiune* se execută**.**

Avantajele aduse de implicarea reprezentării prin reguli de producție sunt datorate de modularitatea proprie fiecărei reguli (orice regulă poate fi privită ca o entitate independentă de celelalte), modularitatea în realizarea formalismului de rezolvare a problemei (toate regulile pot fi văzute ca elemente ale unui ansamblu, care se combină), caracterul natural de exprimare (experţii umani de cele mai multe ori îşi formulează cunoștințele într-o manieră asemănătoare regulilor) și accesabilitatea BR (facilitatea în mentenanţa şi manipularea cunoştinţelor). Contrar, limitările care apar prin utilizarea regulilor de producție sunt în special în cazul unor BC mari, pentru care este dificilă prezicerea unui rezultat și urmărirea raționamentului. Iar în situația în care se impune o anumită ordine prin intermediul regulilor de control, acest lucru va influența procesul decizional.

Rețelele semantice reprezintă o abordare în reprezentare a cunoştinţelor, superioară regulilor de producţie. Ele au apărut ca o consecinţă a modului de surprindere a structurilor relaţionale de mare complexitate dintre datele care caracterizează procesele naturale. Acestea apar sub forma unor graf-uri complexe alcătuite din noduri care sunt legate prin arce, care arată relaţiile dintre noduri. Nodurile sunt folosite pentru reprezentarea de obiecte, concepte, atribute, evenimente şi stări, iar arcele sunt cele care arată relaţiile între noduri.

Integrarea rețelelor semantice în BC a unui sistem expert se poate face printr-o structură de forma:

[Nod sursă], [relație], [Nod destiație].

Punctele forte aduse prin utilizarea rețelelor semantice în reprezentarea cunoștințelor sunt următoarele:

- Reprezentare clară a relațiilor - rețelele semantice conectează entitățile cu care lucrează sistemul expert.
- Suportă raționamentul intuitiv - structura imită modelele cognitive umane pentru înțelegerea sistemelor complexe.

- Interogare și navigare eficiente - nodurile și arcele permit trecerea rapidă de la un concept la cele conexe acestuia.
- Modular și scalabil - pot fi adăugate noduri și arce noi fără a perturba structura existentă.
- Suportă ierarhii și moștenire - nodurile pot fi structurate în clase și subclase.
- Util pentru diagnosticarea defecțiunilor și raționamentul cauzal - relațiile pot fi parcurse înapoi pentru a deduce cauza sau înainte pentru a prezice efectul.
- Se integrează bine cu algoritmii bazați pe grafuri - mulți algoritmi pot opera pe rețele semantice.
- Facilitează reutilizarea și partajarea cunoștințelor - deoarece nodurile reprezintă entități și concepte din lumea reală, rețeaua poate fi reutilizată în diferite SE.
- Bun pentru documentație și instruire - structura vizuală și logică a unei rețele semantice poate fi utilizată pentru documentația educațională și inginerească.

Limitările care apar în folosirea rețelelor semantice derivă din faptul că acestea sunt slabe în reprezentarea cunoștințelor procedurale complexe și a problemelor de scalabilitate mare, fără structurare ierarhică.

Cadrele (frames, cadrele Minsky) sunt o metodă avansată de reprezentare a cunoştinţelor care reunesc metodele de reprezentare procedurale, regulile de producţie și reţelele semantice.

Un cadru cuprinde trei tipuri de componente: obiectul căruia îi este atașat un nume prin intermediul identificatorului, forma (atributul) – arată o categorie de caracteristici și faţeta care descrie obiectul cu ajutorul unor perechi de simbol – valoare. Sintaxa de bază a unui cadru este următoarea:

<cadru> = <identificator_cadru>
<descriere_formă> = <identificator_formă>
<descriere_faţetă> = <identificator_faţetă>
<descriere_dată> = <valoare>.

Avantajele folosirii cadrelor în reprezentarea cunoștințelor sunt următoarele prezentate în figura 3.1.

Figura 3.1. Avantajele utilizării cadrelor în BC

Punctele slabe ale cadrelor sunt expresivitate limitată pentru logica complexă, modelare mai dificilă a cunoștințelor neorientate pe obiecte, lipsa unui mecanism standard de inferență, probleme de scalabilitate în sistemele mari, conflicte de moștenire și mai puțin transparentă decât sistemele bazate pe reguli.

Clasele și obiectele sunt cea mai avansată metodă de reprezentare a cunoștințelor (dintre cele patru metode prezentate în această lucrare). Obiectul este considerat ca fiind elementul de bază, care este caracterizat prin toate atributele şi funcţiile (metodele) care acţionează sau au legătură cu el. În domeniul programării, obiectele sunt incluse în clase (classes) şi poartă numele de instanţe ale acestora. Sintaxa de bază a unei clase poate fi scrisă sub forma:

clasa identificator clasa
identificator atribut 1 caracteristică atribut 1
identificator atribut n caracteristică atribut n
identificator metoda 1 acțiune metoda 1
identificator metoda n acțiune metoda n.

Caracteristicile claselor și obiectelor sunt modularitatea, reutilizabilitatea, scalabilitatea, organizare naturală a datelor, încapsularea și comportamentul dinamic, care sunt punctele forte ale metodei de reprezentare a cunoștințelor bazată pe ele.

Comparativ, dezavantajele acestei metode de reprezentare a cunoștințelor rezultă din complexitatea în implementare, necesitatea utilizării intensivă a resurselor, integrarea mai dificilă a regulilor și mărimea suprafeței de depanare ceea ce fac dificilă urmărirea logicii prin obiecte care interacționează și moștenire poate fi dificilă.

3.3.2. WebProtégé

WebProtégé este un editor de ontologii care este rulat pe o platformă informatică dedicată, dar are la bază instrumentul Protégé. La bază, acest instrument informatic dedicat reprezentării cunoștințelor are Java, și poate funcționa pe diferite sisteme de operare.

WebProtégé permite crearea, editarea și vizualizarea ontologiilor (OWL – Web Ontology Language, RDSs – Relational Database System și alte formate de ontologii). Rețelele semantice fiind variante generalizate ale ontologiilor, acestea pot fi implimentate cu ajutorul Protégé. Interacțiunea cu utilizatorii a WebProtégé se face printr-o interfață grafică dedicată (figura 3.2), care ajută în modelarea conceptelor, claselor, proprietăților și instanțelor.

Acest intrument informatic oferă integrare cu raționament (cum ar fi HermiT, Pellet, Fact++) pentru a verifica consistența logică și este un suport puternic pentru reprezentarea cunoștințelor, inclusiv cadre, clase, relații ierarhice și constrângeri.

Punctele forte ale WebProtégé sunt gratuitatea (este o aplicație open-source), maturitatea, utilizarea pe scară largă în mediul academic și industrial pentru ingineria ontologiilor, flexibilitatea, suportul bun pentru raționament și verificarea consistenței.

Dezavantajele rezultă din faptul că WebProtégé este limitat ca memorie de serverul unde este rulat, precum și de conexiunea și traficul Web. De asemenea, pentru începători, interfața grafică poate fi dificil de lucrat cu, în special în cazul rețelelor de dimensiuni mari. În plus, fiind dedicată reprezentării ontologiilor, pot apărea dificultăți în utilizarea unor metode non-ontologice.

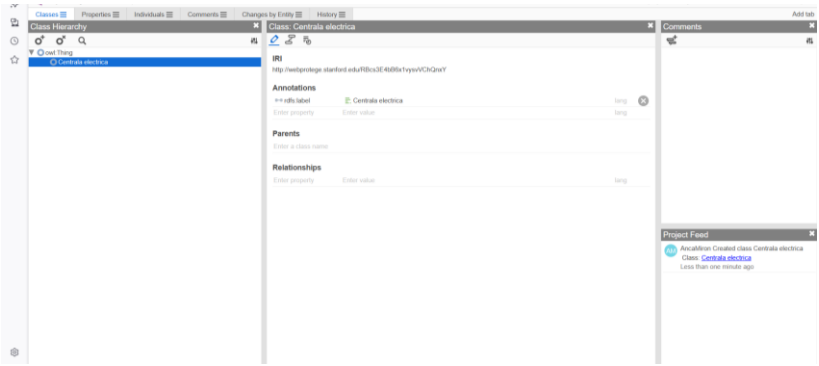


Figura 3.2. Captură – interfața cu utilizatorul a WebProtégé [7]

Protégé / WebProtégé acceptă definirea rețelelor semantice, dar trebuie să se evite sau să se simplifice constrângerile OWL, astfel acest instrument se va utiliza ca un editor grafic de concepte-relații.

Folosirea WebProtégé (sau Protégé) în modelarea unei rețele semantice presupune realizarea următorilor pași:

P1. Crearea conceptelor prin accesarea filei Classes din meniu. Astfel, se vor crea clase precum ”Centrala electrică” sau ”Generator”. Acestea vor reprezenta conceptele, adică nodurile rețelei.

P2. Definirea relațiilor prin accesarea filei Properties / Object Properties. Acestea vor fi tipurile de arce, ele fiind de forma ”produce” sau ”transformă”.

P3. Conectarea conceptelor se realizează prin atribuirea unui domeniu și a unui interval pentru fiecare proprietate. Prin această acțiune se definește o conexiune direcțională, precum într-o rețea semnatică.
Exemplu: produce
Domeniu: Generator
Interval: Electricitate

P4. Adăugarea unei instanțe specifice (etapă opțională) în situația în care se dorește reprezentarea unor elemente particulare. Pentru realizarea acestei etape se accesează fila Individuals, unde se adaugă elementele precum arată exemplul:
Generator_1
Turbină_1

Electricitate_Ieșire
La final se corelează conceptele:
drives(Turbină_1, Generator_1)
produce(Generator_1, Electricitate_Ieșire)

P5. Vizualizarea rețelei prin utilizarea plugin-ului ”OntoGraf”, care este inclus în Protégé. În consecință, se va afișa o rețea grafică cu noduri și arce. Pentru doar vizualizarea rețelei, se vor dezactiva funcțiile de raționament.

Protégé stochează fiecare concept ca un nod într-un graf RDF, astfel fiecare relație este un triplet de forma:

Subiect – Predicat – Obiect.

Când instrumentul informatic funcționează (fără logica ontologiei, deoarece a fost dezactivată) motorul RDF gestionează structura grafului, dar nu utilizează raționament DL. Deci, Protégé devine practic un editor și vizualizator de rețele semantice, susținut de un model RDF. Pentru a folosi mai departe această rețea semantică în rezolvarea problemei de către SE, ei trebuie să i se atașeze o bază de reguli, prin care se introduce logica în BC. Acest lucru se face prin două metode: (1) salvarea rețelei într-un fișier, și apoi folosit de către o aplicație dezvoltată în CLIPS, sau (2) definirea unor reguli prin Protégé, dar trebuie folosit plugin-ul SWRL (Semantic Web Rule Language).

3.4. Sistemul expert de control SECI

Sistemul expert SECI este un sistem expert prototip, dezvoltat în scop didactic, din categoria sistemelor de control, care a fost implementat în patru variante folosind IDE CLIPS și WebProtégé.

SECI este subiectul de studiu al acestei lucrări de laborator, el fiind folosit pentru a se studia modul de construcție și funcționare a unui SE prin implicarea celor patru metode de reprezentare a cunoștințelor, adică reguli de producție, cadre, rețele semantice și clase+obiecte. În consecință se vor dezvolta patru variante a sistemului SECI.

Acest SE poate fi folosit în automatizarea completă a procesului de încălzire a unei case, a asigurării securității, siguranței și creșterea

eficienței energetice. Întrucât SECI are capacitatea în funcție de datele de intrare primite de la senzori să controleze confortul, securitatea, siguranța și eficiența energetică a unei locuințe. În continuare este prezentată în detaliu varianta scrisă a SECI, punându-se accent pe componentele SE și testarea acestuia.

3.4.1. Baza de cunoștințe

Baza de cunoștințe cuprinde BR și BF. Baza de reguli este compusă din 24 de reguli, care se referă la controlul temperaturii, a luminozității, și a temperaturii, dar și la măsuri de securitate și siguranță, și la creșterea eficienței energetice. În continuare sunt enumerate primele 10 reguli, care se referă la cunoștințe legate de realizarea confortului, securitate și siguranță.

R1. DACĂ temperatura < 20 °C ȘI ocupare = Adevărat ȘI ora_zilei în 06:00-09:00, ATUNCI porniți încălzirea, valoare setată = 21,5 °C.
R2. DACĂ temperatura < 20 °C ȘI ocupare = Adevărat ȘI ora_zilei în 18:00-22:00, ATUNCI porniți încălzirea, valoare setată = 19 °C.
R3. DACĂ ocupare = Fals ȘI ora_zilei în 23:00-06:00, ATUNCI porniți încălzirea în ECO_MODE 18 °C, luminile stinse.
R4. DACĂ iluminare < 150 lx ȘI ocupare = Adevărat, ATUNCI porniți luminile, luminozitate redusă = 80%.
R5. DACĂ stare_ușă = deschisă ȘI ocupare = Fals, ATUNCI activați alarma, notificați utilizatorul.
R6. DACĂ contor_energie > 3000 Wh ȘI receptoare critice = Fals, ATUNCI opriți sarcinile necritice (încălzitor de apă, încărcător auto electric).
R7. DACĂ temperatură < 5 °C ȘI fereastră_deschisă = Adevărat ȘI încălzitor pornit, ATUNCI notificați utilizatorul: risipă de energie.
R8. DACĂ temperatură > 26 °C ȘI ocupare = Adevărat, ATUNCI porniți răcirea, valoare setată = 24 °C.
R9. DACĂ nivel_CO2 > 1000 ppm ȘI ocupare = Adevărat, ATUNCI porniți ventilația.
R10. DACĂ umiditate < 30% ȘI ocupare = Adevărat, ATUNCI pornire umidificator.

Baza de fapte cuprinde informații despre temperatură, ocupare, oră, iluminare, stare ușă, contor de energie, sarcini critice, fereastră deschisă, încălzitor, nivel de CO2 și umiditate.

Baza de cunoștințe va fi implementată în IDE CLIPS, variantele cu reguli de producție, cadre și obiecte, respectiv în WebProtege în varianta cu rețele semantice, care apoi este importantă în CLIPS.

3.4.2. Motorul de inferențe

Motorul de inferență a SE are ca strategie de control, strategia de control înainte. Luându-se în considerare faptul că SECI este implementat în CLIPS IDE, atunci rezultă că MI este implicit, folosind metodologia de operare prezentată anterior, adică strategia de control înainte, algoritmul Rete și rezolvarea conflictelor după indicatorul de prioritate, numărul de fapte și ordinea din baza de reguli.

3.4.3. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SECI este interfața oferită de CLIPS IDE, care conține doar elemente de tip text. Deci, utilizatorul va comunica cu SE prin mesaje text, iar acesta va afișa rezultatul la fel, sub formă de text (cuvinte, numere, simboluri). De altfel, scopul acestei lucrări de laborator este BC, deci nu s-a insistat pe acest modul al SE.

3.4.4. Modulul explicativ

Modulul explicativ al SECI oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, adică regulile executate. Acest aspect se poate obține implicit prin evaluarea treptată a pașilor realizați de aplicație până la soluționarea problemei. Precum s-a procedat și în cazul motorului de inferență, nu s-a insistat pe acest aspect, ci pe dezvoltarea BC în cele patru variante.

3.4.5. Implementarea SECI

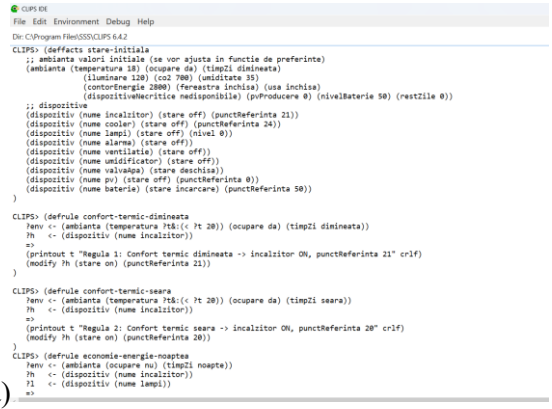
SECI a fost implemntat în CLIPS IDE în trei versiuni (menționate anterior) și WebProtege + CLIPS IDE o versiune. Figura 3.3. ilustrează capturi cu BC reguli de producție (a), cadre (b) clase și

obiecte (c), și rețeaua semantică corespunzătoare dezvoltată în WebProtege (d).

Liniile de comandă implementate în CLIPS IDE au fost salvate sub forma unui fișier text, care reprezintă materialul de lucru din cadrul orelor aplicative. În aceast material didactic sunt prezentate doar o parte din componentele SECI implementate.

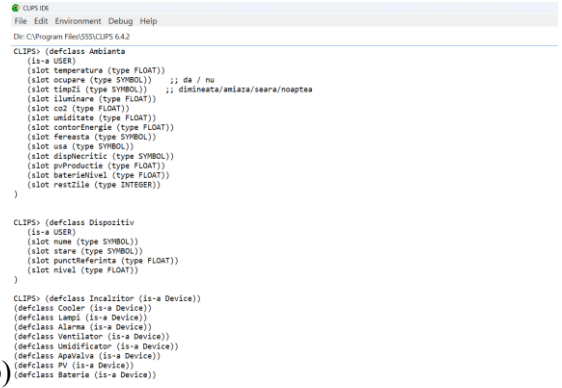
Regulile de inferență sunt prezentat în paragraful următor. Pentru a se observa diferența dintre cele cele trei metode de reprezentare a cunoștințelor, se prezintă aceeași regulă în cele trei variante.

(a)



```
CLIPS IDE
File  Edit  Environment  Debug  Help

Dir: C:\Program Files\SSS\CLIPS 6.4.2

CLIPS> (deffacts stare-initiala
   ;; ambianta valori initiale (se vor ajusta in functie de preferinte)
   (ambianta (temperatura 18) (ocupare da) (timpZi dimineata)
             (iluminare 120) (co2 700) (umiditate 35)
             (contorEnergie 2800) (fereastra inchisa) (usa inchisa)
             (dispozitiveNecritice nedisponibile) (pvProducere 0) (nivelBaterie 50) (restZile 0))
   ;; dispozitive
   (dispozitiv (nume incalzitor) (stare off) (punctReferinta 21))
   (dispozitiv (nume cooler) (stare off) (punctReferinta 24))
   (dispozitiv (nume lampi) (stare off) (nivel 0))
   (dispozitiv (nume alarma) (stare off))
   (dispozitiv (nume ventilatie) (stare off))
   (dispozitiv (nume umidificator) (stare off))
   (dispozitiv (nume valvaApa) (stare deschisa))
   (dispozitiv (nume pv) (stare off) (punctReferinta 0))
   (dispozitiv (nume baterie) (stare incarcare) (punctReferinta 50))
)

CLIPS> (defrule confort-termic-dimineata
   ?env <- (ambianta (temperatura ?t&:(< ?t 20)) (ocupare da) (timpZi dimineata))
   ?h   <- (dispozitiv (nume incalzitor))
   =>
   (printout t "Regula 1: Confort termic dimineata -> incalzitor ON, punctReferinta 21" crlf)
   (modify ?h (stare on) (punctReferinta 21))
)

CLIPS> (defrule confort-termic-seara
   ?env <- (ambianta (temperatura ?t&:(< ?t 20)) (ocupare da) (timpZi seara))
   ?h   <- (dispozitiv (nume incalzitor))
   =>
   (printout t "Regula 2: Confort termic seara -> incalzitor ON, punctReferinta 20" crlf)
   (modify ?h (stare on) (punctReferinta 20))
)
CLIPS> (defrule economie-energie-noaptea
   ?env <- (ambianta (ocupare nu) (timpZi noapte))
   ?h   <- (dispozitiv (nume incalzitor))
   ?l   <- (dispozitiv (nume lampi))
   =>
```

(b)



```
CLIPS IDE
File  Edit  Environment  Debug  Help

Dir: C:\Program Files\SSS\CLIPS 6.4.2

CLIPS> (defclass Ambianta
   (is-a USER)
   (slot temperatura (type FLOAT))
   (slot ocupare (type SYMBOL))    ;; da / nu
   (slot timpZi (type SYMBOL))     ;; dimineata/amiaza/seara/noaptea
   (slot iluminare (type FLOAT))
   (slot co2 (type FLOAT))
   (slot umiditate (type FLOAT))
   (slot contorEnergie (type FLOAT))
   (slot fereasta (type SYMBOL))
   (slot usa (type SYMBOL))
   (slot dispNecritic (type SYMBOL))
   (slot pvProductie (type FLOAT))
   (slot baterieNivel (type FLOAT))
   (slot restZile (type INTEGER))
)


CLIPS> (defclass Dispozitiv
   (is-a USER)
   (slot nume (type SYMBOL))
   (slot stare (type SYMBOL))
   (slot punctReferinta (type FLOAT))
   (slot nivel (type FLOAT))
)

CLIPS> (defclass Incalzitor (is-a Device))
(defclass Cooler (is-a Device))
(defclass Lampi (is-a Device))
(defclass Alarma (is-a Device))
(defclass Ventilator (is-a Device))
(defclass Umidificator (is-a Device))
(defclass ApaValva (is-a Device))
(defclass PV (is-a Device))
(defclass Baterie (is-a Device))
```

Figura 3.3. Captură – BC a SECI

(c)

```
CLIPS IDE
File  Edit  Environment  Debug  Help
Dir: C:\Program Files\SSS\CLIPS 6.4.2
CLIPS> (defclass Ambianta
   (is-a USER)
   (slot temperatura (type FLOAT))
   (slot ocupare (type SYMBOL))
   (slot timpZi (type SYMBOL))
   (slot iluminare (type FLOAT))
   (slot co2 (type FLOAT))
   (slot umiditate (type FLOAT))
   (slot contorEnergie (type FLOAT))
   (slot fereasta (type SYMBOL))
   (slot usa (type SYMBOL))
   (slot dispozitiveNecritice (type SYMBOL))
   (slot pvProductie (type FLOAT))
   (slot baterieNivel (type FLOAT))
   (slot restZile (type INTEGER))
)

CLIPS> (defclass Dispozitiv
   (is-a USER)
   (slot nume (type SYMBOL))
   (slot stare (type SYMBOL))
   (slot punctReferinta (type FLOAT))
   (slot nivel (type FLOAT))
   (message-handler turn-on (is-public)
      (printout t ?self:nume " -> turned ON" crlf)
      (bind ?self:state on))
   (message-handler turn-off (is-public)
      (printout t ?self:nume " -> turned OFF" crlf)
      (bind ?self:state off))
   (message-handler set-eco (is-public) (?sp)
      (printout t ?self:nume " -> set ECO mode " ?sp crlf)
      (bind ?self:stare eco)
      (bind ?self:punctReferinta ?sp))
   (message-handler set-punctReferinta (is-public) (?sp)
      (printout t ?self:nume " -> setpoint " ?sp crlf)
      (bind ?self:punctReferinta ?sp))
   (message-handler set-nuvel (is-public) (?d)
      (printout t ?self:nume " -> dim " ?d crlf)
```

(d)

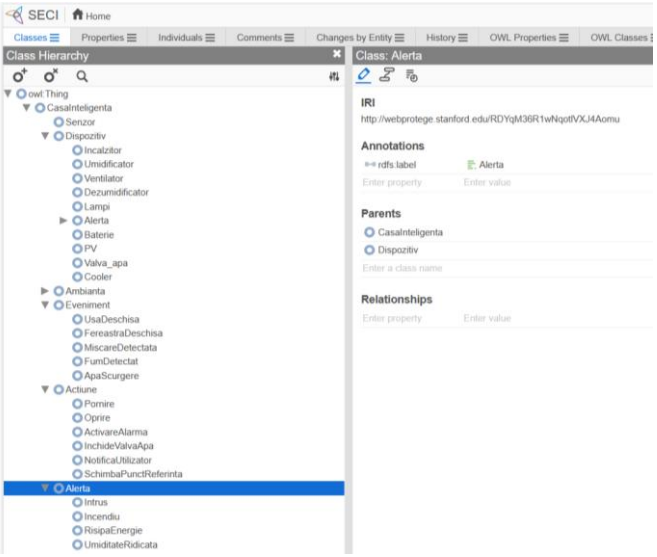


Figura 3.3. Captură – BC a SECI

Sintaxa CLIPS pentru regula 1:

(defrule confort-termic-dimineata

?env <- (ambianta (temperatura ?t&:(< ?t 20)) (ocupare da) (timpZi dimineata))

?h <- (dispozitiv (nume incalzitor))

```
    =>
    (printout t "Regula 1: Confort termic dimineata -> incalzitor
ON, punctReferinta 21" crlf)
    (modify ?h (stare on) (punctReferinta 21))
  )

  (defrule incalzire-dimineata
    (object (is-a Ambianta) (temperatura ?t&:(< ?t 20)) (ocupare da)
(timpZi dimineata))
    =>
    (printout t "Incalzitor ON (dimineata), punct de referinta 21"
crlf)
    (set-slot-value incalzitor1 stare on)
    (set-slot-value incalzitor1 punctReferinta 21)
  )

  (defrule incalzire-dimineata
    (object (is-a Ambianta) (temperatura ?t&:(< ?t 20)) (ocupare
nu) (timpZi dimineata))
    =>
    (printout t "Confort termic dimineata -> Se pornește încălzitorul
" crlf)
    (send heater set-puntReferinta 21)
    (send heater turn-on)
  )
```

3.4.6. Testarea SECI

Testarea SE se realizează prin rularea lui, pentru a i se verifica eficacitatea. De asemenea, se poate urmări procesul de soluționare prin selectarea din ”Debug” a ”Agenda”, ”Facts” și ”Instances”, respectiv ”Watch” – All.

În figura 3.6. este ilustrată testarea SECI realizată prin rularea aplicației cu linia de comandă (run), pentru varianta cu reguli de producție și faptele definite anterior.

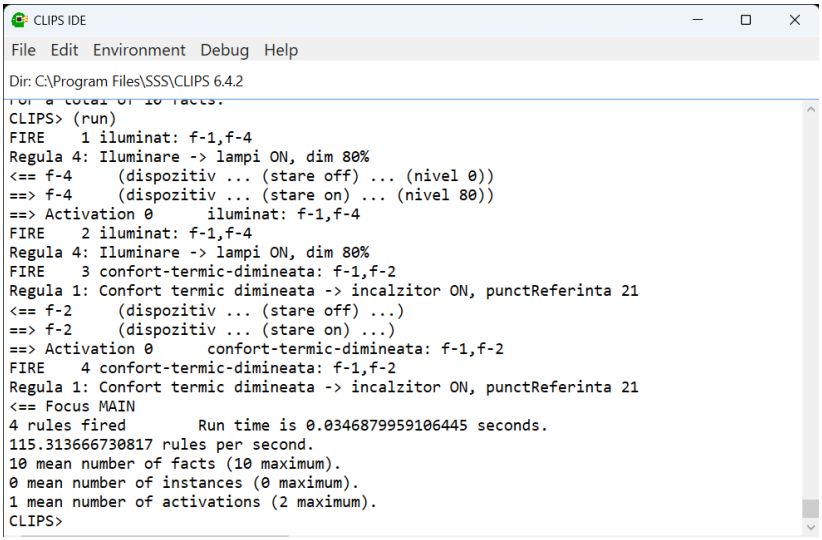


Figura 3.4. Captură SECI – testarea sistemului expert

3.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea celor trei variante a SECI, prin folosirea fișierelor SEE-L3-SECI-r.cpl, SEE-L3-SECI-f.cpl și SEE-L3-SECI-c.cpl, în CLIPS IDE.
2. Construirea rețelei semantice corespunzătoare SECI în WebProtege și apoi implementarea ei în CLIPS IDE și rularea variantei 4 a SECI.
3. Activarea opțiunilor în CLIPS IDE necesare ilustrării etapelor realizate de aplicație în rezolvarea problemei, conform celor menționate anterior (vezi 3.4.6).
4. Analiza atentă a aplicației pentru identificarea celor trei componente ale SECI, adică BC, ME și MIU. Se va pune accent pentru identificarea diferențelor în construcția BC în cele patru variante ale SECI.
5. Depanarea și testarea versiunilor SECI implementate prin folosirea faptelor definite în cadrul SE.

6. Testarea versiunilor SECI prin rularea SE folosind datele de intrare predefinite în SE la care se vor adăuga altele particulare preferințelor utilizatorului.
7. Rezultatele se trec în tabelul 3.1 astfel – pentru fiecare set de date se vor specifica rezultatele pentru toate versiunile SECI.

Tabelul 3.1. Valorile testării sistemului expert SECI

Date de intrare	**Rezultatele obținute**	**Observații**

3.6. Interpretarea rezultatelor

Datele din tabelul 3.1. se vor folosi pentru a concluziona următoarele informații, cu care se va completa coloana de ”Observații” a tabelului:

- Ordinea executării regulilor;
- Numărul de reguli implicate;
- Timpul de execuție.

3.7. Concluzii

Se vor formula concluzii asupra structurii BC și construirea ei în funcție de metoda de reprezentare a cunoștințelor. De asemenea, se vor evidenția avantajele și limitările celor patru metode de reprezentare a cunoștințelor folosite pentru dezvoltarea SECI în cele patru versiuni ale sale.

.....

.....

.....

.....

.....

.....

.....

.....

.....

4

MOTORUL DE INFERENȚE
STRATEGIA DE CONTROL ÎNAINTE

4.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta metodologia de funcționare a motorul de inferențe a sistemelor expert, în vederea înțelegii modalității de implementare și operare a acestui algoritm rezolutiv în vederea rezolvării unor probleme specifice ingineriei energetice.

Pentru atingerea acestui scop va fi folosit ca subiect de studiu, un sistem expert de clasificare și acționare dedicat localizării și izolării defectelor în sistemele de transport și distribuție a energiei electrice. Sistemul expert a fost denumit SELID (Sistem Expert pentru Localizarea și Izolarea Defectelor), care a fost dezvoltat prin folosirea limbajului de programare Python.

4.2. Obiectivele lucrării

Obiectivele lucrării de laborator sunt definirea și analiza metodologiei de funționare a motorului de inferențe. Aceasta presupune prezentarea strategiei de control înainte, a algoritmilor de căutare și a metodelor de rezolvare a conflictelor din cadrul motorului de inferențe a sistemelor expert. În continuare, se va demonstra modul de funcționarea unui sistem expert particular considerând strategia de control înainte, trei metode de rezolvare a conflictelor și doi algoritmi de căutare. Pentru a aprofunda aspectele

teoretice, se va testa sistemul expert studiat în cele șase variante, pentru a se scoate în evidență diferențele dintre diferitele versiuni ale sistemului expert. La final se vor evidenția avantajele și limitările sistemul expert studiat, în urma testării și analizei funcționării versiunilor sistemului expert studiat (doi algoritmi de căutare a câte trei metode de rezolvare a conflictelor).

4.3. Noţiuni teoretice

4.3.1. Motorul de Inferențe

Motorul de Inferenţe, MI este cel care dă funcţionalitate unui SE, și se bazează pe inferenţe. Acesta prelucrează cunoștințele din baza de reguli și fapte folosind diferite procedee, care se referă la o strategie de control, algoritmi de căutare și metode de rezolvare a conflictelor. Strategia de control se referă la un set de norme care ghidează căutarea regulilor, care pot fi aplicabile pentru rezolvarea unei probleme. Un sistem expert poate avea o singură strategie de căutare. Algoritmii de căutare ai unui MI cuprind totalitatea operatorilor și regulilor necesare pentru a căuta în spaţiul bazei de reguli în vederea găsirii acelor reguli care sunt în concordanţă cu starea problemei la un moment dat. Iar metodele de rezolvare a conflictelor au rolul de a alege regula care se va executa dintre regulile selectate, întrucât MI poate executa o singură regulă la un moment dat. MI poate cuprinde mai mulți algoritmi de căutare și mai multe metode de rezolvare a conflictelor. În această lucrare se va studia strategia de control înainte, căutarea exhausivă, căutarea bazată pe agendă, și trei metode de rezolvare a conflictelor: prima regulă aplicabilă, regula cea mai specializată și regula cea mai productivă.

Strategia de control înainte

Această strategie de control foloseşte o metodologie de parcurgere a datelor de la fapte înspre obiectiv, din această cauză este aplicabilă în cazul problemelor la care se cunosc foarte bine datele de intrare. Astfel, în momentul parcurgerii bazei de fapte, și în etapa de selectare a regulilor aplicabile, vor fi alese acele reguli pentru care faptele din enunţ se află în corpul premiselor regulilor. Figura 4.1. ilustrează schema logică de bază a strategiei de control înainte.

Algoritmul de căutare – ”căutarea exhaustivă”

Prin utilizarea acestui algoritm de căutare, baza de reguli va fi parcursă în totalitate, regulă cu regulă. Avantajul acestui algoritm de căutare este simplitatea, dar în defavoarea eficienței.

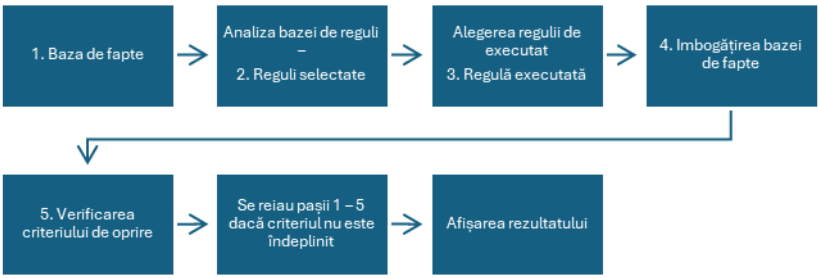


Figura 4.1. Schema logică a strategiei de control înainte

Pseudocodul algoritmului strategiei de control înainte, în cazul în care se folosește algoritmul de căutare exhaustivă este prezentat în rândurile următoare:

```
1. Inițializare:
    Baza_fapte ← fapte inițiale
    Baza_reguli ← mulțimea tuturor regulilor
    Reguli_executate ← ∅
    Modificări ← Adevărat
2. While Modificări = Adevărat:
     Modificări ← Fals
      Pentru fiecare regulă R din Baza_reguli:
       If (R nu este în Reguli_executate):
         Evaluează partea IF (antecedente):
          Dacă toate condițiile din R sunt îndeplinite de Baza_fapte:
             Execută regula R:
              Adaugă concluzia(ile) lui R la Baza_fapte
              Înregistrează R în Reguli_Executate
              Modificări ← Adevărat
    Sfârșit While.
3. Dacă este specificat Scop:
    Dacă Scop ∈ Baza_fapte:
     Returnează "Obiectiv atins", Reguli_executate, Baza_fapte
      Else:
       Returnează "Obiectiv neatins", Reguli_executate, Baza_fapte
      Else:
```

Returnează "Inferență completă", Reguli_executate, Baza_fapte.

Algoritmul de căutare – ”căutarea bazată pe agendă ” presupune folosirea unor seturi de conflicte pentru a crește eficiența motorului de inferențe. Mai exact, algoritmul folosește o agendă care este actualizată dinamic atunci când faptele se schimbă. Astfel, în loc să verifice toate regulile de fiecare dată, reevaluează doar regulile afectate de faptele nou adăugate. Pseudocodul acestui algoritm este următorul:

```
1. Inițializare:
    Baza_fapte ← fapte inițiale
    Baza_reguli ← toate regulile
    Reguli_executate ← ∅
    Urmă_baza_reguli ← ∅
2. Repetare:
     Pasul 1: Se construiește setul de conflicte Set_conflict ← ∅
       Pentru fiecare regulă R din Baza_reguli:
       Dacă (R nu se află în Reguli_executate) ȘI (toate R.condiție ∈
Baza_fapte):
          Adăugați R la Set_conflict
     Pasul 2: Verificați dacă există vreo regulă aplicabilă
       Dacă Set_conflict este gol:
  Ieșiți din buclă # Nu mai există reguli aplicabile
Pasul 3: Rezolvarea conflictelor
     Reguli_Selectate ← regulă din Set_conflict cu cea mai mare prioritate
conform Criteriu_prioritate
Pasul 4: Executarea regulii selectate
     Adăugați Reguli_selectate.acțiune la Baza_fapte
     Adăugați Reguli_selectate la Reguli_executate
     Adăugați Reguli_selectate la urmă_bază_reguli
  Dacă Scop ∈ Baza_fapte:
    Ieșire din buclă
3. Până când Set_conflict = ∅
4. Returnează:
Urmă_baza_reguli, Baza_fapte
```

Metoda de rezolvare a conflictelor – ”prima regulă aplicabilă”

Rezolvarea conflictelor prin metoda menționată presupune ca dintre regulile selectate, se va alege spre execuție (declanșare) prima regulă scrisă în baza de reguli, adică se folosește ordinea din baza de

reguli. Dacă algoritmul decizional se construiește sub forma unei funcții, pseudocodul este următorul:

```
Funcția Prima_Regulă(Set_conflict, Baza_reguli):
  Pentru fiecare regulă R din Baza_reguli:
    Dacă R ∈ Set_conflict:
      returnează R
```

Metoda de rezolvare a conflictelor – ”regula cea mai productivă”

Rezolvarea conflictelor prin metoda menționată presupune ca dintre regulile selectate, se va alege spre execuție (declanșare) regula care are cele mai multe acțiuni, adică elemente după ”Atunci”. Dacă algoritmul decizional se construiește sub forma unei funcții, pseudocodul este următorul:

```
Funcția Regula_Productivă(Set_conflict):
   Nr_maxim_acțiuni ← 0
   Regulă_selectată ← Nici una
   Pentru fiecare regulă R din Set_conflict:
      Număr_concluzii ← numără(R.acțiuni)
      Dacă Număr_concluzii > Nr_maxim_acțiuni:
         Nr_maxim_acțiuni ← Număr_acțiunii
         Regulă_selectată ← R
   returnează Regulă_selectată
```

Metoda de rezolvare a conflictelor – ”regula cea mai specializată”

Rezolvarea conflictelor prin metoda menționată presupune ca dintre regulile selectate, se va alege spre execuție (declanșare) regula care are cele mai multe condiții, adică după ”Dacă”. În cazul în care algoritmul decizional se construiește sub forma unei funcții, pseudocodul este următorul:

```
Funcția Regula_specializată(Set_conflict):
   Număr_max_condiții ← 0
   Regula_selectată ← Nici una
   Pentru fiecare regulă R din Set_conflict:
      Nr_condiții ← numărăr(R.premise)
      Dacă Nr_condiții > Număr_max_condiții:
         Număr_max_condiții ← Nr_condiții
         Regula_selectată ← R
   returnează Regula_selectată
```

4.3.2. Python

Python este un limbaj de programare de nivel înalt, interpretat, de uz general, creat de Guido van Rossum și lansat pentru prima dată în 1991. Acesta a fost dezvoltat cu scopul de a îndeplinii următoarele condiții – lizibilitate ridicată (sintaxa sa arată ca engleza naturală), simplitate (mai puține linii de cod în comparație cu C/C++ sau Java), și flexibilitate (utilizată în multe domenii: știința datelor, inteligență artificială, dezvoltare web, automatizare etc.). Principalele caracteristici ale acestui limbaj de programare sunt evidențiate în figura 4.2.

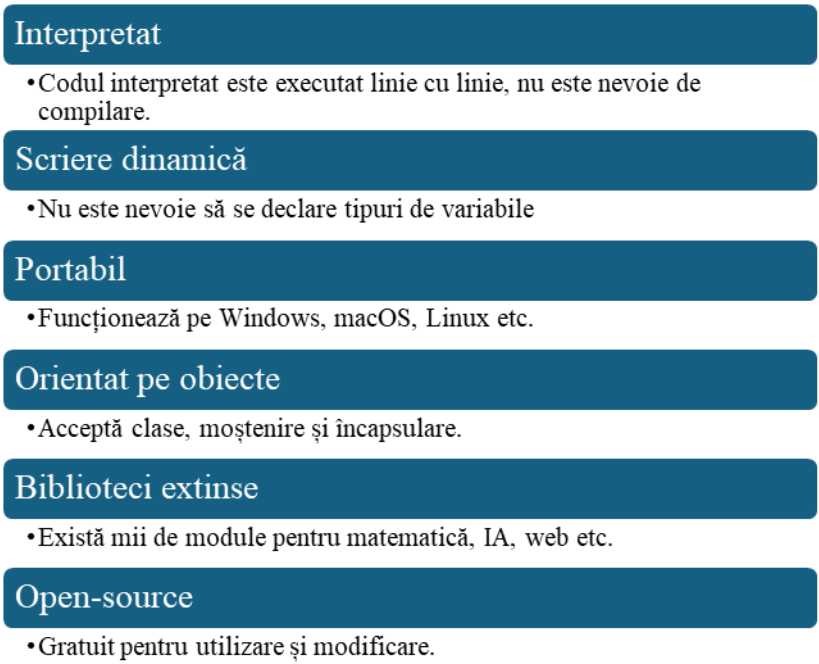


Figura 4.2. Caracteristicile Python

Python este foarte popular deoarece este ușor de citit și de scris, are un suport comunitar imens, poate funcționa și interacționa cu alte limbaje (C/C++, Fortran, Java), și este excelent pentru dezvoltarea de prototipuri și folosirea lui în cercetare.

Cuvintele predefinite în Python sunt prezentate și explicate în tabelul 4.1.

Tabelul 4.1. Cuvinte cheie în Python

Cuvânt cheie	**Explicație**
If	Testare condiție, dacă
elseif	Condiția else if, altfel dacă
Else	Altfel, ramura implicită a condiției
For	Ciclu repetitiv pe o secvență de date
while	Ciclu repetitiv atâta timp cât condiția este adevărată
break	Ieșire din ciclu repetitiv
continue	Trece peste acțiunile din ciclul repetitiv și continuă restul algoritmului
match, case	Potrivirea modelelor
Def	Definirea unei funcții
class	Definirea unei clase
return	Folosit pentru returnarea unei valori a funcției
And	ȘI logic
Or	SAU logic
True	Valoarea de adevărat boolean
False	Valoarea de fals boolean
none	Nul / nici o valoare
Not	NU logic
global	Declararea variabilelor globale
nonlocal	Declararea variabilelor care se vor folosi în scop închis
Del	Ștergerea unei variabile
assert	Verificarea unei condiții
class	Definirea unei clase
Self	Referirea la obiectul curent
Is	Testarea identității unui obiect
==, +, //, **	Operatori simbolici
import	Importarea unui modul
As	Redenumirea unui obiect sau modul importat

4.4. Sistemul expert de clasificare SELID

Sistemul expert SELID este un sistem expert prototip, care a fost construit în scop didactic. Acesta face parte din categoria sistemelor expert de clasificare și control, care a fost implementat în șase variante folosind Python.

SELID este subiectul de studiu al acestei lucrări de laborator, rolul lui didactic este de a ajuta în prezentarea modului de funcționare și construcție a motorului de inferență. Astfel, sunt implementate șase variante ale SELID, care se caracterizează prin următoarele:

- Toate variantele au aceeași bază de cunoștințe;
- Motorul de inferențe este diferit pentru cele șase variante, așa cum a fost deja menționat:
    - Strategia de control înainte;
    - Căutarea exhaustivă, rezolvarea conflictelor – prima regulă, regula ce mai productivă, respectiv regula cea mai specializată;
    - Căutarea bazată pe agendă, rezolvarea conflictelor – prima regulă, regula ce mai productivă, respectiv regula cea mai specializată;
- Modulul de interfață cu utilizatorul și modulul explicativ au o structură rudimentară, și sunt aceleași pentru toate cele șase variante.

SELID poate fi folosit ca asistent, dar și sistem de control care poate lua deciziile și acționa independent, cu scopul de a localiza și izola defectele din sistemele electroenergetice de transport și distribuție. Acest sistem primește date de la aparatele de măsură din sistem, iar în funcție de aceste valori, va realiza anumite acțiuni sau va notifica inginerul pentru a-l ajuta pe acesta în luarea deciziilor. În continuare sunt prezentate în detaliu variantele scrise ale SELID, punându-se accent pe componentele SE și testarea acestuia.

4.4.1. Baza de cunoștințe

Baza de cunoștințe cuprinde BR și BF. Baza de reguli este compusă din 25 de reguli, care se referă la localizarea și izolarea defectelor, astfel datele de intrare sunt primite de la aparatele de măsură și senzori care arată starea aparatelor de protecție, a elementelor sistemului electroenergetic și caracteristicile energiei electrice. Primele 10 reguli din baza de reguli au următoarea formă:

Regula 1 – DACĂ I_l > 1,2 * I_nom, ATUNCI defect_posibil = Adevărat.

Regula 2 – DACĂ I_l > 1,2 * I_nom ȘI dt > 10 s, ATUNCI defect_posibil = Adevărat, înregistrați linia, notificați operatorul.

Regula 3 – DACĂ I_l > 30 mA, ATUNCI defect_de_linie = Adevărat.

Regula 4 – DACĂ I_l > 30 mA ȘI U > 1,1 * U_nom, ATUNCI tip_defect = supratensiune_de_la_masă, izolați linia, înregistrați defect.

Regula 5 – DACĂ I_l_A > 1,2 * I_nom ȘI I_l_B > 1,2 * I_nom ȘI I_l_C normal, ATUNCI tip_defect = fază-fază, izolați ambele linii, înregistrați defecțiune, notificați operatorul.

Regula 6 – DACĂ I_l_A > 1,2 * I_nom ȘI I_masă > 30 mA, ATUNCI tip_defect = fază-masă, izolați linia A.

Regula 7 – DACĂ I_l_A > 1,2 * I_nom ȘI I_l_B > 1,2 * I_nom ȘI I_masă > 30 mA ȘI dU > 10%, ATUNCI tip_defect = masă_fazică_multiplă, izolați ambele linii, înregistrați defectul, alertați SCADA.

Regula 8 – DACĂ U < 0,9 * U_nom, ATUNCI locație posibilă_defect = în aval.

Regula 9 – DACĂ U < 0,9 * U_nom ȘI declanșare_întrerupător = Fals ȘI dt > 5 s, ATUNCI confirmare_defect_aval, înregistrați tensiune.

Regula 10 – DACĂ declanșarea_întrerupătorului = Adevărat, ATUNCI izolarea_secțiunii_defectate.

Baza de fapte cuprinde date cu care lucrează SE. Inițial, aceasta conține informații primite de la aparatele de măsură, precum curentul de linie, tensiunea de alimentare, starea aparatelor de protecție sau comutație, și durata defectului.

Baza de cunoștințe va fi implementată în Python într-o singură variantă, și anume folosind clase și obiecte.

4.4.2. Motorul de inferențe

Motorul de inferențe (față de cele precedente a sistemelor expert a lucrărilor 1-3, care foloseau SE cadru, în cazul cărora MI era predefinit) va fi construit în totalitate, cu scopul de a evidenția procesul de operare a acestuia. De asemenea, vor fi șase variante ale MI, menționate mai sus, dar care vor avea toate la bază strategia de control înainte. Python a fost folosit pentru implementarea MI, în cele șase variante, a căror pseudocod a fost prezentat anterior în această lucrare.

4.4.3. Modulul explicativ

Modulul explicativ al SELID oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, adică folosirea regulile executate. Acest aspect se poate obține prin evaluarea treptată a pașilor realizați de aplicație până la soluționarea problemei. Precum s-a procedat și în cadrul lucrării precedente, nu s-a insistat pe acest aspect, ci pe dezvoltarea MI în cele șase variante. Față de cazurile precedente, acest modul a fost construit integral în Python.

4.4.4. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SELID este de tip minimalist, care conține doar elemente de tip text. Deci, utilizatorul va comunica cu SE prin mesaje text, iar acesta va afișa rezultatul la fel, sub formă de text (cuvinte, numere, simboluri). De altfel, scopul acestei lucrări de laborator este MI, deci nu s-a insistat pe acest modul al SE. La fel ca ME, și MIU a fost construit integral în Python.

4.4.5. Implementarea sistemului expert

SELID a fost implemntat în Python în șase versiuni (menționate anterior). Figura 4.3. ilustrează capturi cu BC, în care cunoștințele au fost reprezentate folosind clase și obiecte. Motorul de inferență a fost implementat ca algoritmi rezolutivi; figura 4.4. ilustrează o captură a versiunii care cuprinde căutarea exhaustivă și rezolvarea conflictelor prin prima regulă aplicabilă.

Liniile de comandă implementate în Python au fost salvate sub forma unui fișier text, care reprezintă materialul de lucru din cadrul orelor aplicative. În aceast material didactic sunt prezentate doar o parte din componentele SELID implementate.

```
# ============================================================
# Baza de reguli
# ============================================================
def creaza_reguli(es):
    es.add_regula("Detectat supracurent",
        ["I_l > 1.2 * I_nom"],
        ["defect_supracurent = True", "log = 'supracurent de linie'"])

    es.add_regula("Detectat defect de masa",
        ["I_masa > 0.03"],
        ["defect_de_masa = True", "anunta = 'operator'"])

    es.add_regula("Defect faza-faza",
        ["I_l_A > 1.2 * I_nom",
         "I_l_B > 1.2 * I_nom",
         "I_l_C <= I_nom"],
        ["tip_defect = 'faza-faza'", "log = 'defect faza'"])

    es.add_regula("Defect faza-nul",
        ["I_l_A > 1.2 * I_nom",
         "I_masa > 0.03"],
        ["tip_defect = 'faza-nul'", "izoleaza = 'linia A'"])

    es.add_regula("Detectata cadere de tensiune",
        ["U < 0.9 * U_nom"],
        ["localizare_defect = 'aval'", "log = 'tensiune mica'"])
```

Figura 4.3. Implementarea BC a SELID

```
*SEE-L4-SELID-e-prima.py - C:/Users/AncaMironUTCluj/Personal data/2024-2025/Suport did...
File  Edit  Format  Run  Options  Window  Help

        while executata:
            executata = False
            reguli_eligibile = []
            for regula in self.reguli:
                rezultat = regula.evalueaza(self.fapte)
                print(f"Regula testata: {regula.nume} → {rezultat}")
                if rezultat:
                    reguli_eligibile.append(regula)

            if reguli_eligibile:
                print(f"\n--- Ciclu {pas} Se executa prima regula eligibila---")
                regula = reguli_eligibile[0]
                inainte = dict(self.fapte)
                regula.executa(self.fapte)
                dupa = dict(self.fapte)
                self.explicatie.append((regula.nume, inainte, dupa))
                print(f"Regula executata: {regula.nume}")
                executata = True
                pas += 1

        print("\n=== Proces rezolutiv finalizat ===")
```

Figura 4.4. Implementarea MI a SELID

Codul Python pentru versiunea MI, care se bazează pe căutarea după agendă și metoda regula cea mai productivă este următorul:

```
    def ruleaza(self):
        for regula in self.reguli:
            regula._executata = False
        self.explicatie.clear()
        pas = 1
        while True:
            agenda = []
            for regula in self.reguli:
                if regula.evalueaza(self.fapte):
                    modificari = regula.prezice_modificari(self.fapte)
                    if modificari > 0:
                        agenda.append((modificari, regula))
            if not agenda:
                break
            agenda.sort(key=lambda x: x[0], reverse=True)
            modificari, regula_selectata = agenda[0]
            inainte = dict(self.fapte)
            regula_selectata.executa(self.fapte)
            dupa = dict(self.fapte)
            self.explicatie.append((regula_selectata.nume, inainte,
dupa, modificari))
            print(f"\n--- Pas {pas}: Regula executata
'{regula_selectata.nume}' produce {modificari} noi fapte ---")
            pas += 1
        print("\n=== Proces rezolutiv finalizat ===")
```

4.4.6. Testarea sistemului

Testarea SELID se realizează prin verificarea funcționării modulelor lui, pentru a i se determina eficacitatea, în cazul celor șase variante. În figura 4.5. este ilustrată testarea SELID realizată prin rularea aplicației cu linia de comandă (run), pentru varianta cu reguli de producție și faptele definite anterior.

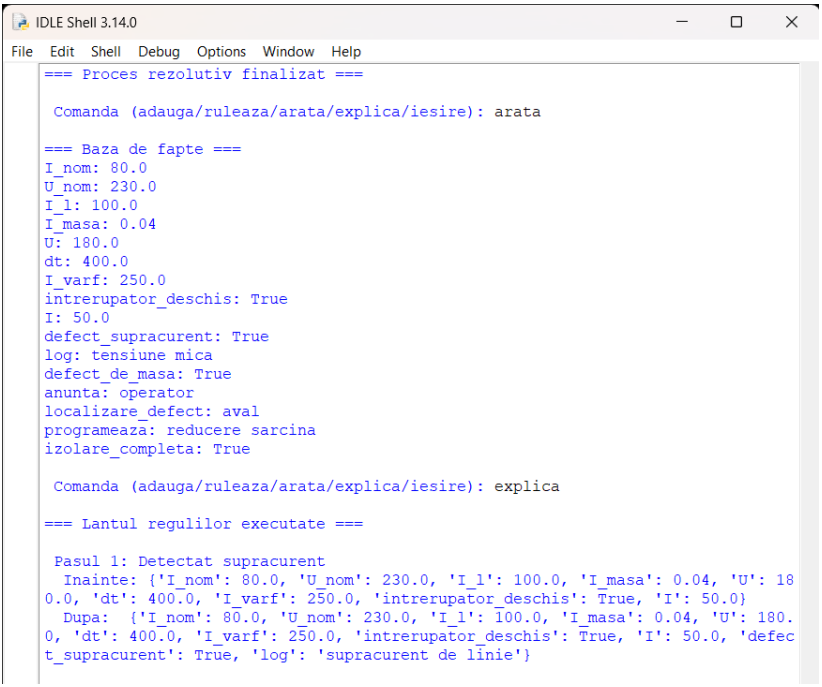


Figura 4.5. Testarea SELID

4.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea celor șase versiuni ale SELID, prin folosirea fișierelor SEE-L4-SELID_*.py, în IDLE Pyhton.
2. Analiza atentă a aplicației pentru identificarea celor patru module de bază ale SELID, adică BC, MI, ME și MIU. Se va pune accent pentru identificarea diferențelor în construcția MI în cele șase versiuni ale SELID.
3. Depanarea și testarea versiunilor SELID implementate prin folosirea faptelor implicite ale SE.
4. Testarea versiunilor SELID prin rularea SE folosind datele de intrare implicite în SE la care se vor adăuga altele particulare preferințelor utilizatorului.

5. Rezultatele se trec în tabelul 4.2 astfel – pentru fiecare set de date se vor specifica rezultatele pentru o versiune a SELID.

Tabelul 4.2. Valorile testării sistemului expert SELID

Date de intrare	**Rezultatele obținute**	**Observații**

4.6. Interpretarea rezultatelor

Datele din tabelul 4.2. se vor folosi pentru a trage concluzii cu privire la ordinea executării regulilor și numărul de reguli implicate,

informații cu care se va completa coloana ”Observații” din tabelul 4.2.

4.7. Concluzii

Se vor formula concluzii asupra algoritmului MI implementat și funcționarea lui în funcție de abordarea în rezolvarea conflictelor și tipului de căutare implicată. De asemenea, se vor evidenția avantajele și limitările celor doi algoritmi de căutare și a celor trei metode de rezolvare a conflictelor folosite în construcția MI a SELID în cele șase versiuni ale sale.

.....

.....

.....

.....

.....

.....

.....

.....

.....

5

MOTORUL DE INFERENȚE
STRATEGIA DE CONTROL ÎNAPOI

5.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta metodologia de funcționare a motorul de inferențe din cadrul sistemelor expert, în vederea înțelegii modalității de implementare și lucru a acestui algoritm rezolutiv în vederea rezolvării unor probleme specifice ingineriei energetice. Față de lucrarea precedentă, unde s-a folosit strategia de control înainte, aici se pune accentul pe strategia de control înapoi.

Pentru atingerea acestui scop va fi utilizat ca subiect de studiu, un sistem expert de anticipare dedicat realizării prognozei consumului de energie electrică la un consumator de energie electrică. Sistemul expert a fost denumit SEPCE (Sistem Expert pentru Prognoza Consumului de Energie electrică), care a fost dezvoltat prin folosirea limbajului de programare Python.

5.2. Obiectivele lucrării

Obiectivele lucrării de laborator sunt definirea și analiza metodologiei de funționare a motorului de inferențe, în cazul în care este folosită strategia de control înapoi. Aceasta presupune prezentarea strategiei de control înapoi, a algoritmilor de căutare și a metodelor de rezolvare a conflictelor specifice acestui raționament.

În continuare, lucrarea cuprinde noțiuni teoretice care descriu modul de funcționare a unui sistem expert particular considerând strategia de control înapoi, patru algoritmi de căutare și patru metode de rezolvare a conflictelor. Aspectele teoretice sunt completate cu testarea sistemul expert studiat (Sistem Expert pentru Prognoza Consumului de Energie - SEPCE) în cele patru variante, pentru a se scoate în evidență diferențele dintre diferitele versiuni ale sistemului expert. La final se vor scoate în evidență și nota puntele forte și cele slabe ale SEPCE, obținute în urma testării și analizei funcționării versiunilor sistemului expert (patru algoritmi de căutare, fiecare cu metoda particulară de rezolvare a conflictelor).

5.3. Noţiuni teoretice

Noțiunile teoretice necesare pentru realizarea acestei lucrări de laborator sunt legate de strategia de control înapoi, și algoritmii de căutare implicați, respectiv metodele de rezolvare a conflictelor folosite. Informații despre aspecte abordate și în lucrările precedente (precum caracteristicile Python) nu se reiau, ci doar ceea ce se introduce nou.

5.3.1. Motorul de inferențe

Strategia de control înapoi utilizează un raţionament centrat pe scop. Astfel, se începe de la scop, şi se caută faptele care permit atingerea acestuia. Această strategie se aplică atât sistemelor expert care folosesc reguli de producție, cât şi celor care folosesc graf-uri şi arbori. În cazul sistemelor expert care au BC compusă din reguli de producție, în timpul procesului de selecţie a regulilor, la o anumită etapă sunt selectate acele reguli care au în concluzii scopul specificat iniţial. Procesul se repetă până toate sub-scopurile obţinute sunt demonstrate, sau în urma etapei de filtraj mulţimea regulilor declanşabile este vidă, adică este “eşec”.

Diferenţa dintre strategia de control înainte și strategia de control înapoi constă în faptul că, în cazul primei strategii baza de fapte este modificată, iar starea problemei se schimbă în funcţie de noile fapte. Comparativ, strategia de control înapoi lasă nealterată baza de fapte iniţială, deoarece foloseşte doar baza de reguli, și anume consecinţele regulilor de producție. Pseudocodul strategiei de control înapoi, care

folosește căutarea exhaustivă și rezolvarea conflictelor prin alegerea celei mai specializate reguli este:

```
Funcția StrategiaÎnapoi (scop):
   Dacă scopul este în Baza_fapte:
      Returnează Adevărat
   Set_Reguli ← toate regulile care în partea de concluzie au scopul
   Dacă Set_Reguli este gol:
      Returnează Fals
   Selectează regula R din Set_Reguli care este cea mai specializată
      (adică are cele mai multe condiții, vezi lucrarea 4).
   Pentru fiecare premisă P din R.condiții:
      Dacă nu StrategieÎnapoi(P):
         Returnează Fals
   Adaugă scopul în Baza_fapte
   Returnează Adevărat
```

Pentru a crește eficiența motorului de inferențe, se implementează alături de strategia de control înapoi, algortimi de căutare mai avansați. În această lucrare sunt folosiți următorii algoritmi de căutare: căutarea în adâncime, căutarea în lățime și căutarea optimului.

Căutarea în adâncime (depth-first search) este cea mai frecvent folosită metodă de căutare în aplicaţiile practice. Algoritmul căutării în adâncime funcţionează astfel: la fiecare pas se generează o succesiune a nodurilor care definesc o posibilitate de parcurs a bazei de reguli. Dacă ultima regulă din acest lanț decizional este starea finală dorită, adică scopul, atunci acest lanț este considerat ca fiind calea în rezolvarea problemei. În caz contrar, se va reveni la ultimul nod care nu a fost parcurs şi se va forma o cale cu acesta. Pseudocodul motorul de inferențe bazat pe strategia de control înapoi, căutarea în adâncime și rezolvarea conflictelor prin selectarea primei reguli aplicabile este:

```
   Funcția StrategiaÎnapoi_CA(scop, vizitate):
      Dacă scopul este în Baza_fapte:
         Returnează Adevărat
      Dacă scopul în vizitate:
         Returnează Fals
      Adaugă scopul la vizitate
      Reguli_selectate ← toate regulile care au drept concluzii scopul
      Dacă Reguli_selectate este gol:
```

```
        Returnează Fals
    Pentru fiecare regulă R în Reguli_selectate (în ordinea definită):
        Toate_adevărat ← Adevărat
        Pentru fiecare premisă P din R.condiții:
            Dacă nu StrategieÎnapoi_CA(P, vizitate):
                Toate_adevărat ← Fals
                Pauză
        Dacă Toate_adevărat:
            Adaugă scopul în Baza_fapte
            Returnează Adevărat
    Returnează Fals
```

Căutarea în lăţime (breadth-first search) este o metodă de parcurgere a unei baze de cunoștințe mai puțin populară decât căutarea în lăţime, dar este adecvată problemelor a căror reprezentare arborescentă a regulilor arată o adâncime mare (ramuri lungi). Modul de lucru al căutării în lăţime este următorul: se porneşte de la scop și se parcurge baza de reguli, după ce au fost parcurse toate regulile corespunzătoare unui nivel, apoi se trece la nivelul următor (inferior). Pseudocodul motorului de inferență construit pe baza strategiei de control înapoi, algoritmul de căutare în lățime și rezolvarea conflictelor prin selectarea spre execuție a regulii celei mai productive este următorul:

```
Funcția StrategiaÎnapoi_CL(scop):
    Crează un șir S
    Adaugă scopul în S
    vizitate ← ∅
    Atâta timp cât S nu este gol:
        scop_curent ← ȘtergeȘirul(S)
        Dacă scop_curent este în Baza_fapte:
            Continuă
        Dacă scop_curent este în vizitate:
            Continuă
        Adaugă scop_curent în vizitate
        Reguli_selectate ← toate regulile a căror concluzie = scop_curent
        Dacă Reguli_selectate este gol:
            Continuă
        Ordonează crescător Reguli_selectate după numărul de condiții
        Pentru fiecare regulă R din Reguli_selectate:
            Toate_adevărat ← Adevărat
```

```
        Pentru fiecare premisă P din R.condiții:
            Dacă P nu este în Baza_fapte:
                Adaugă în listă(P)
                Toate_adevărat ← Fals
        Dacă Toate_adevărat:
            Adaugă scop_cutent în Baza_fapte
    Returnează (scop în Baza_fapte)
```

Algoritmii de căutare anteriori nu iau în considerare costul realizării unei acțiuni. Pentru a elimina acest dezavantaj, se folosesc algoritmi care integrează principiile căutării euristice precum căutarea soluției optime. În cazul acestei metode, fiecărei reguli candidate care ar putea satisface un obiectiv i se atribuie un scor euristic care reprezintă cât de promițătoare este - pe baza unor factori precum încrederea în regulă, numărul de antecedente deja satisfăcute sau costul estimat al verificării. Motorul de inferență selectează întotdeauna regula cu cea mai mare prioritate euristică, explorând mai întâi cea mai probabilă cale către succes. Această abordare euristică de rezolvare a conflictelor evită căutările exhaustive și revenirile inutile prin clasificarea inteligentă a regulilor, în loc să le aplice arbitrar. Pseudocodul MI care are la bază strategia de control înapoi, căutarea soluției optime și rezolvarea conflictelor prin alegerea regulii care are scorul cel mai bun este:

```
Funcția StrategiaÎnapoi_CSO(scop):
    Șir_prioritate ← gol
    Inserează (scop, prioritate=0) în Șir_prioritate
    vizitate ← ∅
    Atâta timp cât Șir_prioritate nu este gol:
        scop_curent ← Scoate_Prioritate_Maximă(Șir_prioritate)
        Dacă scop_curent în Baza_fapte:
            Continuă
        Dacă scop_curent în vizitate:
            Continuă
        Adaugă scop_curent în vizitate
        Reguli_selectate ← toate regulile care au concluzia = scop_curent
        Dacă Reguli_selectate este gol:
            Continuă
        Pentru fiecare regulă R din Reguli_selectate:
            scor_euristic ← CalculeazăEuristic(R)
            Inserează (R, scor_euristic) în Șir_scor_reguli
```

```
    Atât timp cât Șir_prioritate_reguli nu este gol:
      R ← Scoate_prioritate_maximă(Șir_prioritate_reguli)
      Toate_adevărat ← Adevărat
      Pentru fiecare premisă P din R.condiții:
        Dacă P nu este în Baza_fapte:
          Inserează (P, CalculeazăEuristicăptrScop(P)) în Șir_prioritate
          Toate_adevărat ← Fals
      Dacă Toate_adevărat:
        Adaugă scop_curent în Baza_fapte
        Pauză
Returnează (scop în Baza_fapte)
```

5.4. Sistemul expert de prognoză SEPCE

SEPCE este un sistem expert prototip, care a fost construit în scop didactic. Acesta face parte din categoria sistemelor expert de anticipare (sau prognoză), care a fost implementat în patru variante folosind Python.

SEPCE este subiectul de studiu al acestei lucrări de laborator, rolul lui didactic este de a ajuta în prezentarea modului de funcționare și construcție a motorului de inferență, atunci când acesta are la bază strategia de control înapoi. Astfel, sunt implementate patru versiuni ale SEPCE, care se caracterizează prin următoarele:

- Toate variantele au aceeași bază de cunoștințe;
- Motorul de inferențe este diferit pentru cele patru versiuni, astfel:
    - Strategia de control înapoi;
    - Căutarea exhaustivă și metoda de rezolvare a conflictelor după regula cea mai specializată;
    - Căutarea în lungime și rezolvarea conflictelor prin folosirea metodei – prima regulă aplicabilă;
    - Căutarea în lățime și rezolvarea conflictelor prin implicarea metodei – cea mai productivă regulă;
    - Căutarea soluției optime și rezolvarea conflictelor după o regulă euristică – alegerea regulii cu cel mai ridicat scor;

- Modulul de interfață cu utilizatorul și modulul explicativ au o structură rudimentară, și sunt aceleași pentru toate cele șase variante.

SEPCE poate fi folosit pentru estimarea consumului de energie electrică a unui consumator de energie electrică, în vederea creșterii eficienței energetice a acestuia prin luarea unor măsuri preventive.

5.4.1. Baza de cunoștințe

Baza de cunoștințe este formată din baza de reguli și baza de fapte. Baza de reguli este compusă din 30 de reguli, care sunt forma finală a procesului de reprezentare a cunoștințelor care ajută în determinarea consumului de energie estimat (E_estim) în funcție de factorii (precum ora din zi, ziua din săptămână, temperatura exterioară, luna din an etc.) care influențează consumul de energie. Baza de reguli cu primele 10 reguli are următoarea compoziție:

R1. DACĂ ora în 06:00–09:00, ATUNCI P_estim = 1.2 * P_med.
R2. DACĂ ora în 18:00–21:00, ATUNCI P_estim = 1,3 * P_med.
R3. DACĂ ora în 23:00–05:00, ATUNCI P_estim = 0,7 * P_med.
R4. DACĂ zi = sâmbătă sau duminică, ATUNCI P_estim = 0,85 * P_med_zi_l.
R5. DACĂ temp_ext < 5 °C, ATUNCI P_estim += 15%.
R6. DACĂ temp_ext > 30 °C, ATUNCI P_estim += 10%.
R7. DACĂ luna = decembrie sau ianuarie sau februarie, ATUNCI P_estim += 10%.
R8. DACĂ luna = iunie sau iulie sau august, ATUNCI P_estim += 8%.
R9. DACĂ precipitații > 10 mm/h, ATUNCI P_estim += 5%.
R10. DACĂ ocupare = Ridicată ȘI ora în 17:00–20:00, ATUNCI P_estim += 15%.

Baza de fapte reprezintă componenta BC, care cuprinde datele de intrare și apoi informațiile care caracterizează stările intermediare în procesul de estimare a consumului de energie electrică. Datele de intrare, adică faptele inițiale sunt: temperatura experioară, ora din zi, ziua din săptămână, luna din an, cantitatea de precipitații, gradul de ocupare a spațiului, zi lucrătoare / vacanță.

5.4.2. Motorul de inferențe

Motorul de inferențe, la fel ca în cazul lucrării precedente este construit în totalitate, cu scopul de a evidenția procesul de operare a acestuia. De asemenea, sunt implementate cele patru versiuni ale MI, menționate mai sus, dar care sunt toate bazate pe strategia de control înapoi. Python a fost folosit pentru implementarea MI, în cele patru versiuni, a căror pseudocod a fost prezentat anterior în această lucrare.

5.4.3. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SEPCE este de tip minimalist, care conține doar elemente de tip text. Deci, utilizatorul va comunica cu SE prin mesaje text, iar acesta va afișa rezultatul la fel, sub formă de text (cuvinte, numere, simboluri). De altfel, scopul acestei lucrări de laborator este MI, deci nu s-a insistat pe acest modul al SE. La fel ca ME, și MIU a fost construit integral în Python.

5.4.4. Modulul explicativ

Modulul explicativ al SEPCE oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, adică lanțul regulilor declanșate în rezolvarea problemei de prognoză. Acest aspect se poate obține prin evaluarea treptată a pașilor realizați de aplicație până la soluționarea problemei. Precum s-a procedat și în cadrul lucrării precedente, nu s-a insistat pe acest aspect, ci pe dezvoltarea MI în cele patru versiuni. Față de cazurile precedente, acest modul a fost construit integral în Python.

5.4.5. Implementarea SEPCE

SEPCE a fost implemntat în Python în cele patru versiuni (menționate anterior). Figura 5.1. ilustrează capturi cu BC, în care cunoștințele au fost reprezentate folosind clase și obiecte. Motorul de inferență a fost implementat sub forma unor algoritmi rezolutivi. Figura 5.2. ilustrează o captură a versiunii care cuprinde căutarea exhaustivă și rezolvarea conflictelor prin selectarea regulii celei mai specializate.

Figura 5.1. Captură SEPCE – BC

```
# --- Strategie de control înapoi ---
def StrategiaÎnapoi(scop, adâncime=0):
    if scop in fapte:
        return True

    reguli_aplicabile = [r for r in reguli if r["concluzie"] == scop]
    if not reguli_aplicabile:
        return False

    # Selectarea regulii celei mai specializate
    regula_bună = None
    scor_bun = -1
    for r in reguli_aplicabile:
        scor = sum(eval_reg(p) for p in r["premize"])
        if scor > scor_bun:
            regula_bună = r
            scor_bun = score

    if scor_bun == 0:
        return False
```

Figura 5.2. Captură SEPCE – MI

Liniile de comandă implementate în Python au fost salvate sub forma unui fișier text, care reprezintă materialul de lucru din cadrul orelor aplicative. În aceast material didactic este prezentată doar o parte din componentele SEPCE implementate. Codul Python pentru versiunea MI, care se bazează pe căutarea în lungime și selectarea primei reguli aplicabile este următorul:

```
def StrategiaÎnapoiCA(scop, vizitate):
    if scop in Baza_fapte:
        return True
    if scop in vizitate:
        return False
    vizitate.add(scop)
    reguli_aplicabile = [r for r in Reguli if r["concluzie"] == scop]
    if not reguli_aplicabile:
        return False

    for regula in reguli_aplicabile:
        Toate_adevărate = True
        for premize in regula["conditii"]:
            if not StrategiaÎnapoiCA(premize, vizitate):
                Toate_adevărate = False
                break

        if Toate_adevărate:
            Baza_fapte[scop] = regula["actiune"](FACT_BASE)
            Lant_Explicatie.append(regula["regula_nume"])
            return True
    return False
```

5.4.6. Testarea SEPCE

Testarea SEPCE se realizează prin verificarea funcționării modulelor lui, pentru a i se determina eficacitatea, în cazul celor patru variante ale sale. În figura 5.3. este ilustrată testarea SEPCE realizată prin rularea aplicației (accesarea Run Module), pentru prima variantă, și anume strategia de control înapoi, căutarea exhaustivă și regula cea mai specializată.

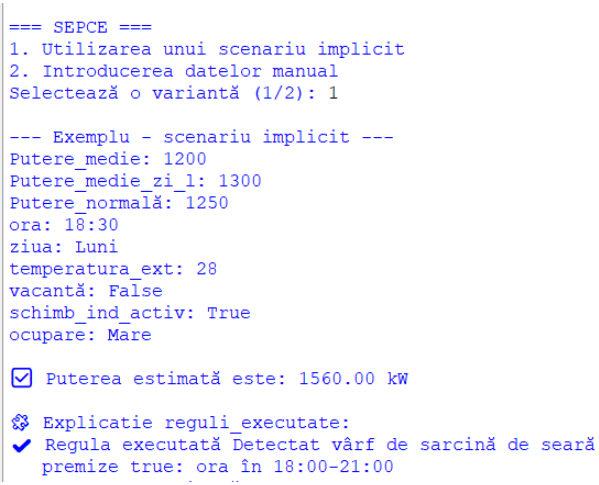


Figura 5.3. Captură SEPCE – testarea cu fapte implicite

5.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea celor patru versiuni ale SEPCE, prin folosirea fișierelor SEE-L5-SEPCE_*.py, în IDLE Python.
2. Analiza atentă a aplicației pentru identificarea celor patru module de bază ale SEPCE, adică BC, MI, ME și MIU. Se va pune accent pe identificarea diferențelor în construcția MI pentru cele patru versiuni ale SEPCE.
3. Depanarea și testarea versiunilor SEPCE implementate prin folosirea faptelor implicite ale SE.
4. Testarea versiunilor SEPCE prin rularea SE folosind datele de intrare implicite în SE la care se vor adăuga altele particulare preferințelor utilizatorului.
5. Rezultatele se trec în tabelul 5.1. astfel – pentru fiecare set de date se vor specifica rezultatele pentru o versiune a SEPCE.

Tabelul 5.1. Valorile testării sistemului expert SEPCE

Date de intrare	Rezultatele obținute	Observații

5.6. Interpretarea rezultatelor

Datele din tabelul 5.1. se vor folosi pentru a trage concluzii cu privire la ordinea executării regulilor și numărul de reguli implicate, informații cu care se va completa coloana ”Observații” din tabelul 5.1.

5.7. Concluzii

Se vor formula concluzii asupra algoritmului MI implementat și funcționarea lui în funcție de abordarea în rezolvarea conflictelor și tipului de căutare implicată. De asemenea, se vor evidenția avantajele și limitările celor patru algoritmi de căutare și a celor patru metode de rezolvare a conflictelor folosite în construcția MI a SEPCE în cele patru versiuni ale sale.

.....

.....

.....

.....

.....

.....

.....

.....

.....

6

MODULUL DE INTERFAȚĂ CU UTILIZATORUL

6.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta modul de construcție a modulului de interfață cu utilizatorii (bazat pe utilizatori) a sistemelor expert.

Pentru atingerea acestui scop va fi utilizat ca subiect de studiu, un sistem expert de control specializat în controlul unei centrale hibride de producere a energiei electrice bazate pe unități de generare care folosesc surse regenerabile de energie, precum energia eoliană și solară. Sistemul expert a fost denumit SEMER (Sistem Expert dedicat Managementului Energiei Regenerabile), care a fost dezvoltat prin folosirea limbajului de programare Python.

6.2. Obiectivele lucrării

Obiectivele lucrării de laborator sunt prezentarea modulului de interfață cu utilizatorii (MIU) a unui sistem expert, dacă acesta este construit special pentru utilizatorii specifici. Aceasta presupune descrierea caracteristicilor de bază ale MIU, precum și a modelării utilizatorilor sistemului expert. În continuare, lucrarea cuprinde noțiuni teoretice despre modulul de interfață cu utilizatorul a sistemelor expert și despre abordarea în dezvoltarea MIU conform trăsăturilor importante ale utilizatorilor specifici. Aspectele teoretice sunt completate cu partea practică, care presupune implementarea și

testarea sistemul expert studiat (Sistem Expert dedicat Managementului Energiei Regenerabile - SEMER). La final se vor scoate în evidență și nota puntele forte și cele slabe ale SEMER, obținute în urma testării și analizei funcționării acestuia. Se va pune accent pe elementele componente ale MIU.

6.3. Noţiuni teoretice

Noțiunile teoretice necesare pentru realizarea acestei lucrări de laborator sunt legate de modulul de interfață cu utilizatorul și în consecință modelarea utilizatorului specific. Informații despre aspecte abordate și în lucrările precedente (precum caracteristicile Python) nu se reiau, ci doar informațiile care se introduc pentru prima dată în cadrul orelor aplicative.

6.3.1. Modulul de interfață cu utilizatorii

Modulul de interfață cu utilizatorii (sau simplu interfața cu utilizatorii) este o componentă foarte importantă a unui SE. Studiile în domeniul SE au arătat faptul că aprecierea utilizatorilor asupra unui SE depinde de măsura de asemănare între modul de reprezentare a informaţiilor de către SE şi modelul pe care utilizatorul îl consideră corespunzător pentru rezolvarea problemei. Acest fapt este de asemenea în strânsă legătură cu modul în care toate informaţiile sunt făcute cunoscute utilizatorului, adică de interfaţa cu utilizatorul.

Interfaţa grafică cu utilizatorul, este acea variantă specială a unei interfeţe cu utilizatorul care se caracterizează prin faptul că interacţiunea cu operatorul uman se face prin intermediul unor elemente grafice (ferestre, icoane etc.). Alte variante de interfeţe cu utilizatorul utilizate în domeniul SE sunt interfeţele care se bazează pe text, pe comenzi etc., în care comunicarea dintre calculator și utilizator se realizează cu ajutorul tastaturii sau a mouse-ului.

Așa cum s-a specificat anterior, recunoaşterea şi acceptarea precoce a unui SE de către utilizatori este în legătură directă cu eficienţa şi imaginea sistemului, a caracteristicilor interfeţei şi a implicării utilizatorilor. Astfel, importanța modulului de interfață cu utilizatorii reiese din următoarele afirmații:

- MIU permite utilizatorilor fără expertiză tehnică sau specifică domeniului să beneficieze de sistemul expert.
- O interfață bine concepută asigură o interacțiune fluidă, reducând erorile și frustrarea utilizatorilor.
- Prezentarea clară și logică a informațiilor consolidează încrederea utilizatorilor în rezultatele sistemului.
- MIU optimizează procesul decizional prin minimizarea interacțiunilor inutile și prezentarea promptă a rezultatelor.

În dezvoltarea MIU este necesar să se considere un echilibru între aspectele de comunicare, control și acces ale SE. Problema comunicării se referă la dialogul dintre sistem și utilizator, în particular la cine iniţiază dialogul în situaţiile cheie ale procesului de soluţionare. În ceea ce priveşte controlul, interfaţa prezintă elemente care se referă la cine ia deciziile în situaţiile critice. De exemplu, există SE care lucrează cu utilizatori neexperimentaţi, la care sistemul este cel care ia deciziile. Accesul se referă la toate elementele unei interfeţe care permit utilizatorului să înţeleagă tot mecanismul și raţionamentul prin care s-a rezolvat problema. La interfaţa unui SE aceste elemente cuprind: ferestre de comandă, meniu Help etc.

Prima etapă în creionarea MIU este modelarea utilizatorului specific. Această activitate (modelarea utilizatorului) este un proces lung de determinare a trăsăturilor comune unui grup ţintă de utilizatori, în care trebuie să se aibă în vedere:

- Vârsta medie a utilizatorilor;
- Genul;
- Locul de muncă;
- Caracteristicile şi condiţiile în care se lucrează;
- Cunoștințele deţiune de utilizator;

Astfel, metodologia de realizare a interfeţelor bazate pe dialog porneşte de la modelarea utilizatorului specific. Scopul modelării utilizatorului este de a determina consecinţele comportamentului acestuia, respectiv cum să se achiziţioneze informaţia, să se reprezinte și cum să se afişeze rezultatele (soluţia şi raţionamentul).

6.4. Sistemul expert de control SEMER

SEMER este un sistem expert prototip, care a fost construit în scop didactic. Acesta face parte din categoria sistemelor expert de control, care a fost implementat folosind limbajul de programare Python. Precum s-a procedat la primele lucrări de laborator, acest SE este subiectul de studiu al prezentei lucrări de laborator, rolul lui didactic este de a ajuta în prezentarea modului de funcționare și construcție a modulului de interfață cu utilizatorul al sistemului expert.

SEMER poate fi folosit ca asistent în procesul de control și management al procesului de producere a energiei electrice din surse regenerabile. Scopul final este creșterea eficienței energetice a sistemului electroenergetic.

6.4.1. Baza de cunoștințe

Baza de cunoștințe este formată din baza de reguli și baza de fapte. Baza de reguli este compusă din 25 de reguli, care reprezintă baza de informații a sistemului de management energetic. Baza de reguli cu primele 10 reguli are următoarea compoziție

R1. Urmărirea punctului de putere maximă a panourilor fotovoltaice
DACĂ iradiere_solară > 200 W/m², ATUNCI reglați invertorul fotovoltaic la puterea maximă de ieșire.
R2. Viteza de conectare a turbinei eoliene
DACĂ viteză_vânt >= 3 m/s, ATUNCI porniți generatorul turbinei eoliene.
R3. Viteza de deconectare a turbinei eoliene
DACĂ viteză_vânt > 25 m/s, ATUNCI opriți turbina eoliană pentru a preveni deteriorarea.
R4. Încărcarea bateriei de la panoul solar
DACĂ putere_solară > cerere_sarcină ȘI încărcare_baterie < 90%, ATUNCI încărcați bateria.
R5. Descărcați bateria
DACĂ cerere_sarcină > putere_solară ȘI încărcare_baterie > 20%, ATUNCI descărcați bateria pentru a alimenta sarcina.
R6. Alimentare la rețea

DACĂ putere_solară + putere_vânt > cerere_sarcină ȘI încărcare_baterie = 100%, ATUNCI introduceți excesul de energie în rețea.
R7. Import rețea
DACĂ cerere_sarcină > putere_solară + putere_vânt + putere_baterie, ATUNCI importați energie din rețea.
R8. Controlul pasului turbinei eoliene
DACĂ Viteză_vânt > 15 m/s, ATUNCI ajustați pasul palelor pentru a limita viteza generatorului.
R9. Protecție la supraîncărcare a invertorului solar
DACĂ Curent_invertor_PV > Curent_nominal, ATUNCI reduceți puterea, anunțați operatorul.
R10. Avertisment SOC baterie descărcată
DACĂ SOC baterie < 20%, ATUNCI alertă: stocare insuficientă.

Baza de fapte reprezintă componenta BC, care cuprinde datele de intrare și apoi informațiile care caracterizează stările intermediare în procesul de estimare a consumului de energie electrică. SEMER, fiind un sistem expert de control, datele de intrare, adică faptele sunt obținute prin intermediul aparatelor de măsură și comutație, care oferă informații în timp real cu privire la: starea bateriei de înmagazinare a energiei electrice (SOC), viteza vântului, iradierea solară, puterea solară, puterea vântului, necesarul de putere a consumatorilor (cererea de putere), curentul invertorului a PV, etc., iar ca date de ieșire oferă sfaturi provind ceea ce trebuie realizat pentru a asigura necesarul de energie electrică, dar și creșterea eficienței energetice.

6.4.2. Motorul de inferențe

Motorul de inferențe, la fel ca în cazul lucrării precedente este construit în totalitate, având în vedere că implementarea în Python presupune acest lucru. Față de ultimele două lucrări, care au avut ca temă principală MI, această lucrare se focalizează pe MIU, în consecință, nu s-a pus accent pe MI, ci s-a folosit varianta cea mai uzuală a acestuia, și anume strategia de control înainte, căutarea exhaustivă și rezolvarea conflictelor prin selectarea primei reguli aplicabile (se ia în considerare ordinea în care au fost scrise regulile în baza de cunoștințe).

6.4.3. Modulul explicativ

Modulul explicativ al SEMER oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, adică lanțul regulilor declanșate în rezolvarea problemei de control. Acest aspect se poate obține prin evaluarea treptată a pașilor realizați de aplicație până la soluționarea problemei. Precum s-a procedat și în cadrul lucrării precedente, nu s-a insistat pe acest aspect, ci pe dezvoltarea MIU. La fel ca în ultimele două lucrări, acest modul a fost construit integral în Python.

6.4.4. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SEMER este construit special pentru utilizatorul specific, în consecință acesta conține elemente de tip text și grafic. Astfel, utilizatorul va comunica cu SE prin mesaje text și a unor elemente grafice, iar datele de ieșire oferite de SE sunt prezentate sub formă de text (cuvinte, numere, simboluri) dar și a unor elemente de tip grafic – butoane și grafice.

6.4.5. Implementarea în Python

SEMER a fost implemntat în Python pentru a se putea obține interfața personalizată și centrată pe utilizator. Figurile 6.1., 6.2. și 6.3. ilustrează capturi cu BC, MI, respectiv ME. Așa cum se observă în figuri, pentru fiecare modul s-a definit o clasă.

SEE-L6-SEMER-rom.py - C:/Users/AncaMironUTCluj/Personal data/2024-2025/Suport didactic/Sisteme Expert in Energetica/SEE_Indrumar laborator/SEE-L6-SEMER-rom.py (3.14.0)

File Edit Format Run Options Window Help

```
# Definirea regulilor

        # 1. Urmarire iradiere maxima
        def cond_solar_mppt(f):
            return f['iradiere_solara'] > 200
        def act_solar_mppt(f, g):
            changed = False
            if f.get('pv_invertor_iesire') != 'normal':
                changed = set_fact('pv_invertor_iesire', 'normala')
                action_log('Urmarirea punctului maxim solar: seteaza invertorul pe valoarea maxima de iesire')
            return changed
        self.kb.add_regula(regula('Iradiere maxima', cond_solar_mppt, act_solar_mppt))

        # 2. Pornirea turbinei eoliene
        def cond_wind_cut_in(f):
            return f['viteza_vant'] >= 3 and not f.get('functionare_turbina')
        def act_wind_cut_in(f, g):
            changed = set_fact('functionare_turbina', True)
            action_log('Pornire generator eolian: pornire turbina eoliana')
            return changed
        self.kb.add_regula(regula('Pornire generator eolian', cond_wind_cut_in, act_wind_cut_in))

        # 3. Deconectare turbina eoliana
        def cond_wind_cut_out(f):
            return f['viteza_vant'] > 25 and f.get('functionare_turbina')
        def act_wind_cut_out(f, g):
            changed = set_fact('functionare_turbina', False)
            action_log('Oprire generator eolian: deconectare turbina eoliana pentru a proveni deteriorarea', level='warn')
            return changed
        self.kb.add_regula(regula('Deconecatre turbina eoliana', cond_wind_cut_out, act_wind_cut_out))
```

Figura 6.1. Captură SEMER - BC

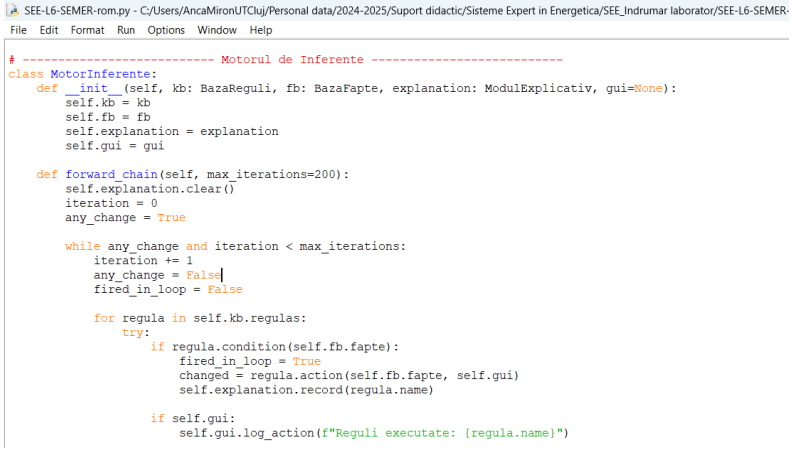


Figura 6.2. Captură SEMER - MI

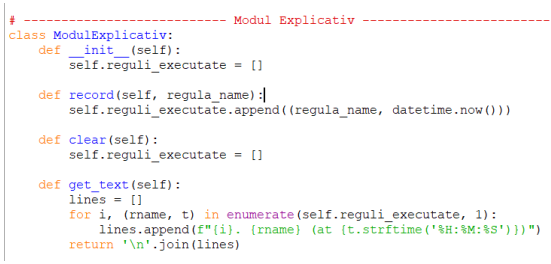


Figura 6.3. Captură SEMER - ME

Modulul de interfață cu utilizatorul este ilustrat în figura 6.4. Se poate observa din imagine folosirea tkinter, care se află în librăria standard, și nu necesită instalare suplimentară.

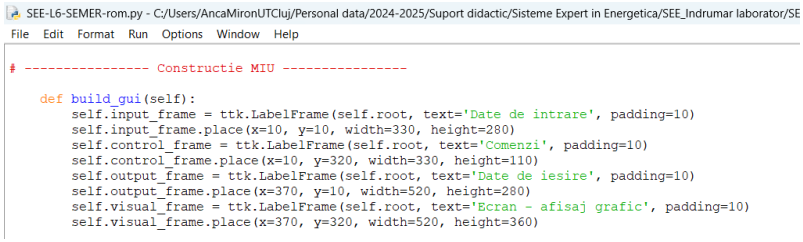


Figura 6.4. Interfața grafică cu uitilizatorul a SEMER

Liniile de comandă implementate în Python au fost salvate sub forma unui fișier text, care reprezintă materialul de lucru din cadrul orelor aplicative. În aceast material didactic este prezentată doar o parte din componentele SEMER implementate. Parte din codul Python dezvoltat pentru MIU, care a fost folosit pentru turbina eoliană este:

```
def desenare_turbina_eoliana(self, canvas):
    parts = {}
    w = 90
    h = 60
    parts['mast'] = canvas.create_rectangle(w-5, h, w+5, h+60, fill='sienna')
    parts['hub'] = canvas.create_oval(w-8, h-8, w+8, h+8, fill='grey')
    parts['blades'] = [
        canvas.create_line(w, h, w+50, h, width=3),
        canvas.create_line(w, h, w-25, h+43, width=3),
        canvas.create_line(w, h, w-25, h-43, width=3),
    ]
    parts['angle'] = 0
    return parts
```

6.4.6. Testarea SEMER

Testarea SEMER se realizează prin verificarea funcționării modulelor lui, pentru determinarea eficacității. În figura 6.5. este ilustrată testarea SEMER realizată prin rularea aplicației (accesarea Run Module), pentru datele de intrare implicite:

- Iradiere solară – 500 W/m^2;
- Viteza vântului – 6 m/s;
- Cerere energie – 5 kW (necesarul de kW pentru a răspunderii cererii);
- Incărcare baterie – 50 %;
- Capacitatea sistemului – 10 kW (cantitatea de kW produsă de sistemul de generare hibrid);
- Prognoză iradiere – 50 %;

- Tarif maxim – Fals;
- Frecvența rețelei – 50 Hz;
- Temperatura la nivelul PV – 30 °C.

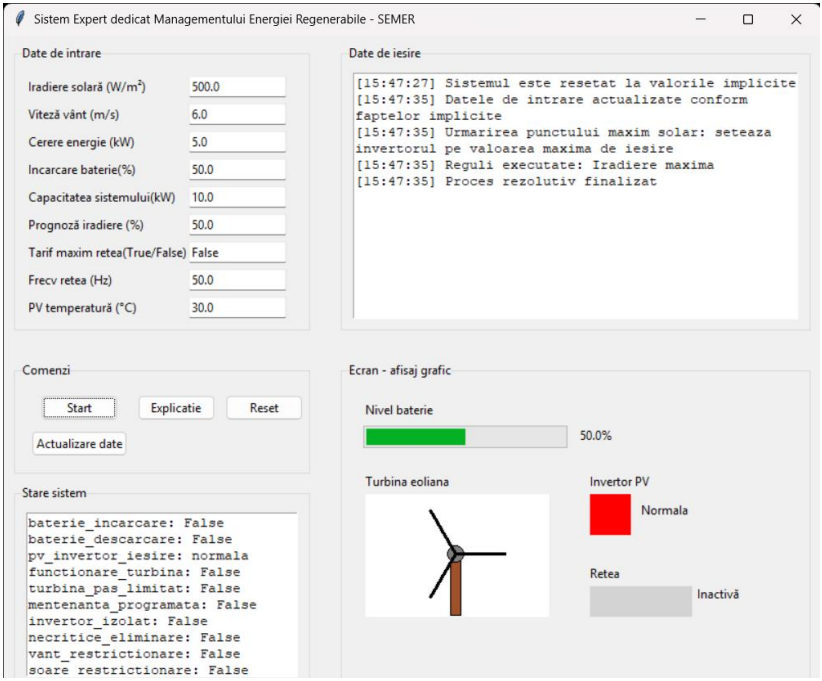


Figura 6.5. Testarea SEMER – captură interfața cu utilizatorii

6.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea celor patru versiuni ale SEMER, prin folosirea fișierului SEE-L6-SEMER.py, în IDLE Python.
2. Analiza detaliată a aplicației pentru identificarea celor patru module de bază ale SEMER, adică BC, MI, ME și MIU. Se va pune accent pe identificarea elementelor modului de interfață cu utilizatorii al SEMER.

3. Depanarea și testarea SEMER implementate prin folosirea faptelor implicite ale SE.
4. Testarea suplimentară a SEMER folosind datele de intrare introduse de utilizator.
5. Se vor aduce modificări la BC și ME, se va salva noua versiune ca o copie a versiunii originale.
6. Rezultatele se trec în tabelul 6.1. astfel – pentru fiecare set de date se vor specifica rezultatele obținute și informațiile oferite de modulul explicativ, precum și versiunea – originală sau modificată.

Tabelul 6.1. Valorile testării sistemului expert SEMER

Date de intrare	**Rezultatele obținute**	**Observații**

6.6. Interpretarea rezultatelor

Datele din tabelul 6.1. se vor folosi pentru a trage concluzii cu privire la ordinea executării regulilor și numărul de reguli implicate, informații cu care se va completa coloana ”Observații” din tabelul 6.1.

6.7. Concluzii

Se vor formula concluzii asupra MIU implementat și funcționarea acestuia. De asemenea, se vor evidenția avantajele și limitările acestuia, comparativ cu sistemele expert folosite în cadrul lucrărilor anterioare, care nu aveau o interfață grafică dedicată, ci una minimalistă, bazată pe text.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

7

MODULUL EXPLICATIV

7.1. Scopul lucrării

Scopul acestei lucrări este de a prezenta modul de construcție a modulului explicativ (dedicat utilizatorilor) a sistemelor expert.

Pentru atingerea acestui scop va fi utilizat ca subiect de studiu, un sistem expert de clasificare capabil să determine tipul consumatorilor de energie electrică (precum casnici, industriali și comerciali) în funcție de caracteristicile de consum. Sistemul expert a fost denumit SECCE (Sistem Expert dedicat Clasificării Consumatorilor de Energiei electrică), care a fost dezvoltat prin folosirea sistemului expert cadru CLIPS IDE.

7.2. Obiectivele lucrării

Obiectivele lucrării de laborator sunt prezentarea modulului explicativ (ME) a unui sistem expert, dacă acesta este construit special pentru utilizatorii și deci folosește anumite tipuri de explicații. Aceasta presupune descrierea caracteristicilor de bază ale ME, precum și tipurilor de explicații. În continuare, lucrarea cuprinde noțiuni teoretice despre modulul explicativ. Aspectele teoretice sunt completate cu partea practică, care presupune implementarea și testarea sistemul expert studiat (SECCE). La final se vor scoate în evidență și nota puntele forte și cele slabe ale SECCE, obținute în urma testării și analiza funcționării acestuia. Se va pune accent pe ME.

7.3. Noţiuni teoretice

Noțiunile teoretice necesare pentru realizarea acestei lucrări de laborator sunt legate de modulul explicativ și tipurile de explicații. Informațiile despre aspectele abordate și în lucrările precedente (precum caracteristicile CLIPS IDE) nu se reiau, ci doar informațiile care se introduc pentru prima dată în cadrul orelor aplicative.

7.3.1. Modulul explicativ

Modul explicativ este o componentă de bază a unui SE, întrucât acesta prin copierea comportamentului expertului uman trebuie să ofere posibilitatea furnizării de explicații utilizatorului. Într-adevăr, rolurile ME în cadrul și în munca cu un SE sunt următoarele:

- Oferă justificare pentru concluzii;
- Crește încrederea arătând utilizatorilor cum a ajuns sistemul expert la concluzia sa;
- Ajută la instruirea personalului mai puțin experimentat prin expunerea logicii din spatele deciziilor experților;
- Asistă la depanarea bazei de cunoștințe ajutând dezvoltatorii să înțeleagă care reguli s-au declanșat și de ce;
- Sprijină auditarea și conformitatea în mediile reglementate.

Experienţa în dezvoltarea SE, și a ME a arătat că utilizatorii SE au nevoie de informaţii despre procedurile, raţionamentele, scopul, controlul şi cunoştinţe în general. Aceste necesităţi, clasifică explicaţiile necesare utilizatorilor în următoarele tipuri, care în multe surse bibliografice apar menţionate doar primele trei [1]:

- De ce ? – dau informaţii care justifică o acţiune întreprinsă de SE;
- Cum ? – dau informaţii despre drumul parcurs în baza de cunoștințe care a dus la soluţia găsită;
- Strategice – dau utilizatorului informaţii despre meta-cunoștințe (controlul și strategiile folosite de sistem);
- Ce ? – concepute pentru a da informaţii despre obiectele şi variabilele folosite de SE;

- Ce dacă ? – permite reaplicarea raţionamentelor cu alţi parametrii ai variabilelor de intrare.

Sistemele experte recente au introdus explicaţiile de tipul "feedforward" şi "feedback" ataşate celor trei explicaţii de bază – de ce, cum şi strategice – figura 7.1.

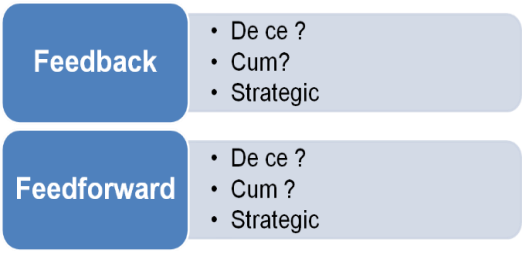


Figura. 7.1. Tipuri de explicaţii

Cele două tipuri de explicaţii se diferenţiază prin faptul că explicaţiile feedforward se referă la informaţii care sunt date utilizatorilor înainte de realizarea unei acţiuni, comparativ cu explicaţiile feedback care se focalizează pe ieşiri.

În dezvoltarea ME trebuie să se aibă în vedere faptul că există mai mulţi factori care pot să îl influenţeze [1]: tipul atribuţiilor (sarcinilor) care trebuie să le rezolve SE și contextul în care se află SE la un moment dat, categoria explicaţiilor, interfaţa şi modul de prezentare a explicaţiilor și utilizatorii.

7.4. Sistemul expert de clasificare SECCE

SECCE este un sistem expert prototip, care a fost construit în scop didactic. Acesta face parte din categoria sistemelor expert de clasificare, care a fost implementat folosind sistemul expert cadru CLIPS IDE. Precum s-a procedat la primele lucrări de laborator, acest SE este subiectul de studiu al prezentei lucrări de laborator, rolul lui didactic este de a ajuta în prezentarea modului de funcționare și construcție a modulului explicativ al sistemului expert.

SECCE poate fi folosit pentru a obține o clasificare a consumatorilor de energie electrică în vederea obținerii unei imagini

clare a necesarului de energie electrică și deci posibilitatea creșterii eficienței energetice a sistemului electroenergetic.

În continuare sunt prezentate componentele SECCE, la care se adaugă descrierea aspectelor importante privind implementarea și testarea SE.

7.4.1. Baza de cunoștințe

Baza de cunoștințe este formată din baza de reguli și baza de fapte. Baza de reguli este compusă din 24 de reguli, care reprezintă fundația procesului de clasificare, întrucât acestea fac legătura dintre categoriile de consumatori și factorii prin care se determină aceste categorii. Baza de reguli cu primele 10 reguli are următoarea compoziție:

R1 — Consumator casnic - vârf de seară
DACĂ vârf_seara > 1,4 * vârf_dimineața ȘI min_noapte < 0,3 * vârf_seara, ATUNCI categoria = casnic.

R2 — Consumator casnic - forma tipică a consumului zilnic
DACĂ vârf_dimineața între 0,4 și 0,8 din vârf_seara ȘI indice_variață > 0,5, ATUNCI categoria = casnic

R3 — Consumator casnic - consum ridicat în weekend
DACĂ medie_weekend >= medie_zi_săptămână * 0,8 ȘI vârf_seara semnificativ, ATUNCI categoria = casnic

R4 — Consumator casnic - consum scăzut pe timp de noapte
DACĂ min_noapte < 0,25 * sarcină_max ȘI medie_zi < vârf_seara, ATUNCI categoria = casnic

R5 — Consumator comercial – consum dominat de zi
DACĂ media_zi > 1,5 * vârf_seară ȘI min_noapte < 0,4 * medie_zi, ATUNCI categoria = comercial

R6 — Consumator comercial - model de închidere în weekend
DACĂ media_weekend < 0,3 * media_zi_luna_săptămânii ȘI media_zi > vârf_seară, ATUNCI categoria = comercial

R7 — Consumator comercial – consum în timpul zilei
DACĂ indicele_de_planare ridicat între orele 8:00–18:00 ȘI vârf_seară mic, ATUNCI categoria = comercial

R8 — Consumator comercial - semnătură de închidere comercială anticipată
DACĂ scădere bruscă după 18:00 (rata_schimbărilor_bruste > 1,8) ȘI media_weekend scăzută, ATUNCI categoria = comercial

R9 — Consumator industrial – curbă de sarcină plată pe 24 de ore
DACĂ de la vârf la mediu < 1,2 ȘI min. noapte > 0,7 * sarcină maximă, ATUNCI categoria = industrial_continuu

R10 — Consumator industrial - funcționare stabilă cu variație redusă
DACĂ indice_varianță < 0,2 ȘI medie_zi ≈ medie_noapte, ATUNCI categoria = industrial_continuu

Baza de fapte reprezintă componenta bazei de cunoștințe, care cuprinde datele de intrare și apoi informațiile prin care sunt caracterizate stările intermediare în procesul de clasificare a consumatorilor în funcție de semnătura consumului de energie electrică. SECCE, fiind un sistem expert de clasificare, datele de intrare, adică faptele sunt obținute de la utilizatori. Aceste fapte fac referire la consumul de energie electrică, mai exact: vârf_dimineață, vârf_seară, încărcare_max., minim_noapte, medie_zi, medie_weekend, medie_zi_săptămână, medie_vârf, indice_sezon, raport_schimbări_bruște, medie_noapte, vârf_zi, indice_varianță, indice_aplatizare.

7.4.2. Motorul de inferențe

Motorul de inferențe este predefinit, având în vedere că se folosește un sistem expert cadru. Luându-se în considerare faptul că SECCE este implementat în CLIPS IDE, atunci rezultă că MI are la bază strategia de control înainte, algoritmul Rete (ca algoritm de căutare) și rezolvarea conflictelor prin implicarea a trei metode, și anume după indicatorul de prioritate, numărul de fapte și ordinea din baza de reguli.

7.4.3. Modulul de interfață cu utilizatorul

Modulul de interfață cu utilizatorul al SECCE este de tip text. Aceasta deoarece CLIPS IDE nu oferă facilități grafice în această direcție, ci doar text. În plus, tema acestei lucrări de laborator este

ME, și deci, nu s-a insistat pe acest aspect. În concluzie, comunicarea cu utilizatorul a SECCE se face doar prin text.

7.4.4. Modulul explicativ

Modulul explicativ al SECCE oferă posibilitatea utilizatorului să afle informații despre cum s-a rezolvat problema, adică lanțul regulilor declanșate în rezolvarea problemei de control. Acest aspect se poate obține prin evaluarea treptată a pașilor realizați de aplicație până la soluționarea problemei. În plus, au fost adăugate reguli, prin care utilizatorii pot să afle explicații care să răspundă la întrebăriile menționate în 7.3.1.

7.4.5. Implementarea SECCE

SECCE a fost implemntat în CLIPS IDE, așa cum ilustrează și figura 7.2., unde este prezentată o parte a bazei de cunoștințe.

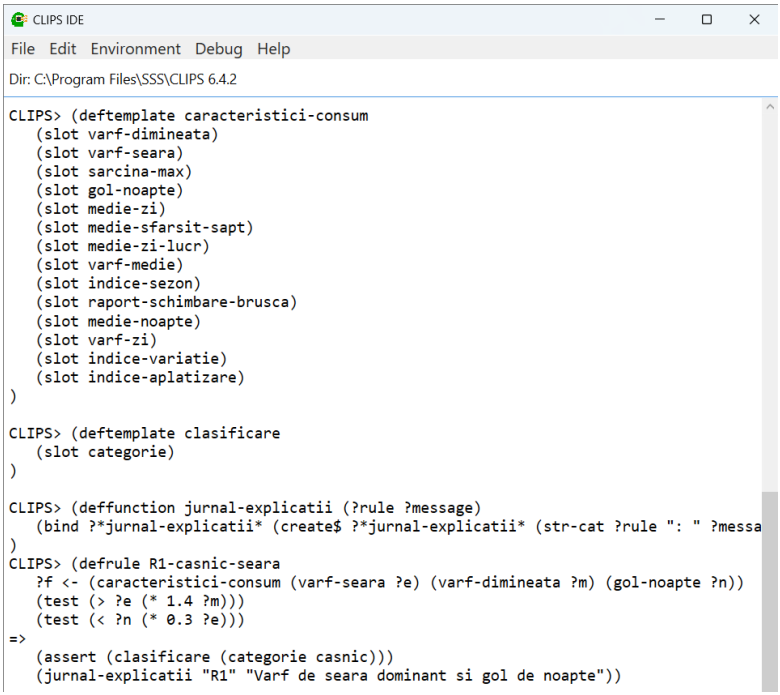


Figura 7.2. Captură SECCE – BC

Liniile de comandă implementate în CLIPS IDE au fost salvate sub forma unui fișier text, care reprezintă materialul de lucru din cadrul orelor aplicative. În aceast material didactic sunt prezentate doar o parte din componentele SECCE implementate.

Baza de cunoștințe a SECCE a fost populată cu reguli de producție, care au fost grupate în funcție de categoria consumatorului de energie electrică. În continuare sunt prezentate două reguli, una pentru consumatorul casnic, respectiv pentru consumatorul suspicios.

```
defrule R4-consumator-casnic-gol-noapte
  ?f <- (caracteristici-consum (gol-noapte ?n) (sarcina-max ?mx) (medie-zi ?d))
  (test (< ?n (* 0.25 ?mx)))
  (test (< ?d ?mx))
=>
  (assert (clasificare (categorie casnic)))
  (jurnal-explicatii "R4" "Gol de noapte si media zilnica este sub sarcina maxima"))

(defrule R23-consumator-suspicios
  ?f <- (caracteristici-consum (raport-schimbare-brusca ?scr))
  (test (> ?scr 3))
=>
  (assert (clasificare (categorie suspicios)))
  (jurnal-explicatii "R23" "Modificari bruse si iregulare in consum"))
```

MIU minimalist, este componenta prin care se cere utilizatorului introducerea datelor:

```
(deffunction introducere-caracteristici-consum ()
  (printout t "Introduceti varf de dimineata: ") (bind ?m (read))
  (printout t "Introduceti varf de seara: ") (bind ?e (read))
  (printout t "Introduceti sarcina maxima: ") (bind ?mx (read))
  (printout t "Introduceti golul de noapte: ") (bind ?n (read))
  (printout t "Introduceti media zilnica: ") (bind ?d (read))
  (printout t "Introduceti media de weekend: ") (bind ?w (read))
```

```
    (printout t "Introduceti media zi lucratoare: ") (bind ?wd (read))
    (printout t "Introduceti raport varf-medie: ") (bind ?p2a (read))
    (printout t "Introduceti indice sezonier: ") (bind ?s (read))
    (printout t "Introduceti raport modificare brusca: ") (bind ?scr
(read))
    (printout t "Introduceti media noaptra: ") (bind ?na (read))
    (printout t "Introduceti varf de zi: ") (bind ?dp (read))
    (printout t "Introduceti indice de variatie: ") (bind ?v (read))
    (printout t "Introduceti indice de aplatizare: ") (bind ?fli (read))
    (assert (caracteristici-consum
      (varf-dimineata ?m)
      (varf-seara ?e)
      (sarcina-max ?mx)
      (gol-noapte ?n)
      (medie-zi ?d)
      (medie-sfarsit-sapt ?w)
      (medie-zi-lucr ?wd)
      (varf-medie ?p2a)
      (indice-sezon ?s)
      (raport-schimbare-brusca ?scr)
      (medie-noapte ?na)
      (varf-zi ?dp)
      (indice-variatie ?v)
      (indice-aplatizare ?fli)
    ))
  )
```

Componenta ME a fost inclusă sub forma unor funcții, care corespund tipurilor de explicații menționate în lucrare. Codul pentru explicația de tip ”de ce ” este:

```
  (deffunction explica-de-ce ()
    (printout t "Explicatie (reguli executate):" crlf)
    (foreach ?x ?*jurnal-explicatii*
      (printout t " - " ?x crlf)
    )
  )
```

7.4.6. Testarea SECCE

Testarea SE se realizează prin rularea lui, pentru a i se verifica eficacitatea. De asemenea, se poate urmări procesul de soluționare prin selectarea din ”Debug” a ”Agenda”, ”Facts” și ”Instances”, respectiv ”Watch” – All.

În figura 7.3. este ilustrată testarea SECCE realizată prin rularea aplicației cu linia de comandă (run), cu datele de intrare clar evidențiate. Se poate observa din imagine, că testarea a fost realizată prin introducerea unor fapte care determină declanșarea tuturor regulilor și afișarea ca rezultat a tuturor categoriilor de consumatori. Aceste date nu vor fi folosite în cadrul orelor aplicative.

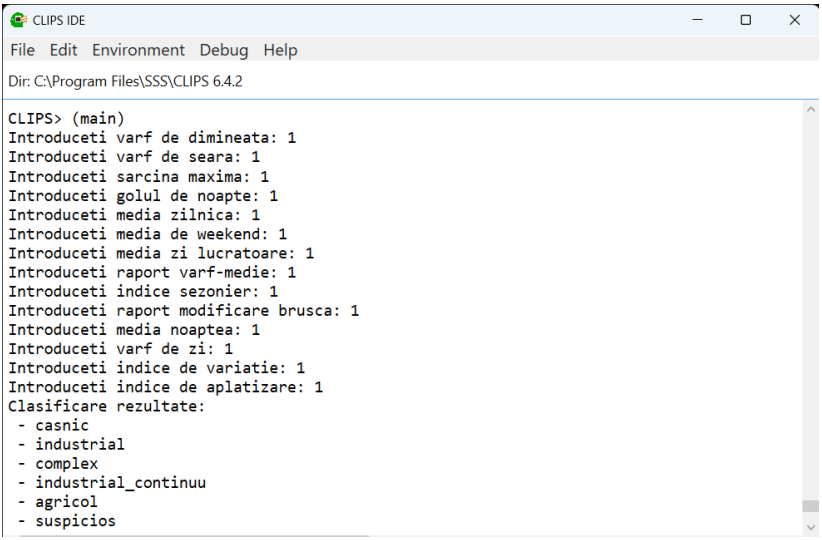


Figura 7.3. Captură SECCE – testarea sistemului expert

7.5. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

1. Implementarea SECCE, prin folosirea fișierului SEE-L7-SECCE.txt, în CLIPS IDE.

2. Analiza atentă a aplicației pentru identificarea celor patru module de bază ale SECCE, adică BC, MI, ME și MIU. Se va pune accent pe identificarea elementelor modului explicativ.
3. Depanarea și testarea SECCE implementat prin folosirea unor fapte care să ducă la apariția unei singure categorii de consumator.
4. Testarea suplimentară a SECCE prin introducerea unor fapte, care să cauzeze alegerea pe rând a fiecărui tip de consumator.
5. Se vor aduce modificări la BC și ME, și se va salva noua versiune ca o copie a versiunii originale.
6. Rezultatele se trec în tabelul 7.1. astfel – pentru fiecare set de date se vor specifica rezultatele obținute și informațiile oferite de modulul explicativ, precum și versiunea – originală sau modificată.

Tabelul 7.1. Valorile testării sistemului expert SECCE

Date de intrare	**Rezultatele obținute**	**Observații**

7.6. Interpretarea rezultatelor

Datele din tabelul 7.1. se vor folosi pentru a trage concluzii cu privire la ordinea executării regulilor și numărul de reguli implicate, informații cu care se va completa coloana ”Observații” din tabelul 7.1. În plus, se va specifica dacă SE a răspuns tuturor întrebărilor.

7.7. Concluzii

Se vor formula concluzii asupra ME implementat și funcționarea acestuia. De asemenea, se vor evidenția avantajele și limitările acestuia, comparativ cu sistemele expert folosite în cadrul lucrărilor anterioare, care nu aveau un modul explicativ dedicat, ci unul de

bază, care oferea informații cu precădere despre pașii realizați de algoritmul rezolutiv.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

BIBLIOGRAFIE

[1]. Mihai Gavrilaş, Inteligenţa Artificială şi Aplicaţii în Energetică, Cap. 1, Vol I, Ed. Gh. Asachi, Iaşi, 2002.

[2]. *CLIPS Reference Manual, Volume 1, Basic Programming Guide, Version 6.4.2, Secret Society Software, LLC, 2025.

[3]. Joseph C. Giarratano, CLIPS User's Guide, version 6.30, accesat ultima data octombrie 2025, https://www.clipsrules.net/documentation/v631/ug631.pdf.

[4]. *https://www.clipsrules.net/ - acces CLIPS domeniu public în octombrie 2025.

[5]. *SWI-Prolog Reference Manual, version June 2020, accesat ultima dată în octombrie 2025, https://www.swi-prolog.org/pldoc/doc_for?object=manual.

[6]. Anca Miron, Sisteme Expert în Energetică. Suport de curs. UTPRESS, Cluj-Napoca 2025, ISBN: 978-606-737-802-3.

[7]. Tania Tudorache, Csongor Nyulas, Natalya F. Noy & Mark A. Musen, WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web, Semantic Web (Journal), Vol. 4, No. 1, 2013, pages 89–99

[8]. Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, et. All, SciPy 1.0 — Fundamental Algorithms for Scientific Computing in Python, 2019.