SISTEM EXPERT CADRU, BAZAT PE REGULI DE PRODUCȚIE DE TIP BACKWARD- CHAINING

Monica Paladi
Adrian Zagar
Silviu Timus

Universitatea Politehnică București

Rezumat: Acest articol prezintă un sistem expert cadru, bazat pe reguli de producție, avînd facilități de explicare. Sistemul are o politică de tip backward-chaining. Asigură facilități de afișare prietenoasă atît a regulilor inițiale, cît și a formelor interne ale dicționarului utilizat și ale regulilor. Permite încărcarea/ștergerea selectivă din memorie a faptelor nou achiziționate, a tuturor faptelor sau ale regulilor. Implementează metode de preprocesare a regulilor care optimizează puternic procesul de căutare (orice căutare are un timp de calcul proporțional cu 0(1)). Mediul integrat include un sistem de meniuri și ferestre cu help incorporat. Sistemul este dezvoltat în C++ pe IBM-PC, sub MS-DOS.

1. Introducere

Existența unor probleme cu caracter special, dificil sau imposibil de abordat cu metode deterministe, a dus la dezvoltarea puternică, în ultimii ani, a unor tipuri diverse de sisteme expert și de sisteme capabile să construiască sisteme expert.

Utilizarea unui sistem expert este recomandabilă:

- pentru probleme nedeterministe;
- pentru probleme nedecidabile;
- pentru probleme deterministe cu o structură generală stabilă, dar a căror structură de detaliu se modifică. [1]

Pentru aceste tipuri de probleme, sistemele expert duc la obținerea unor rezolvări convenabile (soluții nu neapărat exacte/optime, dar suficient de bune).

Există multe criterii de clasificare a sistemelor expert. Amintim:

a) după modul de reprezentare a cunoștințelor:
- logica propozițională sau calcul cu predicate;
- bazate pe reguli;
- bazate pe obiecte structurate;
- rețele semantice;
- sisteme expert particulare;

b) după tipul problemelor rezolvate:
- sisteme expert pentru clasificare;
- sisteme expert pentru diagnoză;
- sisteme expert pentru probleme cu caracter constructivist [2].

În lucrarea de față este prezentat un sistem expert cadru (care permite construcția de sisteme expert) bazat pe reguli, pentru rezolvarea problemelor de diagnoză. Sistemul permite folosirea unei baze de reguli pentru mai multe seturi de fapte, permițînd memorarea/ștergerea cunoștințelor nou achiziționate. Acest lucru permite preprocesarea bazei de reguli (operația cea mai costisitoare în timp) o singură dată și utilizarea multiplă a acesteia.

Cu acest sistem se pot realiza sisteme expert de diagnoză. Această clasă de sisteme are multiple aplicații în domenii variate (tehnică, medicină, economie etc.).

Arhitectura generală a unui sistem bazat pe reguli de producție este constituită din:

- o colecție de fapte și de reguli dintr-un anumit domeniu al cunoașterii umane;
- un mecanism de control al inferenței acestor reguli [4], [5].

2. Baza de reguli

Baza de reguli se încarcă sub formă de fișier extern, care poate fi editat cu orice editor de texte.

Regulile trebuie să respecte sintaxa descrisă de următoarea gramatică predictivă:

```
BR  →R P
P   →;BR |
R   →"if" E "then" E "cnf=" număr
E   →T A
T   →F B
A   →"or" E | lambda
B   →"and" T | lambda
F   →"("E")" | "not" F | simbol
```

Deci baza de reguli este constituită dintr-un număr nedeterminat de reguli separate prin ";", ultima regulă fiind terminată prin ".".

O regulă este constituită din cuvîntul cheie 'if' urmat de precondiție, de cuvîntul cheie 'then', concluzie, apoi cuvîntul cheie 'cnf=' și factorul de certitudine (0-100).

Factorul de certitudine are semnificația unei probabilități matematice (în procente) asupra valabilității regulii respective.

Precondiția unei reguli este o expresie conținînd fapte (delimitate prin simbolul ' ' (backquote), paranteze și cuvintele cheie 'not', 'or', 'and'.

Sunt permise oricîte niveluri de imbricare a expresiilor. Precedența operatorilor (în absența parantezelor) este cea normală (not, and, or).

Concluzia unei reguli este o expresie de tipul precondiției. Concluzia va fi redusă la forma normal conjuctivă și va fi luată în considerare doar prima conjucție.

Deci, o concluzie de tipul "NOT ('a' OR 'b')" este validă, deoarece se va face întîi reducerea la "NOT 'a' AND NOT 'b'" și abia apoi va fi analizată.

De reținut că, în nici un loc spațiile, tab-urile, și 'Ân' nu sunt considerate delimitatori; ele se ignoră, fiind folosite doar pentru aranjarea prietenoasă în pagină a regulilor.

3. Faptele

În baza de reguli, ca și în baza de date, apar fapte.

Faptele sunt șiruri de caractere (oarecare), delimitate de simbolurile ' '. În interiorul celor doi delimitatori, faptele pot avea orice formă. În cazul întîlnirii unuia dintre șirurile >,<,<>,=, faptul respectiv va fi considerat ca fiind o comparație între două fapte cu caracter de variabilă de tip real (float), cele două variabile rezultate fiind tratate astfel în restul bazei de reguli.

Determinarea tipului de fapt (normal sau variabilă) se face automat la analiza sintactică a bazei de reguli, nefiind necesară o secțiune de declarare.

Fiecare fapt este caracterizat printr-un coeficient de certitudine, (0-100), care reflectă probabilitatea (în procente) ca acel fapt să fie adevărat.

Excepție fac variabilele, caracterizate prin valoare (un număr real). Odată aflată valoarea unei variabile, ea este considerată sigură (factor de certitudine 100).

4. Baza de date

După încărcarea unei baze de reguli (imediat sau după un timp de utilizare, caracterizat de achiziționarea unor noi date), poate fi încărcat un fișier de date. Acest fișier poate fi editat cu orice editor de texte extern.

Sintaxa fișierelor de date respectă gramatica:

BD →<fapt> = <valoare> R

R →'\n' BD | lambda

După cum se observă, fișierul conține o succesiune de atribuiri de tipul "fapt=valoare", separate prin '\n'.

Un fapt are structura din secțiunea 3; valoare este un număr real ce corespunde coeficientului de certitudine sau valorii pentru un fapt static, respectiv pentru o variabilă.

De reținut: un fapt din fișierul de date trebuie să existe și în baza de reguli (condiție normală, avînd în vedere inutilitatea cazului contrar).

În interiorul delimitatorilor BACKQUOTE se iau în considerație toate caracterele din șir, inclusiv spațiile.

Excepție fac spațiile alăturate (în ambele sensuri) operatorilor relaționali >,<,<>,=.

5. Sistemul de meniuri

Meniul mediului oferă următoarele opțiuni:

- GOAL : permite introducerea unui fapt care va fi evaluat (valoare, în cazul unui fapt de tip variabilă, respectiv factor de certitudine, în cazul unui fapt de tip static) de către sistemul expert;
- HOW : după aflarea unui scop, explică modul de determinare a rezultatului;
- SHOW : meniu de tip POPUP, oferă următoarele opțiuni:
  - INTERNAL : afișează forma internă a unei reguli;
  - INIȚIAL : afișează forma inițială a unei reguli;
  - DICTIONARY: afișează toate faptele existente în sistem cu valorile lor (doar faptele atomice, nu și cele compuse (macro- uri);
  - FACT : afișează forma internă a unui fapt atomic sau a unei conjuncții de fapte;
- LOAD : meniu de tip POPUP, oferă următoarele opțiuni:
  - RULE : încarcă un fișier ce conține baza de reguli;
  - DB : încarcă un fișier ce conține baza de date;
- RESET : meniu de tip POPUP, oferă următoarele opțiuni:
  - ALL : reinițializează tot sistemul;
  - NEW FACTS : șterge din memorie rezultatele aflate în procesul de deducere a scopurilor anterioare;
  - ALL FACTS : șterge din memorie toate valorile faptelor păstrînd doar baza de reguli;
- QUIT : iese din sistemul expert, revenind în MS-DOS.

Toate opțiunile au atașată cîte o pagină de help, activată prin apăsarea tastei F1 cînd bara-cursor este poziționată pe opțiunea respectivă.

6. Structuri de date și algoritm de funcționare

Sunt folosite următoarele structuri de date:

- pentru reguli inițiale:

```
typedef struct regula inițială
{
   fnc f; // precondiția
   int v[10]; // indexul în vectorul de fapte al
   // conjuncțiilor din precondiție
   cnj c; // concluzia
   float cnf; // factorul de certitudine
}regini;
```

- pentru reguli interne:

```
class regula
{
   public:
      int Condiții [ MaxCond ];
                  // indexul în vectorul de fapte al
                  // conjuncțiilor din precondiție
      int NrCond;
```

```
                    // numărul de elemente valide
                    // în Condiții
        int fapt; // postcondiția
        int cîte; // cîte componente au fost
                    // evaluate la evaluarea regulii
        float CNF [ MaxCond ];
                    // factorul de certitudine al
                    // condițiilor față de regulă
        int DeCareRegAparțin [ MaxCond ];
                    // regula inițială din care
                    // provin conjuncțiile
        regula( ){NrCond=0;};
        float Evaluare();
        void init(int*,int,float*,int*);
        void Prezentare( );
    };
```

- pentru fapte:

```
    class fapt
    {
      public:
        int tipcomparație; // pentru faptevaluabil
        char* text; // numele faptului
        int Componente [ MaxComp ];
                    // indexul în vectorul de fapte al
                    // componentelor (pentru
                    conjuncții)
        int NrComp;
                    // numărul de elemente valide
                    // în Componente
        int Evaluat; // dacă e deja evaluat
        float CNF; // factor de certitudine
                    // sau valoare
        regula* RegAsoc;
                    // regula internă (dacă există)
        fapt(      )(RegAsoc=NULL;text=NULL;
        NrComp=Evaluat=CNF=0;}; șfapt( )
        {
           if(RegAsoc!=NULL) delete RegAsoc;
        }
        virtual float Evaluare(int) {return 0;};
        void init(regula*,char*,int*,int ,int ,float );
        virtual void afișare( );
        virtual void prezentare( );
};
class faptstatic:public fapt
{
        public:
        faptstatic( ) { };
        virtual float Evaluare(int);
};
class faptvariabilă:public fapt
{
    public:
        faptvariabilă( ) { };
        virtual float Evaluare(int);
        virtual void afișare( );
};
class faptevaluabil:public fapt
{
    public:
        faptevaluabil( ) { };
        virtual float Evaluare(int);
        void init(regula*,char*,int*,int ,int ,float ,int);
        virtual void prezentare( );
};
```

- variabile globale:

```
        regini reguli[100]; // reguli inițiale
        int stiva[100],contors,simbol,contorw,contordic=0;
        int contorreg=0,contorfapte=0;
                    // contoare și stiva pentru fnc
        char w[2000];
                    // buffer pentru citire fișier
        char dicționar[MaxFapt][100];
                    // dicționar
```

La încărcarea unei baze de reguli, se parcurg următoarele etape:

a) Se face analiza sintactică/lexicală a fișierului
- la întîlnirea unui fapt, analizorul lexical îl pune în dicționar și în vectorul de fapte (dacă e cazul), și întoarce poziția faptului în vectorul de fapte.
- analizorul sintactic completează vectorul de reguli (reguli inițiale), punînd precondițiile sub forma normal conjunctivă și concluziile sub forma de conjuncție;
- la întîlnirea unei erori lexicale sau sintactice, se specifică tipul de eroare și simbolul (cuvîntul cheie) așteptat și se abandonează încărcarea bazei de reguli.

b) Se parcurge vectorul de reguli inițiale și, pentru fiecare conjuncție din precondiția fiecărei reguli, se introduce conjuncția în vectorul de fapte (dacă este cazul) și se notează în regulă poziția pe care se află.

c) Pentru fiecare fapt atomic din vectorul de fapte se completează regula asociată faptului respectiv (dacă există), prin combinarea regulilor inițiale care conțin în concluzie faptul respectiv, completîndu-se cîmpurile corespunzătoare.

La introducerea unui scop, se lansează funcția de evaluare a faptului respectiv și se afișează rezultatul obținut. Deoarece orice fapt derivă din clasa abstractă "fapt", funcția apelată va fi specifică pentru fiecare tip de fapt (polimorfism).

Deoarece fiecare fapt (respectiv fiecare regulă internă) are în componență indexuri în vectorul de fapte, orice trimitere la un fapt sau la o funcție asociată se rezolvă într-un timp proporțional cu 0(1), ceea ce face ca funcționarea sistemului să fie foarte performantă.

Performanțele cresc și datorită faptului că orice conjuncție sau comparație este considerată fapt separat, ceea ce evită recalcularea unei conjuncții calculate anterior.

7. Transformarea în Forma Normal Conjunctiva (FNC)

O componentă esențială a optimizărilor efectuate de sistem este transformarea expresiilor în forma normal conjunctivă.

Forma normal conjunctivă (FNC) constă în reprezentarea expresiilor sub formă de sumă de produse (disjuncții de conjuncții). Această formă de reprezentare permite evaluarea mult mai rapidă a unei expresii, față de cazul obișnuit (evaluarea unei forme postfixate cu un analizor de tip stivă). Mai mult, în cazul de față, considerarea conjuncțiilor din FNC ca fapte de sine stătătoare permite evitarea evaluării repetare a unei conjuncții (deci se evită, atît refacerea unor înmulțiri, cît mai ales apelul inutil a unor funcții).

Structura de date folosită în acest scop este:

```
class cnj
{
    public:
        int v[10];      // componente = indexuri în
                        // vectorul de fapte
        int nr;         // număr de componente
        cnj operator*(cnj&);
        int comp(cnj&);
        void afișare( );
};
class fnc
{
    public:
        cnj v[10];      // componente
        int nr;         // număr de componenete
        fnc operator+(fnc&);
        fnc operator*(fnc&);
        fnc operator ș ( );
        void afișare( );
};
```

Pentru a transforma o expresie avînd paranteze, OR, NOT, AND, și numere, în FNC, se transformă expresia în forma poloneză (postfixată), înlocuind simbolurile reprezentînd fapte cu forme normal conjunctive cu o conjuncție, conjuncție formată dintr-un singur element.

Prin evaluarea formei poloneze cu un evaluator tip stivă, folosind operatorii definiți în clasa FNC, se obține ca rezultat o formă normal conjuctivă echivalentă cu expresia inițială, avînd deja efectuate o serie de reduceri (eliminarea conjuncțiilor de tip ...a AND ... AND NOT a...).

8. Exemplu

Fișier: reg2.txt - baza de reguli

if \deja plouă\ then \ iau umbrela \ cnf=100;
if \va ploua \ then \ iau umbrela \ cnf=100;
if \e înnorat \ and not \ e foarte frig \ then \ va ploua \ cnf=80;
if \e iarnă \ then \ e foarte frig \ cnf=60;
if not \ e soare \ and \ e zi \ and not \ e eclipsă \ then \ e înnorat \ cnf=100;
if \ e soare \ and \ e cald \ then \ iau umbrela \ cnf=95;
if \ temperatura > 25 \ then \ e cald \ cnf=100;
if \ temperatura < -5 \ then \ e foarte frig \ cnf=100;
if \ ora > 7 \ and \ ora < 19 \ then \ e zi \ cnf=100;
if \ buletin meteo: va ploua \ then \ va ploua \ cnf=30.

Fișier: reg2.bd - baza de date

\ ora \ =14
\ e cald \ =46

Descriem în continuare o sesiune de lucru simplă, folosind datele de test de mai sus:

- După intrarea în sistemul integrat se aleg din meniu opțiunile:
  Load rule (se introduce numele reg2.txt în cutia de editare)
  Load database file (se introduce numele reg2.bd).
- Se alege opțiunea Dictionary din meniu, se poate vizualiza forma internă a regulilor; cuvîntul cheie UNKNOWN arată că faptul pe care îl caracterizează nu a fost încă evaluat.

Se alege un scop: opțiunea Goal din meniu; se deschide o cutie de dialog în care se poate introduce scopul în limbaj natural (adică însuși textul faptului); să alegem, de exemplu, faptul

\ iau umbrela \,

acesta fiind în fond scopul principal al minisistemului expert implementat cu ajutorul celor două fișiere de test.

Prezentare teoretică a algoritmului BACKWARD-CHAINING:

- Algoritmul de backward-chaining găsește regula de indice 0, adică prima, în care \iau umbrela \ este concluzie; va încerca să evalueze precondiția ei, adică faptul \deja plouă \; deoarece nu există nici o regulă care să aibă acest nou goal în concluzie, sistemul va întreba direct pe utilizator cît de sigur este că deja plouă.
- Se alege ca nou goal scopul \va ploua \ pentru a afla din a doua regulă factorul de certitudine pentru scopul de bază (\iau umbrela \); acest nou goal se poate evalua cu ultima regulă, deci se va lua ca sub-goal precondiția acesteia, sistemul va cere CNF pentru \buletin meteo: va ploua \...

Funcționarea efectivă a implementării prezentate:

- Nu se folosesc regulile inițiale, cum ar apare din prezentarea comportării modelului didactic de mai sus; pentru eficiență s-a adoptat o formă internă a regulilor, care permite optimizarea puternică a calculelor; forma internă a regulilor a fost prezentată mai sus.
- În regulile aduse la forma internă, concluziile sunt reduse la un singur fapt, iar fiecare fapt apare în concluzia unei singure reguli sau chiar

în nici o concluzie; în consecință, evaluarea nu se mai desfășoară în cicluri care implică repetarea evaluărilor și deci viteza redusă, ci arborescent (fiecare sub-goal este un sub-arbore, în rădăcină e goal-ul cerut de utilizator).

Să descriem o posibilă conversație cu sistemul expert:

U= comenzi/date introduse de utilizator,

S= mesaje de la sistem.

U: - goal: iau umbrela

S: - Introduceți CNF pentru \deja ploua\:

U: - 50 (valoare care arată că nu se știe nimic despre aceasta)

S: - Introduceți CNF pentru \e soare\:

U: - 50 (valoare care arată că nu se știe nimic despre aceasta)

S: - Introduceți CNF pentru \e eclipsă\:

U: - 0 (utilizatorul nu vede vremea de afară, dar știe că nu este eclipsă, altfel ar fi scris în ziar);

S: - Introduceți CNF pentru \e iarnă\:

U: - why? (utilizatorul are dubii asupra logicii sistemului)

S: - Păi desigur ca pentru a aplica regula nr 3 !

Aici trebuie remarcat că deși sistemul expert nu folosește regulile în forma lor inițială, el simulează totuși bine acest lucru, răspunzînd la "WHY?" în termeni depinzînd de forma inițială. Acest lucru este esențial pentru ca interfața să fie "USER-FRIENDLY". În acest scop s-au memorat regula inițială și CNF-ul cu care provine fiecare fapt al precondiției fiecărei reguli din forma internă, în câmpurile DeCareRegAparțin[ ] și CNF[ ] din clasa Regula.

U: - 40 (destul de probabil, dar parcă mai degrabă nu)

S: - Valoare pentru \temperatură\:

Sistemul expert știe să ceară inițializarea acelor variabile despre care nu a putut afla nimic din baza de date sau din reguli.

U: - 4 (plus 4 grade)

S: - Introduceți CNF pentru \buletin meteo: va ploua\:

U: - 50 (nu putem să ne bazăm prea tare pe prognoza meteo)

S: Confidence for \iau umbrela\: 76% (deci s-o luăm)

U: HOW? (opțiune din meniu) - how did you find this out?

Sistemul deschide o cutie în care arată succesiunea pașilor de raționament, astfel: regula aplicată, concluzie, factor de încredere atribuit concluziei; de remarcat, din nou, că numerele regulilor sunt de fapt cele ale regulilor inițiale, deși nu acestea se folosesc efectiv, pentru a îmbina eficiența cu cerințele de interfață prietenoasă.

- regula 8 ⇒ \e zi\, 100 %
- regula 4 ⇒ \e înnorat\, 50 %
- regula 3,7 ⇒ \e foarte frig\, 24 %

Explicație: din regulile inițiale 3 și 7 sistemul a creat o singură regulă internă având "e foarte frig" drept concluzie. Aplicarea acestei reguli interne se raportează user-ului ca fiind aplicarea regulilor inițiale 3 și 7, astfel acesta va beneficia de o informare mult mai relevantă din punctul său de vedere.

- regula 2,9 ⇒ \va ploua\, 41 %
- regula 0,1,5 ⇒ \iau umbrela\, 76.88 %

În final, utilizatorul poate alege opțiunea Dictionary din meniu pentru a constata că toate sub-goal-urile calculate pe acest traseu au fost memorate în structurile interne ale sistemului. În caz de nevoie ele vor fi preluate de acolo deja calculate, ceea ce va spori considerabil viteza. Dacă se dorește renunțarea la această facilitate, se poate folosi meniul RESET.

Pentru detalii de funcționare, utilizatorul poate oricînd apăsa tasta F1, obținînd informații despre opțiunea curent selectată din meniu.

9. Comparații și concluzii

Sistemul expert prezentat permite o editare prietenoasă a regulilor și faptelor, precum și afișarea unor mesaje concludente în cazul erorilor lexicale/sintactice.

Din punct de vedere al performanțelor:

- viteza de preprocesare a regulilor este relativ scăzută, datorită alocărilor dinamice și a interpretorului pentru forme normal conjunctive:
- viteza în timpul utilizării bazei de reguli preprocesate este mare datorită optimizărilor, comparabile cu optimizările aduse de sistemul RETE pentru OPS5 [5].

Trebuie precizat că evitarea unor calcule inutile (evaluarea unor reguli pentru fapte deja certe, respectiv a unor elemente din conjuncții deja evaluate ca false) este dependentă de ordinea de scriere a regulilor în baza de reguli, respectiv a faptelor în conjuncții. Acest dezavantaj apare datorită modului de implementare a analizorului sintatic;

- numărul de reguli suportat este limitat de capacitatea memoriei convenționale. Datorită folosirii operatorului "new" al limbajului C++, acest număr este în această versiune 100.

De remarcat însă că este destul de ușor de realizat extinderea programului în vederea utilizării memoriei extinse sau chiar a virtualizării memoriei. Datorită modului de construcție a vectorului de fapte și a caracteristicii de localitate a regulilor unui sistem expert normal, se poate menține în memoria convențională doar o zonă contiguă din vectorul de fapte, zona fiind descărcată/încărcată la nevoie. În acest caz, datorită indexării, timpul de determinare a "segmentului" necesar este proporțional cu 0(1).

Prin acest procedeu, viteza scade, datorită introducerii unor teste înaintea utilizării oricărui fapt, dar numărul de reguli poate fi mărit de 100 de ori.

Datorită localității regulilor, timpul necesar pentru maparea segmentului necesar nu afectează sensibil performanțele generale.

- Lungimea maximă a unui fapt, numărul maxim de fapte etc. sunt limitate în versiunea actuală de existența unor variabile definite ca macrouri (#define) (s-a folosit în anumite cazuri alocare statică, datorită unor erori datorate operatorului "new" implementat în Borland C++ versiunea 2.0).
- Resurse hardware necesare:
  - calculator compatibil IBM-PC AT, cu cel puțin 640 Ko RAM.

Bibliografie

1. **GIUMALE, C.**: Note de curs, 1993.
2. **FLOREA, A.**: Note de curs, 1992.
3. **FIESCHI, M.**: Towards Validation of Expert Systems as Medical Decision Aids. In: International Journal of Bio-Medical Computing, vol.26, no.1,2, July 1990.
4. **NEGOIȚĂ, C.V.**: Expert Systems and Fuzzy Systems, the Bejamin/Cummings Publishing Company, Inc., New York, 1985.
5. **BROWNSTON, L., FARRELL, R., KANT, E., MARTIN, N.**: Programming Expert Systems in OPST. An Introduction to Rule-Based Programming, Addison-Wesley, 1985.