BRAND. Broad Research in Accounting, Negotiation, and Distribution
ISSN 2067-8177, Volume 1, Issue 1, 2010

# Accounting Knowledge Representation in PROLOG Language

*Bogdan Pătruț*
"Vasile Alecsandri" University of Bacău, Romania
bogdan@edusoft.ro

**Abstract**

This paper presents some original techniques for implementing accounting knowledge in PROLOG language. We will represent rules of operation of accounts, the texts of accounting operations, and how to compute the depreciation.

**Keywords:** accounting, knowledge representation, PROLOG, depreciation, natural language processing

## 1. Introduction

The PROLOG programming language[1] was developed specifically for problems of artificial intelligence. PROLOG is based on first order predicate calculus. In a program we must to define a set of predicates, and each predicates has its own clauses. The clauses represent facts and rules that make the predicate to be true. A rule is seen as an implication $A_1 \wedge A_2 \wedge ... \wedge A_n \rightarrow B$, written in reverse order: $B \leftarrow A_1 \wedge A_2 \wedge ... \wedge A_n$. For the operator „←" it is used the word „if" or the symbols „:-".

In our research, we have developed several modules that can be used in the PROLOG language representation of accounting knowledge. This paper is based on the result of [3], [4].

We used PROLOG for:

- represent the operating rules of the accounts;
- to implement an economic operation of the Romanian language directly in the formula accounts it represents;
- to represent some knowledge of a procedural nature, such as the calculation of depreciation.

## 2. Representation of rules of operation of accounts and creation of log book on accounting formulas

In order to achieve a log book must write all accounting items, so we need accounting formulas. These represents a codification of economic operations carried out at different times. An accounting analysis is essential, so we need a codification of rules of operation of accounts (see Table 1).

We define predicates for Prolog:

- determine the type of account (active or passive (capital or liability));
- link between an account and accordingly the patrimonial element;
- changes to economic elements of patrimony (increase or decrease);
- crediting or debiting an account.

Table 1 The rules of operations of accounts[2]

| | |
|---|---|
| 1 | If X is an active account, X reflects Y and Y increases, then X is debited. |
| 2 | If X is an active account, X reflects Y and Y decreases, then X is credited.. |
| 3 | If X is a liability account, X reflects Y and Y increases, then X is credited. |
| 4 | If X is a liability account, X reflects Y and Y decreases, then X is debited.. |

[1] More about Prolog and its uses in artificial intelligence systems can be found in [1].
[2] The table reffers to active, capital or liability accounts; source [2].

The predicates `active_account(integer)` and `liability_account(integer)` each have a single argument, which is an accounting symbol. The clauses of these predicates give us a picture of all types of accounts, in the general plan of accounts.

The predicate `reflects(integer, string)` concerns the link between an accounting symbol and the accounting matter which it reflects in accounting. For example, the clause `reflects(401,"Debts to suppliers")` means that the account 401 reflects debts to suppliers.

Predicates `increases(string, real, integer)` and `decreases(string, real, integer)` each have three arguments. First refers to a patrimony element, the second to an amount (of increasing/decreasing) of that account, and the third represents the time moment of the operation[3].

For example, the clause `decreases("Debts to suppliers", 300000, 2)` represents that the debts to suppliers decreases with 300.000 RON, at moment 2.

Finally, the predicates `is_debited(integer, real, integer)` and `is_credited(integer, real, integer)` specify the ammount (the second argument) which is debited/credited to an account (the first argument) when given time for the third argument.

The Program 1 is a representation in PROLOG of the rules of operation of accounts. The predicate `record` will generate the list of all accounting formulas (records).

Program 1. PROLOG representation of the rules of operation of accounts

```
predicates
      active_accont (integer)
      liability_account (integer)
      reflects(integer,string)
      increases(string,real,integer)
      decreases(string,real,integer)
      is_debited(integer,real,integer)
      is_credited(integer,real,integer)
      record(integer,integer,real,integer)
clauses
      active_account (626).
      active_account(531).
      liability_account(401).

      reflects(626,"Post and telecommunications expenses").
      reflects(401,"Suppliers").
      reflects(531,"Cash").

      is_debited(X,S,M) if active_account(X),
                             reflects(X,Y), increases(Y,S,M).
      is_debited(X,S,M) if liability_account(X),
                             reflects(X,Y), decreases(Y,S,M).
      is_credited(X,S,M) if liability_accont (X),
                             reflects(X,Y), decreases(Y,S,M).
      is_credited(X,S,M) if liability_account(X),
                             reflects(X,Y), increases(Y,S,M).

      record(D,C,S,M) if is_debited(D,S,M), is_credited(C,S,M),
            write(M,". ",D,"=",C," Amount: ",S,"  lei "),nl.

      increases("Post and telecommunication expenses", 300000,1).
      increases("Suppliers",300000,1).
      decreases("Suppliers", 300000,2).
      decreases("Cash", 300000,2).
goal
      record(X,Y,S,M), fail.
```

[3] Obviously, when time could be more than a whole number, but we considered the type integer to simplify exposure.

## 3. Translating the accounting operations from natural language to equations

In computer science literature, the procedure of understanding the text of an accounting operation can be called "parsing". It consists by a syntactic analysis, followed by an "interpretation" and an "execution" of the analyzed text.

Thus, we propose that by providing a description of our operations and accounting system, to receive the appropriate accounting equation.

The program we developed is below. It underpins an intelligent agent that may associate an accounting equation to any sentences of natural language, freely expressed, which describes the operation of accounts.

For example, we would like to get answers to the right for phrases like those on the left (Table 2).

Table 2. Examples of translations from Romanian language into accounting equations

| | |
|---|---|
| Se subscrie un capital social în valoare de 3000 lei.<br>(Subscribed share capital is worth 3000 lei.) | 456 = 1011, 3000 lei |
| Se depune, sub formă de aport în natură, un teren în valoare de 2000 lei.<br>(Is deposited as a contribution in kind, a field value of 2000 lei.) | 2111 = 456, 2000 lei |
| Se înregistrează capitalul social vărsat de 3000 lei.<br>(Record paid up share capital 3,000 lei.) | 1011 = 1012, 3000 lei |

How can we achieve this? First will be a transformation of the sentences from their original string form, into a list of strings, knowing that in PROLOG we can implement effective predicates for working with lists. The predicate *make_list* will transform the original text of the accounting operation into a list of words, i.e. *make_list* separates the original phrase into words. Then, the list of words will draw the "word" which is the numerically amount (using the predicate *get_amount*). This will be determined easily, because the amount starts with a digit. For things to be simplified, we will remove the words that are not considered relevant for that phrase. Thus we will not consider any prepositions or other words that appear in a given dictionary. Predicate which simplifies a list goes through the list and examines each word. Every word is searched in a dictionary (using the *dictionary* predicate, with its clauses) and if there is, then it is replaced by its standard version, all taken from the dictionary. Other variants are considered synonymous. In the event that that word isn't in the dictionary, then it will not appear in the simplified list. After deletion of the "insignificant" words, there is an alphabetical ordering of words left, made by the *sort* predicate, which implements (recursively) a process of sorting by insertion. Thus, phrases like "se varsă capitalul" or "capital vărsat" have the same meaning.

Finally, it is called the *translation* predicate, which carries out the desired translation, yielding the accounting entry.

E.g.: consider the phrase "se subscrie un capital social în valoare de 3000 lei.".

After applying the first transformation (*make_list*), we obtain the list ["se", "subscrie", "un" "capital", "social", "în", "valoare", "de", "3000", "lei"]. After that, the predicate *get_amount* will select "3000" (lei) as amount, which will be displayed (at the end of execution of the program) along with the rest of accounting entry.

Applying predicate *simplify* to the list of words obtained before, just stick with the list ["subscrie", "capital"], because it chooses inarticulately standard "capital" instead of "capitalul" and other words that don't appear in our dictionary will be considered insignificant. Then this list is ordered alphabetically, resulting in the final list ["capital", "subscrie"], which, according to the predicate *translation* leads to the accounting entry 456 = 1011, with the amount of 3000 lei.

The program below is purely demonstrative (program 2.). This system including this fragment has included a database containing a multitude of variants of "translation" and other kind of sentence processing, for exceptional cases or for optimization.

Program 2. Demonstrating how to implement a PROLOG translator that converts accounting operations from Romanian language to accounting entries

```
domains
  accont=integer
  slist=string*
predicates
  dictionary(string,lista)
  member(string,lista)
  simplify(lista,lista)
  mak_list(string,lista)
  translate(lista,cont,cont)
  transform(string,cont,cont,integer)
  repeat
  run
  sort(lista,lista)
  insert(string,lista,lista)
  get_ammount(lista,integer)
clauses
  dictionary(terenuri,[terenuri,teren,suprafata,suprafete]).
  dictionary(constructii,[constructii,constructir,cladire,cladiri]).
  dictionary(subscrie,[subscrie,subscris]).
  dictionary(aport,[aport,contributie,adus]).
  dictionary(varsa,[varsa,varsat]).
  dictionary(capital,[capital, capitalul]).
  dictionary(rezerve,[rezerve]).
  dictionary(incorporeaza,[incorporeaza]).
  dictionary(rascumparare,[rascumparare]).
  dictionary(actiuni,[actiuni]).
  translate([capital,subscrie],456,1011).
  translate([capital,varsa],1011,1012).
  translate([aport,terenuri],2111,456).
  translate([aport,constructii],2121,456).
  translate([incorporeaza,rezerve],1061,1012).
  translate([actiuni,rascumparare],502,5121).
  member(C,[C|_]) if !.
  member(C,[_|Rest]) if membru(C,Rest).
  make_list("",[]) if !.
  make_list(S,L) if fronttoken(S,Cuv,S1), make_list(S1,L1),L=[Cuv|L1].
  get_amount([],0) if !.
  get_amount([X|_],Suma) if str_int(X,Suma), !.
  get_amount([_|Rest],Suma) if get_ammount(Rest,Suma).
  simplify([],[]) if !.
  simplify([Cap|Rest],L) if
         dictionary(CapStandard,Sinonime), member(Cap,Sinonime),
         simplificy(Rest,L1),L=[CapStandard|L1],!.
  simplify([_|Rest],L) if simplifica(Rest,L).
  transform(Propozitie,CD,CC,Suma) if
         make_list(Propozitie,L), get_ammount(L,Suma),
         simplificy(L,L1), sort(L1,L2), !, translate(L2,CD,CC).
  sort([X|Rest],S) if sort(Rest,R),insert(X,R,S).
  sort ([],[]).
  insert(X,[Y|Rest],[Y|Rest1]) if X>Y, insert(X,Rest,Rest1).
  insert(X,Rest,[X|Rest]).
  repeat.
  repeat if repeat.
  run if repeat, write(">>"), readln(Propozitie),
         transform(Propozitie,ContDebitat,ContCreditat,Suma),
         write(ContDebitat," = ",ContCreditat,", ",Suma," lei"), nl, fail.
```

## 4. Calculation of diminishing depreciation of fixed assets, without influence of the obsolescence

The essential characteristic of assets (fixed assets) is repeated participation over several years in economic and financial activity of the enterprise. They gradually exhausted as aging and gradual recovery in the form of depreciation.

In terms of accounting, depreciation has more than one interpretation [5] [6]:
a) Finding the lost book value of assets suffered as a result of their depreciation time (physical wear and obsolence);
b) The process of property transfer or sharing of costs, during the economic year of use of depreciable assets;
c) Source of funding for renewal of assets;
d) In International Accounting Standards, depreciation is defined as "systematic allocation of the depreciable amount of an asset over its useful life" [7] [8] [9].

All tangible and intangible assets, with some exceptions, are the subject of depreciation. Criteria used to assess depreciation are the forms which takes lifetime, namely:
a) the period during which it is estimated that the company will use the asset subject to amortization, which can be identified with the normal operation for fixed assets, depreciation or narrow down by law, if intangible assets;
b) the volume of production achieved, i.e. the units of production or similar units that are expected to be obtained by using business asset.

The methodology of calculation of depreciation depends on the criterion used to evaluate depreciation, depreciation regime adopted etc. If depreciation after the normal operating, the computing elements are:
a) AC = annual depreciation rate, determined by the relation AC = 100/DU, by DU understanding normal use during the years covered in the catalog of useful life and classification of assets. If the revaluation of fixed assets, based on legislation, annual depreciation percentage rate (AC) is calculated by 100 compared to the normal duration of use remaining. Share so calculated will apply the remaining amount and value date of entry;
b) VI = depreciable value, i.e. value of the fixed assets, or if necessary, update the remaining book value, less the estimated residual value (sometimes can be ignored in the calculation of depreciation).

Depreciation scheme used depends on how the asset is expected to bring economic advantages to the undertaking. In the case of linear depreciation, operating costs are included (uniformly) some fixed amount, set in proportion to the number of years of normal use life of fixed assets.

In the case of diminishing depreciation, amortization process is accelerating in the first years of operation of the asset by multiplying the linear damping rate with one of the following factors: 1.5 where the normal duration of use is between 2 and 5 years, 2.0 if it is between 6 and 10 years respectively for a duration of 2.5 normal use of the asset more than 10 years. Then, switch to linear depreciation for the remaining years [6]. In our program (program 3) the *compute_coeficient (real, real)* sets the multiplier. The first argument is the duration of operation, and the second multiplier. In Prolog, we have:

```
compute_coeficient(D,1.5) if 2<=D, D<=5,!.
compute_coeficient(D,2) if 5<D, D<=10, !.
compute_coeficient(D,2.5) if D>10.
```

Coefficient calculated with this predicate will be sent (as an argument, namely that the argument of the last position (CD)) to the *depreciation* predicate. The *depreciation* predicate has four actual type arguments. This was chosen because this type of data type integer was not sufficiently comprehensive to represent large numbers (in the millions). Predicate is used to calculate depreciation and annual depreciation and remaining value display end. The header of this predicate is *depreciation(S, N, A, R, CD)*. The first argument is the amount remaining amortized

(S), the second is the number of years remaining (N), the third annual depreciation will be even diminishing (A) which will be displayed along with the remaining (R).

1) In the case of diminishing depreciation, without the influence of the obsolescence, (AD1) in first year it is applied the depreciation rate to the input value. For subsequent years, the same rate, but each times the remaining value. This calculation continues operating until the resulting annual depreciation is equal to or less than linear annual depreciation calculated for the remaining period of operation. Depreciation predicate is recursive, so that after calculation of depreciation for the current year, it comes to calculating the following year and so on. The first possibility, we have the following depreciation clause predicate:

```
depreciation(S,N,A,R,CD) if
  A1=S*CD/100,A2=S/N, A1>=A2, A=A1, R=S-A,
  write(": A=",A,", R=",R), nl,N1=N-1,
depreciation(R,N1,_,_,CD).
```

Therefore, A1 is the variation diminishing value depreciation and amortization in the amount of variant A2 is linear.

Comparing A1 to A2, if A1> = A2, the variant is still diminishing, in which A is A1 and his place will be taken by R S = SA.

2) Since the A1 = A2, until the expiration of normal service to switch to annual depreciation linear to the end.

Thus we have two clauses for predicate depreciation (second clause is the last year, end table):

```
    depreciation (S,N,A,R,CD) if
    A1=S*CD/100, A2=S/N, A1<A2, A=A2, R=S-A,
    write(". A=",A,", R=",R), nl, N1=N-1, R>0,
depreciation(R,N1,_,_,CD).
depreciation (S,N,A,R,CD) if
    A1=S*CD/100, A2=S/N, A1<A2, A=A2, R=S-A, R=0, !.
```

To illustrate, consider an asset with value input VI = 30000000 during normal use lei DU = 10 years. Therefore, the depreciation will be linear AC = 100/10 = 10% and 10% diminishing balance depreciation rate x 2 = 20% (coefficient is 2). 4.3 Prolog program will display the following:

```
DIMINISHING DEPRECIATION
WITHOUT THE INFLUENCE OF OBSOLESCENCE
Give the input value: 30000000
Give the normal usage in years:
10
: A=6000000, R=24000000
: A=4800000, R=19200000
: A=3840000, R=15360000
: A=3072000, R=12288000
: A=2457600, R=9830400
: A=1966080, R=7864320
. A=1966080, R=5898240
. A=1966080, R=3932160
. A=1966080, R=1966080
. A=1966080, R=0
Press the SPACE bar
```

We prefixed with the ":" years when diminishing depreciation rate is applied, respectively. " during the linear version is used.

Program 3. Prolog representation of the procedure for calculating the diminishing depreciation, without the influence of the obsolescence

```
predicates
    depreciation (real,real,real,real,real)
    % suma, nr. de ani ramasi, ad, val ramasa, cota de am. degresiva
    calcul_coeficient(real,real)
    % durata de functionare => coeficientul de multiplicare
clauses
    depreciation (S,N,A,R,CD) if
        A1=S*CD/100,A2=S/N, A1>=A2, A=A1, R=S-A,
        write(": A=",A,", R=",R), nl, N1=N-1,
        depreciation (R,N1,_,_,CD).
    depreciation (S,N,A,R,CD) if
        A1=S*CD/100, A2=S/N, A1<A2, A=A2, R=S-A,
        write(". A=",A,", R=",R), nl, N1=N-1, R>0, amortizare(R,N1,_,_,CD).
    depreciation (S,N,A,R,CD) if
        A1=S*CD/100, A2=S/N, A1<A2, A=A2, R=S-A, R=0, !.
    compute_coeficient(D,1.5) if 2<=D, D<=5,!.
    compute_coeficient(D,2) if 5<D, D<=10, !.
    compute_coeficient(D,2.5) if D>10.
goal
    makewindow(1,7,0,"",0,0,25,80), clearwindow,
          write("DIMINISHING DEPRECIATION"),nl,
          write("WITHOUT THE INFLUENCE OF OBSOLESCENCE"),nl,
    write("Give the input value: "), readreal(VI),
          write("Give the normal usage in years:"), readreal(DU),
    CA=100/DU, compute_coeficient(DU,Coef), CD=CA*Coef,
    depreciation(VI,DU,_,_,CD).
```

## 5. Conclusions

We consider it necessary accounting knowledge representation in intelligent systems using known methods: the calculation of-order predicates, semantic networks and production rules, but is useful and discover new ways of representation. Knowledge includes rules, conceptual representations, ideas, principles, abstractions, and how the representation and organization of knowledge is an essential element of any intelligent system.

Knowledge representation is to find and achieve a correspondence between reality and a symbolic domain that allows reasoning. Therefore, the representation of knowledge is captured essential elements of a problem domain.

In our opinion, Prolog can be used to represent many accounting knowledge, including procedural (computer) knowledge. In our research, we have developed several modules that can be used in Prolog representation of accounting knowledge, such as implementation of economic operations (written in natural language) directly in the formula book, or representation of depreciation calculation.

## 6. References

[1] Bratko, I. 2006, *Prolog Programming for Artificial Intelligence*, Addison Wesley; 4rd edition
[2] Zaharie, D., Năstase, P., Albescu, F., Bojan, I., Mihai, F., Covrig, L., 1999, *Sisteme expert. Teorie şi aplicații*, Editura Dual Tech, Bucureşti
[3] Pătruț, B., 2004, *Analiza şi înțelegerea textelor operațiunilor contabile, folosind limbajul Prolog*, în Informatica Economică, nr. 3(23)-2002, A.S.E. Bucureşti
[4] Pătruț, B., 2003, *Tehnologia Microsoft Agent pentru învățarea regulilor de funcționare a conturilor*, in Informatica Economica, nr. 4(28)/2003, A.S.E. Bucureşti
[5] Colasse, B., 2005, *Contabilitate generală*, Editura Moldova Iaşi
[6] Pătruț, V., Rotilă, A., Darie, V., Drehuță, E., Gorbănescu, C., 1998, *Manualul expertului contabil şi al contabilului autorizat*, Editura Agora, Bacău

[7] Epstein, B. J., Mîrza, A. A., 2005, *Interpretarea şi aplicarea Standardelor Internaţionale de Contabilitate şi Raportare Financiară*, Wiley IFRS
[8] * * *, 2000, *Standardele Internaţionale de Contabilitate 2000*, International Accounting Standards Committee, London UK, Editura Economică
[9] Duţescu, A., 2001, *Ghid pentru înţelegerea şi aplicarea Standardelor Internaţionale de Contabilitate*, Editura CECCAR, Bucureşti