ALLO - LIMBAJ ALGEBRIC PENTRU OPTIMIZARE LINEARĂ

dr. ing. Neculai Andrei
mat. Gheorghe Borcan

Institutul de Cercetări în Informatică

Rezumat: Lucrarea prezintă limbajul de modelare matematică ALLO, proiectat şi implementat de către Institutul de Cercetări în Informatică, Bucureşti, România, în cadrul Laboratorului de Modelare Matematică şi Optimizare. Limbajul ALLO este o componentă de baza a Sistemului Avansat de Modelare şi Optimizare - SAMO funcţional pe sistemele WINDOWS.
în câteva centre puternice din lume au fost implementate limbaje de modelare recunoscute: AIMMS, AMPL, GAMS, LINDO, LPFORM, LPL, LPMODEL, MODLER, MPL, PLAM. Unele dintre acestea au fost integrate în sisteme de modelare şi optimizare care sunt functionale pe diverse computere sau platforme de calcul.
Limbajul ALLO este dedicat modelării matematice lineare şi este un limbaj apropiat de limbajul ştiinţific preponderent algebric ceea ce îl face uşor de asimilat. Acesta permite definirea de variabile auxiliare (tip: întreg, real, domeniu, fişier), variabile ale modelului, variabile funcţii obiectiv sau restricţii. De asemenea, orice variabilă (excepţie cele tip domeniu sau fişier) poate fi chiar de forma masiv de variabile simple. Variabilele auxiliare pot avea condiţii de validare şi pot să fie iniţializate cu valori ce se regăsesc în fişiere secundare de date. Limbajul care dispune de blocurile repetitive FOR şi SUM, permite concentrarea definiţiilor alcătuite din expresii numerice sau simbolice. Limbajul permite definirea unor domenii dinamice, care sunt calculate în momentul evaluării instrucţiunii din care fac parte lucru deosebit de util. în limbajul ALLO, se pot scrie prototipuri complexe, care, prin personalizare (realizată prin schimbarea datelor din unele fişiere secundare care nu afectează fişierul principal prototip) conduc la modele concrete rezolvate cu un anumit solver. Translatorul ALLO generează un model în forma numită Standard MPS, ce este acceptat ca intrare de către solverele recunoscute în domeniul optimizării liniare.

Cuvinte cheie: modelare matematică, model linear, programare lineară, optimizare, standard MPS, gramatică formală, gramatica LL(1), limbaj asociat unei gramatici formale, limbaj specializat pe problemă, analiză lexicală, analiză sintactică, analiză semantică, scaner, parser, generator de cod, compilator, translator, interpretor.

1. ALLO Limbaj Algebric pentru Optimizare Lineară

Limbajul ALLO este un limbaj specializat pentru modelarea matematică lineară. El permite modelatorului să scrie cu uşurinţă modele lineare, dacă se cunoaşte o formulare algebrică în care toate constantele implicate în construcţie sunt cunoscute.

În limbajul ALLO, modelatorul poate să-şi definească variabilele auxiliare, variabilele modelului, funcţiile modelului precum şi restricţiile acestuia. Variabilele auxiliare pot primi valori în momentul definirii şi pot fi supuse unor condiţii de validare. Valorile sunt în fişierul sursă principal sau în diverse fişiere secundare (cu date citate în fişierul sursă principal) ceea ce permite construirea de prototipuri.

Printre variabilele auxiliare regăsim domenii întregi sau la domenii reale care pot fi cunoscute a priori sau numai în faza de " execuţie", a unor instrucţiuni de definire variabile model, funcţii model sau restricţii model.

Modelatorul poate defini blocuri repetitive FOR şi SUM, care permit iniţializarea unor masive auxiliare, scrierea unor condiţii concentrate de validare variabile auxiliare de tip masiv, scrierea unor condiţii concentrate de mărginire variabile model de tip masiv, definirea unor expresii simbolice concentrate, care participă la construirea şi definirea funcţiilor obiectiv şi a restricţiilor scalare sau de tip masiv.

Tipurile de date auxiliare sunt: întreg, real, domeniu intreg, domeniu real, tipul indice şi tipul nume de fişier. Un nume de fişier ia ca valoare o specificaţie de fişier DOS şi poate fie citat în celelalte instrucţiuni de definire variabile auxiliare (când dorim să-i atribuim valoarea/valorile din fişierul concret pe care îl reprezintă). Definirea unui nume de fişier (secundar cu date) poate fi făcută oriunde în secţiunea variabilelor auxiliare. Când este referit în definiţia unei variabile auxiliare înseamnă că aceasta va primi valori iniţiale din fişierul în cauză. Revenirea dintr-un astfel de fişier pe textul din fişierul principal se produce ca urmare a satisfacerii comenzii de iniţializare a variabilei auxiliare respective sau ca urmare a unei erori sintactice sau semantice.

Variabilele auxiliare de tip întreg sau real pot fi organizate în masive de variabile simple de acelaşi fel. După necesitate şi variabilele model, funcţiile obiectiv sau restricţiile model pot fi grupate respectiv, în masive. Un tip special de variabilă auxiliară (care nu trebuie să fie definit explicit) este variabila indice utilizată în instrucţiunile cu blocuri FOR sau SUM.

Limbajul ALLO este un limbaj cu libertate maximă faţă de zone albe formate din linii vide, taburi, spaţii şi comentarii care pot să se desfăşoare pe una sau mai multe linii sursă.

Limbajul ALLO nu este senzitiv la litere mari sau mici. El impune restricţii de utilizare numai în scopul precizat asupra unui set de cuvinte cheie, care pot fi scrise cu litere mari/mici sau în combinaţie după cum agreează modelatorul.

Translatorul ALLO este un task independent, integrat în sistemul de optimizare lineară SAMO, funcţional pe sistemele WINDOWS. Sistemul SAMO dispune şi de un manager de fişiere, un editor de fişiere text şi un optimizator pentru rezolvarea problemelor lineare. Rezultatul translatării unui model ALLO este un model echivalent Standard MPS recunoscut de solverele utilizate curent în domeniu.

Limbajul ALLO a apărut dintr-o necesitate practică. Cu ajutorul lui am scris relativ uşor modele de mari dimensiuni pe care nu le-am fi putut scrie altfel decât cu un efort considerabil. ªi, dacă mai avem în vedere etapa de perfecţionare a oricărui model de asemenea anvergură, ne dăm mai uşor seama de rolul unui astfel de instrument.

2. Un exemplu de model scris în limbajul ALLO

Considerăm un atelier care utilizează următoarele materii prime de bază: fier, nichel, cupru pentru a produce: piuliţe, şuruburi, nituri, cuie şi şaibe. Procesul se desfăşoară în patru perioade de producţie, iar maximul de producţie pe o perioada este în valoare de 140.5. Unităţile de materii prime pe unitatea de produs finit este dat de tabelul de mai jos:

	piuliţe	şuruburi	nituri	cuie	şaibe
fier	0.79	0.85	0.89	0.86	0.58
nichel	0.19	0.10	0.07	0.14	0.20
cupru	0.02	0.05	0.04	0.00	0.22

Profitul estimat (dacă valoarea este pozitivă) sau costul (dacă valoarea este negativă) produsului final şi pe fiecare perioadă de timp este dat de tabelul de mai jos:

	Perioada 1	Perioada 2	Perioada 3	Perioada 4
piuliţe	1.73	1.80	1.60	2.30
şuruburi	1.82	1.93	1.72	2.51
nituri	1.87	1.96	1.65	1.94
cuie	1.78	1.84	1.69	1.98
şaibe	1.05	1.16	0.98	1.37

Consumul maxim permis pe întreg procesul de producţie din cele patru perioade si pentru fiecare tip de materie primă trebuie să fie în stoc la începutul primei perioade. Se cunoaşte preţul de stocare pe perioadă şi pe unitate de material. Pierderile estimate (dacă valoarea este pozitivă) sau costurile impuse (dacă valoarea este negativă) pe materiile prime după ultima perioadă sunt, de asemenea, precizate. Aceste date sunt în tabelul:

	fier	nichel	cupru
Consum maxim	46.8	7.32	5.16
Preţ stocare	0.03	0.025	0.023
Pierderi/costuri	0.02	-0.01	-0.02

Problema care se pune este determinarea unui plan de producţie pentru fiecare perioadă de timp şi cantitatea de materiale din stoc la începutul fiecărei perioade în aşa fel ca valoarea obţinută (la finalul celor patru perioade) din profitul estimat, minus costurile de stocare plus valoarea materialelor rămase neutilizate sa fie maximă. Aprovizionarea cu materii prime este permisă numai la începutul ciclului de producţie, format din cele patru perioade. De aceea, problema stocurilor de materii prime şi a penalizărilor este prezentă şi deosebit de importantă în realizarea optimului.

O reprezentare ALLO a modelului menionat

```
MODEL Plan_Producţie
/* planprod.alo-----------------------------------------------------------------------------------------*/
/*                                                                                                       */
/* Sursa: Fourer, R., Gay, D.M.,Kernighan, B.W., AMPL: A Mathematical Programming Language. Technical*/
/* Report 87-03, Department of Industrial Engineering and Management Sciences, Northwestern University,*/
/*Evanston, Illinois 60201, January 1987, pp.1-63                                                       */
/*-----------------------------------------------------------------------------------------------------*/
FILE fpp = "planprod.dat"
   INTEGER
      nrfprod READ fpp IS nrfprod > 0; /* Număr produse finite */
      nrrmat READ fpp IS nrrmat  > 0; /* Număr materii prime */
      nrperp READ fpp IS nrperp  > 0  /* Perioade pe ciclu pr. */
   RANGE
      fprod    = [1,nrfprod  ];      /* Domeniu coduri produse finite */
      rmat     = [1,nrrmat   ];      /* Domeniu coduri materii prime */
      perp     = [1,nrperp   ];      /* Domeniul perioade producţie */
      perpext = [1,nrperp+1]         /* Domeniu extins perioade prod. */
   REAL
      prodmax READ fpp IS prodmax > 0.0;
         /* Producţia maximă permisă pe orice perioadă de producţie */
      ump [rmat,fprod] READ fpp IS FOR [r IN rmat,f IN fprod] ump [r,f] >= 0.0;
      /* ump[r,f] Unităţi materie primă r pentru o unitate de produs f */

      profdc [fprod,perp] READ fpp;
      /* Profit estimat (dacă >=0.0)/costul impus (dacă <0.0) */
      /* al produselor finale şi pe fiecare perioadă de timp */

      stinimax [rmatţ READ fpp IS FOR [r IN rmatţ stinimax[rţ>=0.0;
      /* Stocul iniţial maxim permis pe fiecare materie primă */

      storcost [rmatţ READ fpp IS FOR [r IN rmatţ storcost [rţ >= 0.0;
      /* Costuri stocare pe unitate materie primă, pe orice perioadă */

      rvdcost [rmatţ READ fpp
         /* Valorile reziduale estimate (dacă >=0.0)/costul impus (dacă */
         /* < 0.0) al materiilor prime după ultima perioadă a ciclului */
VARIABLES
   xprod [fprod,perpţ; /* Unităţi de produse finite pe fiecare perioadă */
   xsmat[rmat,perpextţ IS FOR[r IN rmatţ xsmat[r,1ţ <= stinimax[rţ
         /* Stocul materii prime necesar la startul fiecarei perioade */
         /* La începutul primei perioade trebuie respectată condiţia */
OBJECTIVES
   profit IS profit:= SUM[f IN fprod,p IN perpţ(profdc[f,pţ*xprod[f,pţ) -
                     SUM[r IN rmat,p IN perpţ (storcost[rţ*xsmat[r,pţ) +
                     SUM[r IN rmatţ (rvdcostşrţ*xsmat[r,nrperp+1ţ)
         /* Profitul estimat pe întreg ciclul de producţie este profitul pe */
   MAXIMIZE profit
CONSTRAINTS
   limprod [perpţ IS /* Limita producţiei totale pe fiecare perioadă */
      FOR [p IN perpţ limprodşpţ:= SUMşf IN fprodţ(xprod şf,pţ) <= prodmax;
   pmbal [rmat,perpţ IS /* Balanţa materială a procesului pe perioade */
      FOR [r IN rmat, p IN perpţ pmbalşr,pţ:=
            xsmat[r,p+1ţ = xsmat[r,pţ - SUM[f IN fprodţ(umpşr,fţ * xprodşf,pţ)
END
/* fpp = planprod.dat - fişier secundar cu date de personalizare prototip */
/* nrfprod */   5
```

```
/* nrrmat */  3
/* nrperp */  4
/* prodmax */ 140.5
/* ump şrmat,fprodv */
                          /* piuliţe   şuruburi   nituri   cuie   şaibe */

   /* fier  */              0.79,     0.85,      0.89,    0.86,   0.58,
   /* nichel */             0.19,     0.10,      0.07,    0.14,   0.20,
   /* cupru  */             0.02,     0.05,      0.04,    0.00,   0.22;

/* profdc [fprod,perp] */ /*   1        2        3        4      */
   /* piuliţe */            1.73,     1.80,     1.60,    2.30,
   /* şuruburi*/            1.82,     1.93,     1.72,    2.51,
   /* nituri  */            1.87,     1.96,     1.65,    1.94,
   /* cuie    */            1.78,     1.84,     1.69,    1.98,
   /* şaibe   */            1.05,     1.16,     0.98,    1.37;

                          /* fier     nichel    cupru*/

   /* stinimax [rmat] */    46.8,     7.32,     5.16;
   /* storcost [rmat] */    0.03,     0.025,    0.023;
   /* rvdcost [rmat] */     0.02,     -0.01,    -0.02;

   /* Eof planprod.dat */
```

Un extras din raportul optimizator din rezolvarea aceastei probleme

```
Name of the problem: Plan_producţie
Maximize the function: profit

Synopsis of original matrix
===================

No. constraints                 =  16

No. structural variables .      =  35

No. non-zeros elements .        =  100

Density of non-zeros in matrix  = 17.86%

No. unity elements .            =  44

Density of unity elements       = 7.86%

Row section

No. equations .                 =  12

No. less than or equal .        =  4

Column section

No. normal variables            =  32

No. upper bounded variables     =  3

Feasible Solution found in        0 iterations
Optimal  Solution found in       25 iterations
Objective function value        = 131.7778

NoCol  Name  Pos   Value       InputCost  LowerLimit  UpperLimit  ReducedCost
1 xprod1  lb   .                .1730D+01     .          none      -.52336D+00
suprimarea noastră
7 xprod7  lb   .                .1720D+01     .          none      -.76085D+00
8 xprod8  bs  .550588D+02       .2510D+01     .          none        .
9 xprod9  lb   .                .1870D+01     .          none      -.66540D+00
suprimarea noastră
```

20 xprod20	lb	.	.1370D+01	.	none	-.37298D+00
21 xsmat1	ub	.468000D+02	-.3000D-01	.	.4680D+02	.28158D+01
22 xsmat2	bs	.468000D+02	-.3000D-01	.	none	.
23 xsmat3	bs	.468000D+02	-.3000D-01	.	none	.
24 xsmat4	bs	.468000D+02	-.3000D-01	.	none	.
25 xsmat5	lb	.	.2000D-01	.	none	-.29158D+01
26 xsmat6	bs	.550588D+01	-.2500D-01	.	.7320D+01	.
27 xsmat7	bs	.550588D+01	-.2500D-01	.	none	.
28 xsmat8	bs	.550588D+01	-.2500D-01	.	none	.
29 xsmat9	bs	.550588D+01	-.2500D-01	.	none	.
30 xsmat10	lb	.	-.1000D-01	.	none	-.11000D+00
31 xsmat11	bs	.275294D+01	-.2300D-01	.	.5160D+01	.
32 xsmat12	bs	.275294D+01	-.2300D-01	.	none	.
33 xsmat13	bs	.275294D+01	-.2300D-01	.	none	.
34 xsmat14	bs	.275294D+01	-.2300D-01	.	none	.
35 xsmat15	lb	.	-.2000D-01	.	none	-.11200D+00

Acum putem să tragem următoarele concluzii:

1. La începutul ciclului de producţie al celor patru perioade trebuie să plecăm cu următoarele stocuri de materii prime:

	Cantitate	Maxim admis
fier	46.8	46.8
nichel	5.5	7.32
cupru	2.75	5.16

2. Nu producem nimic în primele trei perioade ale ciclului, iar în a patra perioadă producem numai şuruburi în cantitatea de 55.05 din xprod8 (de fapt xprod$2,4$ liniarizată cu procedura: ultimul indice se schimbă cel mai repede după care penultimul etc). Aceasta este o soluţie corectă a modelului pus în discuţie cu datele şi pretenţia expuse. în practică, atelierul nu poate sta trei perioade pentru ca apoi să fabrice un singur produs. Managerul găseşte soluţii de renunţare la a obţine profit maxim (cu orice preţ) pentru a asigura continuitatea procesului de producţie fiind obligat să introducă noi condiţii în model. O condiţie poate cere minim producţie în fiecare perioadă. O alta ţine de asigurarea unui minim din fiecare tip de produs final precum şi de satisfacerea unor cereri prezente (chiar dacă nu complet profitabile) cu scopul de a nu pierde clientela. Toate acestea ţin de strategia de existenţă şi de profitabilitate a atelierului pe un orizont mai mare decât a celor patru perioade de productie prezente în model. În final, strategia şi chiar tactica de moment pot fi modelate în anumite condiţii care îmbogăţesc modelul expus şi pot duce la câteva variante de funcţionare pe parcursul celor patru perioade de studiu. Pe baza acestora, managerul poate să decidă ce cale are de urmat.

3. Elementele limbajului ALLO

3.1 Setul de caractere ALLO

Clasa	Cod ASCII	Simbol extern	Descrire
Litere	065-090 097-122	A - Z a - z	Litere Mari Litere Mici
Cifre	048-057	0 - 9	Cifre zecimale
Caractere de control	013 010 013,010	<CR> figurativ <LF> <CR><LF>	Cariage Return Line Feed Sfârşit linie text sursă
Caractere	009	<TAB>	Tab orizontal echivalat vizual cu unul sau

pentru spaţiere	032	<SP>	mai multe spaţii Caracterul spaţiu
Comentarii	047,042 042,047	/* */	început comentariu Sfârşit comentariu
Car.leg. nume	095	_	Caracter de legătură în nume formate din entităţi separate
Delimitatori	046 034 044 059	. " , ;	Punct zecimal Delimitator în şir Separator elemente listă Sfârşit instrucţiune sau declaraţie

Paranteze	040
	041
	091
	093
	123
	125
		(
		)
		[
		]
		{
		}
			început expresie parantezată sau început listă de argumente din apel funcţie
			început expresie parantezată sau început listă de argumente din apel funcţie
			Semnalează început: constantă domeniu, listă dimensiuni, specificaţie de ciclare FOR/SUM
			Semnifică sfârşitul celor menţionate deasupra
			Deschide o secvenţă de instrucţiuni sau o listă de valori iniţiale
			Închide elementele menţionate

Operatori Aritmetici	043 045 042 047	+ - * /	Plus /Adunare Minus unar / Scădere Înmulţire Împarţire
Operatori Relaţionali	060 061 062 060,061 062,061 060,062 073,078	< = > <= >= <> IN	Mai mic strict Egal cu Mai mare strict Mai mic sau egal Mai mare sau egal Diferit Este în domeniul
Operatori de definire, condiţionare, atribuire valori	061 058,061 073,083	= := IS	Atribuie valoare/lista valori Este prin definiţie Are definiţia / valorile /condiţia date de...

3.2 Cuvinte cheie ALLO

Nr.	Nume	Descriere rol
01	MODEL	Început model
02	END	Sfârşit model
03	VARIABLES	Început secţiune variabile model
04	OBJECTIVES	Început secţiune funcţii obiectiv

05	CONSTRAINTS	Început secţiune restricţii
06	FILE	Instrucţiune definire fişiere secundare cu date
07	RANGE	Instrucţiune definire domenii întregi sau reale
08	INTEGER	Instrucţiune definire variabile aux. întregi
09	REAL	Instrucţiune definire variabile auxiliare reale
10	READ	Comandă citire date din fişier secundar
11	IN	Început condiţie apartenenţă domeniu de valori
12	IS	Clauză de definire globală
13	FOR	Început secvenţă de ciclare
14	SUM	Început funcţie de sumare
15	MINIMIZE	Instrucţiune de selecţie funcţie obiectiv
16	MAXIMIZE	Instrucţiune de selecţie funcţie obiectiv
Acţiuni intermediare de tip funcţii		
17	ABS	Valoarea absolută
18	AND	Şi logic
19	APX	Ridicare la putere cu baza variabilă
20	ATR	Schimbare valori pentru variabile auxiliare
21	DIP	Diferenţă pozitivă
22	IFP	Test pozitivitate, întoarcere valoare selectată
23	IFS	Test logic cu întoarcere valoare selectată
24	LOR	Sau logic
25	LOG	Logaritm zecimal
26	MAX	Maxim a două valori
27	MIN	Minim a două valori
28	MOD	Restul împărţirii întregi
29	NOT	Negare logică
30	SEARCH	Căutare existenţă val., întoarcere rezultat, loc
31	SIG	Semnul valorii
32	SQR	Radical de ordin doi
33	XOR	Sau exclusiv

Pe baza alfabetului şi a vocabularului se construiesc noi cuvinte din vocabular sau asamblaje din ce în ce mai complexe până la asamblajul cu maximă încărcătură sintactică şi semantică, numit program sau model. Elementele de bază ale limbajului ALLO sunt: nume sau identificatori; constante; comentarii; tablouri; indici; variabile; operatori aritmetici; operatori funcţionali; paranteze; delimitatori; expresii; lista de argumente de apel funcţie; listă dimensională; listă de indici; asamblaj de ciclare; instrucţiuni; listă de instrucţiuni; listă de condiţii; secţiuni; program.

4. Sintaxa limbajului ALLO

Precizări de metalimbaj

::= ne spune că elementul din stânga se construieşte după cum urmează sau este prin definiţie;

| ne indică faptul că putem alege pentru construcţia elementului din stânga simbolului ::= pe oricare element din lista prezentată şi ale carei elemente sunt separate de acest simbol.

Spaţiul dintre două elemente de metalimbaj sau dintre un element de metalimbaj şi un atom înseamnă operaţia de concatenare dintre două şiruri ce aparţin clasei (mulţimii) din care fac parte cele două elemente. Un element de metalimbaj semnifică în fapt o clasă de asamblaje (şiruri pe alfabet în cele din urmă) ce se obţin prin aplicarea regulilor de obţinere a acestuia.

Cuvintele rezervate apar cu litere mari dar ele pot fi scrise după cum se doreşte.

Eventuale elemente nedefinite în această sintaxă se pot uşor deduce din tot ceea ce am prezentat.

Sintaxa limbajului ALLO

```
Program          ::= MODEL nume_model
                         sect_var_aux
                     VARIABLES
                         sect_var_model
                     OBJECTIVES
                         sect_var_fobj
                     CONSTRAINTS
                         sect_var_restr
                   END
sect_var_aux     ::= s_instr
s_instr          ::= instr | instr s_instr
instr            ::= instr_file | instr_domeniu |
                     instr_integer | instr_real
instr_file       ::= FILE s_decl_file
s_decl_file      ::= decl_file | decl_file ; s_decl_file
decl_file        ::= var_fisier = spec_fişier_dos
spec_fisier_dos  ::= sir
instr_domeniu    ::= RANGE s_decl_domeniu
s_decl_domeniu   ::= decl_domeniu | decl_domeniu; s_decl_domeniu
decl_domeniu     ::= var_domeniu = const_domeniu
instr_integer    ::= INTEGER s_decl_integer
s_decl_integer   ::= decl_integer | decl_integer; s_decl_integer
decl_integer     ::= var_integer part_init part_valid
part_init        ::= lambda | READ var_fisier | = val_init
val_init         ::= expr_aritm | { l_expr_aritm }
l_expr_aritm     ::= expr_aritm | expr_aritm , l_expr_aritm
part_valid       ::= lambda | IS proced_valid
proced_valid     ::= expr_cond_valid |
                     FOR [ l_expr_apart ] proced_valid |
                     { s_proced_valid }
s_proced_valid   ::= proced_valid | proced_valid ; s_proced_valid
l_expr_apart     ::= expr_apart | expr_apart , l_expr_apart
expr_apart       ::= var_indice IN domeniu
expr_cond_valid  ::= expr_aritm opr_rel expr_aritm
instr_real       ::= REAL s_decl_real
s_decl_real      ::= decl_real | decl_real ; s_decl_real
decl_real        ::= var_real part_init part_valid
sect_var_model   ::= s_instr_var_model
s_instr_var_model ::= instr_var_model | instr_var_model ; s_instr_var_model
instr_var_model  ::= var_model part_cond_marg
part_cond_marg   ::= lambda | IS proced_marg
proced_marg      ::= ref_var_model cond_var_model |
                     FOR [ l_expr_apart ] proced_marg |
                     { s_proced_marg }
ref_var_model    ::= var_model_scalară | ref_element_masiv_var_model
cond_var_model   ::= opr_marg expr_marg_var
expr_marg_var    ::= expr_aritm | domeniu
s_proced_marg    ::= proced_marg | proced_marg ; s_proced_marg
sect_var_fobj    ::= s_instr_fobj instr_select_fobj
s_instr_fobj     ::= instr_fobj | instr_fobj ; s_instr_fobj
instr_fobj       ::= var_fobj IS proced_def_fobj
proced_def_fobj  ::= ref_fobj := expr_simb |
                     FOR [ l_expr_apart ] proced_def_fobj |
                     { s_proced_def_fobj }
s_proced_def_fobj ::= proced_def_fobj | proced_def_fobj ; s_proced_def_fobj
ref_fobj         ::= var_fobj_scalară | ref_element_masiv_fobj
```

instr_select_fobj ::= MINIMIZE ref_fobj | MAXIMIZE ref_fobj
sect_var_restr ::= s_instr_restr
s_instr_restr ::= instr_restr | instr_restr ; s_instr_restr

instr_restr ::= var_restr IS proced_defrestr
proced_defrestr ::= ref_var_restr := expr_simb opr_marg expr_marg_restr
| FOR [l_expr_apart] proced_defrestr
| { s_proced_defrestr }
s_proced_defrestr ::= proced_defrestr | proced_defrestr; s_proced_defrestr
ref_var_restr ::= var_restr_scalara | ref_element_masiv_restr
expr_marg_restr ::= expr_simb | domeniu
expr_aritm ::= termen rest_sumă
termen ::= factor rest_produs
rest_suma ::= lambda | opr_adit rest_sumă
rest_produs ::= opr_mult factor rest_produs | lambda
factor ::= numar | ref_var_aux | apel_fun |
(expr_aritm) |
SUM [l_expr_apart] (expr_aritm)
ref_var_aux ::= ref_var_întreg | ref_var_real
ref_var_intreg ::= var_întreg_scalară | ref_element_masiv_întreg
ref_var_real ::= var_real_scalară | ref_element_masiv_real
expr_simb ::= termen_simb rest_sumă_simb
termen_simb ::= factor_simb rest_produs_simb
rest_sumă_simb ::= lambda | opr_adit rest_sumă_simb
rest_produs_simb ::= lambda | opr_mult factor_simb rest_produs_simb
factor_simb ::= număr | ref_var_aux | ref_var_model | apel_fun |
(expr_simb) | SUM [l_expr_apart] (expr_simb)
apel_fun ::= opr_fun (l_param_act)
l_param_act ::= expr_aritm | expr_aritm , l_param_act

nume ::= litera | nume literă | nume cifră | nume carlegatură
var_întreg ::= var_întreg_scalară | var_întreg_masiv
var_întreg_scalara ::= nume
var_întreg_masiv ::= masiv
var_indice ::= nume
var_real ::= var_real_scalară | var_real_masiv
var_real_scalară ::= nume
var_real_masiv ::= masiv
var_domeniu ::= nume
var_fişier ::= nume
var_model ::= var_model_scalară | var_model_masiv
var_model_scalară ::= nume
var_model_masiv ::= masiv
var_fobj ::= var_fobj_scalară | var_fobj_masiv
var_fobj_scalară ::= nume
var_fobj_masiv ::= masiv
var_restr ::= var_restr_scalară | var_restr_masiv
var_restr_scalară ::= nume
var_restr_masiv ::= masiv
secv_cifră ::= cifră | cifră secv_cifră
număr_întreg ::= semn secv_cifră
semn ::= - | + | lambda
număr_real ::= semn p_int . p_frac exponent
p_int ::= secv_cifră | lambda
p_frac ::= secv_cifră | lambda
exponent ::= lambda | E număr_întreg | e număr_întreg
const_domeniu ::= [expr_aritm , expr_aritm]
sir ::= " secv_car "

```
secv_car              ::= car | car secv_car
masiv                 ::= nume_masiv ş l_dim ţ
nume_masiv            ::= nume
l_dim                 ::= dim | dim , l_dim
dim                   ::= domeniu
domeniu               ::= const_domeniu | var_domeniu
ref_element_masiv     ::= nume_masiv [ l_expr_ind ]
l_expr_ind            ::= expr_ind | expr_ind , l_expr_ind
expr_ind              ::= expr_aritm
opr_aritm             ::= opr_unar Ă opr_binar
opr_unar              ::= -
opr_binar             ::= - | + | * | /
opr_adit              ::= + | -
opr_mult              ::= * | /
opr_fun               ::= ABS | AND | APX | ATR | DIP | IFP | IFS | LOR | LOG
                          | MAX | MIN | MOD | NOT | SEARCH | SIG | SQR | XOR
opr_rel               ::= < | <= | = | >= | > | <>
opr_marg              ::= <= | = | >= | IN
==================================================================
```

5. Exemplu Prototip Tanglewood

MODEL Tanglewood

```
/*-------------------------------------------------------------------------------------------------------------*/
/* Dată Creare    : 09 Martie 1995                                                                            */
/* Dată Modificare : 03 Noiembrie 1997                                                                        */
/*-------------------------------------------------------------------------------------------------------------*/
/* Sursa : MATURANA S. V., Survey and Analysis of LINGO,Mgt 298D Course Proj;                                 */
/*         John E. Anderson Graduate School of Management,Univ of California,                                 */
/*         Los Angeles, CA 90024                                                                              */
/*                                                                                                            */
/*      O companie produce scaune în câteva din centrele sale de producţie din localităţi                     */
/*distincte şi le vinde în câteva oraşe. Cheresteaua necesară producţiei sale este achiziţionată              */
/* de la mai multe depozite care o vinde cu condiţia unui minim de achiziţie. Fiecare centru                 */
/* de producţie poate produce un număr de scaune situat între anumite limite. Pentru livrare                 */
/*scaune sunt solicitate cantităţi care să nu depăşească anumite limite. Se cere un plan de*                  /
/*producţie, de aprovizionare şi desfacere care să producă un profit maxim. Modelul prezent                  */
/* răspunde la întrebările:                                                                                   */
/*                                                                                                            */
/*        1. De unde şi cât trebuie să cumpere cherestea fiecare centru productie;                            */
/*        2. Câte scaune trebuie să producă fiecare centru de producţie;                                      */
/*        3. Câte scaune trebuie să vândă compania la fiecare centru de livrare;                              */
/*        4. Din care centre de producţie şi în ce cantitate trebuie să se livreze                            */
/*           scaunele pentru satisfacerea cererii la fiecare centru de livrare;                               */
/*        5. Care este numarul de scaune fabricat şi vândut de companie şi care va                            */
/*           fi cantitatea de cherestea pe care trebuie s-o cumpere;                                          */
/*        6. Cât încasează compania pe fiecare centru de producţie de la fiecare                              */
/*           centru de achiziţie pentru scaunele livrate.                                                     */
/*        7. Cât incasează compania pe scaunele vândute din fiecare centru de fabricaţie                      */
/*        8. Cât trebuie să încaseze compania de la fiecare centru de achiziţie.                              */
/*        9. Cât incasează compania pe toate scaunele livrate centrelor de achiziţie                          */
/*       10. Care este profitul rezultat al companiei.                                                        */
/*                                                                                                            */
/*      Modelul prezent este modificat faţă de sursa prin introducerea unor variabile statistice              */
/*suplementare care uşurează şi forma de scriere. Acest model este în fapt un MIP model şi ar fi             */
trebuit tratat şi rezolvat cu un MIP */ solver. Totusi pe aceste date soluţia este corectă chiar              */
```

```
lucrând cu variabile*/ continue. Soluția conține rezultate întregi la toate variabilele ce se referă   */
la număr de scaune                                                                                    */
/*-----------------------------------------------------------------------------------------------------*/

FILE ftw="tangwood.dat"

INTEGER
    nrdep READ ftw IS nrdep >0;  /* Număr depozite de cherestea */
    nrfab READ ftw IS nrfab >0;  /* Număr centre de fabricație scaune */
    nrdes READ ftw IS nrdes >0   /* Număr centre desfacere scaune */
RANGE
    dep=[1,nrdep];  /* Domeniu coduri depozite cherestea */
    fab=[1,nrfab];  /* Domeniu coduri centre de fabricație */
    des=[1,nrdes]   /* Domeniu coduri centre desfacere scaune */
REAL
    spret[des] READ ftw IS FOR[ k IN des] spret[k]>0.0;
        /* Preț de vânzare scaune la fiecare centru de desfacere */
     pscost[fab] READ ftw IS FOR[ k IN fab] pscost[k]>0.0;
        /* Cost de producție scaun la fiecare centru de producție */
    tscost[fab,des] READ ftw IS FOR [j IN fab, k IN des] tscost[j,k] >= 0.0;
        /* Cost de transport scaun de la fiecare centru de producție */
        /* la fiecare centru de vânzare */

    tmcost[dep,fab] READ ftw IS FOR [j IN dep, k IN fab] tmcost[j,k] >= 0.0;
        /* Cost transport cherestea de la fiecare depozit la fiecare centru */
    mcost[dep] READ ftw IS FOR[k IN dep] mcost[k]>0.0;
        /* Cost cherestea la fiecare depozit */
    qamin[dep] READ ftw IS FOR[k IN dep] qamin[k]>0.0;
        /* Cantitate minima de achiziție cherestea la fiecare depozit */
    qms READ ftw IS qms >0.0;
        /* Cantitatea de cherestea necesară pentru un scaun */
    fmin[fab] READ ftw IS FOR[k IN fab] fmin[k]>=0.0;
        /* Număr minim scaune posibil de fabricat la fiecare centru */
    fmax[fab] READ ftw IS FOR[k IN fab] fmax[k]>0.0;
        /* Număr maxim scaune posibil de fabricat la fiecare centru */
    smin[des] READ ftw IS FOR[k IN des] smin[k]>0.0;
        /* Număr minim scaune solicitat de fiecare centru de vânzare */
    smax[des] READ ftw IS FOR[k IN des] smax[k]>0.0
        /* Număr maxim scaune solicitat de fiecare centru de vânzare */

VARIABLES

xdf [dep,fab];
    /* Cantitate cherestea cumpărată de fiecare centru de fabricatie */
xd [dep] IS FOR[j IN dep] xd[j] >= qamin[j];
    /* Cantitate cherestea achiziționată de la fiecare depozit */
xt; /* Cantitate cherestea achiziționată de companie */
xf [fab];
    /* Cantitate repartizată de companie fiecarui centru fabricație */
yfs [fab,des];
    /* Număr scaune livrate de la fiecare centru de */
    /* fabricație pentru fiecare centru de achiziție */
yf [fab] IS FOR[j IN fab] yf[j] IN [fmin[j],fmax[j]];
    /* Număr scaune produse de către fiecare centru de fabricație */
ys [des] IS FOR[j IN des] ys[j] IN [smin[j],smax[j]];
    /* Numar scaune livrate de companie fiecarui centru de achiziție */
yt;/* Număr total de scaune fabricate și vândute de companie */
zfsc [fab,des];
    /* încasări companie de la fiecare centru de achiziție pe */
    /* scaunele livrate de către fiecare centru de fabricație */
```

```
zfc [fab];
    /* încasări companie pe scaunele fiecarui centru de fabricaţie */
zsc [des];
    /* încasări companie de la fiecare centru de achiziţie scaune */
ztc
    /* încasări companie pe toate scaunele vândute */

OBJECTIVES

profit IS profit := ztc -
    SUM[j IN fab,k IN des] ((pscost[j]+ tscost[j,k]) * yfs[j,k]) -
    SUM[j IN dep,k IN fab] ((tmcost[j,k]+mcost[j]) * xdf[j,k])

        /* Profitul estimat este preţul obţinut de companie pe toate scaunele */
        /* vândute din care trebuie scăzut costul lor de fabricaţie, costul pe*/
        /* cheresteaua cumpărată şi utilizată , costul transport cherestea de */
        /* la depozite la centrele de fabricaţie,costul transport scaune de la*/
        /* aceste centre la centrele de livrare convenite cu beneficiarii lor */

MAXIMIZE profit

CONSTRAINTS

    rad [dep] IS FOR[j IN dep] rad[j] := xd[j]= SUM[k IN fab] (xdf[j,k]);
        /* Definiţia cantitate cherestea achiziţionată din fiecare depozit */
    rat IS rat := xt= SUM[j IN dep] (xd[j]);
        /* Definiţia cantitate cherestea achiziţionată de companie */
    raf [fab] IS FOR[j IN fab] raf[j] := xf[j]= SUM[k IN dep] (xdf[k,j]);
        /* Cantitate cherestea achiziţionată de fiecare centru fabricaţie */
    rprod [fab] IS FOR[j IN fab] rprod[j] := yf[j]= SUM[k IN des] (yfs[j,k]);
        /* Producţii scaune pe fiecare centru de fabricaţie */
    rdem [des] IS FOR[j IN des]
        rdem[jţ := ys[jţ= SUM[k IN fabţ(yfs[k,jţ);
        /* Număr scaune achiziţionat de fiecare centru de livrare */
    rbal [fab] IS FOR[j IN fab] rbal[j] := xf[j]= qms*yf[j];
        /* Balanţa materială din fiecare centru producţie */
    ryt IS ryt:= yt=  SUM[j IN fab] (yf[j]);
        /* Total număr scaune fabricat şi vândut de către companie */
    rbalt IS rbalt:= yt = SUM[j IN des] (ys[j]);
        /* Balanţa scaunelor preluate de toate centrele de producţie */
        /* ale companiei, care sunt scaune livrate tuturor centrelor */
    rfsc [fab,des] IS FOR[j IN fab,k IN des]
        rfsc[j,k] := zfsc[j,k]= spret[k]*yfs[j,k];
        /* Ecuaţii încasari pe centre producţie de la centre de achiziţie */
    rfc[fab] IS FOR[j IN fab] rfc[j] := zfc[j]= SUM[k IN des] (zfsc[j,k]);
        /* Ecuaţii încasări companie pe fiecare centru producţie */
    rsc [des] IS FOR[j IN des] rsc[j] := zsc[j]= SUM[k IN fab] (zfsc[k,j]);
        /* Ecuaţii încasări companie de la fiecare centru achiziţie */
    rzt IS rzt:= ztc= SUM[j IN fab] (zfc[j])
        /* Ecuaţia încasări companie pentru toate scaunele vândute */

END

/* Tangwood.dat----------------------*/

/* dep */ 2
/* fab */ 4
/* des */ 4
/* spret [des] */  20.,15.,20.,18.;
/* pscost[fab] */  5.,7.,3.,4.;
/* tscost[fab,des] */
```

```
                                1.,1.,2.,0.,
                                3.,6.,7.,3.,
                                3.,1.,5.,3.,
                                8.,2.,1.,4.;
/* tmcost[dep,fab] */
                                0.01,0.02,0.04,0.04,
                                0.04,0.03,0.02,0.02;
/* mcost[dep] */ 0.1,0.075;
/* qamin [dep] */ 8.0,8.0;
/* qms */       20.0
/* fmin[fab] */   0.,400.,500.,250.;
/* fmax[fab] */ 500.,750.,1000.,250.;
/* smin[des] */ 500.,100.,500.,500.;
/* smax[des] */ 2000.,400.,1500.,1500.;
/* Eof Tangwood.dat------------------*/
```

Bibliografie

1. **AHO, A.H., ULLMAN, J.D.**: The Theory of Parsing, Translations and Compiling, Vol.I and II, Prentice Hall, Inc., 1972.
2. **ANDREI, N., BĂRBULESCU, M.**: ALLO - A Language for Linear Optimization. Technical Report LSSO-92-2, Large Scale Systems Optimization, Research Institute for Informatics, Bucharest, September 1992.
3. **ANDREI, N., BĂRBULESCU, M., BORCAN, GHE.**: Integrated System and Languages for Large-Scale. Mathematical Programming F1. Research Institute for Informatics, Technical Report, December 993.
4. **ANDREI, N., BORCAN, GHE.**, Doneanu, R., Bărbulescu, M.: Integrated System and Languages for Large-Scale Mathematical Programming F3. Research Institute for Informatics, Technical Report, December 1994.
5. **ANDREI, N., BORCAN, GHE.**: Sisteme integrate şi limbaje asociate pentru programarea matematică de mari dimensiuni. Raport de Cercetare, 1995.
6. **ANDREI, N., BORCAN, GHE.**: Sistem avansat de modelare şi optimizare bazat pe limbaje de programare matematica. Raport de Cercetare, 1996.
7. **ANDREI, N., BORCAN, GHE.**: Proiectul versiunii 2 a limbajului ALLO. Raport de Cercetare, 1998.
8. **BROOKE A., KENDRICK D., MEERAUS A.**: GAMS: UserÑ Guide, The Scientific Press, Redwood City CA, 1988.
9. **FOURER, R., GAY, D.M., KERNIGHAN, B.W.**: AMPL: A Mathematical Programming Language. Technical Report 87-03, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60201, January 1987, pp.1-63.
10. **HÜRLIMANN, T.**: LPL: A Mathematical Programming Language. OR Spektrum, Vol.15, No.1, 1993, pp.43-55.
11. **KNUTH, E. DONALD.**: The Art of Computers Programming. Addison-Wesley Publishing Company, Vol.1 - 7, 1968-1973. Vol.6, Languages Theory, Vol.7, Compilers.
12. **LUCAS, C., MITRA, G.**: Computer-Assisted Mathematical Programming (Modeling) System (CAMPS). The Computer Journal, vol.31, no.4, 1988, pp.364-375.
13. **MEERAUS, A.**: General Algebraic Modeling System (GAMS): UserÑ Guide, Version 1.0, Development Research Center, World Bank, 1984.
14. **MURPHY, F.H., STOHR, E.A., ASTHANA, A.**: Representation Schemes for Linear Programming Models. Management Science, Vol.38, No.7, July 1992, pp.964-991.
15. **ŞERBĂNAŢI, LUCA, DAN, et al.**: Logical Project for LPTR Compiler, University "Politechnica" - Bucharest ,Contract ICI nr.48/7.2, 1977.
16. **ŞERBĂNAŢI, LUCA, DAN**: Programming Languages and Compilers. Romanian Academy Publish House, 1987.
17. Williams, H.P.: Model Building in Mathematical Programming (second edition), John Wiley & Sons, Chichester, 1985.