

Très bon travail, il manque juste la question de compréhension.

Rapport Travaux Pratiques : Programmation par Contraintes - TP 2 : **Contraintes Logiques**

Nicolas Desfeux
Aurélien Texier

1^{er} mars 2011

Dans ce T.P., nous allons utiliser les contraintes et les domaines finis pour résoudre un problème. Nous allons tenter, à partir des informations (contraintes) qui nous sont fournies, de déduire d'autres informations.

Pour cela, nous allons définir plusieurs prédicats qui nous permettront au final d'obtenir les informations que nous cherchons.

Question 1.1 Nous allons définir les différents domaines dont nous allons avoir besoin au cours de notre problème.

Listing 1 – "Domaines"

```
1 :- local domain(pays(anglais, espagnol, ukrainien, norvegien, japonais)).
2 :- local domain(couleur(rouge, verte, blanche, jaune, bleue)).
3 :- local domain(boisson(cafe, the, lait, jusdOrange, eau)).
4 :- local domain(voiture(bmw, toyota, ford, honda, datsun)).
5 :- local domain(animal(chien, serpents, renard, cheval, zebre)).
```

OK

Question 1.2 Nous avons également besoin de définir des prédicats qui permettent de contraindre le domaine des variables qui composent une maison.

Listing 2 – "Domaines maisons"

```
1 domaines_maison(m(Pays, Couleur, Boisson, Voiture, Animal, Numero)) :-
2     Pays &:: pays,
3     Couleur &:: couleur,
4     Boisson &:: boisson,
5     Voiture &:: voiture,
6     Animal &:: animal,
7     Numero #:: 1..5.
```

OK

Question 1.3 Ce prédicat permet de définir la liste des maisons, tout en posant les différentes contraintes que l'on a précédemment définies. C'est également grâce à ce prédicat que l'on pose les numéros des maisons.

Listing 3 – "Prédicat *rue(?Rue)*"

```

1  rue(Rue) :- length(Rue,5),
2                (foreach(Elem,Rue), for(I, 1, 5), fromto([],InP,OutP,
                    FinP), fromto([],InC,OutC,FinC), fromto([],InB,
                    OutB,FinB), fromto([],InV,OutV,FinV), fromto([],
                    InA,OutA,FinA)
3                do
4                    domaines_maison(Elem),
5                    Elem = m(P,C,B,V,A,I),
6                    OutP=[P|InP],
7                    OutC=[C|InC],
8                    OutB=[B|InB],
9                    OutV=[V|InV],
10                   OutA=[A|InA]
11                ),
12                ic_symbolic:alldifferent(FinP),
13                ic_symbolic:alldifferent(FinC),
14                ic_symbolic:alldifferent(FinB),
15                ic_symbolic:alldifferent(FinV),
16                ic_symbolic:alldifferent(FinA).
17
18  /* Tests
19  [eclipse 42]: rue(R).
20
21  R = [m(_296{[anglais, espagnol, ukrainien, norvegien, japonais]}, _400{[rouge
    , verte, blanche, jaune, bleue]}, _504{[cafe, the, lait, jusdOrange, eau
    ]}, _608{[bmw, toyota, ford, honda, datsun]}, _712{[chien, serpents,
    renard, cheval, zebre]}, 1), m(_894{[anglais, espagnol, ukrainien,
    norvegien, japonais]}, _998{[rouge, verte, blanche, jaune, bleue]}, _1102
    {[cafe, the, lait, jusdOrange, eau]}, _1206{[bmw, toyota, ford, honda,
    datsun]}, _1310{[chien, serpents, renard, cheval, zebre]}, 2), m(_1492{[
    anglais, espagnol, ukrainien, norvegien, japonais]}, _1596{[rouge, verte,
    blanche, jaune, bleue]}, _1700{[cafe, the, lait, jusdOrange, eau]},
    _1804{[bmw, toyota, ford, honda, datsun]}, _1908{[chien, serpents, renard
    , cheval, zebre]}, 3), m(_2090{[anglais, espagnol, ukrainien, norvegien,
    japonais]}, _2194{[rouge, verte, blanche, jaune, bleue]}, _2298{[cafe,
    the, lait, jusdOrange, eau]}, _2402{[bmw, toyota, ford, honda, datsun]},
    _2506{[chien, serpents, renard, cheval, zebre]}, 4), m(_2688{[anglais,
    espagnol, ukrainien, norvegien, japonais]}, _2792{[rouge, verte, blanche,
    jaune, bleue]}, _2896{[cafe, the, lait, jusdOrange, eau]}, _3000{[bmw,
    toyota, ford, honda, datsun]}, _3104{[chien, serpents, renard, cheval,
    zebre]}, 5)]
22
23  There are 30 delayed goals. Do you want to see them? (y/n)
24  Yes (0.10s cpu)
25  */

```

Tres bien

Question 1.4 Ce prédicat va permettre un affichage clair des maisons.

Listing 4 – "Prédicat *ecrit_maison(?Rue)*"

```

1  écrit_maisons(Rue) :-      (foreach(Elem,Rue)
2                                do
3                                writeln(Elem)
4                                ).
5  % Permet de tester l'affichage
6  % Ruetest = [m(anglais , rouge , cafe ,bmw ,chien ,1) ,m(norvegien , bleue , lait , honda ,
7                zebre ,2)]
8  /* Test
9  [eclipse 48]: Ruetest = [m(anglais , rouge , cafe ,bmw ,chien ,1) ,m(norvegien , bleue ,
10                lait ,honda , zebre ,2)] , écrit_maisons(Ruetest) .
11  m(anglais , rouge , cafe , bmw , chien , 1)
12  m(norvegien , bleue , lait , honda , zebre , 2)
13  Ruetest = [m(anglais , rouge , cafe , bmw , chien , 1) , m(norvegien , bleue , lait ,
14                honda , zebre , 2)]
15  Yes (0.00s cpu)
16  */

```

OK

Question 1.5 On définit ici un prédicat qui permet de récupérer la liste des variables du problème.

On utilise ensuite un prédicat de labeling, qui va permettre de labeliser les variables par rapport aux différents domaines que l'on a défini.

Listing 5 – "Prédicat *getVarList(+List)*"

```

1  getVarList([],[]).
2  getVarList([m(P,C,B,V,A,_I)|Rest],[P,C,B,V,A | Liste]) :- getVarList(Rest,
3                        Liste).
4  labeling_symbolic(Liste) :- (foreach(Elem,Liste)
5                                do
6                                ic_symbolic:
7                                    indomain(
8                                        Elem)
9                                ).
10 /* Test
11 rue(R),getVarList(R,L),labeling_symbolic(L).
12
13 R = [m(anglais , rouge , cafe , bmw , chien , 1) , m(espagnol , verte , the , toyota ,
14         serpents , 2) , m(ukrainien , blanche , lait , ford , renard , 3) , m(norvegien ,
15         jaune , jusdOrange , honda , cheval , 4) , m(japonais , bleue , eau , datsum ,
16         zebre , 5)]
17 L = [anglais , rouge , cafe , bmw , chien , espagnol , verte , the , toyota , serpents
18     , ukrainien , blanche , lait , ford , renard , norvegien , jaune , jusdOrange ,
19     honda , ...]
20 Yes (0.00s cpu , solution 1 , maybe more) ? ;
21

```

Attention à la signature...

Bien

```

17 R = [m(anglais, rouge, cafe, bmw, chien, 1), m(espagnol, verte, the, toyota,
    serpents, 2), m(ukrainien, blanche, lait, ford, renard, 3), m(norvegien,
    jaune, jusdOrange, honda, zebre, 4), m(japonais, bleue, eau, datsun,
    cheval, 5)]
18 L = [anglais, rouge, cafe, bmw, chien, espagnol, verte, the, toyota, serpents
    , ukrainien, blanche, lait, ford, renard, norvegien, jaune, jusdOrange,
    honda, ...]
19 Yes (0.00s cpu, solution 2, maybe more) ? ;
20
21 R = [m(anglais, rouge, cafe, bmw, chien, 1), m(espagnol, verte, the, toyota,
    serpents, 2), m(ukrainien, blanche, lait, ford, renard, 3), m(norvegien,
    jaune, jusdOrange, datsun, cheval, 4), m(japonais, bleue, eau, honda,
    zebre, 5)]
22 L = [anglais, rouge, cafe, bmw, chien, espagnol, verte, the, toyota, serpents
    , ukrainien, blanche, lait, ford, renard, norvegien, jaune, jusdOrange,
    datsun, ...]
23
24 */

```

Question 1.6 Le prédicat *resoudre* va permettre de trouver une solution respectant les contraintes de domaines, mais pas les contraintes du problèmes.

Listing 6 – "Prédicat *resoudre(?Rue)*"

```

1 resoudre(R) :- rue(R), getVarList(R, L), labeling_symbolic(L), ecrit_maisons(R).
2
3 /* Test OK
4 [eclipse 3]: resoudre(R).
5 m(anglais, rouge, cafe, bmw, chien, 1)
6 m(espagnol, verte, the, toyota, serpents, 2)
7 m(ukrainien, blanche, lait, ford, renard, 3)
8 m(norvegien, jaune, jusdOrange, honda, cheval, 4)
9 m(japonais, bleue, eau, datsun, zebre, 5)
10
11 R = [m(anglais, rouge, cafe, bmw, chien, 1), m(espagnol, verte, the, toyota,
    serpents, 2), m(ukrainien, blanche, lait, ford, renard, 3), m(norvegien,
    jaune, jusdOrange, honda, cheval, 4), m(japonais, bleue, eau, datsun,
    zebre, 5)]
12 Yes (0.10s cpu, solution 1, maybe more) ? ;
13 m(anglais, rouge, cafe, bmw, chien, 1)
14 m(espagnol, verte, the, toyota, serpents, 2)
15 m(ukrainien, blanche, lait, ford, renard, 3)
16 m(norvegien, jaune, jusdOrange, honda, zebre, 4)
17 m(japonais, bleue, eau, datsun, cheval, 5)
18
19 R = [m(anglais, rouge, cafe, bmw, chien, 1), m(espagnol, verte, the, toyota,
    serpents, 2), m(ukrainien, blanche, lait, ford, renard, 3), m(norvegien,
    jaune, jusdOrange, honda, zebre, 4), m(japonais, bleue, eau, datsun,
    cheval, 5)]
20 Yes (0.10s cpu, solution 2, maybe more) ? ;
21 m(anglais, rouge, cafe, bmw, chien, 1)
22 m(espagnol, verte, the, toyota, serpents, 2)
23 m(ukrainien, blanche, lait, ford, renard, 3)

```

```

24 m(norvegien , jaune , jusdOrange , datsun , cheval , 4)
25 m(japonais , bleue , eau , honda , zebre , 5)
26
27 R = [m(anglais , rouge , cafe , bmw , chien , 1), m(espagnol , verte , the , toyota ,
    serpents , 2), m(ukrainien , blanche , lait , ford , renard , 3), m(norvegien ,
    jaune , jusdOrange , datsun , cheval , 4), m(japonais , bleue , eau , honda ,
    zebre , 5)]
28 Yes (0.10s cpu , solution 3 , maybe more) ? ;
29 m(anglais , rouge , cafe , bmw , chien , 1)
30 m(espagnol , verte , the , toyota , serpents , 2)
31 m(ukrainien , blanche , lait , ford , renard , 3)
32 m(norvegien , jaune , jusdOrange , datsun , zebre , 4)
33 m(japonais , bleue , eau , honda , cheval , 5)
34
35 R = [m(anglais , rouge , cafe , bmw , chien , 1), m(espagnol , verte , the , toyota ,
    serpents , 2), m(ukrainien , blanche , lait , ford , renard , 3), m(norvegien ,
    jaune , jusdOrange , datsun , zebre , 4), m(japonais , bleue , eau , honda ,
    cheval , 5)]
36 */

```

Question 1.7 Maintenant, nous définissons les différentes contraintes, et un prédicat pouvant les utiliser.

Listing 7 – "Définition des contraintes et prédicat de résolution"

```

1  contraintes(R) :-      ( foreach(m(P,C,B,V,A,I),R)
2                          do
3                          (P &= anglais) #= (C &= rouge),
4                          (P &= espagnol) #= (A &= chien),
5                          (B &= cafe) #= (C &= verte),
6                          (P &= ukrainien) #= (B &= the),
7                          (A &= serpents) #= (V &= bmw),
8                          (C &= jaune) #= (V &= toyota),
9                          (B &= lait) #= (I #= 3),
10                         (P &= norvegien) #= (I #= 1),
11                         (B &= jusdOrange) #= (V &= honda),
12                         (P &= japonais) #= (V &= datsun)
13                         ).
14
15  contraintes2(R) :-      ( foreach(m(P1,C1,_B1,V1,_A1,I1),R),param(R)
16                          do
17                          ( foreach(m(_P2,C2,_B2,_V2,A2,I2),R),param(I1,C1,V1,P1)
18                              do
19                              ((C1 &= verte) and (C2 &= blanche)) => (I1 #=I2+1),
20                              ((V1 &= ford) and (A2 &= renard)) => ((I2 #=I1+1) or (I2 #=I1-1)),
21                              ((V1 &= toyota) and (A2 &= cheval)) => ((I2 #=I1+1) or (I2 #=I1-1)),
22                              ((P1 &= norvegien) and (C2 &= bleue)) => ((I2 #=I1+1) or (I2 #=I1-1))
23                              )
24                          ).
25
26  resoudre2(R) :- rue(R),getVarList(R,L),contraintes(R),contraintes2(R),
    labeling_symbolic(L),ecrit_maisons(R).
27  /* Test

```

```

28
29 [eclipse 34]: resoudre2(R).
30 m(norvegien , jaune , eau , toyota , renard , 1)
31 m(ukrainien , bleue , the , ford , cheval , 2)
32 m(anglais , rouge , lait , bmw , serpents , 3)
33 m(espagnol , blanche , jusdOrange , honda , chien , 4)
34 m(japonais , verte , cafe , datsun , zebre , 5)
35
36 R = [m(norvegien , jaune , eau , toyota , renard , 1), m(ukrainien , bleue , the ,
      ford , cheval , 2), m(anglais , rouge , lait , bmw , serpents , 3), m(espagnol ,
      blanche , jusdOrange , honda , chien , 4), m(japonais , verte , cafe , datsun ,
      zebre , 5)]
37 Yes (0.01s cpu , solution 1, maybe more) ? ;
38
39 No (0.01s cpu)
40
41 */

```

Question 1.8 On obtient donc la réponse aux questions posées par le problème :

Le japonais possède un zèbre et le norvégien boit de l'eau

OK

Et la question de compréhension ?

1 Code Complet, avec l'ensemble des tests

Listing 8 – "TP2"

```

1 :- lib(ic).
2 :- lib(ic_symbolic).
3
4 % Question 2.1
5
6 :- local domain(pays(anglais, espagnol, ukrainien, norvegien, japonais)).
7 :- local domain(couleur(rouge, verte, blanche, jaune, bleue)).
8 :- local domain(boisson(cafe, the, lait, jusdOrange, eau)).
9 :- local domain(voiture(bmw, toyota, ford, honda, datsun)).
10 :- local domain(animal(chien, serpents, renard, cheval, zebre)).
11
12 domaines_maison(m(Pays, Couleur, Boisson, Voiture, Animal, Numero)) :-
13     Pays &:: pays,
14     Couleur &:: couleur,
15     Boisson &:: boisson,
16     Voiture &:: voiture,
17     Animal &:: animal,
18     Numero #:: 1..5.
19
20 rue(Rue) :- length(Rue, 5),
21     (foreach(Elem, Rue), for(I, 1, 5), fromto([], InP, OutP,
22         FinP), fromto([], InC, OutC, FinC), fromto([], InB,
23         OutB, FinB), fromto([], InV, OutV, FinV), fromto([],
24         InA, OutA, FinA)
25     do
26         domaines_maison(Elem),
27         Elem = m(P, C, B, V, A, I),
28         OutP=[P|InP],
29         OutC=[C|InC],
30         OutB=[B|InB],
31         OutV=[V|InV],
32         OutA=[A|InA]
33     ),
34     ic_symbolic:alldifferent(FinP),
35     ic_symbolic:alldifferent(FinC),
36     ic_symbolic:alldifferent(FinB),
37     ic_symbolic:alldifferent(FinV),
38     ic_symbolic:alldifferent(FinA).
39
40 /* Tests
41 [eclipse 42]: rue(R).
42
43 R = [m(_296{[anglais, espagnol, ukrainien, norvegien, japonais]}, _400{[rouge,
44     verte, blanche, jaune, bleue]}, _504{[cafe, the, lait, jusdOrange, eau
45     ]}, _608{[bmw, toyota, ford, honda, datsun]}, _712{[chien, serpents,
46     renard, cheval, zebre]}, 1), m(_894{[anglais, espagnol, ukrainien,
47     norvegien, japonais]}, _998{[rouge, verte, blanche, jaune, bleue]}, _1102
48     {[cafe, the, lait, jusdOrange, eau]}, _1206{[bmw, toyota, ford, honda,
49     datsun]}, _1310{[chien, serpents, renard, cheval, zebre]}, 2), m(_1492{[

```

```

    anglais , espagnol , ukrainien , norvegien , japonais }}, _1596{[rouge , verte ,
    blanche , jaune , bleue ]}, _1700{[cafe , the , lait , jusdOrange , eau ]},
    _1804{[bmw , toyota , ford , honda , datsun ]}, _1908{[chien , serpents , renard
    , cheval , zebre ]}, 3), m(_2090{[anglais , espagnol , ukrainien , norvegien ,
    japonais ]}, _2194{[rouge , verte , blanche , jaune , bleue ]}, _2298{[cafe ,
    the , lait , jusdOrange , eau ]}, _2402{[bmw , toyota , ford , honda , datsun ]},
    _2506{[chien , serpents , renard , cheval , zebre ]}, 4), m(_2688{[anglais ,
    espagnol , ukrainien , norvegien , japonais ]}, _2792{[rouge , verte , blanche ,
    jaune , bleue ]}, _2896{[cafe , the , lait , jusdOrange , eau ]}, _3000{[bmw ,
    toyota , ford , honda , datsun ]}, _3104{[chien , serpents , renard , cheval ,
    zebre ]}, 5)]
41
42 There are 30 delayed goals. Do you want to see them? (y/n)
43 Yes (0.10s cpu)
44 */
45
46 ecrit_maisons(Rue) :-      (foreach(Elem,Rue)
47                               do
48                               writeln(Elem)
49                               ).
50 % Permet de tester l'affichage
51 % Ruetest = [m(anglais , rouge , cafe ,bmw ,chien ,1) ,m(norvegien , bleue , lait , honda ,
    zebre ,2) ]
52
53 /* Test
54 [eclipse 48]: Ruetest = [m(anglais , rouge , cafe ,bmw ,chien ,1) ,m(norvegien , bleue ,
    lait , honda ,zebre ,2) ] ,ecrit_maisons( Ruetest) .
55 m(anglais , rouge , cafe , bmw , chien , 1)
56 m(norvegien , bleue , lait , honda , zebre , 2)
57
58 Ruetest = [m(anglais , rouge , cafe , bmw , chien , 1) , m(norvegien , bleue , lait ,
    honda , zebre , 2) ]
59 Yes (0.00s cpu)
60
61 */
62
63 getVarList([],[]).
64 getVarList([m(P,C,B,V,A,_I)|Rest],[P,C,B,V,A | Liste]) :- getVarList(Rest,
    Liste).
65
66 labeling_symbolic(Liste) :- (foreach(Elem,Liste)
67                               do
68                               ic_symbolic:
                                indomain(
                                    Elem)
69                               ).
70
71 /* Test
72
73 rue(R),getVarList(R,L),labeling_symbolic(L).
74
75 R = [m(anglais , rouge , cafe , bmw , chien , 1) , m(espagnol , verte , the , toyota ,
    serpents , 2) , m(ukrainien , blanche , lait , ford , renard , 3) , m(norvegien ,

```



```

    jaune, jusdOrange, honda, cheval, 4), m(japonais, bleue, eau, datsun,
    zebre, 5)]
76 L = [anglais, rouge, cafe, bmw, chien, espagnol, verte, the, toyota, serpents
    , ukrainien, blanche, lait, ford, renard, norvegien, jaune, jusdOrange,
    honda, ...]
77 Yes (0.00s cpu, solution 1, maybe more) ? ;
78
79 R = [m(anglais, rouge, cafe, bmw, chien, 1), m(espagnol, verte, the, toyota,
    serpents, 2), m(ukrainien, blanche, lait, ford, renard, 3), m(norvegien,
    jaune, jusdOrange, honda, zebre, 4), m(japonais, bleue, eau, datsun,
    cheval, 5)]
80 L = [anglais, rouge, cafe, bmw, chien, espagnol, verte, the, toyota, serpents
    , ukrainien, blanche, lait, ford, renard, norvegien, jaune, jusdOrange,
    honda, ...]
81 Yes (0.00s cpu, solution 2, maybe more) ? ;
82
83 R = [m(anglais, rouge, cafe, bmw, chien, 1), m(espagnol, verte, the, toyota,
    serpents, 2), m(ukrainien, blanche, lait, ford, renard, 3), m(norvegien,
    jaune, jusdOrange, datsun, cheval, 4), m(japonais, bleue, eau, honda,
    zebre, 5)]
84 L = [anglais, rouge, cafe, bmw, chien, espagnol, verte, the, toyota, serpents
    , ukrainien, blanche, lait, ford, renard, norvegien, jaune, jusdOrange,
    datsun, ...]
85
86 */
87
88 contraintes(R) :-      (foreach(m(P,C,B,V,A,I),R)
89                          do
90                          (P &= anglais) #= (C &= rouge),
91                          (P &= espagnol) #= (A &= chien),
92                          (B &= cafe) #= (C &= verte),
93                          (P &= ukrainien) #= (B &= the),
94                          (A &= serpents) #= (V &= bmw),
95                          (C &= jaune) #= (V &= toyota),
96                          (B &= lait) #= (I #= 3),
97                          (P &= norvegien) #= (I #= 1),
98                          (B &= jusdOrange) #= (V &= honda),
99                          (P &= japonais) #= (V &= datsun)
100                          ).
101
102 contraintes2(R) :-      (foreach(m(P1,C1,_B1,V1,_A1,I1),R),param(R)
103                          do
104                          (foreach(m(_P2,C2,_B2
105                                  ,_V2,A2,I2),R),
106                                  param(I1,C1,V1,P1
107                                  )
108                                  do
109                                  ((C1 &= verte) and (C2 &= blanche)) => (I1 #=I2+1),
110                                  ((V1 &= ford) and (A2 &= renard)) => ((I2 #=I1+1) or (I2 #=I1
111                                  -1)),
112                                  ((V1 &= toyota) and (A2 &= cheval)) => ((I2 #=I1+1) or (I2 #=
113                                  I1-1)),

```

```

109          ((P1 &= norvegien) and (C2 &= bleue)) => ((I2 #=I1+1) or (I2
110              #=I1-1))
111              )
112          ).
113  resoudre(R) :- rue(R), contraintes(R), getVarList(R,L), labeling_symbolic(L),
114      ecrit_maisons(R).
115  resoudre2(R) :- rue(R), getVarList(R,L), contraintes(R), contraintes2(R),
116      labeling_symbolic(L), ecrit_maisons(R).

```