TP5 - Contraindre puis chercher

Paul Chaignon - Ulysse Goarant

28 février 2014

Listing 1 – production.ecl

```
:- lib(ic).
:- lib(branch_and_bound).

/**
 * Question 5.1
 */
/**
 * techniciens(?Techniciens)
 */
techniciens(Techniciens):-
  Techniciens = [](5, 7, 2, 6, 9, 3, 7, 5, 3).

/**
 * quantites(?Quantites)
 */
quantites(Quantites):-
  Quantites = [](140, 130, 60, 95, 70, 85, 100, 30, 45).

/**
 * benefices(?Benefices)
 */
benefices(Benefices):-
  Benefices = [](4, 5, 8, 5, 6, 4, 7, 10, 11).

/**
 * fabrique(?Fabrique)
 */
fabrique(Fabriquer):-
  techniciens(Techniciens),
  dim(Techniciens, [Taille]),
  dim(Fabriquer, [Taille]),
  Fabriquer #:: 0..1.


/**
 * Question 5.2
 */
/**
 * nb_ouvriers(?Fabriquer, ?NbOuvriers)
 * Recupere le nombre d'ouvriers necessaire.
 */
nb_ouvriers(Fabriquer, NbOuvriers):-
  techniciens(Techniciens),
```

```
44    dim(Techniciens, [Taille]),
45    (for(I, 1, Taille), fromto(0, In, Out, NbOuvriers), param(Fabriquer,
          Techniciens) do
46      Out #= Fabriquer[I] * Techniciens[I] + In
47    ).
48
49  /**
50   * benefices_totaux(?Fabriquer, ?BeneficesTotaux)
51   * Calcule les benefices totaux pour chaque telephone.
52   */
53  benefices_totaux(Fabriquer, BeneficesTotaux):-
54    quantites(Quantites),
55    benefices(Benefices),
56    dim(Benefices, [Taille]),
57    dim(BeneficesTotaux, [Taille]),
58    (for(I, 1, Taille), param(BeneficesTotaux, Fabriquer, Quantites,
          Benefices) do
59      BeneficesTotaux[I] #= Fabriquer[I] * Quantites[I] * Benefices[I]
60    ).
61
62  /**
63   * profit_total(?Fabriquer, ?Profit)
64   * Calcule le profit total.
65   */
66  profit_total(Fabriquer, Profit):-
67    benefices_totaux(Fabriquer, BeneficesTotaux),
68    (foreachelem(Benef, BeneficesTotaux), fromto(0, In, Out, Profit) do
69      Out #= Benef + In
70    ).
71
72
73  /**
74   * Question 5.3
75   */
76  /**
77   * pose_contraintes(?Fabriquer, ?NbTechniciensTotal, ?Profit)
78   */
79  pose_contraintes(Fabriquer, NbTechniciensTotal, NbOuvriers, Profit):-
80    % Le nombre d'ouvriers est positif et inferieur au nombre maximal de
          techniciens.
81    nb_ouvriers(Fabriquer, NbOuvriers),
82    NbOuvriers #=< NbTechniciensTotal,
83    NbOuvriers #>= 0,
84    profit_total(Fabriquer, Profit).
85
86  /**
87   * resoudre(?Fabriquer, ?NbTechniciensTotal, ?NbOuvriers, ?Profit)
88   */
89  resoudre(Fabriquer, NbTechniciensTotal, NbOuvriers, Profit):-
90    fabrique(Fabriquer),
91    pose_contraintes(Fabriquer, NbTechniciensTotal, NbOuvriers, Profit),
92    labeling(Fabriquer).
93
94
95  /**
96   * Question 5.4
97   */
98  equation(X):-
```

```
99    [X, Y, Z, W] #:: [0..10],
100   X #= Z + Y + 2*W,
101   X #\= Z + Y + W,
102   labeling([X, Y, Z, W]).
103
104 % Labeling on X:
105 minimize(equation(X), X).
106   Found a solution with cost 1
107   Found no solution with cost -1.0Inf .. 0
108   X = 1
109   Yes (0.00s cpu)
110 % Labeling on X, Y, Z, W:
111 minimize(equation(X), X).
112   Found a solution with cost 2
113   Found no solution with cost -1.0Inf .. 1
114   X = 2
115   Yes (0.00s cpu)
116
117
118 /**
119  * Question 5.5
120  */
121 resoudre_inv(Fabriquer, NbTechniciensTotal, NbOuvriers, ProfitInv):-
122   resoudre(Fabriquer, NbTechniciensTotal, NbOuvriers, Profit),
123   ProfitInv #= -Profit.
124 resoudre_opti(Fabriquer, NbTechniciensTotal, NbOuvriers, Profit):-
125   minimize(resoudre_inv(Fabriquer, NbTechniciensTotal, NbOuvriers,
          ProfitInv), ProfitInv),
126   Profit is -ProfitInv.
127
128
129 /**
130  * Question 5.6
131  */
132 resoudre_licenciements(Fabriquer, NbTechniciensTotal, NbOuvriers,
       Profit):-
133   Profit #> 1000,
134   minimize(resoudre(Fabriquer, NbTechniciensTotal, NbOuvriers, Profit),
          NbOuvriers).
135
136
137
138 /**
139  * Tests
140  */
141 fabrique(Fabriquer).
142   Fabriquer = [](_11162{[0, 1]}, _11180{[0, 1]}, _11198{[0, 1]},
          _11216{[0, 1]}, _11234{[0, 1]}, _11252{[0, 1]}, _11270{[0, 1]},
          _11288{[0, 1]}, _11306{[0, 1]})
143   Yes (0.00s cpu)
144
145 fabrique(Fabriquer), nb_ouvriers(Fabriquer, NbOuvriers).
146   Fabriquer = [](_376{[0, 1]}, _394{[0, 1]}, _412{[0, 1]}, _430{[0, 1]},
          _448{[0, 1]}, _466{[0, 1]}, _484{[0, 1]}, _502{[0, 1]}, _520{[0,
          1]})
147   NbOuvriers = NbOuvriers{0 .. 47}
148   There are 9 delayed goals. Do you want to see them? (y/n)
149   Yes (0.00s cpu)
```

```
150
151 fabrique(Fabriquer), benefices_totaux(Fabriquer, Benefs).
152   Fabriquer = [](_376{[0, 1]}, _394{[0, 1]}, _412{[0, 1]}, _430{[0, 1]},
          _448{[0, 1]}, _466{[0, 1]}, _484{[0, 1]}, _502{[0, 1]}, _520{[0,
         1]})
153   Benefs = [](_685{0 .. 560}, _1249{0 .. 650}, _1813{0 .. 480}, _2377{0
         .. 475}, _2941{0 .. 420}, _3505{0 .. 340}, _4069{0 .. 700}, _4633{0
          .. 300}, _5197{0 .. 495})
154   There are 9 delayed goals. Do you want to see them? (y/n)
155   Yes (0.01s cpu)
156
157 fabrique(Fabriquer), profit_total(Fabriquer, Profit).
158   Fabriquer = [](_376{[0, 1]}, _394{[0, 1]}, _412{[0, 1]}, _430{[0, 1]},
          _448{[0, 1]}, _466{[0, 1]}, _484{[0, 1]}, _502{[0, 1]}, _520{[0,
         1]})
159   Profit = Profit{0 .. 4420}
160   There are 17 delayed goals. Do you want to see them? (y/n)
161   Yes (0.00s cpu)
162
163 fabrique(Fabriquer), pose_contraintes(Fabriquer, 22, Profit).
164   Fabriquer = [](_390{[0, 1]}, _408{[0, 1]}, _426{[0, 1]}, _444{[0, 1]},
          _462{[0, 1]}, _480{[0, 1]}, _498{[0, 1]}, _516{[0, 1]}, _534{[0,
         1]})
165   Profit = Profit{0 .. 4420}
166   There are 26 delayed goals. Do you want to see them? (y/n)
167   Yes (0.00s cpu)
168
169 resoudre(Fabriquer, 22, NbOuvriers, Profit).
170   Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 0, 0)
171   NbOuvriers = 0
172   Profit = 0
173   Yes (0.00s cpu, solution 1, maybe more)
174   Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 0, 1)
175   NbOuvriers = 3
176   Profit = 495
177   Yes (0.00s cpu, solution 2, maybe more)
178   Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 1, 0)
179   NbOuvriers = 5
180   Profit = 300
181   Yes (0.00s cpu, solution 3, maybe more)
182   ...
183
184 Profit #> 2500, resoudre(Fabriquer, 22, NbOuvriers, Profit).
185   Profit = 2665
186   Fabriquer = [](0, 1, 1, 0, 0, 1, 1, 0, 1)
187   NbOuvriers = 22
188   Yes (0.00s cpu, solution 1, maybe more)
189   ...
190 Profit #> 2665, resoudre(Fabriquer, 22, NbOuvriers, Profit).
191   No (0.01s cpu)
192
193 resoudre_opti(Fabriquer, 22, NbOuvriers, Profit).
194   Found a solution with cost 0
195   Found a solution with cost -495
196   Found a solution with cost -795
197   Found a solution with cost -1195
198   Found a solution with cost -1495
199   Found a solution with cost -1535
```

```
  Found a solution with cost -1835
  Found a solution with cost -1955
  Found a solution with cost -1970
  Found a solution with cost -2010
  Found a solution with cost -2015
  Found a solution with cost -2315
  Found a solution with cost -2490
  Found a solution with cost -2665
  Found no solution with cost -1.0Inf .. -2666
  Fabriquer = [](0, 1, 1, 0, 0, 1, 1, 0, 1)
  NbOuvriers = 22
  Profit = 2665
  Yes (0.01s cpu)

resoudre_licenciements(Fabriquer, 22, NbOuvriers, Profit).
  Found a solution with cost 10
  Found a solution with cost 9
  Found a solution with cost 8
  Found a solution with cost 7
  Found no solution with cost -1.0Inf .. 6
  Fabriquer = [](1, 0, 1, 0, 0, 0, 0, 0, 0)
  NbOuvriers = 7
  Profit = 1040
  Yes (0.00s cpu)
```

Question 5.4

Si on labelle uniquement sur X le solveur nous trouve une valeur minimum de 1 ce qui est faux. Comme les variables sont des entiers : $X = 1 \Rightarrow z = 1 \; ou \; y = 1$. Or $x \neq z+y+w$, donc ce n'est pas possible. Le solveur a encore des *delayed goals* et n'a donc pas vérifié que la valeur associée à X est possible.

Il faut donc toujours labeller sur toutes les variables dans le but pour ne pas avoir de *delayed goals*.