Rapport Travaux Pratiques : Acquisition de connaissance 2 - TP 2: Programmation Logique Inductive

Nicolas Desfeux
Aurélien Texier

23 février 2011

Table des matières

1 Prise en main Les trains de Michalsky

L'objectif de cette section est la prise en main de l'outil Aleph. Pour cela, nous allons essayer de reproduire l'exemple du cours. Les fichiers qui nous sont fournis contiennent tout le paramètrage et les exemples nécessaires pour réaliser cette exemple. Nous allons vous donner ici le résultat de l'éxécution :

Listing 1 – "Exemple - Trains de Michalsky"

```
1
2
3  A L E P H
4  Version 5
5  Last modified: Sun Mar 11 03:25:37 UTC 2007
6
7  Manual: http://www.comlab.ox.ac.uk/oucl/groups/machlearn/Aleph/index.html
8
9  [consulting pos examples] [train.f]
10 [consulting neg examples] [train.n]
11 [select example] [1]
12 [sat] [1]
13 [eastbound(east1)]
14
15 [bottom clause]
16 eastbound(A) :-
17    has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
18    short(E), short(C), closed(C), long(D),
19    long(B), open_car(E), open_car(D), open_car(B),
20    shape(E,rectangle), shape(D,rectangle), shape(C,rectangle), shape(B,
          rectangle),
21    wheels(E,2), wheels(D,3), wheels(C,2), wheels(B,2),
22    load(E,circle,1), load(D,hexagon,1), load(C,triangle,1), load(B,rectangle
          ,3).
23
24 [literals] [25]
25 [saturation time] [0.002]
26 [reduce]
27 [best label so far] [[1,0,2,1]/0]
28 eastbound(A).
29 [5/5]
30 eastbound(A) :-
31    has_car(A,B).
32 [5/5]
33 eastbound(A) :-
34    has_car(A,B).
35 [5/5]
36 eastbound(A) :-
37    has_car(A,B).
38 [5/5]
39 eastbound(A) :-
40    has_car(A,B).
41 [5/5]
42 eastbound(A) :-
```

```
43      has_car(A,B), long(B).
44  [3/5]
45  eastbound(A) :-
46      has_car(A,B), open_car(B).
47  [5/5]
48  eastbound(A) :-
49      has_car(A,B), shape(B,rectangle).
50  [5/5]
51  eastbound(A) :-
52      has_car(A,B), wheels(B,2).
53  [5/5]
54  eastbound(A) :-
55      has_car(A,B), load(B,rectangle,3).
56  eastbound(A) :-
57      has_car(A,B), has_car(A,C).
58  [5/5]
59  eastbound(A) :-
60      has_car(A,B), has_car(A,C).
61  [5/5]
62  eastbound(A) :-
63      has_car(A,B), has_car(A,C).
64  [5/5]
65  eastbound(A) :-
66      has_car(A,B), short(B).
67  [5/5]
68  eastbound(A) :-
69      has_car(A,B), closed(B).
70  [5/2]
71  eastbound(A) :-
72      has_car(A,B), shape(B,rectangle).
73  [5/5]
74  eastbound(A) :-
75      has_car(A,B), wheels(B,2).
76  [5/5]
77  eastbound(A) :-
78      has_car(A,B), load(B,triangle,1).
79  [5/2]
80  eastbound(A) :-
81      has_car(A,B), has_car(A,C).
82  [5/5]
83  eastbound(A) :-
84      has_car(A,B), has_car(A,C).
85  [5/5]
86  eastbound(A) :-
87      has_car(A,B), long(B).
88  [3/5]
89  eastbound(A) :-
90      has_car(A,B), open_car(B).
91  [5/5]
92  eastbound(A) :-
93      has_car(A,B), shape(B,rectangle).
94  [5/5]
95  eastbound(A) :-
```

```
96      has_car(A,B), wheels(B,3).
97    [3/2]
98    eastbound(A) :-
99      has_car(A,B), load(B,hexagon,1).
100   eastbound(A) :-
101     has_car(A,B), has_car(A,C).
102   [5/5]
103   eastbound(A) :-
104     has_car(A,B), short(B).
105   [5/5]
106   eastbound(A) :-
107     has_car(A,B), open_car(B).
108   [5/5]
109   eastbound(A) :-
110     has_car(A,B), shape(B,rectangle).
111   [5/5]
112   eastbound(A) :-
113     has_car(A,B), wheels(B,2).
114   [5/5]
115   eastbound(A) :-
116     has_car(A,B), load(B,circle,1).
117   [3/3]
118   eastbound(A) :-
119     has_car(A,B), closed(B), shape(B,rectangle).
120   eastbound(A) :-
121     has_car(A,B), closed(B), wheels(B,2).
122   eastbound(A) :-
123     has_car(A,B), closed(B), load(B,triangle,1).
124   [2/0]
125   [----------------------------------------]
126   [found clause]
127   eastbound(A) :-
128     has_car(A,B), closed(B), load(B,triangle,1).
129   [pos cover = 2 neg cover = 0] [pos-neg] [2]
130   [clause label] [[2,0,4,2]]
131   [clauses constructed] [34]
132   [----------------------------------------]
133   eastbound(A) :-
134     has_car(A,B), closed(B), has_car(A,C).
135   eastbound(A) :-
136     has_car(A,B), closed(B), has_car(A,C).
137   eastbound(A) :-
138     has_car(A,B), load(B,triangle,1), has_car(A,C).
139   eastbound(A) :-
140     has_car(A,B), load(B,triangle,1), has_car(A,C).
141   eastbound(A) :-
142     has_car(A,B), wheels(B,3), has_car(A,C).
143   eastbound(A) :-
144     has_car(A,B), open_car(B), shape(B,rectangle).
145   eastbound(A) :-
146     has_car(A,B), open_car(B), wheels(B,2).
147   eastbound(A) :-
148     has_car(A,B), open_car(B), has_car(A,C).
```

```
149 eastbound(A) :-
150     has_car(A,B), open_car(B), has_car(A,C).
151 eastbound(A) :-
152     has_car(A,B), open_car(B), has_car(A,C).
153 eastbound(A) :-
154     has_car(A,B), shape(B,rectangle), wheels(B,2).
155 eastbound(A) :-
156     has_car(A,B), shape(B,rectangle), has_car(A,C).
157 eastbound(A) :-
158     has_car(A,B), shape(B,rectangle), has_car(A,C).
159 eastbound(A) :-
160     has_car(A,B), shape(B,rectangle), has_car(A,C).
161 eastbound(A) :-
162     has_car(A,B), wheels(B,2), has_car(A,C).
163 eastbound(A) :-
164     has_car(A,B), wheels(B,2), has_car(A,C).
165 eastbound(A) :-
166     has_car(A,B), wheels(B,2), has_car(A,C).
167 eastbound(A) :-
168     has_car(A,B), has_car(A,C), short(C).
169 eastbound(A) :-
170     has_car(A,B), has_car(A,C), closed(C).
171 eastbound(A) :-
172     has_car(A,B), has_car(A,C), shape(C,rectangle).
173 eastbound(A) :-
174     has_car(A,B), has_car(A,C), wheels(C,2).
175 eastbound(A) :-
176     has_car(A,B), has_car(A,C), load(C,triangle,1).
177 eastbound(A) :-
178     has_car(A,B), has_car(A,C), has_car(A,D).
179 eastbound(A) :-
180     has_car(A,B), has_car(A,C), has_car(A,D).
181 eastbound(A) :-
182     has_car(A,B), has_car(A,C), long(C).
183 eastbound(A) :-
184     has_car(A,B), has_car(A,C), open_car(C).
185 eastbound(A) :-
186     has_car(A,B), has_car(A,C), shape(C,rectangle).
187 eastbound(A) :-
188     has_car(A,B), has_car(A,C), wheels(C,3).
189 eastbound(A) :-
190     has_car(A,B), has_car(A,C), load(C,hexagon,1).
191 eastbound(A) :-
192     has_car(A,B), has_car(A,C), has_car(A,D).
193 eastbound(A) :-
194     has_car(A,B), has_car(A,C), short(C).
195 eastbound(A) :-
196     has_car(A,B), has_car(A,C), open_car(C).
197 eastbound(A) :-
198     has_car(A,B), has_car(A,C), shape(C,rectangle).
199 eastbound(A) :-
200     has_car(A,B), has_car(A,C), wheels(C,2).
201 eastbound(A) :-
```

```
    has_car(A,B), has_car(A,C), load(C,circle,1).
eastbound(A) :-
    has_car(A,B), short(B), closed(B).
[5/0]
[----------]
[found clause]
eastbound(A) :-
    has_car(A,B), short(B), closed(B).
[pos cover = 5 neg cover = 0] [pos-neg] [5]
[clause label] [[5,0,4,5]]
[clauses constructed] [70]
[----------]
[clauses constructed] [70]
[search time] [0.004]
[best clause]
eastbound(A) :-
    has_car(A,B), short(B), closed(B).
[pos cover = 5 neg cover = 0] [pos-neg] [5]
[atoms left] [0]
[positive examples left] [0]
[estimated time to finish (secs)] [0.0]

[theory]

[Rule 1] [Pos cover = 5 Neg cover = 0]
eastbound(A) :-
    has_car(A,B), short(B), closed(B).

[Training set performance]
           Actual
        +        -
     + 5         0         5
Pred
     - 0         5         5

       5         5         10

Accuracy = 1.0
[Training set summary] [[5,0,0,5]]
[time taken] [0.006]
[total clauses constructed] [70]
```

Cet exemple nous permet d'identifier facilement notre espace de recherche, qui partira de la bottom clause (clause la moins générale, créé par Aleph) :

```
[ bottom clause ]
eastbound(A) :-
has_car(A,B),has_car(A,C),has_car(A,D),has_car(A,E),
short(E),short(C),closed(C),long(D),
long(B),open_car(E),open_car(D),open_car(B),
shape(E,rectangle),shape(D,rectangle),shape(C,rectangle),shape(B,rectangle),
wheels(E,2),wheels(D,3),wheels(C,2),wheels(B,2),
load(E,circle,1),load(D,hexagon,1),load(C,triangle,1),load(B,rectangle,3).
```

et qui se poursuivra tant que l'outil n'aura pas trouvé la règle la plus efficace, et qu'il n'aura pas atteint la clause la plus spécifique. Il parcourt l'espace de recherche en faisant donc un bottom-down. La clause la plus générale est :

```
1 eastbound(A).
```

C'est donc de cette clause qu'il est parti pôur inférer la règle recherchée. Il infère jusqu'à trouver une règle qui caractérise correctement tout les exemples :

```
1 [----------]
2 [found clause]
3 eastbound(A) :-
4     has_car(A,B), short(B), closed(B).
5 [pos cover = 5 neg cover = 0] [pos-neg] [5]
6 [clause label] [[5,0,4,5]]
7 [clauses constructed] [70]
8 [----------]
```

2 Une affaire de famille

Pour cette section, c'est à nous de créer les fichiers nécéssaire à l'inférence de la règle. On cherche à obtenir le prédicat *fillede*/2, tel que fille(X,Y) soit vérifié si X est fille de Y. Pour cela, nous avons du d'abord écrire le fichier lu par Aleph qui permet de définir les prédicats utilisables, les relations déjà connues,... Cela définit le background knowledge de notre étude.

Listing 2 – "family.b"

```
1  % test family
2
3
4  :- set(i,2).
5  :- set(verbose,1).
6
7  :- modeh(1,fillede(+pers,+pers)).
8  :- modeb(1,pere(+pers,+pers)).
9  :- modeb(*,pere(+pers,-pers)).
10 :- modeb(1,pere(-pers,+pers)).
11 :- modeb(*,pere(-pers,-pers)).
12 :- modeb(1,mere(+pers,+pers)).
13 :- modeb(*,mere(+pers,-pers)).
14 :- modeb(1,mere(-pers,+pers)).
15 :- modeb(*,mere(-pers,-pers)).
16 :- modeb(1,femme(+pers)).
17 :- modeb(*,femme(-pers)).
18 :- modeb(1,homme(+pers)).
19 :- modeb(*,homme(-pers)).
20
21 :- determination(fillede/2,pere/2).
22 :- determination(fillede/2,mere/2).
23 :- determination(fillede/2,femme/1).
24 :- determination(fillede/2,homme/1).
25
```

```
26 % type definitions
27
28
29 %Relations
30
31 femme(ann).
32 femme(mary).
33 femme(rosy).
34 femme(eve).
35 femme(lisa).
36 homme(tom).
37 homme(bob).
38
39 mere(ann,mary).
40 mere(ann,tom).
41 mere(mary,rosy).
42 pere(tom,eve).
43 pere(tom,lisa).
44 pere(tom,bob).
```

Une fois cela fait, il nous a fallut définir des exemples positifs(X est fille de Y), et négatifs (X n'est pas fille de Y).

Listing 3 – "Exemples positifs"

```
1 fillede(mary,ann).
2 fillede(rosy,mary).
3 fillede(eve,tom).
4 fillede(lisa,tom).
```

Listing 4 – "Exemples négatifs"

```
1 fillede(tom,ann).
2 fillede(bob,tom).
3 fillede(bob,mary).
4 fillede(bob,ann).
5 fillede(tom,mary).
```

L'éxécution a permis de trouver une règle :

Listing 5 – "*fillede* - version 1"

```
1 Contre exemple :
2 fillede(tom,ann).
3 fillede(bob,tom).
4 fillede(bob,mary).
5 fillede(bob,ann).
6 fillede(tom,mary).
7 exemple :
8 fillede(mary,ann).
9 fillede(rosy,mary).
10 fillede(eve,tom).
11 fillede(lisa,tom).
12
13 A L E P H
```

```
Version 5
Last modified: Sun Mar 11 03:25:37 UTC 2007

Manual: http://www.comlab.ox.ac.uk/oucl/groups/machlearn/Aleph/index.html

[consulting pos examples] [family.f]
[consulting neg examples] [family.n]
[select example] [1]
[sat] [1]
[fillede(mary,ann)]
[repeated literals] [38/52]

[bottom clause]
fillede(A,B) :-
    pere(C,D), pere(C,E), pere(C,F), mere(B,A),
    mere(B,C), mere(A,G), femme(B), femme(A),
    femme(G), femme(D), femme(E), homme(C),
    homme(F).
[literals] [14]
[saturation time] [0.002]
[reduce]
[best label so far] [[1,0,2,1]/0]
fillede(A,B).
[4/5]
fillede(A,B) :-
    homme(C).
[4/5]
fillede(A,B) :-
    pere(C,D).
[4/5]
fillede(A,B) :-
    pere(C,D).
[4/5]
fillede(A,B) :-
    pere(C,D).
[4/5]
fillede(A,B) :-
    homme(C).
[4/5]
fillede(A,B) :-
    femme(C).
[4/5]
fillede(A,B) :-
    femme(C).
[4/5]
fillede(A,B) :-
    femme(C).
[4/5]
fillede(A,B) :-
    mere(B,A).
[2/1]
fillede(A,B) :-
    mere(B,C).
```

```
[2/4]
fillede(A,B) :-
    mere(A,C).
fillede(A,B) :-
    femme(B).
[2/4]
fillede(A,B) :-
    femme(A).
[4/0]
[-----------------------------------------]
[found clause]
fillede(A,B) :-
    femme(A).
[pos cover = 4 neg cover = 0] [pos-neg] [4]
[clause label] [[4,0,2,4]]
[clauses constructed] [14]
[-----------------------------------------]
[clauses constructed] [14]
[search time] [0.001]
[best clause]
fillede(A,B) :-
    femme(A).
[pos cover = 4 neg cover = 0] [pos-neg] [4]
[atoms left] [0]
[positive examples left] [0]
[estimated time to finish (secs)] [0.0]

[theory]

[Rule 1] [Pos cover = 4 Neg cover = 0]
fillede(A,B) :-
    femme(A).

[Training set performance]
          Actual
        +        -
     + 4         0         4
Pred
     - 0         5         5

       4         5         9

Accuracy = 1.0
[Training set summary] [[4,0,0,5]]
[time taken] [0.003]
[total clauses constructed] [14]
```

```
[Rule 1] [Pos cover = 4 Neg cover = 0]
fillede(A,B) :-
    femme(A).
```

Cette règle n'est pas fausse si l'on considère uniquement les exemples qui nous avons fournit. Malgré tout, on voit bien que le sens de la règle ne correspond pas au sens que l'on veut donner

au prédicat. Nous avons donc utilisé une nouvelle liste de contre-exemples :

Listing 6 – "Exemples négatifs"

```
1 fillede(tom,ann).
2 fillede(bob,tom).
3 fillede(bob,mary).
4 fillede(bob,ann).
5 fillede(tom,mary).
6 fillede(ann,mary).
```

Et à l'éxécution nous avons obtenu après inférence deux nouvelles régles qui correspondent à ceux que l'on cherchait :

```
1 [Rule 1] [Pos cover = 2 Neg cover = 0]
2 fillede(A,B) :-
3    mere(B,A), femme(A).
4
5 [Rule 2] [Pos cover = 2 Neg cover = 0]
6 fillede(A,B) :-
7    pere(B,A), femme(A).
```

3 Les figures du Poker

L'objectif est de faire apprendre à une machine les différentes figures du Poker. Des fichiers squelettes nous été fournis. En les remplissant, nous avons ainsi pu obtenir :

Listing 7 – "entete.pl"

```
1
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 % Paramètres d'exécution
4 %
5
6 :- set(clauselength, 100).       % on borne la taille des clauses
7 :- set(minacc, 0.99).            % [0-1] on autorise x% de bruit au maximum
8 :- set(noise, 1).                % nb max d'e- pouvant etre couverts par une
      hypothese
9 %:- set(minposfrac, 0.03).       % pourcentage min de couverture de E+
10 :- set(minpos, 2).              % nb e+ minimal pour une hypothese
11 :- set(minscore, 0.1).          % score minimal pour une hypothese
12 :- set(verbose, 0).             % pas de blabla, on travaille
13 :- set(i, 3).                   % longueur max de connexion avec les var de
       tete
14 :- set(depth, 30).              % profondeur de preuve (pour couverture)
15 :- set(newvars, 20).            % nombre max var introduite
16 :- set(nodes, 100000).          % nb maximum de clauses construites dans
      la recherche
17 :- set(check_useless, true).    % pour avoir des var qui servent a qq chose
       ds bot
18 :- set(record, true).           % on garde une trace...
19 :- set(recordfile, 'test_modif_v2'). % ...dans ce fichier
20 :- set(evalfn, coverage).       % fonction d'evaluation de la qualite des
      clauses
```

```
21 :- set( lazy_on_contradiction , true).
22 :- set( search , bf ).       % shorter first: bf, best first: heuristic
23
24
25
26
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%% Definition du langage d'hypothese Lh
28 % Modes
29 %        modeh are mode declarations for head literals
30 %        modeb are mode declarations for body literals
31 %        + represents input arguments
32 %        - represents output arguments
33 %        + or - are followed by type of each argument
34 %        in this example, the type "int" is a built-in unary predicate
35
36 %:- modeh(1,carre(+jeu)).
37 %:- modeh(1,paire(+jeu)).
38 %:- modeh(1,double_paire(+jeu)).
39 %:- modeh(1,brelan(+jeu)).
40 :- modeh(1,suite(+jeu)).
41 %:- modeh(1,couleur(+jeu)).
42 %:- modeh(1,full(+jeu)).
43 %:- modeh(1,quinte_flush(+jeu)).
44 %:- modeh(1,quinte_royale(+jeu)).
45
46 :- modeb(1,cartes(+jeu,set(-c,-c,-c,-c))).
47 :- modeb(1,meme_hauteur(+c,+c)).
48 :- modeb(1,meme_couleur(+c,+c)).
49 :- modeb(1,a_suivre(+c,+c)).
50
51
52 %:- determination(carre/1,cartes/2).
53 %:- determination(carre/1,meme_hauteur/2).
54
55 %:- determination(paire/1,cartes/2).
56 %:- determination(paire/1,meme_hauteur/2).
57
58 %:- determination(double_paire/1,cartes/2).
59 %:- determination(double_paire/1,meme_hauteur/2).
60
61 :- determination(suite/1,cartes/2).
62 :- determination(suite/1,a_suivre/2).
63
64 %:- determination(couleur/1,cartes/2).
65 %:- determination(couleur/1,meme_couleur/2).
66
67 %:- determination(full/1,cartes/2).
68 %:- determination(full/1,meme_hauteur/2).
69
70 %:- determination(quinte_flush/1,cartes/2).
71 %:- determination(quinte_flush/1,meme_couleur/2).
72 %:- determination(quinte_flush/1,a_suivre/2).
73
```

```
74 %:- determination(quinte_royale/1,cartes/2).
75 %:- determination(quinte_royale/1,meme_couleur/2).
76 %:- determination(quinte_royale/1,a_suivre/2).
77
78 %:- determination(brelan/1,cartes/2).
79 %:- determination(brelan/1,meme_hauteur/2).
80
81
82
83
84 meme_hauteur(C1,C2) :-          valeur(C1,A),valeur(C2,A),\=(C1,C2).
85 meme_couleur(C1,C2) :-          couleur(C1,A),couleur(C2,A),\=(C1,C2).
86 a_suivre(C1,C2) :-              valeur(C1,sept),valeur(C2,huit),\=(C1,C2).
87 a_suivre(C1,C2) :-              valeur(C1,huit),valeur(C2,neuf),\=(C1,C2).
88 a_suivre(C1,C2) :-              valeur(C1,neuf),valeur(C2,dix),\=(C1,C2).
89 a_suivre(C1,C2) :-              valeur(C1,dix),valeur(C2,valet),\=(C1,C2).
90 a_suivre(C1,C2) :-              valeur(C1,valet),valeur(C2,dame),\=(C1,C2).
91 a_suivre(C1,C2) :-              valeur(C1,dame),valeur(C2,roi),\=(C1,C2).
92 a_suivre(C1,C2) :-              valeur(C1,roi),valeur(C2,as),\=(C1,C2).
93
94 cartes(Jeu,set(A,B,C,D,E)) :-   a_carte(Jeu,A),
95                                 a_carte(Jeu,B),\=(A,B),
96                                 a_carte(Jeu,C),\=(A,C),\=(B,C),
97                                 a_carte(Jeu,D),\=(A,D),\=(B,D),\=(C,D),
98                                 a_carte(Jeu,E),\=(A,E),\=(B,E),\=(C,E),\=(D,E
                                        ).
```

Listing 8 – "preparpokerdata.prl"

```
1  #!/usr/bin/perl -w
2
3  # Author: Vincent Claveau
4  # Contact: vincent.claveau@irisa.fr
5  # Last modif date: 22 January 2008
6
7  # Purpose: sert a preparer les donnees poker pour Aleph
8
9  # Comment:
10
11 # Code :
12
13
14 ##########################################################################
15 ##                              INITIALISATIONS                         ##
16 ##########################################################################
17
18 use strict;
19 use locale;
20 binmode(STDOUT,':utf8'); # si le terminal est en utf8
21 binmode(STDERR,':utf8'); # si le terminal est en utf8
22
23 my $unique = $$;
24 my $prg = $0;
25 if ($0 =~ /.+\/([^\/]+\.[^\.\/]+)$/)
```

```
26    { $prg = $1 }
27
28  my $verbose = 2;
29
30  my $infini = 1000000;
31
32
33  ######################################################################
34  # FICHIERS
35
36  my $file_in;
37
38  my $prefix = "poker";
39
40  ##############################################################################
41  # FLAG
42
43
44  ##############################################################################
45  # VARIABLES DIVERSES
46
47  # A CHANGER SELON VOTRE SITUATION
48
49  my $entete = "/home/nicolas/Bureau/tpacqdeco2/poker/entete.pl";
50  my $aleph = "/home/nicolas/Bureau/tpacqdeco2/aleph.pl";
51
52  ##############################################################################
53  # USAGE
54
55  my $usage = "Usage␣:␣$prg
56  Mandatory
57  argument
58  *\t␣-out␣␣␣␣␣␣␣<prefixe␣des␣fichiers␣de␣sortie>
59  *\t␣-in␣␣␣␣␣␣␣␣<fichier␣de␣donnees␣a␣transformer>
60
61  ␣\t␣-h␣␣␣␣␣␣␣␣:␣aide
62  \n";
63
64
65  die $usage unless ($#ARGV > -1);
66
67  while ($#ARGV > -1 and $ARGV[0] =~ /^-/) {
68    $_ = shift;
69    if (/^-out/)                { $prefix = shift }
70    elsif (/^-in/)              { $file_in = shift }
71
72    elsif (/^-?-he/)            { die $usage }
73
74    else                        { die "unknown␣option␣$_!\n␣$usage\n" }
75  }
76
77
```

```
78  ##############################################################################
79  ##############################################################################
80  ##
        ##
81  ##                                      BODY
                                                ##
82  ##
        ##
83  ##############################################################################
84  ##############################################################################
85
86  open(BK,">␣$prefix.b") or die "$prg␣unable␣to␣write␣$prefix.b␣$!\n";
87  print BK ":-␣['$entete'].\n\n";
88
89  open(IN,$file_in) or die "$prg␣unable␣to␣read␣$file_in␣$!\n";
90  open(POS,">␣$prefix.f") or die "$prg␣unable␣to␣write␣$prefix.f␣$!\n";
91  open(NEG,">␣$prefix.n") or die "$prg␣unable␣to␣write␣$prefix.n␣$!\n";
92  my $nb_ex = 0;
93  while(<IN>)
94    {
95      if (/^([1-4]),([0-9]+),([1-4]),([0-9]+),([1-4]),([0-9]+),([1-4]),([0-9]+)
            ,([1-4]),([0-9]+),([0-9])/)
96        {
97          my $couleur1 = Trans_couleur($1); my $valeur1 = Trans_valeur($2);
98          my $couleur2 = Trans_couleur($3); my $valeur2 = Trans_valeur($4);
99          my $couleur3 = Trans_couleur($5); my $valeur3 = Trans_valeur($6);
100         my $couleur4 = Trans_couleur($7); my $valeur4 = Trans_valeur($8);
101         my $couleur5 = Trans_couleur($9); my $valeur5 = Trans_valeur($10);
102         my $main = Trans_main($11);
103
104         $nb_ex++;
105
106         # ici completer en inscrivant les exemples (pos et neg)
107         # et leur description dans le BK, $nb_ex peut servir a identifier
108         # chaque instance
109
110         # pour comparer deux chaines : $toto eq "blablabla"
111         # pour tirer au hasard un element sur 10 : if (int rand 10 == 0)
112
113                 if($main eq 'suite'){
114                         print POS "suite(jeu$nb_ex).\n"
115                 }
116                 elsif(($main eq 'rien' and int rand 100==0) or
117                         (not $main eq 'rien' and int rand 10==0)){
118                                 print NEG "suite(jeu$nb_ex).\n"
119                 }
120
```

```
121                 if($main eq 'carre'){
122                         print POS "carre(jeu$nb_ex).\n"
123                 }
124                 elsif(($main eq 'rien' and int rand 100==0) or
125                         (not $main eq 'rien' and int rand 10==0)){
126                                         print NEG "carre(jeu$nb_ex).\n"
127                         }
128
129                 if($main eq 'couleur'){
130                         print POS "couleur(jeu$nb_ex).\n"
131                 }
132                 elsif(($main eq 'rien' and int rand 100==0) or
133                         (not $main eq 'rien' and int rand 10==0)){
134                                         print NEG "couleur(jeu$nb_ex).\n"
135                 }
136
137                 if($main eq 'full'){
138                         print POS "full(jeu$nb_ex).\n"
139                 }
140                 elsif(($main eq 'rien' and int rand 100==0) or
141                         (not $main eq 'rien' and int rand 10==0)){
142                                         print NEG "full(jeu$nb_ex).\n"
143                 }
144
145                 if($main eq 'quinte_flush'){
146                         print POS "quinte_flush(jeu$nb_ex).\n"
147                 }
148                 elsif(($main eq 'rien' and int rand 100==0) or
149                         (not $main eq 'rien' and int rand 10==0)){
150                                         print NEG "quinte_flush(jeu$nb_ex).\n
                                                "
151                 }
152
153                 if($main eq 'quinte_royale'){
154                         print POS "quinte_royale(jeu$nb_ex).\n"
155                 }
156                 elsif(($main eq 'rien' and int rand 100==0) or
157                         (not $main eq 'rien' and int rand 10==0)){
158                                         print NEG "quinte_royale(jeu$nb_ex).\
                                                n"
159                 }
160
161                 if($main eq 'brelan'){
162                         print POS "brelan(jeu$nb_ex).\n"
163                 }
164                 elsif(($main eq 'rien' and int rand 100==0) or
165                         (not $main eq 'rien' and int rand 10==0)){
166                                         print NEG "brelan(jeu$nb_ex).\n"
167                 }
168
169                 if($main eq 'double_paire'){
170                         print POS "double_paire(jeu$nb_ex).\n"
171                 }
```

```
172                 elsif(($main eq 'rien' and int rand 100==0) or
173                         (not $main eq 'rien' and int rand 10==0)){
174                                         print NEG "double_paire(jeu$nb_ex).\n
                                                "
175                 }
176
177                 if($main eq 'paire'){
178                         print POS "paire(jeu$nb_ex).\n"
179                 }
180                 elsif(($main eq 'rien' and int rand 100==0) or
181                                 (not $main eq 'rien' and int rand 10==0)){
182                                         print NEG "paire(jeu$nb_ex).\n"
183                         }
184
185                         print BK "a_carte(jeu$nb_ex,carte1_$nb_ex).\n";
186                         print BK "a_carte(jeu$nb_ex,carte2_$nb_ex).\n";
187                         print BK "a_carte(jeu$nb_ex,carte3_$nb_ex).\n";
188                         print BK "a_carte(jeu$nb_ex,carte4_$nb_ex).\n";
189                         print BK "a_carte(jeu$nb_ex,carte5_$nb_ex).\n";
190                         print BK "couleur(carte1_$nb_ex,$couleur1).\n";
191                         print BK "couleur(carte2_$nb_ex,$couleur2).\n";
192                         print BK "couleur(carte3_$nb_ex,$couleur3).\n";
193                         print BK "couleur(carte4_$nb_ex,$couleur4).\n";
194                         print BK "couleur(carte5_$nb_ex,$couleur5).\n";
195                         print BK "valeur(carte1_$nb_ex,$valeur1).\n";
196                         print BK "valeur(carte2_$nb_ex,$valeur2).\n";
197                         print BK "valeur(carte3_$nb_ex,$valeur3).\n";
198                         print BK "valeur(carte4_$nb_ex,$valeur4).\n";
199                         print BK "valeur(carte5_$nb_ex,$valeur5).\n";
200
201           # fin de la partie a completer
202         }
203       else { print STDERR "$prg: pb ligne au format inconnu :\n$_\n\n" }
204     }
205   close IN;
206   close POS;
207   print NEG "\n";
208   close NEG;
209   close BK;
210
211
212   # un fichier pour tout lancer sans s'embeter a tout taper
213   # il n'y aura plus qu'a faire yap -l xxxx.start
214   # et taper 'induce.' ou 'sat(1), reduce.' a l'invite de yap
215   open(START,"> $prefix.start") or die "$prg unable to write $prefix.start $!\n
          ";
216   print START ":- ['$aleph'].\n";
217   print START ":- read_all($prefix).\n";
218   print START ":- set(recordfile,'$prefix.res').\n";
219   print START ":- set(verbose,0).\n";
220   close START;
221
222
```

```
223
224
225
226 ################################################################################
227 ################################################################################
228 ##
       ##
229 ##                                  PROCEDURES
                                              ##
230 ##
       ##
231 ################################################################################
232 ################################################################################
233
234
235
236 sub Trans_valeur {
237   my $valeur = shift;
238
239   if ($valeur == 1)      { $valeur = "as"}
240   elsif ($valeur == 2)   { $valeur = "deux"}
241   elsif ($valeur == 3)   { $valeur = "trois"}
242   elsif ($valeur == 4)   { $valeur = "quatre"}
243   elsif ($valeur == 5)   { $valeur = "cinq"}
244   elsif ($valeur == 6)   { $valeur = "six"}
245   elsif ($valeur == 7)   { $valeur = "sept"}
246   elsif ($valeur == 8)   { $valeur = "huit"}
247   elsif ($valeur == 9)   { $valeur = "neuf"}
248   elsif ($valeur == 10)  { $valeur = "dix"}
249   elsif ($valeur == 11)  { $valeur = "valet"}
250   elsif ($valeur == 12)  { $valeur = "dame"}
251   elsif ($valeur == 13)  { $valeur = "roi"}
252
253   return $valeur;
254 }
255
256
257 sub Trans_couleur {
258   # peu importe l'ordre en fait
259 ␣␣my␣$valeur␣=␣shift;
260
261 ␣␣if␣($valeur␣==␣1)␣␣␣␣␣␣{␣$valeur␣=␣"coeur"}
262 ␣␣elsif␣($valeur␣==␣2)␣␣␣{␣$valeur␣=␣"pique"}
263 ␣␣elsif␣($valeur␣==␣3)␣␣␣{␣$valeur␣=␣"carreau"}
264 ␣␣elsif␣($valeur␣==␣4)␣␣␣{␣$valeur␣=␣"trefle"}
265
266 ␣␣return␣$valeur;
```

```
}


sub␣Trans_main␣{
␣␣my␣$main␣=␣shift;

␣␣if␣($main␣==␣0)␣␣␣␣␣␣{␣$main␣=␣"rien"␣}
␣␣elsif␣($main␣==␣1)␣␣␣{␣$main␣=␣"paire"␣}
␣␣elsif␣($main␣==␣2)␣␣␣{␣$main␣=␣"double_paire"␣}
␣␣elsif␣($main␣==␣3)␣␣␣{␣$main␣=␣"brelan"␣}
␣␣elsif␣($main␣==␣4)␣␣␣{␣$main␣=␣"suite"␣}
␣␣elsif␣($main␣==␣5)␣␣␣{␣$main␣=␣"couleur"␣}
␣␣elsif␣($main␣==␣6)␣␣␣{␣$main␣=␣"full"␣}
␣␣elsif␣($main␣==␣7)␣␣␣{␣$main␣=␣"carre"␣}
␣␣elsif␣($main␣==␣8)␣␣␣{␣$main␣=␣"quinte_flush"␣}
␣␣elsif␣($main␣==␣9)␣␣␣{␣$main␣=␣"quinte_royale"␣}

␣␣return␣$main;
}
```

Voici le résultat pour l'inférence de la règle suite.

```
[theory]

[Rule 207] [Pos cover = 44 Neg cover = 0]
suite(A) :-
   cartes(A,set(B,C,D,E,F)), a_suivre(E,D), a_suivre(D,B), a_suivre(C,F),
   a_suivre(B,C).

[Training set performance]
              Actual
         +            -
     +  44            0           44
Pred
     -  12473         10974       23447

        12517         10974       23491

Accuracy = 0.469030692605679
[Training set summary] [[44,0,12473,10974]]
[time taken] [11.83]
[total clauses constructed] [190]
yes
```

En commentant et décommentant tour è tour les différentes parties du code, on peut obtenir toutes les règles du poker !