

Rapport TP RCR

TP :

Inférence logique basée sur un solveur SAT

Travail réalisé par : Sophinez Azouaou 181833011664

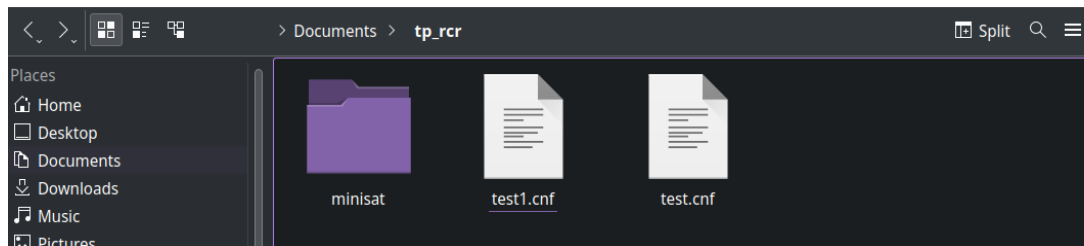
Aouabed Samy Aghiles 181831084214

Section M1. IV,

Groupe 1

Etape 1 :

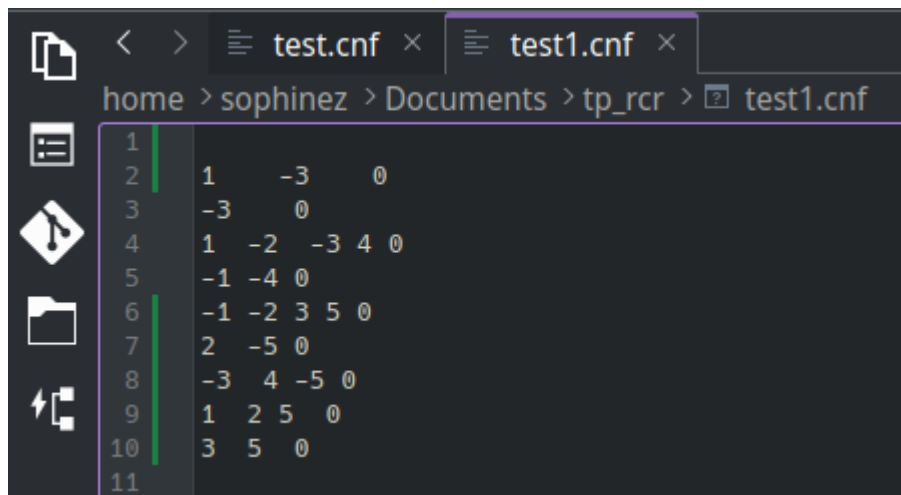
Création d'un répertoire et copie des fichiers cnf :



NB : Le TP est réalisé sur une machine Linux, le solveur Minisat a été utilisé dans ce TP.

Etape 2 : Test de la base test1.cnf + test2.cnf

test1.cnf se présente comme suit :



Exécution du solveur Minisat sur le fichier :

```
~/Documents/tp_rcr minisat test1.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
|
| Number of variables:          5
| Number of clauses:           5
| Parse time:                   0.00 s
| Eliminated vars:              2
| Vars set :                    3
| Eliminated clauses:           0.00 Mb
| Simplification time:          0.00 s
|
===== [ Search Statistics ] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
|           | Vars     | Clauses | Literals | Limit   | Clauses | Lit/Cl |
=====
restarts      : 1
conflicts     : 0 (-nan /sec)
decisions     : 1 (0.00 % random) (inf /sec)
propagations  : 3 (inf /sec)
conflict literals : 0 (-nan % deleted)
Memory used   : 10.27 MB
CPU time      : 0 s
SATISFIABLE
```

La base est satisfiable, et le solveur a fourni un modèle :

```
< > test.cnf x test1.cnf x res1.cnf x
home > sophinez > Documents > tp_rcr > res1.cnf
1 SAT
2 -1 2 -3 -4 5 0
3
```

Le modèle correspond a : $\neg a \vee b \vee \neg c \vee \neg d \vee e$

Test de la base test2.cnf

test2.cnf se présente comme suit :

```
< > test.cnf x test1.cnf x res1.cnf x test2.cnf x
home > sophinez > Documents > tp_rcr > test2.cnf
1
2 2 -3 0
3 -3 0
4 1 -2 -3 4 0
5 -1 -4 0
6 2 -4 0
7 1 3 0
8 -1 -2 3 5 0
9 2 -5 0
10 -3 4 -5 0
11 1 2 5 0
12 3 5 0
13 -5 0
14
```

Après exécution sur le solveur minisat, on a :

```
~/Documents/tp_rcr minisat test2.cnf res2.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
| Number of variables:      5 |
| Number of clauses:       5 |
| Parse time:              0.00 s |
| Simplification time:     0.00 s |
=====
Solved by simplification
restarts      : 0
conflicts    : 0 (0 /sec)
decisions    : 0 (-nan % random) (0 /sec)
propagations : 5 (1133 /sec)
conflict literals : 0 (-nan % deleted)
Memory used  : 10.27 MB
CPU time     : 0.004413 s
UNSATISFIABLE
```

Nous constatons que la base test2.cnf n'est, en effet, pas satisfiable.

Etape 3 :

Partie 1 : traduction de la base de connaissances relative aux connaissances zoologiques vu en cours sous format cnf

→ Nous possédons deux traductions :

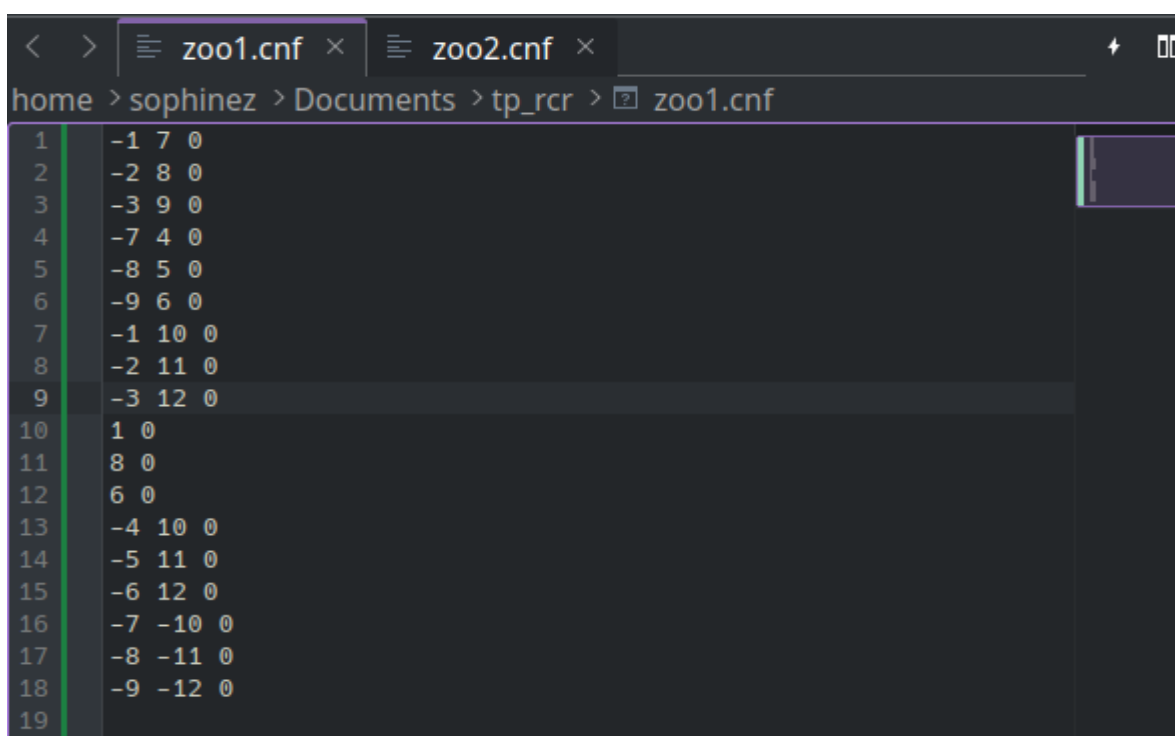
1. Avant ne tenir compte du mot "généralement"
2. Après modification et tenir compte du mor "généralement"

Pour effectuer la traduction, nous mettons :

- Na = 1
- Nb = 2
- Nc = 3
- Ma = 4
- Mb = 5
- Mc = 6
- Cea = 7
- Ceb = 8
- Cec = 9
- Coa = 10
- Cob = 11
- Coc = 12

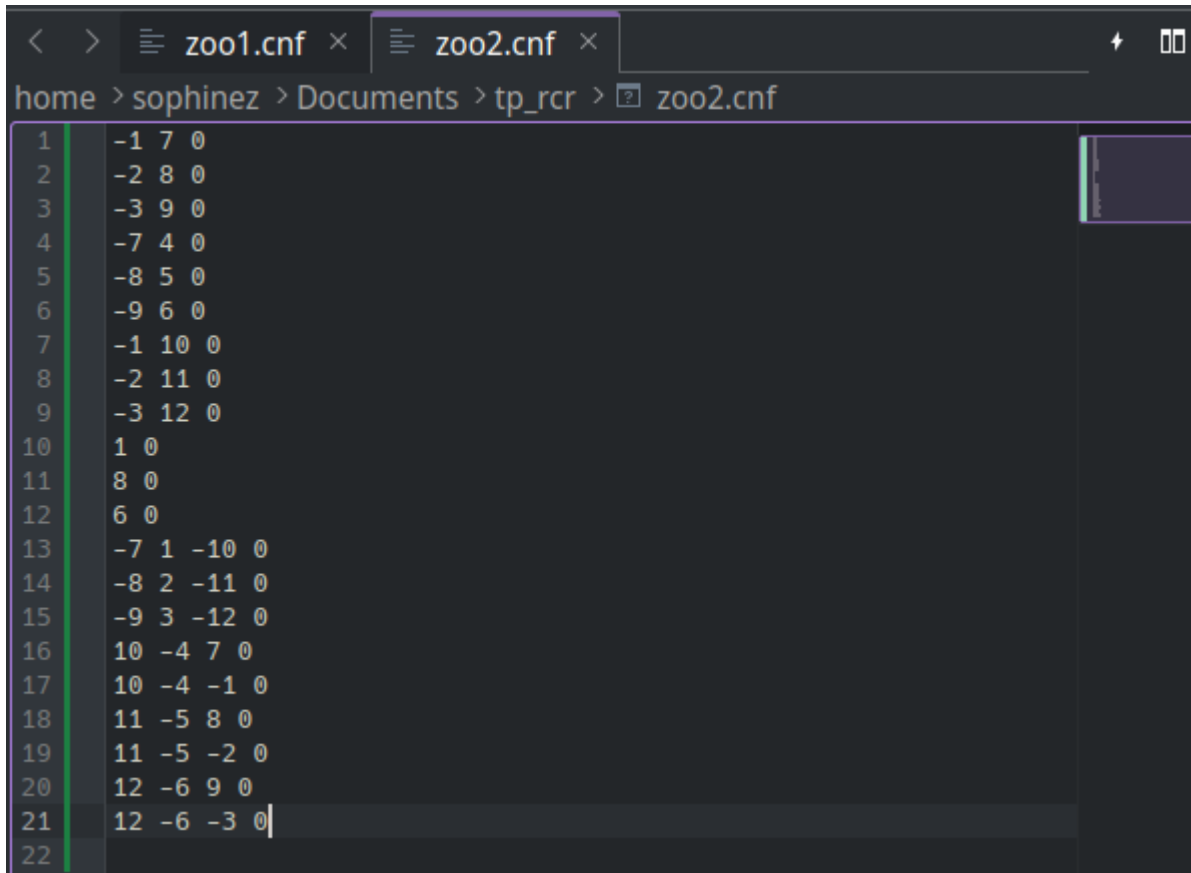
Nous aurons les **traductions** suivantes :

1. Avant modification :



```
< > zoo1.cnf x zoo2.cnf x
home > sophinez > Documents > tp_rcr > ? zoo1.cnf
1 -1 7 0
2 -2 8 0
3 -3 9 0
4 -7 4 0
5 -8 5 0
6 -9 6 0
7 -1 10 0
8 -2 11 0
9 -3 12 0
10 1 0
11 8 0
12 6 0
13 -4 10 0
14 -5 11 0
15 -6 12 0
16 -7 -10 0
17 -8 -11 0
18 -9 -12 0
19
```

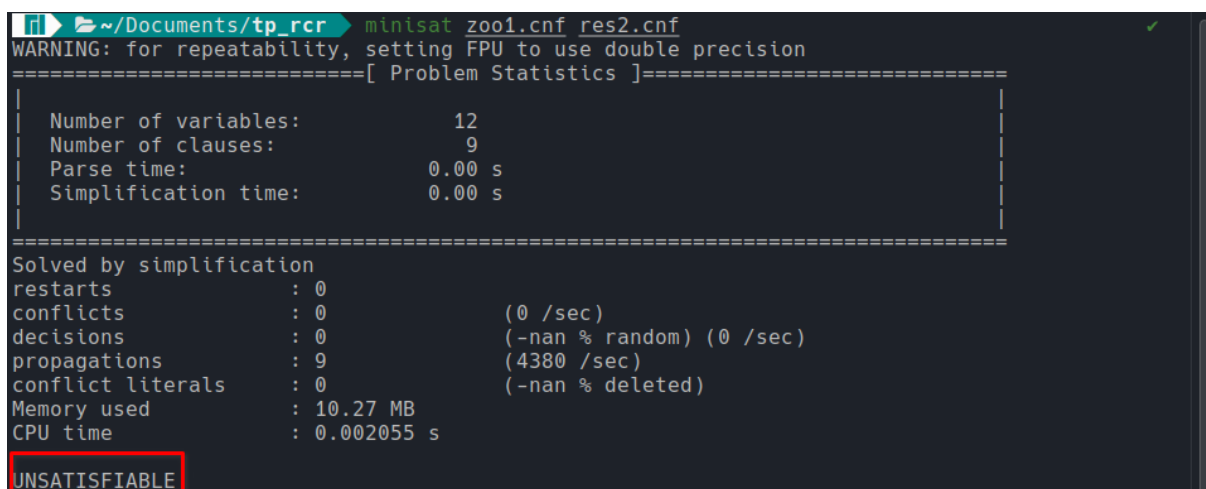
2. Après modification :



```
< > zoo1.cnf x zoo2.cnf x
home > sophinez > Documents > tp_rcr > ? zoo2.cnf
1 -1 7 0
2 -2 8 0
3 -3 9 0
4 -7 4 0
5 -8 5 0
6 -9 6 0
7 -1 10 0
8 -2 11 0
9 -3 12 0
10 1 0
11 8 0
12 6 0
13 -7 1 -10 0
14 -8 2 -11 0
15 -9 3 -12 0
16 10 -4 7 0
17 10 -4 -1 0
18 11 -5 8 0
19 11 -5 -2 0
20 12 -6 9 0
21 12 -6 -3 0
22
```

→ Nous procédons avec l'exécution sur le solveur SAT :

1. Première traduction :



```
~/Documents/tp_rcr minisat zoo1.cnf res2.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
|
| Number of variables:      12
| Number of clauses:       9
| Parse time:              0.00 s
| Simplification time:     0.00 s
|
=====
Solved by simplification
restarts      : 0
conflicts    : 0          (0 /sec)
decisions    : 0          (-nan % random) (0 /sec)
propagations : 9          (4380 /sec)
conflict literals : 0      (-nan % deleted)
Memory used  : 10.27 MB
CPU time     : 0.002055 s
UNSATISFIABLE
```

La base est non satisfiable.

2. Deuxième traduction :

```
~/Documents/tp_rcr minisat zoo2.cnf res3.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
|
| Number of variables:      12
| Number of clauses:       14
| Parse time:              0.00 s
| Eliminated vars:         5
| Vars set :               7
| Eliminated clauses:      0.00 Mb
| Simplification time:     0.00 s
|
===== [ Search Statistics ] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
|           | Vars  Clauses Literals | Limit  Clauses Lit/Cl |
=====
restarts      : 1
conflicts     : 0          (0 /sec)
decisions     : 1          (0.00 % random) (468 /sec)
propagations  : 7          (3273 /sec)
conflict literals : 0      (-nan % deleted)
Memory used   : 10.27 MB
CPU time      : 0.002139 s
SATISFIABLE
```

La base est cohérente après avoir tenu compte explicitement des exceptions, donc nous trouvons qu'elle est satisfiable.

Etape 4 :

Dans cette partie le but est de vérifier l'inférence d'une base de connaissances.

Etant donné une base de connaissance, nous voulons tester si ladite base de connaissance infère une formule F.

Pour faire ceci, nous écrivons un algorithme en python qui suit le raisonnement suivant :

```
from pysat.formula import CNF
from pysat.solvers import Solver
import sys

args = sys.argv

# prendre le fichier depuis lequel on lis (.cnf)
base = CNF(from_file=args[1])

litteral = int(args[2])
litteral_neg = litteral * -1

s = Solver()
for c in base.clauses :
    s.add_clause(c)

s.add_clause([litteral_neg])

# tester si la base infere
if(s.solve()):
    print(f'la base de connaissance n infere pas le literal {litteral}')
else:
    print(f'la base de connaissance infere {litteral}')
```

On l'applique sur la base de connaissance suivante:

```
-1 7 0  
-2 8 0  
-3 9 0  
-7 4 0  
-8 5 0  
-9 6 0  
-1 10 0  
-2 11 0  
-3 12 0  
1 0  
8 0  
6 0  
-7 1 -10 0  
-8 2 -11 0  
-9 3 -12 0  
10 -4 7 0  
10 -4 -1 0  
11 -5 8 0  
11 -5 -2 0  
12 -6 9 0  
12 -6 -3 0
```

Une fois le script exécuté, on obtient le résultat suivant:

```
PS C:\Users\    \Desktop\Nouveau dossier (3)> python .\inference_script.py zoo2.cnf 1  
la base de connaissance infere 1  
PS C:\Users\    \Desktop\Nouveau dossier (3)> █
```