

**TP N° 1 : Partie 1**  
**Inférence**  
**Logique propositionnelle**  
**basée sur un solveur SAT**

**Etape 1:**

- Créer un répertoire UBCSAT par exemple,
- Copier les fichiers: ubcsat (qui représente le solveur) et les deux fichiers CNF.

**Etape 2:**

Afin d'exécuter le solveur SAT, il faut activer l'invite commande, en sélectionnant la touche windows, Accessoires, Invite de commande.

L'exécution du solveur se fait comme suit:

C:\UBCSAT> ubcsat -alg saps -i test.cnf -solve

La base de connaissances doit être sous la forme CNF:

**Exemple 1 :**

Soit la base test1.cnf suivante :

p	cnf	5	9	
2	-3	0		$a \vee \neg c$
-3	0			$\neg c$
1	-2	-3	4	$a \vee \neg b \vee \neg c \vee d$
-1	-4	0		$\neg a \vee \neg d$
-1	-2	3	5	$\neg a \vee \neg b \vee \neg c \vee e$
2	-5	0		$b \vee \neg e$
-3	4	-5	0	$\neg c \vee d \vee \neg e$
1	2	5	0	$a \vee b \vee e$
-3	5	0		$\neg c \vee e$

La première ligne est définie par: p cnf nombre\_variables nombre\_clauses

Le nombre de clauses doit être exact.

L'exemple représente une base ayant 5 variables et 11 clauses. Chaque clause se termine par

0. -1 représente l'instance  $\neg a$

5 représente l'instance e.

La base test1.cnf est satisfiable. Le solveur fournit diverses modèles :  $\neg a \vee b \vee \neg c \vee d \vee e$  ;

$\neg a \vee b \vee \neg c \vee \neg d \vee e$  ;  $\neg a \vee b \vee \neg c \vee d \vee \neg e$ ,...

**Exemple 2 :**

Soit la base test2.cnf suivante :

```
P cnf 5 12
2 -3 0
-3 0
1 -2 -3 4 0
-1 -4 0
```

```

2 -4 0
1 3 0
-1 -2 3 5 0
2 -5 0
-3 4 -5 0
1 2 5 0
3 5 0
-5 0

```

La base test2.cnf n'est pas satisfiable.

### **Etape 3:**

- Traduire la base de connaissances relative aux connaissances zoologiques (céphalopodes) sous forme CNF, puis tester la satisfiabilité de cette base.  
Remarque : Pour la mise sous forme CNF, il faudrait transformer l'implication en une disjonction :  $(a \supset b) \equiv (\neg a \vee b)$ .
- Télécharger des fichiers Benchmarks sous forme CNF afin de tester leur satisfiabilité en utilisant le solveur ubcsat.

### **Etape 4:**

Ecrire un algorithme pour simuler l'inférence d'une base de connaissances.

Soit BC une base de connaissances et soit  $\phi$  une formule. Pour tester si BC infère  $\phi$ , on utilisera le raisonnement par l'absurde. Ce qui revient à tester si  $BC \cup \{\neg \phi\}$  infère  $\perp$ .

Algorithme de raisonnement par l'absurde

Begin

Input :

BC sous forme CNF

Un littéral  $\phi$

$(BC \vdash \phi) \equiv BC \cup \neg \phi \vdash \perp$

Insérer le littéral dans la base

Call SAT( $BC \cup \neg \phi \vdash \perp$ )

if ( $(BC \cup \neg \phi)$  est insatisfiable )

alors

BC  $\vdash \phi$

Else

BC non  $\vdash \phi$

endif

End

<p align="center"><b>TP N° 1 : Partie 2</b>  <b>Inférence</b>  <b>logique des prédicats</b></p>
---

Il s'agit d'exploiter la librairie tweety pour la modélisation des connaissances en logique des prédicats.

La librairie Java tweety est dédiée aux modes logiques dans le domaine de la représentation des connaissances (Logique Propositionnelle, Logique des prédicats, Logique modale, Logique des défauts et Logique de description) <https://tweetyproject.org/>.