Lab07: Pre-processing SAT

1. **Sobre preprocesar instancias de SAT**

   Para este problema usaremos el mismo formato del laboratorio 4 para representar fórmulas booleanas en CNF.

   - `num_variables` contiene el número de variables que pueden aparecer en la fórmula. Las variables siempre están numeradas de 1 a `num_variables`.
   - La fórmula Booleana está representada como una lista de listas (cláusulas). Cada lista interna correponde con una única cláusula de manera que cada literal x_i de la cláusula se representa con un i y cada literal $\neg x_i$ se representa con un $-i$.
   - Por ejemplo, la fórmula Booleana

     $$\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

     estaría representada por

     `num_variables` = 4

     `clauses` = [[1,2,-3],[2,-4],[-1,3,4]].
   - Para facilitar el proceso, las asignaciones `A` serán listas con `len(A) = num_variables + 1`. Por convenio, siempre `A[0] = 0`, ya que no hay ninguna variable x_0. Una asignación posible para las variables de φ podría ser `A=[0,1,0,1,0]`, de forma que `A`(x_1)`=1, A`(x_2)`=0, A`(x_3)`=1, A`(x_4)`=0`.

   Para realizar el ejercicio disponéis del fichero `lab07_pre_processing.py`.

   Debes implementar la función `sat_preprocessing`, que dada una fórmula booleana en el formato anterior y el número de variables sobre el que la fórmula está definida, aplica varias reglas para reducir la fórmula.

   Las reglas son las siguientes:

   (a) Si hay alguna cláusula formada por una única variable (negada o no), asigna a dicha variable el valor de verdad para que la cláusula se haga cierta.

   (b) Si una variable aparece sólo una vez y no se le ha asignado previamente un valor de verdad, asigna a dicha variable el valor de verdad para que la cláusula se haga cierta.

   (c) Elimina todas las cláusulas que se hayan hecho ciertas

   (d) Elimina todos los literales que evalúen a `False`. Es decir, todas las variables x que tengan asignado un 0 y todas las $\neg x$ que tengan asignado un `True`. Como resultado a este proceso, si una cláusula se reduce a [ ], la fórmula de entrada es insatisfactible y se debe devolver la fórmula [[1],[-1]]

Tu implementación debe recoger el hecho de que las reglas se deben aplicar exhaustivamente. Esto es, hasta que no se pueda aplicar más. Ten en cuenta que después de aplicar las reglas (a), (b), (c), y (d) puede ocurrir que haya nuevas variables que aparezcan sólo una vez y que haya cláusulas formadas por una única variable, por lo que se pueden volver a aplicar las reglas.

Cuando el pre-proceso haya acabado, puede ser que la fórmula resultante sea satisfactible trivialmente. En ese caso debes devolver [ ]. Si la fórmula resultante es insatisfactible debes devolver [[1],[-1]]. En otro caso devolverás la lista de cláusulas resultante.

2. **Juego de pruebas y MINISAT**

   El fichero `instances` en la carpeta `Code for Students` contiene un juego de pruebas (`https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`).

   Cada instancia está en un fichero de texto en formato DIMACS (éstandar para la mayoría de los SAT-solvers). Este formato es similar al nuestro pero requiere una cabecera inicial con la palabra "p cnf" y el número de variable y cláusulas de la fórmula.

   MINISAT

   Vamos a ver cómo funciona **minisat** con la primera fórmula del fichero `instancias`: La fórmula `1-unsat.cnf` es insatisfactible. Teclea en un terminal:

   ```
   minisat 1-unsat.cnf
   ```

   En caso de que la fórmula sea satisfactible, puedes obtener una asignación a las variables que la haga cierta. Si escribes

   ```
   minisat 2-sat.cnf asig-2.txt
   ```

   se creará un fichero `asig-2.txt` donde se encuentra la asignación.

   Uso del juego de pruebas

   Puedes usar el juego de pruebas para probar este laboratorio. La fórmula `4-sat.cnf` es la única que admite un preprocesado.

   Para añadir este juego a tus pruebas hay que transformar el formato DIMACS en nuestro formato. Para ello hemos creado una función

   ```
   list_minisat2list_our_sat()
   ```

   Esta función, dado un fichero en formato DIMACS, devuelve una tupla (t1, t2) donde t1 = número de variables y t2 = fórmula en nuestro formato. Para usarla hay que copiar el fichero `tools.pyc` que se encuentra en la carpeta `Code For Students` e incluir from tools import list_minisat2list_our_sat.

   Para ver cómo se preprocesa la fórmula `1-unsat.cnf` puedes usar la función de esta manera:

   ```
   tupla = list_minisat2list_our_sat ('instancias/1-unsat.cnf')
   ```

   y luego ver el resultado de tu preprocesado:

   ```
   print sat_preprocessing(tupla[0], tupla[1])
   ```