Lab09: SAT-solver

1. **Formato de las fórmulas**

   Recuerda la representación de las 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 $-i$.
   - 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`.

2. **Implementación del SAT-solver**

   Implementa la función **solve_SAT**(num_variables, φ) que se aplica a una fórmula CNF, φ, de forma que, tras realizar un pre-processing, usa un árbol de búsqueda para construir una asignación que haga cierta a φ o asegurar que φ es UNSATISFIABLE. Por tanto tu función debe devolver o bien una asignación o UNSATISFIABLE. Para ello, abre el fichero `Lab09_SAT_solver.py` que se encuentra en la carpeta `Code for Students`.

   Valora si es conveniente hacer pre-processing sólo una vez y al principio o cada vez que asignas un nuevo valor a una variable.

3. **Juego de pruebas y MINISAT**

   La carpeta `instances` en la carpeta `Code for Students` contiene un juego de pruebas Cada instancia está en un fichero de texto en formato DIMACS (éstandar para la mayoría de los SAT-solvers).

MINISAT

El funcionamiento de `minisat` está explicado en el laboratorio 8.

Uso del juego de pruebas

Puedes usar el juego de pruebas para probar este laboratorio.

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. Esta función se encuentra en el fichero `tools.py` que está en la carpeta `Code For Students`. Por ello se ha incluido la cabecera

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 ver el resultado de tu SAT-solver: print(**solve_SAT**(tupla[0], tupla[1])).

4. Comprobación del tiempo de ejecución de tu SAT-solver

Se valorará el tiempo de ejecución de tu SAT-solver. Para que te hagas una idea de cuánto debería tardar tu solución, hemos dejado un documento, en la carpeta `instances`, con los tiempos que tarda un SAT-solver hecho por nosotros. Ten en cuenta que estos tiempos dependen del ordenador que estés usando.

Además del código entrega una tabla con los tiempos que tarda tu programa con el juego de pruebas de la carpeta `instances`.