Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Primer Semestre de 2021

IIC 2213 – Lógica para ciencia de la Computación
Tarea 5 - Entrega Lunes 25 de Mayo a las 20:00 - via siding

Recuerda que esta tarea es individual. Puedes discutir sobre la respuesta con tus compañeros (¡y eso está muy bien!), pero no puedes enviar la respuesta o utilizar la respuesta de alguien más. Hace bien escribir un poco de código para recordarnos que los computadores si existen y el mundo no es solo máquinas de turing.

La meta es programar dos programas (llamados usualmente solvers) que puedan decidir si una formula es inconsistente: uno que funcione por fuerza bruta, probando todas las valuaciones, y otro a decisión tuya: resolución, DPLL o lo que quieras.

Formato

Tu programa solo va a recibir fórmulas en CNF. El formato de las formulas es una versión simplificada de DIMACS, el formato que se usa normalmente en solvers profesionales. Acá va un archivo de ejemplo, que codifica a la fórmula $(x_1 \vee \neg x_5 \vee x_4) \wedge (\neg x_1 \vee x_5 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee \neg x_4)$:

```
c esto es un comentario
c
c esto también es un comentario
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
%
0
```

Las reglas del input son:

- Al principio hay un numero arbitrario de líneas que comienzan con c, todas estas son comentarios y deben ignorarse.
- La primera línea siempre es `p cnf <nprop> <nclaus>`, donde `<nprop>` es el número de proposiciones y `<nclaus>` es el número de cláusulas de la fórmula. Puedes asumir que todas las variables siempre aparecen en al menos una cláusula.
- Luego es una línea por cláusula. Cada una de estas es una secuencia de números entre $[-\texttt{nprop}, -1] \cup [1, \texttt{nprop}]$, y que termina en un 0. Un número positivo representa a la proposición asociada a ese número, y un número negativo a la negación de esa proposición.
- Puedes asumir también que los números no se repiten en una misma cláusula, ni tampoco las cláusulas contienen un número y su negativo al mismo tiempo.
- Finalmente, el archivo termina con un caracter % seguido de un 0.

Código

Importante. No puedes usar librerías con funciones específicas para el manejo de lógica o de solvers para lógica.

Debes generar un código en python donde se vean claramente tres funciones:

- Una función dimacs(texto) que reciba el nombre de un archivo que codifica una formula (se asume que el archivo está en la misma carpeta que el código), lo abra y lo transforme en un formato a tu elección (arreglo, lista, diccionario, lo que sea), que va a contener tu fórmula ya procesada.
- Una función fuerzabruta(formula) que reciba una fórmula ya procesada y entregue un 1 si la fórmula es satisfacible o un 0 si no lo es. Este algoritmo debe funcionar probando las valuaciones, una por una, hasta que alguna satisfaga a la fórmula (o retorne 0 si ninguna lo hace).
- Una función mejorado(formula) que reciba una fórmula ya procesada y entregue un 1 si la fórmula es satisfacible o un 0 si no lo es. Este algoritmo debe incorporar al menos una mejora importante a fuerza bruta, puede ser DPLL, resolución, o algo aún mejor.

Pruebas

Junto con este enunciado, hay tres set de tests en formato DIMACS. Luego, tendrás que hacer lo siguiente:

- Probar tus dos algoritmos, midiendo el tiempo que toma cada algoritmo en los tests. Prueba primero por los tests con 20 proposiciones antes de ir a los de 50.
- Anotar esos tiempos en una tabla.
- Producir un análisis (digamos de 2-4 párrafos). ¿Que te parecen estos tiempos? ¿Esperable o sorpresa? ¿por qué?

Formato de entrega

Un archivo pdf con el análisis. Un archivo .py con las tres funciones.

Nota

Una entrega completa significa un 7. Una entrega incompleta, un 1. Verificaremos un porcentaje de los códigos para asegurarnos que compilan y que los tiempos reportados son razonables, y si el código no nos compila (previa confirmación por correo para arreglar algún problema) bajamos la nota a 1.