UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Integrantes:
Marco Silva Huerta
Adrián Aguilera Moreno

Lógica Computacional

Práctica 3

- `fnn`
  - Usamos las funciones: elimImpl, elimEquiv, deMorgan para obetner la forma normal negativa, que estamos eliminando las implicaciones y equivalencias para después aplicar ley de De Morgan y negar hasta que solo variables atómicas lo estén.
- `fnc`
  - Aplicamos la forma normal conjuntiva más dos funciones auxiliares, una para que regresa un True si es que encuentra conjunciones y la otra que hace ley de distribución sobre la formula que se le ha pasado
- `unit` : en busca de aplicar la regla del unitario se emplea una función auxiliar que `elimina` un elemento de una lista. Primero se verifica que sea la cabeza de la lista, en ese caso se mantiene la cola de la lista y se aplica nuevamente el proceso (pues el elemento puede estar más de una vez en la lista, en teoría no queremos esto y buscamos evitar clausulas duplicadas).
- `elim` : esta función aplica la regla de eliminación en el algoritmo `DPLL`. Esta función emplea una función auxiliar `elimPropRep` que descarta a la cabeza de la lista siempre que esta se encuentre contenida en la cola de la misma, en otro caso se conserva la cabeza de la lista y se aplica la llamada recursiva de `elimPropRep` a la cola de la misma. Esta función es muy parecida a `elimina`, pero en este caso no pasamos el elemento como parámetro.
- `red` : con esta función buscamos aplicar la regla de reducción en el algoritmo `DPLL`. Para esto empleamos 3 funciones auxiliares:
  - → `elimVar` : esta función elimina la literal que se le pasa como parámetro de la clausula que, de igual manera, se le pasa como parámetro.
  - → `neg` : Esta función elimina la proposición que se le pasa como parámetro, sin importar la estructura de esta.
  - → `dif` : realiza la diferencia entre listas, esta función se implementa bajo la idea de que nuestras listas tengan operaciones como los conjuntos (esto nos servirá para evitar repeticiones por ejemplo), pero con algún orden.
- `split`
  - La regla consiste en crear dos ramas, una positiva y otra negativa sacando únicamente una variable de la formula en este caso tomamos la primera y hacemos los dos caminos negando la variable de la segunda, hacemos una lista de las formulas y como sabemos que la primera variable de la formula será a la que le hacemos ramas, usando una función *cabeza* obtenemos la letra que después pasamos a variable.
- `conflict` : si se llega a la clausula vacia o no se obtiene algún modelo, entonces se lanza `True` como respuesta. En otro caso, se regresa `False`.
- `succes` : en caso de que nuestra fórmula se vuelva la vacía, hemos encontrado un modelo y `succes` devuelve `True`, en otro caso devuelve `False`.
- `appDPLL` : solo manda a llamar las funciones ya implementadas.

Matriculas:

1. Marco Silva Huerta: 316205326.
2. Adrian Aguilera Moreno: 421005200.