Facultad de Ciencias - UNAM
Lógica Computacional 2020-2
Práctica 2: SAT y el algoritmo DPLL.
Parte 2

Favio E. Miranda Perea
Alejandra Krystel Coloapa Díaz
Pedro Juan Salvador Sánchez Pérez

19 de marzo de 2020
fecha de entrega: 30/03/2020

Haciendo uso del algoritmo DPLL de la nota 4, el objetivo es poder resolver el problema de satisfacibilidad para la lógica proposicional. Hay que recordar que las fórmulas proposicionales deben de estar en forma normal conjuntiva.

En esta segunda parte se centrarán en implementar el mecanismo necesario para usar las reglas del algoritmo DPLL y así poder hacer la búsqueda de un modelo que satisfaga una formula dada. La idea es hacer una búsqueda recursiva (Backtracking) para encontrar el modelo.

1. Ejemplo

Antes de explicar lo que hay que hacer se explicará a qué se desea llegar y a qué se refiere la palabra Solucion.

Recordemos que una Solucion representa una forma sencilla de guardar y tener a la mano en todo momento el Modelo y la Fórmula a satisfacer.

Hay que recordar las propiedades que debe cumplir la fórmula, para más información leer la nota 4.

Lo que deseamos que pueda hacer el algoritmo es que dado una Fórmula F y el Modelo vacío [ ] (La Solucion que representa esto seria $S = ([\,], F)$) se pueda obtener el Modelo que satisfaga a la Fórmula, es decir, una asignación de verdad de las literales de la Fórmula de tal forma que la Fórmula sea verdadera.

A continuación se mostrará un ejemplo de cómo se puede encontrar un Modelo para la Fórmula $F = (\mathrm{P} \vee R) \wedge (\neg P \vee \neg R)$

La ejecución para encontrar el Modelo es el siguiente:

- S = `([ ], [[V P, V R], [Neg (V P), Neg (V R)]])`

  A la Solución le aplicamos primero `split`.

- `([ ], [[V P, V R], [Neg (V P), Neg (V R)]])`
  ▷
  `1. ([V P], [[V P, V R], [Neg (V P), Neg (V R)]])`
  `2. ([Neg (V P)], [[V P, V R], [Neg (V P), Neg (V R)]])`
  Primero exploremos el camino 1.

  Le volvemos aplicar `split`.

- `([V P], [[V P, V R], [Neg (V P), Neg (V R)]])`
  ▷
  `1.1 ([V P, V R], [[V P, V R], [Neg (V P), Neg (V R)]])`
  `1.2 ([V P, Neg (V R)], [[V P, V R], [Neg (V P), Neg (V R)]])`
  Exploramos el camino 1.1.

  No podemos aplicar `split` de nuevo ya que no hay literales en la formula que ni la misma ni su contraria no estén en el Modelo.

  Aplicamos ahora la regla `red.` con la literal `V P.`

- `([V P, V R], [[V P, V R], [Neg (V P), Neg (V R)]])`
  ▷
  `([V P, V R], [[V P, V R], [Neg (V R)]])`
  No podemos aplicar la regla `unit, split.`

  Aplicamos la regla `elim` com la literal `V P.`

- `([V P, V R], [[V P, V R], [Neg (V R)]])`
  ▷
  `([V P, V R], [[Neg (V R)]])`

  Aplicamos la regla `red` con la literal `V R.`

- `([V P, V R], [[Neg (V R)]])`
  ▷
  `([V P, V R], [[]])`

  Llegamos a $\square$, por lo que se cumple la regla de `conflict`, pero no significa que no exista Modelo ya que aún quedan caminos por explorar.

  Ahora exploramos por el camino 1.2

No podemos aplicar `split` de nuevo ya que no hay literales en la formula que ni la misma ni su contraria no estén en el Modelo.

Aplicamos la regla `red` con la literal `V P`

- `([V P, Neg (V R)], [[V P, V R], [Neg (V P), Neg (V R)]])`
  ▷
  `([V P, Neg (V R)], [[V P, V R], [Neg (V R)]])`
  No podemos aplicar las reglas `unit, split`.

Aplicamos la regla `elim` con la literal `V P`

- `([V P, Neg (V R)], [[V P, V R], [Neg (V R)]])`
  ▷
  `([V P, Neg (V R)], [[Neg (V R)]])`

Ya sólo podemos aplicar la regla `elim` con la literal `Neg (V R)`

- `([V P, Neg (V R)], [[Neg (V R)]])`
  ▷
  `([V P, Neg (V R)], [])`

Llegamos a reducir la fórmula a $\varnothing$ por lo que se ha encontrado un modelo para la fórmula, es decir, se cumple la regla `success`.

Entonces regresamos ese modelo.

- `[V P, Neg (V R)]`

Deben recordar que existen muchas formas de darle un orden a la búsqueda del algoritmo, es decir, existen muchas formas diferentes de aplicar las reglas de DPLL, dado una Solucion, podemos aplicar siempre primero la regla `split`, o siempre primero la regla `unit`, es decir, existen muchas formas en el orden en el que se aplican las reglas, este ejemplo dio una de esas formas pero no es definitivo y ustedes deben implementar el que ustedes crean correcto.

2. Ejercicios

1. (1.5 puntos) Implementa la función `dpllsearch` que dada una Solucion regresa otra Solucion que representa al modelo y fórmula resultante de la aplicación de las reglas `unit, red, elim y split`.

2. (2.5 puntos) Implementa la función `dpll` que dada una Solucion regresa otra Solucion que representa al modelo y fórmula resultante después de una aplicación exhaustiva de las reglas del algoritmo dpll, es decir, esta función es la que se encargará de seguir explorando hasta que se llegue al estado de success o conflict.

3. (2 puntos) Implementa la función `main` que dada una Solucion regresa otra Solucion que contiene el posible modelo que satisface a la fórmula dada o de lo contrario lanza el estado de FAIL ya que el algoritmo dpll no encontró algún modelo que satisface la fórmula.
   Ejemplos

   - `DPLL>main ([],[[V "p", V "q"],[Neg (V "q")],[Neg (V "p"), V "q", Neg (V r")]])`
     `([∼("q"),"p",∼(r")],[])`
   - `DPLL>main ([],[[V "p"], [V "q"], [Neg (V "q"), Neg (V r")], [V r"]])`
     `*** Exception: FAIL`
     `modelo construido: ([r","q","p"],[[]])`

4. (4 puntos) Modifica las funciones `unit, red, elim y split` para que puedan seguir aplicando la búsqueda del modelo (Hacer Backtracking) si es necesario en caso de que la literal escogida no de algún resultado, es decir, para que pueda seguir buscando con otras posibles candidatas si no resulta la búsqueda con alguna otra.

3. Puntos extra

1. (3 puntos) Modifica lo necesario para poder obtener la lista de todos los modelos que satisfacen una fórmula dada.

4. Sugerencias generales

- función dpll: Esta es la que se encargará de repetir la búsqueda hasta llegar a algún resultado, es decir, esta necesitará de alguna manera saber cuando ya no es necesario seguir buscando y cuando seguir buscando, por buscar se refiere a aplicar las reglas a la Solucion (dpllsearch).
- función main: esta es la que se utilizará de manera global, es decir, ejecutarás tus ejemplos de la siguiente manera: main ([ ], FORMULA), con FORMULA alguna fórmula deseada, es decir, esta regresa un error si en toda la búsqueda jamás se llegó a un modelo que satisfaga.
- No hay que confundir que alguna Solucion sea verdadera para la función conflict y que no haya solución para un modelo y lanzar el estado FAIL. ya que puede ser que al hacer backtracking en alguna rama de la recursión se llega a conflict pero puede ser que por otra se llegue a success.
- No hay que modificar mucho las funciones que representan a las reglas para que puedan hacer el backtracking. Sólo hay que hacer que en lugar de que busquen solo con una, ahora busquen con una y después otra si es necesario y así.
- ASISTIR A LA SESIÓN ONLINE O ACTIVIDADES PLANEADAS PARA ESTE PERIODO.