Lógica Computacional: Practica 4

José Miguel Toledo Reyes
Omar Fernando Gramer Muñoz

27 de marzo de 2020

1. Realización

La practica se realizó en conjunto entre los dos miembros del equipo, esto con el fin de estandarizar el código y mantener una mejor coherencia en el formato.

Debido a que el algoritmo de DPLL usa las funciones anteriormente implementadas en la práctica 3 tuvimos que hacer unos cambios en ellas para poder usarlas en el algoritmo.

- `unit` : La implementación que dimos de esta función aplica la regla `unit` a todas las clausulas unitarias existentes en la fórmula, es decir, agrega al modelo a todas las cláusulas conformadas por una única literal, por lo tanto, era posible que una contradicción pudiera agregarse al modelo de la solución tras aplicar la función.
  Por ejemplo, si se tenía una fórmula con dos clausulas unitarias p y $\neg p$, entonces ambas eran erroneamente agregadas al modelo, cosa que no debería pasar.
  Así que se decidió crear una función auxilar `filtraUnit` para evitar este tipo de casos.

- `elim` : No tuvimos que hacer ningún cambio a esta función, pero es importante mencionar que la función aplica la regla `elim` sobre la fórmula para cada literal en el modelo.

- `red` : No tuvimos que hacer ningún cambio a esta función, pero es importante mencionar que la función aplica la regla `red` sobre la fórmula para cada literal en el modelo.

- `split` : A diferencia de las funciones anteriores la implementación de `split` solo aplica la regla sobre una literal, esto lo hace aplicando la regla de `split` sobre la primer literal ℓ en la fórmula tal que ℓ o su negación no se encuentra en el modelo.
  La función tal y como estaba implementada en la práctica anterior devuelve una lista con las dos soluciones resultantes al aplicar la regla `split` sobre la solución original.
  Para esta práctica se decidió que la implementación de esta función deberia poder trabajar con un conjunto de soluciones a la vez, es decir, que si se recibe como parametro una lista con n soluciones entonces la esta nueva implementación de la función debe

aplicar la regla `split` sobre cada una de las soluciones en la lista recibida, devolviendo a lo más una lista con el doble de soluciones que la lista original.

- `conflict` : No tuvimos que hacer ningún cambio a esta función. Simplemente devuelve verdadero cuando encuentra una cláusula vacía en la fórmula, o falso en otro caso.
- `success` : No tuvimos que hacer ningún cambio a esta función, Simplemente devuelve verdadero cuando la fórmula es una lista vacía, o falso en otro caso.

La estructuración del algoritmo `DPLL`, así como la implementación de las funciones propias de esta práctica fue la siguiente:

- `main` : Lo único que hace es llamar a la función `dpll` y tras ello verificar si el conjunto de soluciones recibido es vacío, en cuyo caso devuelve un error comentando que la fórmula es insastifacible, o regresando como salida al conjunto en caso de que exista una o más soluciones.
- `dpll` : Esta función se encarga de general todos los modelos posibles para la solución recibida aplicando la regla `unit` , y posteriormente haciendo `split` las veces necesarias (mediante una función auxiliar) para obtener cada combinación de los posibles modelos a probar. Una vez hecho esto llama a la función `dpllsearch` pasandole como atributo la lista con todas las soluciones generadas.
- `dpllsearch` : Finalmente esta función se encarga de filtrar aquellas soluciones cuyos modelos sastifacen a su fórmula, esto lo logra mediante una lista por comprension donde a cada solución en la lista recibida se le aplican las reglas `elim` y `red` para posteriormente identificar con la regla `success` si el modelo era válido.

2. Ejecución del programa

Para poder compilar el programa es necesario tener instalado GHC compiler, el cual puede obtenerse mediante el siguiente comando en Linux.

```
sudo apt-get install haskell-platform
```

Una vez hecho lo anterior, para correr el programa, es necesario establecerse el directorio de trabajo en la carpeta en donde se encuentran el archivo DPLL-2.hs y ejecutar el siguientes comando:

- Ejecución DPLL-2:

  `ghci DPLL-2.hs`

  Antes que nada es necesario importar el archivo LProp.hs de la practica 3 al directorio en el que se encuentra el archivo DPLL-2 ya que ciertas funciones dependen del archivo antes mencionado por lo que es necesario para su funcionamiento.

  Para ejecutar y compilar el programa basta escribir

  `ghci DPLL-2.hs`

  Esto hará que se compile el programa y sea posible su ejecución.Cabe resaltar que para que se pueda compilar de manera correcta el archivo DPLL-2,el archivo LProp.hs tambien se debe de compilar de manera correcta.

  Para ejecutar el programa solo basta escribir:

  `*DPLL-2>  main ([],[[`$Formula \quad propociocinal$`]])`

  Donde:

  - main: Función principal del programa la cual llevará acabo el algoritmo DPLL.

  - Formula : Formula propocional de la cual se obtendran los posibles modelos que la satisfagan.
    La formula debe de esta en forma normal conjuntiva es decir que se representara de la siguiente manera:

    $$[[\ ], [Clausulas, Clausulas, Clausulas, ...]]$$

    En donde cada clausula asu vez es una lista de literales $[l, l, l, l.....]$

En caso de que al ejecutar el programa se proporcione una formula para la cual no sea posible encontrar modelo alguno que la satisfaga, se mostrará en pantalla el siguiente mensaje:

$$***Exception:$$

La fórmula es
insatisfacible
CallStack (from HasCallStack): error, called at DPLL-2.hs:151:27 in main:DPLL

3. Conclusiones

En esta práctica aplicamos los conocimientos que fuimos adquiriendo en las practicas anterirores sobre listas por comprensión ya que fueron de vital importancia para para poder implementar las funciones que usamos para llevar acabo el algortimo DPLL de una manera sencilla y facil de entender.

Asi mismo fue necesario entender de manera correcta como es que funcionaban cada una de las reglas del algoritmo DPLL,Debido a que como optamos por obtener todos los modelos que satisfacian a la formula propocicional, entonces tuvimos que modificar algunas de las implementaciones de las reglas que ya teniamos para que fuese posible obtenerlos de manera sencilla y sin tener que hacer uso de demasiadas funciones auxiliares y asi poder lograr que al usar las reglas en conjunto, hacer que el algoritmo DPLL se lleve acabo de una manera similar a como si una persona lo hiciese manualmente.