Lógica Computacional
Práctica 2: Lógica Proposicional

César Hernández Cruz
Víctor Zamora Gutiérrez
Diego Carrillo Verduzco

24 de agosto de 2018

Adjuntos a esta práctica vienen dos archivos nombrados `Practica2.hs` y `LProp.hs`. La práctica consiste en completar las definiciones de las funciones declaradas en el archivo `Practica2.hs`, a saber:

1. `busca :: Prop ->Interpretacion ->Bool`, función que recibe una variable proposicional (`PVar`) y una interpretación y busca la asignación de verdad de la variable en la interpretación. Ejemplo:
   ```
   >busca (PVar 'A') [('B', True), ('A', False)]
   False
   >busca (PVar 'B') [('C', True), ('B', True), ('A', False)]
   True
   ```

2. `eval :: Prop ->Interpretacion ->Bool`, función que evalúa una fórmula proposicional de acuerdo a una interpretación dada. (HINT: usar la función `busca`) Ejemplo:
   ```
   >eval (PImpl (PVar 'A') (PVar 'B')) [('A', True), ('B', False)]
   False
   >eval (POr (PVar 'A') (PVar 'B')) [('A', False), ('B', True)]
   True
   ```

3. `satisfacible :: Prop ->Bool`, función que determina si existe una asignación de verdad para las variables de una fórmula proposicional tal que la fórmula se evalúe a verdadero. Se recomienda hacer una función que extraiga todas las variables de una fórmula proposicional

(sin repeticiones, por supuesto) y utilizar la función `interpretaciones` que se incluye en el archivo de la práctica. Ejemplo:

```
>satisfacible (POr (PVar 'A') (PVar 'B'))
True
>satisfacible (PAnd (PVar 'A') (PNeg (PVar 'A')))
False
```

4. `fnc :: Prop ->Prop`, función que recibe una fórmula proposicional y devuelve una fórmula equivalente en forma normal conjuntiva. Ejemplo:

```
>fnc (PImpl (PImpl (PNeg (PVar 'P')) (PVar 'Q')) (PImpl (PNeg
(PVar 'R')) (PVar 'S')))
((¬'P' v ('R' v 'S')) ^ ((¬'Q' v ('R' v 'S')))
```

1. Algoritmo para pasar a forma normal conjuntiva

Una fórmula de la lógica proposicional está en forma normal conjuntiva si y sólo si consiste únicamente de conjunciones de disyunciones de literales; esto es, la fórmula es de la forma

$$\mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_n$$

tal que cada $\mathcal{C}_i$ es de la forma

$$a_1 \vee a_2 \vee \ldots \vee a_m$$

tal que cada a_j es o bien una constante proposicional, una variable, o una negación de alguna de las anteriores.

Para pasar de una fórmula proposicional cualquiera a una en forma normal conjuntiva, basta seguir este procedimiento:

1. Obtener la forma normal negativa de la fórmula. La forma normal negativa es una forma tal que no contiene implicaciones ni dobles implicaciones, y en donde las negaciones aparecen únicamente junto a variables o constantes. Podemos definir una función que transforme una fórmula en su forma normal negativa considerando estos casos, suponiendo que previamente hayamos eliminado las implicaciones y dobles implicaciones (recordando que $(A \to B) \equiv (\neg A \vee B)$ y $(A \leftrightarrow B) \equiv ((A \to B) \wedge (B \to A))$):

- Una literal (variables y constantes) ya está en FNN.
- Si es una conjunción o una disyunción, calcular la FNN de las dos subfórmulas y juntarlas con el mismo respectivo operador.
- Si es una negación, hacemos análisis de casos: si es una negación de una literal, ya está en FNN; si es $\neg\neg A$ se eliminan las negaciones y se aplica recursivamente la función a A; si es una negación de conjunción o disyunción se aplican las leyes de De Morgan y se aplica recursivamente la función a las subfórmulas resultantes $(fnn(\neg(\phi \vee \psi)) = (fnn(\neg\phi) \wedge fnn(\neg\psi)$ y su análogo para la conjunción).

2. Ya operando sobre una fórmula en FNN, tenemos varios casos:

   - Si la fórmula es una literal, ya está en FNC.
   - Si la fórmula es una conjunción, se llama recursivamente la función para calcular la FNC sobre las dos subfórmulas y se devuelve la conjunción del resultado $fnc(\phi \wedge \psi) = fnc(\phi) \wedge fnc(\psi)$.
   - Si la fórmula es una disyunción llamamos recursivamente la función para calcular la FNC sobre las dos subfórmulas y luego aplicamos apropiadamente las leyes de distributividad:

   $$\phi \wedge (\psi \vee \chi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi)$$

   $$\phi \vee (\psi \wedge \chi) \equiv (\phi \vee \psi) \wedge (\phi \vee \chi)$$

   Para este fin podemos definir una función $distr(\phi, \psi)$ que actúe de esta forma:

     - Si ϕ y ψ son literales, regresamos $\phi \vee \psi$, pues esto es una cláusula y por lo tanto está en FNC.
     - Si $\phi = (\phi_1 \wedge \phi_2)$, aplicamos la distributividad y recursivamente aplicamos $distr$ sobre las conjunciones resultantes. $distr((\phi_1 \wedge \phi_2), \psi) = (distr(\phi_1, \psi) \wedge distr(\phi_2, \psi))$.
     - Si $\psi = (\psi_1 \wedge \psi_2)$ la definición es análoga a la anterior (salvo que naturalmente la distributividad se aplica por la izquierda). $distr(\phi, (\psi_1 \wedge \psi_2)) = (distr(\phi, \psi_1) \wedge distr(\phi, \psi_2))$

   La definición de la función fnc para este caso es entonces $fnc(\phi \vee \psi) = distr(fnc(\phi), fnc(\psi))$

2. Formato de entrega

La práctica se entregará de la siguiente manera: se entregará **únicamente** el archivo `Practica2.hs` con los ejercicios resueltos. Deberá incluir en un comentario en la primera linea del archivo el nombre del alumno. El archivo debe subirse como respuesta a la asignación de tarea en el Classroom del grupo.