Práctica 1

Lógica Computacional, 2018-2
Facultad de Ciencias, UNAM

Noé Salomón Hernández Sánchez
no.hernan@gmail.com

María del Carmen Sánchez Almanza
carmensanchez@ciencias.unam.mx

Albert Manuel Orozco Camacho
alorozco53@ciencias.unam.mx

17 de febrero de 2018

1. Objetivo

Que el alumno utilice los conocimientos básicos de Haskell, para implementar el comportamiento del cálculo proposicional.

2. Ejercicios

El esqueleto de código del ejercicio semanal se encuentra en `https://github.com/alorozco53/LabLogComp-2018-2/tree/pract1`. Considere el tipo de datos

```
data Prop = VarP String
          | TTrue
          | FFalse
          | Neg Prop
          | Disj Prop Prop
          | Conj Prop Prop
          | Imp Prop Prop
          | Equiv Prop Prop
```

el cual codifica todas y cada una de las fórmulas de la lógica proposicional.

1. Implemente una función que imprima una fórmula de la lógica proposicional mediante operadores infijos. Por ejemplo:

```
*Practica1> Equiv (VarP "p") (Conj (TTrue) (VarP "q"))
("p") <-> ((True) ^ ("q"))
```

Considere el siguiente tipo *alias* para sustituciones:

```
type Sub = String -> Prop
```

Entonces, una sustitución se definirá mediante una función que va de variables a proposiciones. Por ejemplo, la sustitución $[p := q \leftrightarrow \bot,\ q := True]$ se especificaría mediante

```
sub1 :: Sub
sub1 "p" = Equiv (VarP "q") FFalse
sub1 "q" = TTrue
sub1 other = VarP other
```

Obsérvese, entonces, que el último caso se usa para cualquier otra variable que no esté definida en la sustitución (i.e., función *identidad*).

2. Escriba una función `substitute` que, dada una proposición Φ y una sustitución s (de tipo `Sub`), aplique s en Φ.

Considere el siguiente tipo *alias* para estados (interpretaciones):

```
type State = [String]
```

Para una fórmula Φ de la lógica proposicional, asumimos que cualquier variable v presente en Φ posee una interpretación $\mathcal{I}(v) = 1$ *si y sólo si* v está dentro de la lista de tipo `State` dada.

3. Proponga una función `interp` que, dada una fórmula de tipo `Prop` y una interpretación de tipo `State`, devuelva la interpretación de la fórmula en tipo `Bool` de Haskell.

4. Elabore una función `model` que decida si el estado dado **es un modelo** para una fórmula dada.

5. Escriba una función `vars` que devuelva todas las variables contenidas en la fórmula proposicional dada.

6. Dé una función `powerList` que devuelva la *lista potencia* de una lista dada. Por ejemplo, dada la lista `[1, 2]`, su lista potencia sería `[[], [1], [2], [1, 2]]`. *Sugerencias*:
   - Sea A un conjunto finito de tamaño n. Demuestre (con inducción matemática) que el conjunto potencia de A posee 2^n elementos.
   - Implemente la demostración anterior, usando una lista por comprensión en el caso recursivo.

7. Escriba una función `tautology` que determine si una fórmula dada es una *tautología*. *Sugerencia*:
   - Utilice la función `powerList` para calcular todas las interpretaciones de las variables de una fórmula dada.
8. Escriba una función `equivProp` que determine si dos fórmulas dadas son *lógicamente equivalentes*. *Sugerencia*:
   - Utilice el teorema visto en clase que establece que dos fórmulas son lógicamente equivalentes *si y sólo si* cierta fórmula es una tautología. ¿A qué fórmula nos referimos?
9. Escriba una función `logicConsequence` que, dada una lista de fórmulas Γ, y una proposición Ψ, determine si $\Gamma \models \Psi$.

3. Entrega

La fecha de entrega es el próximo **sábado 3 de marzo de 2018** por la plataforma de *Google Classroom* del curso y siguiendo los lineamientos del laboratorio.