Lógica Computacional, 2019-2
Nota de laboratorio 5*
Lógica Clausular Proposicional

Manuel Soto Romero

13 de marzo de 2019
Facultad de Ciencias, UNAM

Las claúsulas son una clase especial de fórmulas relevantes en distintas aplicaciones dentro y fuera de la lógica. En esta nota discuitimos la implementación de funciones que realicen la transformación de cualquier fórmula del lenguaje `PROP` a la llamada forma clausular.

1. Forma Normal Negativa

Definición 1. Una fórmula φ está en *Forma Normal Negativa* si y sólo se cumplen las siguientes:

1. φ no contiene ni equivalencias ni implicaciones.
2. Las negaciones que figuran en φ afectan sólo a fórmulas atómicas.

El objetivo es implementar una función `fnn :: PROP -> PROP` que satisfaga la Definición 1. Para ello se requiere al menos de las siguientes funciones:

```
eliminaEquivalencias :: PROP -> PROP
eliminaImplicaciones :: PROP -> PROP
introduceNegaciones :: PROP -> PROP
```

1.1. Eliminación de equivalencias

Eliminar equivalencias de una fórmula proposicional puede hacerse de diversas maneras, sin embargo, la forma más adecuada y simple es hacerlo mediante la equivalencia:

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

El Listado de código 1 muestra la implementación de la función `eliminaEquivalencias`. Las líneas 11 a 13 muestra la implementación de la equivalencia anterior.

*Basado en las *Notas para el curso de Lógica Computacional* del Dr. Favio E. Miranda, et al., revisión 2017-2.

```haskell
-- Función que elimina equivalencias de una fórmula proposicional.
eliminaEquivalencias :: PROP -> PROP
eliminaEquivalencias (FA a) = (FA a)
eliminaEquivalencias (Neg p) = Neg (eliminaEquivalencias p)
eliminaEquivalencias (Conj p q) =
   Conj (eliminaEquivalencias p) (eliminaEquivalencias q)
eliminaEquivalencias (Disy p q) =
   Disy (eliminaEquivalencias p) (eliminaEquivalencias q)
eliminaEquivalencias (Impl p q) =
   Impl (eliminaEquivalencias p) (eliminaEquivalencias q)
eliminaEquivalencias (Syss p q) =
   Conj (Impl (eliminaEquivalencias p) (eliminaEquivalencias q))
        (Impl (eliminaEquivalencias q) (eliminaEquivalencias p))
```

Listado de código 1: Eliminación de equivalencias

1.2. Eliminación de implicaciones

Similar a la eliminación de equivalencias de una fórmula proposicional, para eliminar implicaciones, lo más adecuado es hacer uso de la siguiente equivalencia:

$$\varphi \to \psi \equiv \neg\varphi \vee \psi$$

El Listado de código 2 muestra la implementación de la función `eliminaImplicaciones`. Las líneas 9 a 10 muestran la implementación de la equivalencia anterior.

```haskell
-- Función que elimina implicaciones de una fórmula proposicional.
eliminaImplicaciones :: PROP -> PROP
eliminaImplicaciones (FA a) = (FA a)
eliminaImplicaciones (Neg p) = Neg (eliminaImplicaciones p)
eliminaImplicaciones (Conj p q) =
   Conj (eliminaImplicaciones p) (eliminaImplicaciones q)
eliminaImplicaciones (Disy p q) =
   Disy (eliminaImplicaciones p) (eliminaImplicaciones q)
eliminaImplicaciones (Impl p q) =
   Disy (Neg (eliminaImplicaciones p)) (eliminaImplicaciones q)
```

Listado de código 2: Eliminación de equivalencias

Observación 1. La función `eliminaImplicaciones` supone que la fórmula que recibe como parámetro no tiene signos de equivalencia (`Syss`).

1.3. Introducción de negaciones

Similar a la eliminación de equivalencias e implicaciones de una fórmula proposicional, para introducir negaciones, lo más adecuado es hacer uso de las siguientes equivalencias:

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi \qquad \neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi \qquad \neg\neg\varphi \equiv \varphi$$

Se deja como ejercicio la implementación de las funciones `introduceNegaciones` y `fnn` que hace uso de todas las funciones definidas.

2. Forma Normal Conjuntiva

Definición 2. Una *literal* ℓ es una fórmula atómica o la negación de una fórmula atómica.

Definición 3. Dada una literal ℓ se define su *literal contraria*, denotada ℓ^c, como sigue:

$$\ell^c = \begin{cases} a & si \quad \ell = \neg a \\ \neg a & si \quad \ell = a \end{cases}$$

Donde a es una fórmula atómica. El parar $\{\ell, \ell^c\}$ es llamado *par de literales complementarias*.

Definición 4. Una *cláusula* C es una literal o una disyunción de literales.

Definición 5. Una fórmula φ está en *Forma Normal Conjuntiva* si y sólo sí es de la forma $C_1 \wedge C_2 \wedge ... \wedge C_n$ donde cada C_i es una cláusula.

El objetivo es implementar una función `fnc :: PROP -> PROP` que satisfaga la Definicion 5. De manera algorítmica puede obtenerse la Forma Normal Conjuntiva de una fórmula proposicional de acuerdo a los siguientes pasos:

1. Obtener la Forma Normal Negativa correspondiente de la fórmula.
2. Si la fórmula es una literal, ya está en Forma Normal Conjuntiva y simplemente se devuelve como salida.
3. Si la fórmula es una conjunción $\varphi_1 \wedge \varphi_2$ se procesa recursivamente como $fnc(\varphi_1) \wedge fnc(\varphi_2)$.
4. Si la fórmula es una disyunción $\varphi_1 \vee \varphi_2$ se procesa recursivamente como $fnc(\varphi_2) \vee fnc(\varphi_2)$ y se transforma la expresión resultante en conjunción mediante las leyes de la distribuitividad.

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

Se requiere entonces la implementación de la siguiente función:

```
distribuyeDisyuncion :: PROP -> PROP
```

2.1. Distribución de conjunciones

Para implementar la función `distribuyeDisyuncion` se supone que las fórmulas de entrada φ_1 y φ_2 son formas conjuntivas.

- Si ambas φ_1 y φ_2 son literales entonces se devuelve $\varphi_1 \vee \varphi_2$.
- Si $\varphi_1 = \varphi_{11} \wedge \varphi_{12}$ entonces se aplica distributividad:

$$\varphi_1 \vee \varphi_2 \equiv (\varphi_{11} \wedge \varphi_{12}) \vee \varphi_2 = (\varphi_{11} \vee \varphi_2) \wedge (\varphi_{12} \vee \varphi_2)$$

  y se vuelve a aplicar recursivamente la función con las nuevas disyunciones.
- Si $\varphi_2 = \varphi_{21} \wedge \varphi_{22}$ la definición es análoga al caso anterior.

Se deja como ejercicio la implementación de las funciones `distribuyeDisyuncion` y `fnc`.

3. Forma Normal Disyuntiva

Definición 6. Una fórmula φ está en *Forma Normal Disyuntiva* si y sólo sí es de la forma $C_1 \vee C_2 \vee ... \vee C_n$ donde cada C_i es una cláusula.

El objetivo es implementar una función `fnd :: PROP -> PROP` que satisfaga la Definicion 6. De manera algorítmica puede obtenerse la Forma Normal Disyuntiva de una fórmula proposicional análogamente a cómo se hizo con la Forma Normal Conjuntiva, con el uso de una función:

```
distribuyeConjuncion :: PROP -> PROP
```

Se deja como ejercicio la implementación de las funciones `distribuyeConjuncion` y `fnd`.