<html><head></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">
```
module Sintaxis where

-- ---------------------------------------------------------------------
-- GramÃ¡tica de fÃ³rmulas prosicionales                                --
-- ---------------------------------------------------------------------

-- ---------------------------------------------------------------------
-- Se definen los siguientes tipos de datos:
-- * SimboloProposicional para representar los sÃmbolos de proposiciones
-- * Prop para representar las fÃ³rmulas proposicionales usando los
--   constructores Atom, Neg, Conj, Disj, Impl y Equi para las fÃ³rmulas
--   atÃ³micas, negaciones, conjunciones, implicaciones y equivalencias,
--   respectivamente.
-- ---------------------------------------------------------------------

type SimboloProposicional = String

data Prop = Atom SimboloProposicional
          | Neg Prop
          | Conj Prop Prop
          | Disj Prop Prop
          | Impl Prop Prop
          | Equi Prop Prop
         deriving (Eq,Ord)

instance Show Prop where
    show (Atom p)   = p
    show (Neg p)    = "no " ++ show p
    show (Conj p q) = "(" ++ show p ++ " /\\ " ++ show q ++ ")"
    show (Disj p q) = "(" ++ show p ++ " \\/ " ++ show q ++ ")"
    show (Impl p q) = "(" ++ show p ++ " --&gt; " ++ show q ++ ")"
    show (Equi p q) = "(" ++ show p ++ " &lt;--&gt; " ++ show q ++ ")"

-- ---------------------------------------------------------------------
-- Se definen las siguientes fÃ³rmulas proposicionales
-- atÃ³micas: p, p1, p2, q, r, s, t y u.
-- ---------------------------------------------------------------------

p, p1, p2, q, r, s, t, u :: Prop
p  = Atom "p"
p1 = Atom "p1"
p2 = Atom "p2"
q  = Atom "q"
r  = Atom "r"
s  = Atom "s"
t  = Atom "t"
u  = Atom "u"

-- ---------------------------------------------------------------------
-- Se define la funciÃ³n
--    no :: Prop -&gt; Prop
-- tal que (no f) es la negaciÃ³n de f.
-- ---------------------------------------------------------------------

no :: Prop -&gt; Prop
no = Neg

-- ---------------------------------------------------------------------
-- Se definen los siguientes operadores
--    (/\), (\/), (--&gt;), (&lt;--&gt;) :: Prop -&gt; Prop -&gt; Prop
-- tales que
--    f /\ g      es la conjunciÃ³n de f y g
--    f \/ g      es la disyunciÃ³n de f y g
--    f --&gt; g     es la implicaciÃ³n de f a g
--    f &lt;--&gt; g    es la equivalencia entre f y g
-- ---------------------------------------------------------------------

infixr 5 \/

infixr 4 /\
infixr 3 --&gt;
infixr 2 &lt;--&gt;
(/\), (\/), (--&gt;), (&lt;--&gt;) :: Prop -&gt; Prop -&gt; Prop
(/\)   = Conj
```
</pre></body></html>

```
(\/)   = Disj
(--&gt;)  = Impl
(&lt;--&gt;) = Equi


-- ------------------------------------------------------------------
-- Interpretaciones                                                 --
-- ------------------------------------------------------------------

-- ------------------------------------------------------------------
-- Se define el tipo de datos InterpretaciÃ³n para representar las
-- interpretaciones como listas de pares (Ãítomo,booleano).
-- ------------------------------------------------------------------

type Interpretacion = [(Prop,Bool)]

-- ------------------------------------------------------------------
-- Se define el tipo de dato Literal como sinÃ³nimo de fÃ³rmula.
-- ------------------------------------------------------------------

type Literal = Prop

-- ------------------------------------------------------------------
-- ClÃ¡usulas                                                      --
-- ------------------------------------------------------------------

-- ------------------------------------------------------------------
-- Se define el tipo de datos ClÃ¡usula como una lista de literales.
-- ------------------------------------------------------------------

type Clausula = [Literal]

</pre></body></html>
```