9-4-2008

1. Implementación de las reglas

- He implementado las reglas de creación de zonas, operadores de conexión y canonización con HaRP. El código se encuentra en el directorio *Reglas*, ficheros con extensión *hsx*. En general, la implementación es muy compacta, aunque en ocasiones es tan densa que resulta difícil de leer. Tengo que ver si hay alguna manera de que queden más legibles.
- He movido las baterías de pruebas al fichero *Reglas/Test.hs*. De momento no se me han ocurrido pruebas nuevas aparte de las que ya comenté, tengo que seguir trabajando en ello ya que creo que es un aspecto muy importante.
- Hasta ahora, los tipos de datos enumerados derivaban automáticamente de la clase *Show*. Para que la representación obtenida sea más atractiva, he creado el módulo *Mostrar.hs*, en el que se declaran algunas instancias manualmente:
  - *Conexion*: la nueva instancia usa caracteres Unicode para poder representar los operadores de la manera más fiel posible.
  - *Solucion*: he declarado el tipo *Solucion* como un *newtype* (un tipo algebraico que sólo puede contener una etiqueta), lo que permite definir instancias. Ahora las soluciones se muestran indicando primero el escenario inicial, y debajo los cambios, indicando los nodos que abarca cada cambio.
- Por ejemplo, el escenario $A_4 \diamond N_4 \leftrightarrow Z_4 \otimes_d Z_4 \otimes_d Z_4 \leftrightarrow R_d^t \leftrightarrow Z_6 \otimes_d N_4 \diamond A_4$ se representaría literalmente como:

```
escenario = (A4, N4, [(Z4,OpD) , (Z4,OpD) , (Z4,Directa) ,
             (Rd_td,Directa) , (Z6,OpD)], N4, A4)
```

  Al visualizarlo en pantalla, tendríamos:

```
Ok, modules loaded: Reglas.Canonizacion, Main, Tipos, Reglas.Conexiones
*Main> escenario
(A4,N4,[(Z4,⊗d),(Z4,⊗d),(Z4,↔),(Rd_td,↔),(Z6,⊗d)],N4,A4)
*Main> visualizar
A4 □ N4 ↔ Z4 ⊗d Z4 ⊗d Z4 ↔ Rd_td ↔ Z6 ⊗d  N4 □ A4
          |_____________|
          Canonizacion1
          |_______|
          Operador3
*Main> ▉
```

  Nota: he hecho las pruebas con Konsole, no sé si funcionará igual con otras terminales.
- Un problema es que los cambios se anotan con respecto al escenario al que se aplican, no con respecto al inicial. Eso quiere decir que, en un escenario al que se han aplicado varios cambio, un mismo nodo puede representar varios nodos del escenario inicial. Habría que tener esto en cuenta a la hora de visualizar los cambios.
- He estado reorganizando un poco el código, ya que cada vez hay más módulos. Esta es una de las partes que más tiempo me ha llevado, ya que había módulos muy liosos.
- Como me quedé un poco estancado, estuve pensando en la manera de representar las redes completas como grafos, y cómo obtener todos los posibles caminos desde el nodo inicial hasta el final. Creo que sería interesante discutirlo, ya que podría afectar a lo que estoy desarrollando ahora.