



Pràctica 1: Programes seqüencials amb Erlang

Programació Concurrent i en Temps Real — iTIC

Antoni Escobet Canal

Sebastià Vila-Marta

9 de setembre de 2014

Índex

1 Organització

Aquesta sessió s'organitza com una seqüència de problemes de dificultat creixent l'objectiu dels quals és implementar petits programes escrits en **Erlang**. Amb l'objectiu de reforçar l'hàbit d'usar sistemes de control de versions, cal desenvolupar la pràctica amb el suport del sistema que ofereix <http://escriny3.epsem.upc.edu>.

1.1 Lliurament

Cal lliurar els exercicis en un tarfile a través d'Atenea en la data fixada. Cal que el desenvolupament es faci usant **Subversion** a través de les facilitats que ofereix <http://escriny3.epsem.upc.edu>.

2 Exercicis

EXERCICI 2.1 Dissenyeu una funció `xenx(L,X)` tal que donada una llista **L** amb elements l_0, l_1, l_2, \dots i un enter **X** retorna una llista amb els elements l_0, l_x, l_{2x}, \dots . A continuació teniu un exemple d'ús:

```
1> L = [a,b,c,d,e,f,g,h,i].
2> xenx(L,3).
[a,d,g]
```

EXERCICI 2.2 Dissenyeu i implementeu un programa **Erlang** que llegeix una expressió en notació postfixa i en calcula el valor. Per simplificar assumiu que només s'admeten enters i les operacions suma i producte. Així, per exemple, si el programa s'executa invocant la funció `start()`, es podria executar així:

```
1> start().
expressio> 12 3 + 2 *.
30
2>
```

Recordeu (o rescateu els vostres apunts de TECPRO) que aquest calcul es fa usant una pila. Així doncs haureu d'usar el mòdul pila de la pràctica anterior. Pel que fa a com llegir l'expressió, consulteu l'ús de la funció `io:scan_eri_exprs()` en el manual d'Erlang.



EXERCICI 2.3 Un autòmat per reconèixer paraules formades per lletres minúscules es pot definir especificant aquests tres elements:

1. Un conjunt d'estats E , identificats per números enters consecutius, entenent que el zero correspon sempre a l'estat inicial i el número més gran a l'estat final.
2. Una funció de transició F tal que, donat l'estat actual i un caràcter, indica quin és el següent estat de l'autòmat.

Per exemple, l'autòmat que reconeix les paraules de la forma **a*b** es defineix amb $E = \{0, 1\}$ i F com la funció que compleix la taula següent:

Estat actual	Caràcter	Següent estat
0	a	0
0	b	1

Donada una definició així d'un autòmat, saber si reconeix o no una paraula és molt simple: recorrem les lletres de la paraula i anem aplicant la funció de transició. Si en algun moment la funció no és aplicable o bé en acabar les lletres de la paraula no som en l'estat final, la paraula no es reconeix.

Usant Erlang és molt senzill definir un autòmat. Es pot fer definint una tupla que conté:

- El enter corresponent a l'estat final. Fixeu-vos que aquest número és suficient per a determinar el conjunt E .
- La funció de transició lleugerament ampliada per tal que, en cas que fos cridada amb arguments fora del domini, retorni l'àtom `undef`.

Així, l'autòmat que hem usat com a exemple es pot definir amb aquesta tupla (estudieu bé la definició de la funció anònima i l'ús del dòlar per a indicar constants de tipus caràcter):

```
{1,  
 fun  
   (0,$a) -> 0;  
   (0,$b) -> 1;  
   (_,_) -> undef  
 end  
}
```

Es demana que dissenyeu i implementeu una funció tal que, donat un autòmat i una cadena de caràcters retorni un booleà que indiqui si l'autòmat reconeix la paraula o no.



EXERCICI 2.4 Amplieu el mòdul de l'exercici anterior afegint la funció `makeparser(A)`. Aquesta funció, donat un autòmat `A`, retorna una funció booleana. La funció booleana retornada és tal que, aplicada a una paraula determina si és reconeguda per l'autòmat. Per exemple, usant l'autòmat de l'exercici anterior podem fer:

```
1> A = {1,
      fun
        (0,$a) -> 0;
        (0,$b) -> 1;
        (-,-) -> undef
      end
    }.
2> T = makeparser(A).
3> T("aaab").
true
4> T("cab").
false
```