Pràctica 2: Programes concurrents amb `Erlang`

Programació Concurrent i en Temps Real — iTIC

Antoni Escobet Canal

14 de setembre de 2022

Índex

1 Organització **1**
1.1 Lliurament 1

2 Exercicis **1**

1 Organització

Aquesta sessió s'organitza com una seqüencia 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://escriny.epsem.upc.edu`.

1.1 Lliurament

Cal lliurar els diferents fitxers font dels exercicis a través d'Atenea en la data fixada. Cal que el desenvolupament es faci usant `Subversion` a través de les facilitats que ofereix `http://escriny.epsem.upc.edu`.

2 Exercicis

Exercici 2.1

Generador i verificació del codi de control del EAN-13. El format EAN-13 està format per 12 dígits (entre el 0 i el 9) més un digit de comprovació. Poden haver-hi guionets i es pot comprovar la seva validesa mitjançant la següent formula:

$(x_1 + x_2 * 3 + x_3 + x_4 * 3 + x_5 + x_6 * 3 + x_7 + x_8 * 3 + x_9 + x_{10} * 3 + x_{11} + x_{12} * 3 + x_{13}) mod 10 == 0$

Si el resultat val 0, llavors és un EAN-13 correcte, en cas contrari no és vàlid.

a) Dissenyeu una funció que donat un codi de 12 nombres calculi el digit de control. Per exemple:

```
1> ex2:generaEAN("978-84-8181-227").
2> 97 88481 81227 5
```

El càlcul que fa, és:
(9 + 7*3 + 8 + 8*3 + 4 + 8*3 + 1 + 8*3 + 1 + 2*3 + 2 + 7*3 + X) mod 10 == 0 X = 5

b) Dissenyeu una funció validEAN(Ean) que retorni cert si és correcte i fals en cas contrari. Un exemple pot ser:

1> ex2:validEAN("97 88481 81227 5").
2> **true**.

El càlcul que fa, és:
(9 + 7*3 + 8 + 8*3 + 4 + 8*3 + 1 + 8*3 + 1 + 2*3 + 2 + 7*3 + 5) mod 10 == 0
Podeu fer la verificació de la vostre funció amb el següent conjunt de proves:

1> ex1:validEAN("97 88481 81227 5"). **true**
2> ex1:validEAN("978-84-8181-227-5"). **true**
3> ex1:validEAN("978-84-8181-227-6"). **false**
4> ex1:validEAN("979-84-8181-227-A"). **false**
5> ex1:validEAN("9788481812275"). **true**
6> ex1:validEAN("00"). **false**
7> ex2:validEAN("3-598-21515-X"). **false**
8> ex2:validEAN(""). **false**
9> ex2:validEAN("14-7"). **true** (analitza 000000000014−7)

Exercici 2.2 Una part important de tots els llenguatges de programació funcional és la possibilitat de utilitzar una funció que s'hagi definit i després passar-la com a paràmetre a una altra funció. Una funció que pot acceptar altres funcions com a paràmetres es denomina funció d'ordre superior . Les funcions d'ordre superior són un potent mitjà d'abstracció i una de les millors eines per dominar a Erlang.

a) Implementeu la funció d'ordre superior fold(L,F,I) que té com a paràmetres una llista L, una funció de dues variables F i un valor inicial I. El seu funcionament és tal que si $L = l_0, l_1, l_2, \ldots, l_n$, llavors calcula l'expressió $F(\ldots F(F(I, l_0)), l_1) \ldots, l_n)$.
Useu aquesta funció per definir dues funcions tal que, donada una llista d'enters en calcula una la suma i l'altre el producte. Quins altres problemes classics podríeu resoldre amb la funció fold?

b) Estudieu amb atenció les funcions predefinides al mòdul lists. Sobretot les funcions lists:foldl i lists:flatten que són un conjunt de funcions d'ordre superior molt corrents en els llenguatges funcionals.
Usant aquestes funcions definiu noves funcions que calculin:

1. sumValPar: La suma dels valors parells d'una llista d'enters. L=[2,5,7,8,10]; 2+8+10=20.
2. sumPosPar: La suma dels valors que ocupen posicions parells en una llista d'enters. L=[3,2,4,6]; 2+6=8.
3. dupValPar: La llista obtinguda duplicant els elements parells de la llista original. L=[3,2,6]; [3,2,2,6,6].
4. valUnNeg: Un predicat sobre una llista que valgui cert només si la llista conté un sol valor negatiu. L=[1,4,-3]; cert.

EXERCICI 2.3 Una forma d'accelerar el càlcul del producte escalar quan tenim vectors de dimensió molt elevada és subdividir els vectors per la meitat, calcular el producte escalar per a cadascun dels vectors en un procés independent i sumar els productes parcials obtinguts dels processos. Aquest és l'objectiu d'aquest exercici.

a) Seguiu els següents passos per implementar la solució:

1. Definiu una funció tal que donades dues llistes (vectors) calcula el producte escalar.
2. Definiu una funció que defineix un procés calculador de productes escalars. Aquesta funció rep tres paràmetres: les dues llistes que corresponen als vectors i l'identificador del procés al que cal retornar el resultat.
3. Finalment definiu la funció pe(), que rep com a paràmetres dues llistes, les subdivideix per la meitat (useu la funció lists:split()), arrenca dos processos de càlcul concurrents cadascun amb una de les meitats del problema i espera a que tornin el resultat.

Per demostrar que els processos s'executen correctament, podeu utilitzar la funció de depuració de l'Erlang. Per usar-la, s'ha de compilar el mòdul amb l'opció: debug info

c(nom_modul, [debug_info]).

I cridar el depurador amb la comanda: debugger:start().

Busqueu informació sobre aquest depurador per poder fer les comprovacions oportunes.

b) Seguint la idea emprada a l'apartat a implementeu una funció que ordeni una llista usant dos subprocessos. L'estratègia consisteix a subdividir la llista original en dues subllistes, ordenar-les en sengles subprocessos usant la funció lists:sort(), i fusionar les subllistes ordenades obtingudes usant la funció lists:merge().

Calculeu el temps d'execució que necessita aquesta funció i compareu-lo amb el temps que tarda si s'utilitza només la funció sort(). Per calcular el temps d'execució, podeu utilitzar la funció: statistics(runtime).