CSP

Table of Contents

Concepts

GroovyCSP

The CSP (Communicating Sequential Processes) concurrency concept provides a message-passing model with synchronous rendezvous-type communication. It is valued mainly for its high level of determinism and the ability to compose parallel processes. GPars' GroovyCSP wraps the "JCSP library":http://www.cs.kent.ac.uk/projects/ofa/jcsp/ and builds on the work of "Jon Kerridge":http://www.iidi.napier.ac.uk/people/op/onepeople/peopleid/51 . For more information about the CSP concurrency model, checkout the CSP section of the User Guide or refer to the links below:

- "CSP definition":http://en.wikipedia.org/wiki/Communicating_sequential_processes
- "Google’s Go programming language with CSP-style concurrency":http://golang.org/

Usage

GroovyCSP — the Groovy API for CSP-style concurrency

```
import groovyx.gpars.csp.PAR

import org.jcsp.lang.CSProcess
import org.jcsp.lang.Channel
import org.jcsp.lang.ChannelOutput
import org.jcsp.lang.One2OneChannel

import groovyx.gpars.csp.plugAndPlay.GPrefix
import groovyx.gpars.csp.plugAndPlay.GPCopy
import groovyx.gpars.csp.plugAndPlay.GPairs
import groovyx.gpars.csp.plugAndPlay.GPrint

class FibonacciV2Process implements CSProcess {
    ChannelOutput outChannel

    void run() {
        One2OneChannel a = Channel.createOne2One()
        One2OneChannel b = Channel.createOne2One()
        One2OneChannel c = Channel.createOne2One()
        One2OneChannel d = Channel.createOne2One()
        new PAR([
            new GPrefix(prefixValue: 0, inChannel: d.in(), outChannel: a.out()),
            new GPrefix(prefixValue: 1, inChannel: c.in(), outChannel: d.out()),
            new GPCopy(inChannel: a.in(), outChannel0: b.out(), outChannel1: outChannel),
            new GPairs(inChannel: b.in(), outChannel: c.out()),
        ]).run()
    }
}

One2OneChannel N2P = Channel.createOne2One()
new PAR([
    new FibonacciV2Process(outChannel: N2P.out()),
    new GPrint(inChannel: N2P.in(), heading: "Fibonacci Numbers")
]).run()
```