Abstract machine for MPL (AMPL)

June 28, 2017

1 Concurrent MPL constructs

Concurrent MPL constructs are the constructs using which concurrent MPL programs are written. In this chapter, various concurrent MPL constructs are discussed.

2 Introduction

Concurrent MPL constructs are plug, id, hput-hcase, get-put, split-fork and close-halt. A few noticeable observations about concurrent MPL constructs are below:

- MPL constructs with the exception of run contruct perform an action on a channel. In MPL concurrency is modeled using *message passing* between processes and channels are the conduit through which the messages are exchanged. So, it is only logical that most constructs deal with channel.
- Most concurrent MPL constructs come in pair, i.e get-put, hput-hcase, split-fork and halt-close. id, run and plug are standalone constructs. The constructs of the pair are dual to each other. This is intuitive because in message passing view of concurrency for someone to respond to an action, someone else should drive that action on the opposite end of the channel and viveversa. Thus one of the constructs of in a pair is a driver construct and the other is a reaction construct. For example, for a process to receive a value on a channel, some other process should have put a value on the channel.

A brief description of the constructs are as below:

run	calls a process
id	equates two channels
plug	connects two processes by a channel
get	gets a value on a channel
put	puts a value on a channel
hput	puts a handle on a channel
hcase	cases on the handles obtained on channel
split	splits a channel into two channels
fork	forks two new processes
close	closes a channel
halt	closes a channel

Table 1: Machine Transitions for the SAMPL

In the next section, the concurrent MPL constructs are described in details with examples.

3 get-put

These are the simplest MPL constructs. `get` receives a value on a channel and `put` puts a value on a channel.