Answers Exercises Chi 3.0 Tutorial

M.A. Reniers*

December 16, 2013

Exercise 1.2.1

–

Exercise 1.2.2.a

```
model M():
   writeln("It works")
end
```

The text `It works` is written to the screen.

Exercise 1.2.2.b

```
model M(string str):
   write("%s\n", str)
end
```

The text `OOPS` is printed.

Exercise 2.5.1

The results are `-5^3 = -125`, `-5 * 3 = -15` and `-5 mod 3 = 1`, respectively. To convince yourself you may consider executing the following program:

```
model M():
   writeln("%f", -5^3);
   writeln("%d", -5 * 3);
   writeln("%d", -5 mod 3)
end
```

Exercise 2.5.2

The exercise is wrong in the tutorial. The type definition should be as follows:

*Comments on these answer can be e-mailed to `M.A. Reniers@tue.nl`. Please note that many times other solutions may exist than presented here.

```
type box = tuple(string name; real weight);

model M():
   box x = ("White", 12.5);

   writeln("%s", x.name);
   writeln("%f", x.weight);
   writeln("%s", x)
end
```

The results are `x.name = White`, `x.weight = 12.500000` and `x = (White, 12.5)`.

Exercise 2.5.3

```
model M():
   list int xs;

   xs = [0,1,2,3,4,5,6];
   writeln("%d", xs[0]);
   writeln("%s", xs[1:]);
   writeln("%d", size(xs));
   writeln("%s", xs + [3]);
   writeln("%s", [4,5] + xs);
   writeln("%s", xs - [2,2,3]);
   writeln("%s", xs - xs[2:]);
   writeln("%s", xs[0] + (xs[1:])[0])
end
```

The outcomes are as follows

- `xs[0] = 0`
- `xs[1:]  = [1, 2, 3, 4, 5, 6]`
- `size(xs) = 7`
- `xs + [3] = [0, 1, 2, 3, 4, 5, 6, 3]`
- `[4,5] + xs = [4, 5, 0, 1, 2, 3, 4, 5, 6]`
- `xs - [2,2,3] = [0, 1, 4, 5, 6]`
- `xs - xs[2:]  = [0, 1]`
- `xs[0] + (xs[1:])[0] = 1`

Exercise 3.6.1

```
model M():
   int i = 3;

   if (i < 0) == true:
      write("%d is a negative number\n", i);
   elif (i <= 0) == false:
      write("%d is a positive number\n", i);
   end
```

```
end

model N():
  int i = 0;

  if i < 0:
     write("%d is a negative number\n", i);
   elif i > 0:
      write("%d is a positive number\n", i);
   end
end
```

Exercise 3.6.2

a.
```
model M():
      list int l = <int>[];

      for i in range(0,10):
         l = l + [i*i]
      end;
      writeln("%s", l);
   end
```

b.
```
model M():
      list int l = <int>[];
      int j = 0;

      while j<10:
         l = l + [j*j];
         j = j + 1
      end;
      writeln("%s", l)
   end
```

Exercise 3.6.3

```
model M():

   list int primes = <int>[];
   int i = 2;
   int sum = 0;
   int lsum = 0;
   bool b;

   while size(primes) < 50:
      b = true;
      for j in primes:
         b = b and (i mod j != 0)
      end;
      if b:
         primes = primes + [i]
      end;
      i = i+1
```

```
   end;
   writeln("%s", primes);

   for j in primes[0:7]:
      sum = sum + j
   end;
   writeln("%s", sum);

   for j in primes[39:]:
      lsum = lsum + j
   end;
   writeln("%s", lsum)
end
```

Exercise 6.3.1

a.
```
model M(real a, b; int N):
      dist real u = gamma(a,b);
      real sum = 0.0;

      for i in range(1,N+1):
         sum = sum + sample u
      end;
      writeln("%s", sum / N)
   end
```

Pick some instances for a, b and N and check that indeed the produced mean is close to $a \cdot b$.

b. -

Exercise 6.3.2

```
model M():
   dist real u = triangle(1.0,2.0,5.0);
   real store;
   real mean = 0.0;
   real variance;

   for i in range (1,1001):
      store = sample u;
      mean = (mean * (i-1) + store) / i;
      if i != 1:
         variance = ((i-2) * variance + (store - mean)^2) / (i-1)
      else: variance = 0.0
      end
   end;
   writeln("mean = %s", mean);
   writeln("variance = %s", variance)
end
```

Exercise 6.3.3

a. Initially the score is `1` as can be seen from the initial value assigned to the variable `sc`.

b. There is (in principal) no maximum end score, although the range of the integers in Chi will be limited and therefore an error may be generated.

c. In that case the game stops.

d.
```
model HoL():
    dist int u = uniform(1, 15);
    int sc = 1;
    bool c = true, d = true;
    int new, oldval;
    string s;

    new = sample u;
    write("Your score is %d\n", sc);
    write("The computer drew %d\n", new);
    while c and d:
       writeln("(h)igher or (l)ower:\n");
       s = read(string);
       oldval = new;
       new = sample u;
       write("The computer drew %d\n", new);
       if new == oldval:
          c = false;
       else:
          c = (new > oldval) == (s == "h");
       end;
       if c:
          sc = 2 * sc;
       else:
          sc = 0;
       end;
       write("Your score is %d\n", sc);
       if c:
          write("Do you want to (s)top, or (c)ontinue?\n");
          s = read(string);
          d = s == "c"
       end
    end;
    write("GAME OVER...\n")
end
```

Exercise 8.5.1

a. De simulator rapporteert dat er een "deadlock" is op tijdstip `0.0`. Dit wil zeggen dat geen van de bestaande processen in staat is een statement te executeren, en zelfs dat ze niet in staat zijn dat later in de tijd wel te doen!

b. De deadlock wordt veroorzaakt doordat process `P(a,b)` staat te wachten op een communicatie over kanaal `a`, terwijl het andere proces, `P(b,a)` staat te wachten op een communicatie over kanaal `b`.

Exercise 8.5.2

a.
```
proc R(chan! int fout):
```

```
   int i;

   while i < 10:
      fout!i;
      i = i+1
   end
end
```

b.
```
proc Cadd(chan? int fin; chan! int fout; int add):
   int x;

   while true:
      fin?x;
      fout!x+add
   end
end
```

c.
```
proc Cmul(chan? int fin; chan! int fout; int mul):
   int x;

   while true:
      fin?x;
      fout!x*mul
   end
end
```

d.
```
proc W(chan? int fin):
   int x;

   while true:
      fin?x;
      write("Uitkomst: %s\n", x)
   end
end
```

e.
```
model SixChildren():
   chan int a, b, c, d, e;

   run R(a), Cadd(a,b,2), Cmul(b,c,3), Cmul(c,d,2), Cadd(d,e,6), W(e)
end
```

f. De gerapporteerde uitkomsten zijn:

```
Uitkomst: 18
Uitkomst: 24
Uitkomst: 30
Uitkomst: 36
Uitkomst: 42
Uitkomst: 48
Uitkomst: 54
Uitkomst: 60
Uitkomst: 66
Uitkomst: 72
Deadlock detected at time 0.00.
```

Exercise 9.5.1

a.
```
proc F(chan? int a; chan! int b):
    int x;
    list int xs;

    while true:
       select a?x:
          xs = xs + [x]
       alt size(xs)>0, b!xs[0]:
          xs = xs [1:]
       end
    end
end
```

De simulatie geeft de volgende uitvoer:

```
E received 0
E received 1
E received 2
E received 3
E received 4
E received 5
E received 6
E received 7
E received 8
E received 9
Deadlock detected at time 0.00.
```

Alle producten bereiken dus het exit proces.

b. De simulatie resulteert in de volgende output:

```
E received 0
E received 1
E received 2
E received 3
E received 4
E received 5
E received 6
E received 7
E received 8
Deadlock detected at time 0.00.
```

Het is nu dus niet zo dat alle producten het exit proces bereiken. De reden daarvoor is dat de parameter `low` nu 1 is. Deze parameter geeft aan hoeveel producten er minimaal in de fabriek aanwezig moeten zijn (de opstartfase uitgezonderd). Preciezer geformuleerd: de controller geeft pas toestemming aan het exit proces om een product te ontvangen van de fabriek wanneer het aantal producten dat in de fabriek aanwezig is meer is dan `low`. Dus bij `low = 1` moeten er twee producten in de fabriek zijn voordat er een door het exit proces ontvangen mag worden.

c. Vervang het fabrieksproces door het volgende proces:

```
proc F(chan? int a; chan! int b):
   int x;
```

```
   list int xs;

   while true:
      select a?x:
         xs = [x] + xs
      alt size(xs)>0, b!xs[0]:
         xs = xs [1:]
      end
   end
end
```

Simulatie met `low=0` en `high=1` geeft dat alle producten netjes afgeleverd worden in de juiste volgorde.

Simulatie met `low=1` en `high=4` geeft dat (1) niet alle producten netjes afgeleverd worden, en (2) dat de volgorde van de producten anders is geworden.

d. Wanneer de generator alleen in staat is om producten in een veelvoud van `high-1` te genereren dan blijven er geen producten achter.

```
proc G(chan! int a; chan? void sg):
   for i in range(9):
      sg?;
      a!i;
   end
end
```

Exercise 10.5.1

a. Het aanpassen van de eenheid van tijd moet resulteren in het aanpassen van alle aanwezige delay statements. In dit geval is dat er maar één. Het delay statement beschrijft een delay van driekwart minuut, oftewel 45 seconden.

```
proc P():
   for i in range(3):
      write("i = %d, time = %f\n", i, time);
      delay 45 # seconds
   end
end
```

b. Dezelfde aanpak: 45 seconden = 45/12 tijdseenheden van 12 seconden.

```
proc P():
   for i in range(3):
      write("i = %d, time = %f\n", i, time);
      delay 45/12 # time units of 12 seconds
   end
end
```

Exercise 10.5.2

In dit model zal de generator proberen om items aan te bieden om de 4 tijdseenheden. Aangezien de server 5 tijdseenheden nodig heeft om een item te verwerken en dus open te staan voor het

ontvangen van het volgende item, en er tussen de generator en de server geen buffer aanwezig is, zal de feitelijke interarrival time 5 tijdseenheden zijn!

De voorspelde througput is dus één item per 5 tijdseenheden, dus 1/5. De flowtime van alle items is 5[1]

Simulatie van het model bevestigt deze voorspellingen.

Exercise 10.5.3

a. De server kan geplaatst worden tussen het proces `F` en het exit proces `E`. Let erop dat de bijbehorende kanalen goed aangebracht worden (zie model `M` hieronder).

```
type item = tuple(int id; real entrytime);

proc G(chan! item a; chan? void sg; int N):
   for i in range(N):
      sg?;
      a!(i,time);
   end
end

proc F(chan? item a; chan! item b):
   item x;
   list item xs;

   while true:
      select a?x:
         xs = xs + [x]
      alt size(xs) > 0, b!xs[0]:
         xs = xs [1:]
      end
   end
end

proc E(chan? item a; chan? void se):
   item x;
   real aft;

   while true:
      se?;
      a?x;
      aft = (aft * x.id + time - x.entrytime) / (x.id + 1);
      write("E received %s\t AFT = %s\n", x,aft);
   end
end

proc C(chan! void sg, se; int low, high):
   int count;

   while true:
      while count < high:
         sg!;
```

[1]Merk op dat de impliciete wachttijd in de generator niet meegenomen is in het vaststellen van de flowtime en de throughput.

```
            count = count + 1;
         end
         while count > low:
            se!;
            count = count - 1;
         end
      end
   end

   proc S(chan? item fs; chan! item sse):
      item x;

      while true:
         fs?x;
         delay 4.0;
         sse!x
      end
   end

   model M(int N):
      chan void sg, se;
      chan item gf, fs, sse;

      run C(sg, se, 0, 1), G(gf, sg, N), F(gf, fs), S(fs,sse), E(sse, se);
   end
```

Simulatie van dit model (met parameter `N` groot genoeg) geeft als conclusie dat de gemiddelde flowtime 8 is.

b. Simulatie van hetzelfde model met aangepaste waarden voor `low` en `high` geeft een gemiddelde flowtime van 16.

c. Vervang proces `F` door de volgende

```
proc F(chan? item a; chan! item b):
   item x;
   list item xs;

   while true:
      select a?x:
         xs = [x] + xs
      alt size(xs) > 0, b!xs[0]:
         xs = xs [1:]
      end
   end
end
```

Simulatie geeft aan dat de flowtime 12 is.

Exercise 11.4.1

Opmerking vooraf: het is bij deze opgave niet duidelijk wat de interarrival time van de items zou moeten zijn.

```
type item = tuple(int id; real at);
type conv_item = tuple(item pid; timer t);

proc G(chan item gb):
   for i in range(1000):
      gb!(i,time);
      delay 5.0
   end
end

proc B(chan item gb, bt):
   list item xs;
   item x;

   while true:
      select gb?x:
         xs = xs +[x]
      alt size(xs)>0, bt!xs[0]:
         xs=xs[1:]
      end
   end
end

proc T(chan item bt, te):
   dist real u = exponential(4.0);
   list conv_item xs;
   item x;

   while true:
      select size(xs)<3, bt?x:
         xs = xs + [(x,timer(sample u))];
         write("Item %s with conveyor time %s arrived on conveyor\n", x, real(xs[-1].t));
      alt size(xs)>0 and ready(xs[0].t), te!xs[0].pid:
         write("Item %s left the conveyor\n", xs[0].pid);
                xs=xs[1:]
      end;
   end
end

proc E(chan item te):
   item x;
   real aft;

   while true:
      te?x;
      aft = ((aft * x.id) + time - x.at) / (x.id + 1);
      write("E received %s\t AFT = %s\n", x, aft);
   end
end

model M() :
   chan item gb, bt, te;

   run G(gb), B(gb,bt), T(bt,te), E(te)
```

```
end
```

Enkele experimenten geven de waarden: `5.20`, `5.26`, `4.93`, `5.45`, `5.20`. [2]

Exercise 11.4.2

Opmerking vooraf: het is bij deze opgave niet duidelijk wat de interarrival time van de items zou moeten zijn.

a.
```
type item = tuple(int id; real at);
type conv_item = tuple(item pid; timer t);

proc G(chan item gb):
   for i in range(1000):
      gb!(i,time);
      delay 5.0
   end
end

proc B(chan item gb, bt):
   list item xs;
   item x;

   while true:
      select gb?x:
         xs = xs +[x]
      alt size(xs)>0, bt!xs[0]:
         xs=xs[1:]
      end
   end
end

proc S(chan item bt, te):
   dist real u = exponential(4.0);
   item x;

   while true:
      bt?x;
      delay sample u;
      te!x
   end
end

proc E(chan item te):
   item x;
   real aft;

   while true:
      te?x;
      aft = ((aft * x.id) + time - x.at) / (x.id + 1);
      write("E received %s\t AFT = %s\n", x, aft);
```

[2] Ik geloof dat de experiment omgeving van Chi gebruikt kan worden om veel van deze experimenten automatisch te doen. Wellicht in een volgende versie van de uitwerkingen hier ook gebruiken.

```
   end
end

model M() :
   chan item gb, bt, te;

   run G(gb), B(gb,bt), S(bt,te), S(bt,te), S(bt,te), E(te)
end
```

In dit geval zitten de metingen rond de 4.

b. In het model van deze opgave is de volgorde van items niet gegarandeerd, terwijl dat in het model van de vorige opgave wel het geval is.