21.1

[A] Type Info
Channel c

Value
P: Unit → out c Unit
Q: Unit → in c Unit
P() ≡ {P() || Q() | {x...m}} P();

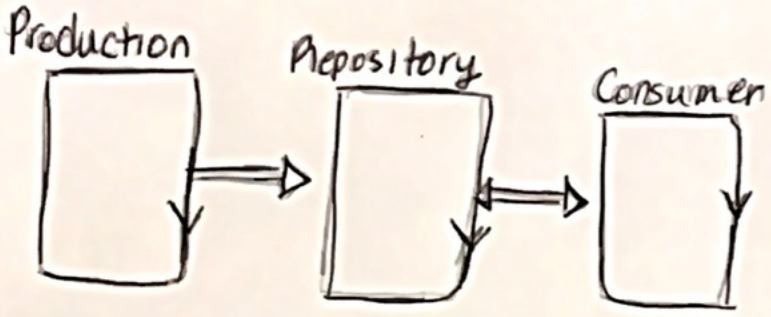
[B]
Type Info
Channel c[1...m]
Value

P() ≡ ||{ P() || Q()x{1...m}} P();

[C] Type Info
Chanel c
Value:
P() ≡ || {P()x{1...m} || Q() x {1...n}}

[D] Type Info
Chanel c[1...m]
value
P() ≡ || {P() x{1...m} || Q() x{1...n}}

a1.2




Type
Object

Channel
pr, rc

Value
Producer: Unit → out pr Object

Producer() ≡ let o = create_object() in pr ! o end; Producer()

Repository: Unit → in pr Unit

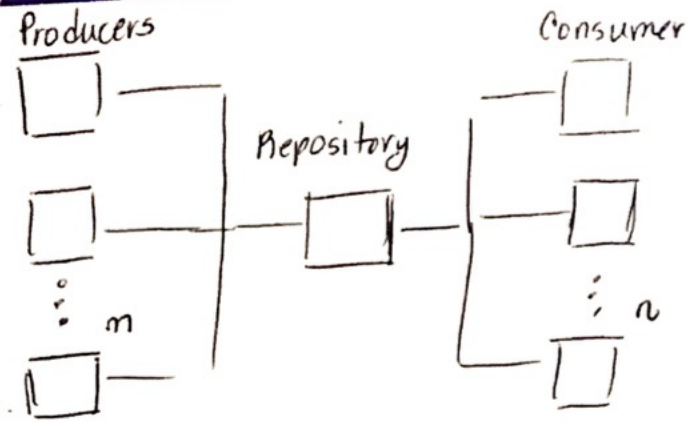
Repository() ≡ let o = pr ? in end let r = rc ? in end in rc !, o end;
Repository()

Consumer: Unit → in, out rc Unit

Consumer() ≡ let r = request() in rc ! let o = rc ? in end Consumer()

System : Producer() || Repository() || Consumer()

21.3




Type

Object, requests, ProduceID

Value

Producers: p : ProducerIndex -> out pr[p] Unit

Producer() ≡ let o = create_object(); let (p,o): (producerIndex, Object()) in pr ! (p,o) end;

Producer()

Repository: Unit -> in { pr[p] | p: Producer index } in; out { rc[c] | c: consumer Index }

Unit

Repository() ≡ let (id,o) = pr[p]? in end let r = rc[c]? in end

in rc[c] ! (id,o) end; Repository()

Consumer : Unit -> in rc[c] unit

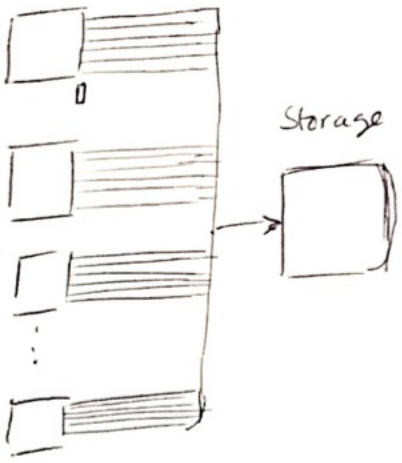
Consumer() ≡ let r = request() in rc[c] ! r end let (id,o) = rc[c]? in end;

Consumer()

System: Unit -> Unit

System() ≡ (|| Producer() | x {1...n}) || Repository() || (|| Consumer() | r: {1...m})

21.4 Comp Process




Type
Data, location

Channel
sd[1...m], fd[1...m], ud[1...m], fv[1...m], rda[1...m], deac[1...m]

Value
Computation : Unit → out sd, fd, ud, rv, rda, deac Unit

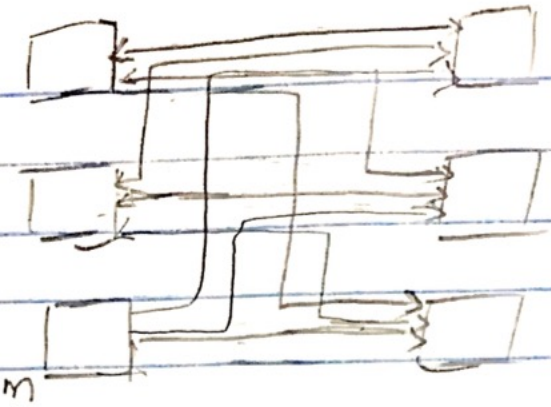
Computation () ≡ let (d, L) = (data, location) in sd
⊓ let (d, L) = (data, location) in fd
⊓ let (d, L) = (data, location) in rv
⊓ let (d, L) = (data, location) in rda
⊓ let (d, L) = (data, location) in deac ; Computation()

Storage : Unit → in sd, fd, ud, ru, rda, deac
Storage () ≡
□ let (d, L) = sd ? in reset () .. save() end
□ let (d, L) = fd ? in request() .. save() end
□ let (d, L) = ud ? in request() .. save() end
□ let (d, L) = rv ? in request() .. save() end
□ let (d, L) = rda ? in request() .. save() end
□ let (d, L) = deac ? in request() ... saved end ;
Storage ()

System : Unit → Unit
System () ≡ Storage() || Comp Process
| x : [1...n] }
System()

Clients Servers

21.7




Client provides identifier to its action

Type
Object, Request

Channel
cs[1...m]

Value

Client: Unit → Unit
Client() ≡ let r = request() in cs end

Server Unit → in cs[r] Unit
Server() ≡ let r = request in cs[c]! r
· Specific_action() end Server()

System Unit → Unit

System() ≡ (Clients() | x{1...n}) || Servers() | x{1...m})