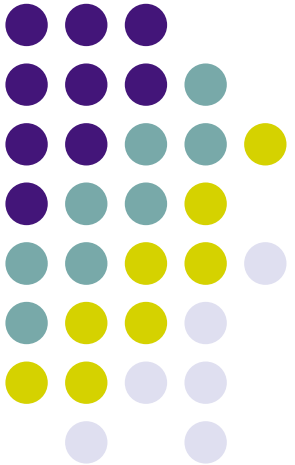
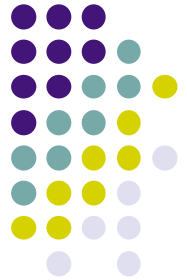
Stateless Agents out of Ports

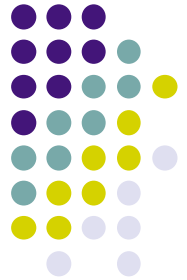


A Math Agent



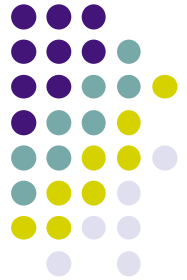
```
proc {Math M}
   case M
   of add(N M A) then A=N+M
   [] mul(N M A) then A=N*M
   [] int(Formula A) then
      A = …
   end
end
```

Making the Agent Work



```
MP = {NewPort S}
proc {MathProcess Ms}
   case Ms of M|Mr then
      {Math M} {MathProcess Mr}
   end
end
thread {MathProcess S} end
```

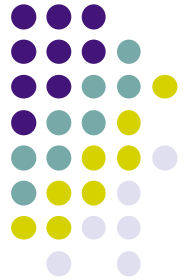
Smells of Higher-Order…



```
proc {ForAll Xs P}
   case Xs
   of nil  then skip
   [] X|Xr then {P X} {ForAll Xr P}
   end
end
```

- Call procedure `P` for all elements in `Xs`

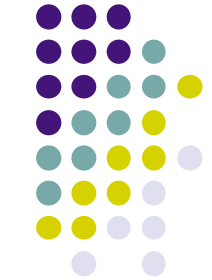
Smells of Higher-Order…



- Using `ForAll`, we have

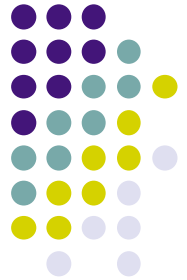
```
proc {MathProcess Ms}
   {ForAll Xs Math}
end
```

Making the Agent Work



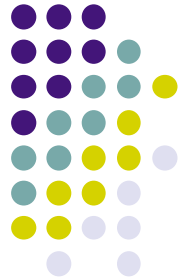
```
MP = {NewPort S}
thread {ForAll S Math} end
```

Making the Agent Work



```
MP = {NewPort S}
thread for M in S do {Math M} end
end
```

Smells Even Stronger…

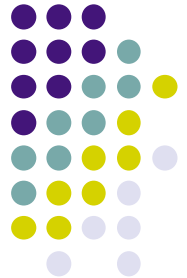


```
fun {NewAgent0 Process}
   Port Stream
in
   Port={NewPort Stream}
   thread {ForAll Stream Process} end
   Port
end
```

Smells Even Stronger…

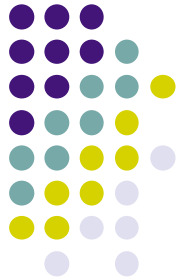
```
fun {NewAgent0 Process}
   Port Stream
in
   Port={NewPort Stream}
   thread
      for M in Stream do {Process M} end
    end
   Port
end
```

Why Do Agents/Processes Matter?



- Model to capture communicating entities
- Each agent is simply defined in terms of how it replies to messages
- Each agent has a thread of its own
  - no screw-up with concurrency
  - we can easily extend the model so that each agent have a state (encapsulated)
- ***Extremely useful to model systems!***

Summary



- Ports for message sending
  - use stream (list of messages) as mailbox
  - port serves as unique address
- Use agent abstraction
  - combines port with thread running agent
  - simple concurrency scheme
- Introduces non-determinism… and state!