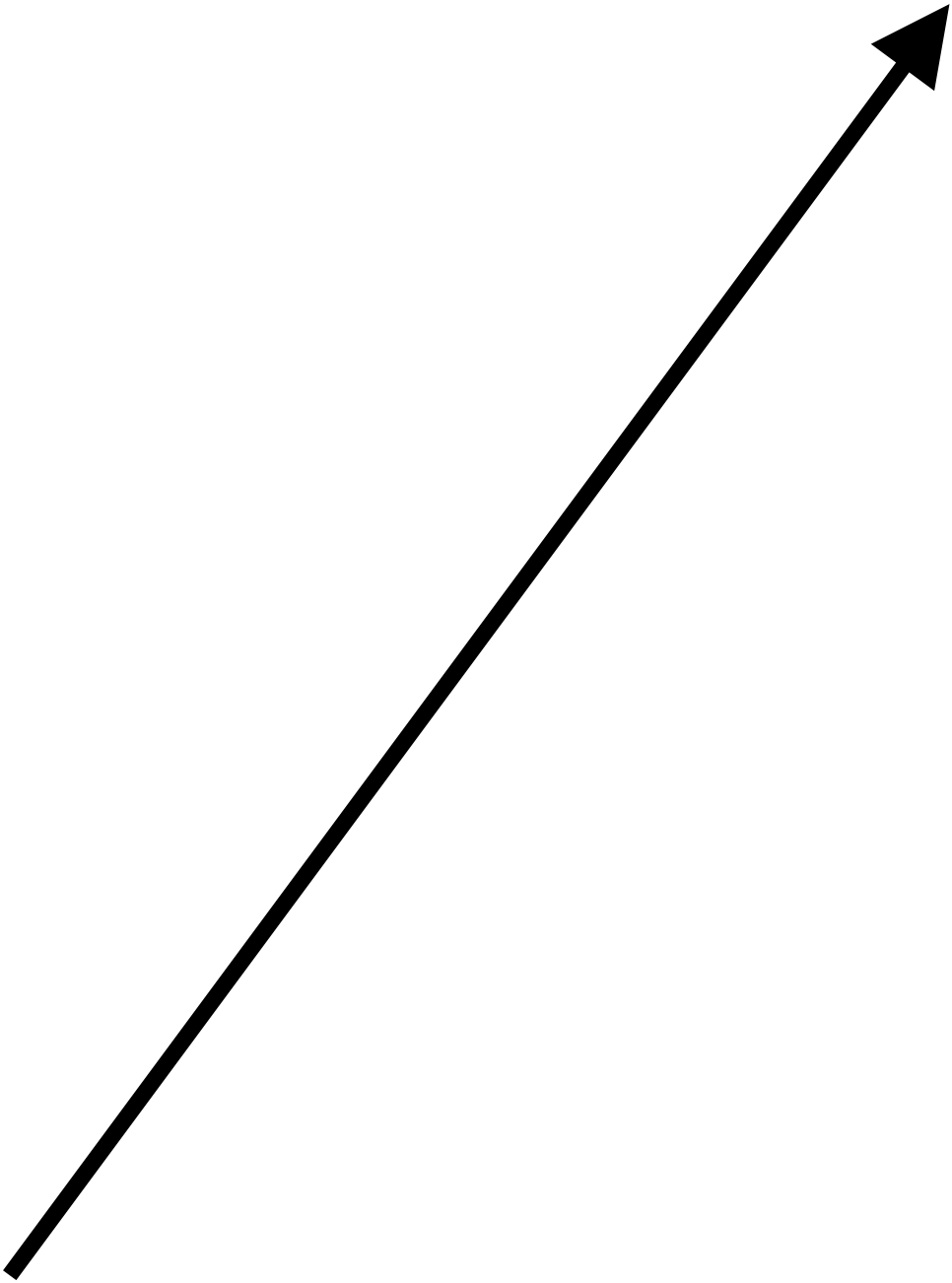
Compute in parallel

(…(**v**)**t**…)**t**n - **v** is a lambda-variable and **t** is a term

Could be thought as: [t1, t2, …] we can compute terms in parallel

```haskell
computeParallel :: [Term] -> IO [CTerm]
computeParallel terms =
  traverse (\term -> async $ krivineMachine term)
```



Spawns a thread that computes term

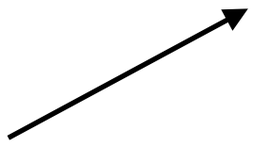
10

Actor System

Actor = process

- **Can receive messages**
- **Send messages**
- **Spawn new actors**
- **Make decisions about next steps (internal state)**

```haskell
computeParallel :: [Term] -> IO [CTerm]
computeParallel terms = do
  traverse (\term -> actorAsync $ krivineMachine term)
  awaitResults
```



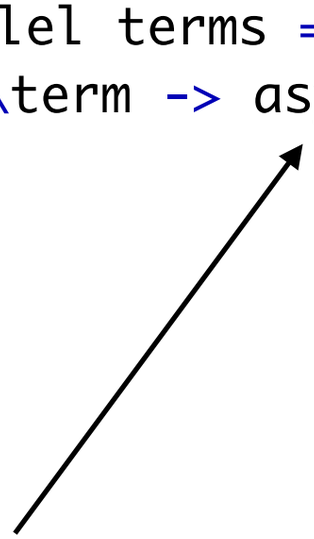
- **create actor that will do computation**
- **Send message to that actor, with term**
- **await results of computations and return them**

Compute in parallel

(…(**v**)**t**…)**t**n - **v** is a lambda-variable and **t** is a term

Could be thought as: [t1, t2, …] we can compute terms in parallel

```haskell
computeParallel :: [Term] -> IO [CTerm]
computeParallel terms =
  traverse (\term -> async $ krivineMachine term)
```



Spawns a thread that computes term