An Intuition for Propagators

George Wilson

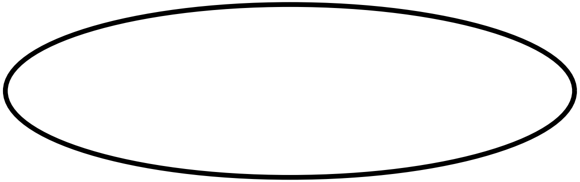
CSIRO's Data61

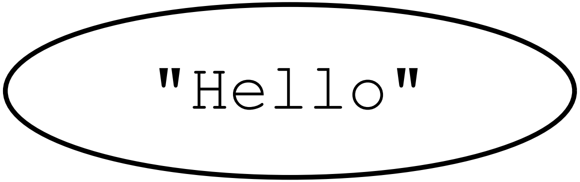
george.wilson@data61.csiro.au

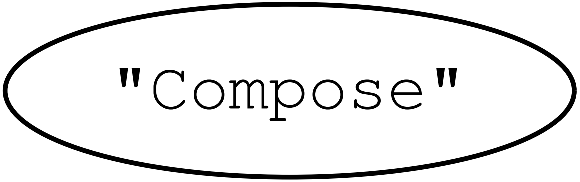
2nd September 2019

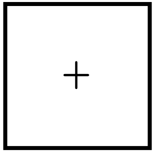
1970s, MIT

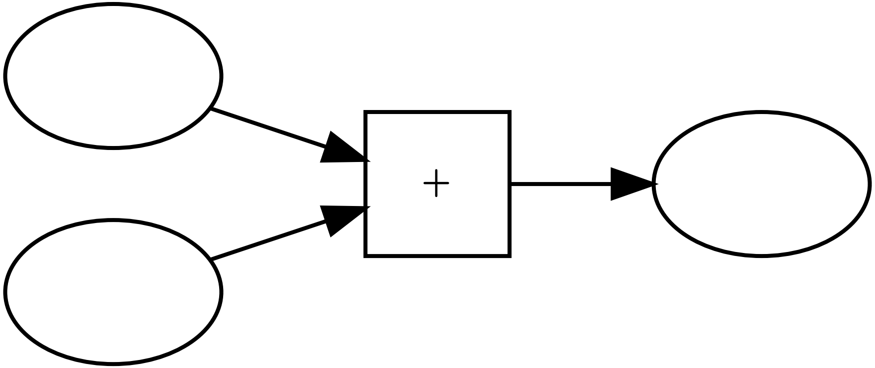
a model of computation for **highly concurrent** machines

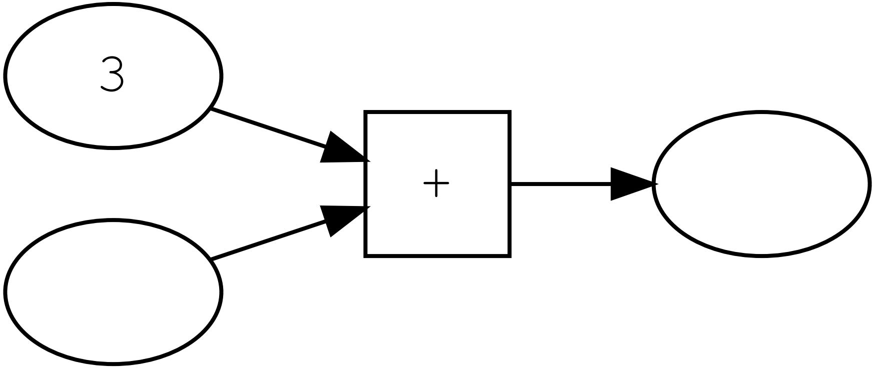


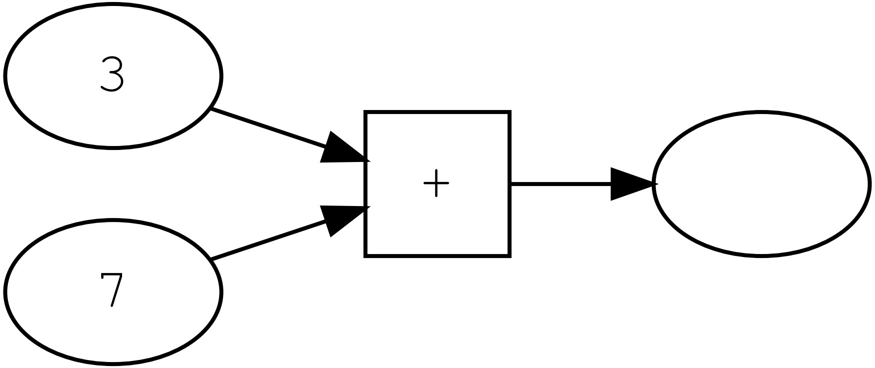

"Hello"

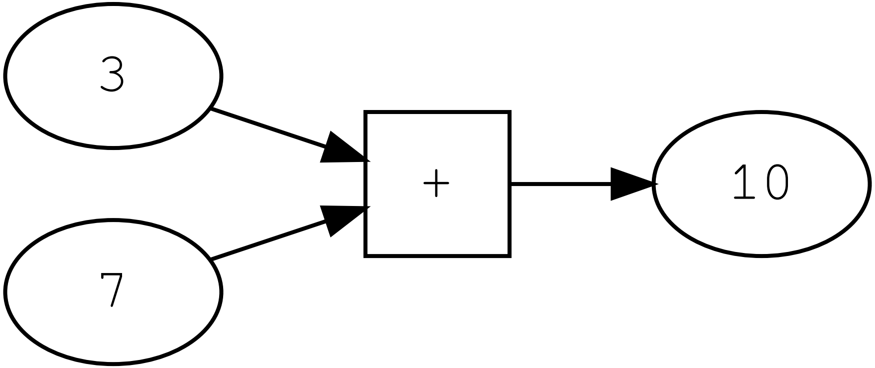

"Compose"

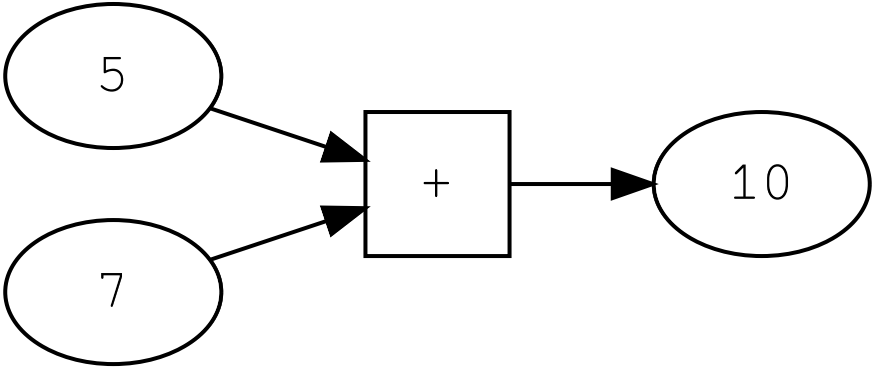


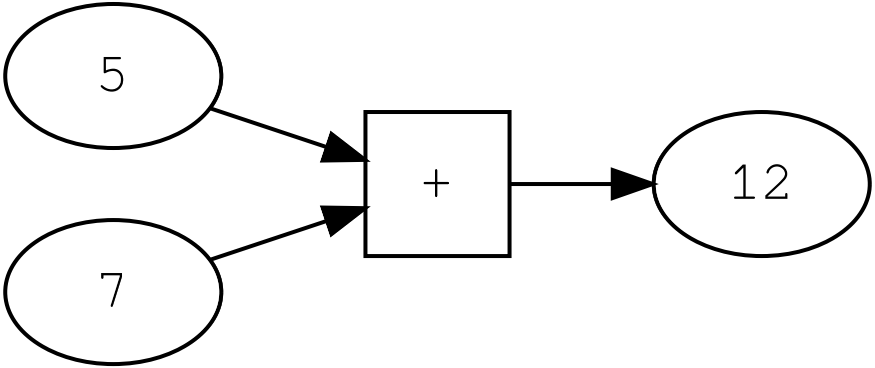



3
+


3
7
+


3
7
+
10


5
7
+
10

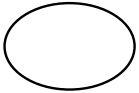

5
7
+
12

```haskell
-- types
data Cell a

data Par a
instance Monad Par
```

```haskell
-- types
data Cell a

data Par a
instance Monad Par


-- Creating a cell
cell    :: Par (Cell a)
```

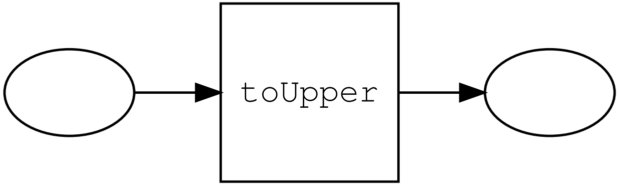
```haskell
-- types
data Cell a

data Par a
instance Monad Par

-- Creating a cell
cell    :: Par (Cell a)

-- Working with Cells
content :: Cell a -> Par (Maybe a)
write   :: Cell a -> a -> Par ()
```

```haskell
-- types
data Cell a

data Par a
instance Monad Par

-- Creating a cell
cell    :: Par (Cell a)

-- Working with Cells
content :: Cell a -> Par (Maybe a)
write   :: Cell a -> a -> Par ()

-- Creating a propagator
watch   :: Cell a -> (a -> Par ()) -> Par ()
```

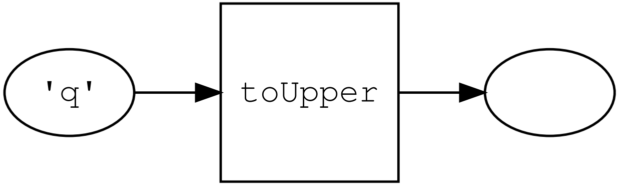


```
do
  input  <- cell
```

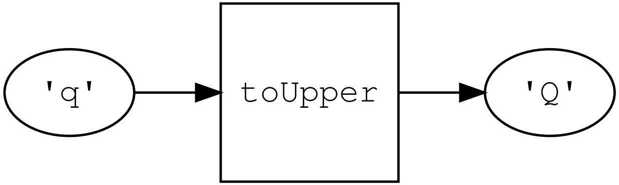
```haskell
do
  input  <- cell
  output <- cell
```



```
do
  input  <- cell
  output <- cell
  watch input (\c ->
    write output (toUpper c))
```



```haskell
do
  input  <- cell
  output <- cell
  watch input (\c ->
    write output (toUpper c))

  write input 'q'
  content output   -- Just 'Q'
```



```haskell
do
  input  <- cell
  output <- cell
  watch input (\c ->
    write output (toUpper c))

  write input 'q'
  content output   -- Just 'Q'
```

Thanks for listening!