Multi-agent programming

- In the last lesson we saw some simple examples of multi-agent programs
  - Producer/consumer
  - Producer/transformer/consumer (pipeline)
- Let’s see two more sophisticated examples
  - Sieve of Eratosthenes: dynamically building a pipeline during its execution
  - Digital logic simulation: using higher-order programming together with concurrency
The Sieve of Eratosthenes

- The Sieve of Eratosthenes is an algorithm for calculating a sequence of prime numbers.
- Each agent in the pipeline removes multiples of an integer.
- Starting with a sequence containing all integers, we end up with a sequence of primes.
A filter agent

- A list function that removes multiples of K:

  ```
  fun {Filter Xs K}
    case Xs of X|Xr then
      if X mod K \= 0 then X|{Filter Xr K}
      else {Filter Xr K} end
    else nil
  end
  end
  ```

- We make an agent by putting it in a thread:

  ```
  thread Ys={Filter Xs K} end
  ```
The Sieve program

- Sieve builds the pipeline during execution:

```plaintext
fun {Sieve Xs}
  case Xs
  of nil then nil
  [] X|Xr then X|{Sieve thread {Filter Xr X} end}
end
end

declare Xs Ys in
thread Xs={Prod 2} end
thread Ys={Sieve Xs} end
{Browse Ys}
```

Concurrent deployment: building the infrastructure of a program during execution
An optimization

- Otherwise too many do-nothing agents are created!

```haskell
fun {Sieve2 Xs M}
    case Xs
    of nil then nil
    [] X|Xr then
        if X=<M then
            X|{Sieve2 thread {Filter Xr X} end M}
        else Xs end
    end
end
```

- We call `{Sieve2 Xs 316}` to generate a list of primes up to 100000 (why?)