

# How to Make Your Process Invisible

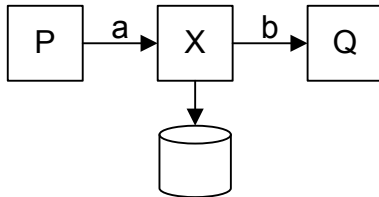
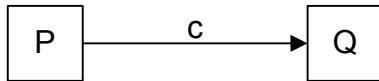
a.k.a The Dialing Philosophers

Neil Brown

Computing Laboratory  
University of Kent  
UK

7 September 2008

# The Invisible Process

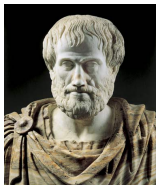




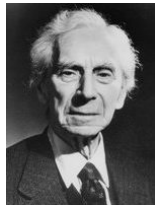
This presentation is concerned with how to take two processes connected via a single channel (here: P and Q connected by c) and interject a third process (here: X) without breaking the synchronisation behaviour between the original two processes (P and Q). We term this interjected process invisible. But rather than use processes, we will turn this into a problem involving eavesdropping on a phone call between two philosophers.

# The Invisible Philosopher

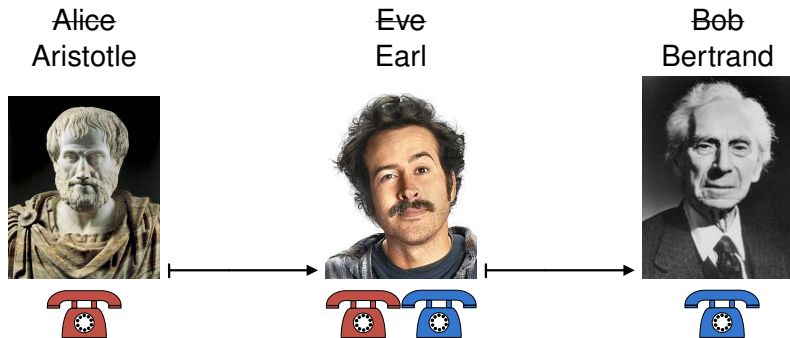
Alice  
Aristotle



Bob  
Bertrand



# The Invisible Philosopher

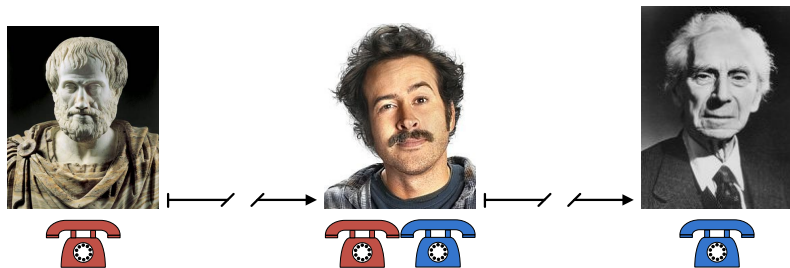


# A System of Telephones

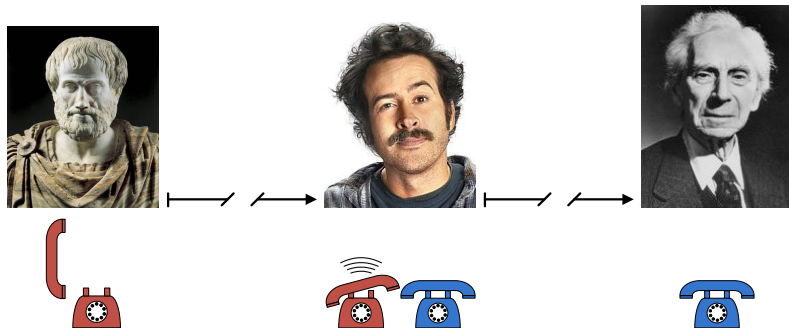
## The First Rules:

- 1** A philosopher (sequential code) may either:
  - do some work
  - commit to making one specific call where he will speak (write)
  - wait for exactly one of several calls where he will listen (read with choice)
  
- 2** A phone conversation (channel communication) comprises:
  - One philosopher picking up the phone, causing the other end to ring
  - The other end being picked up
  - The speaker speaking (sending data)
  - The listener saying good bye
  - Both hanging up

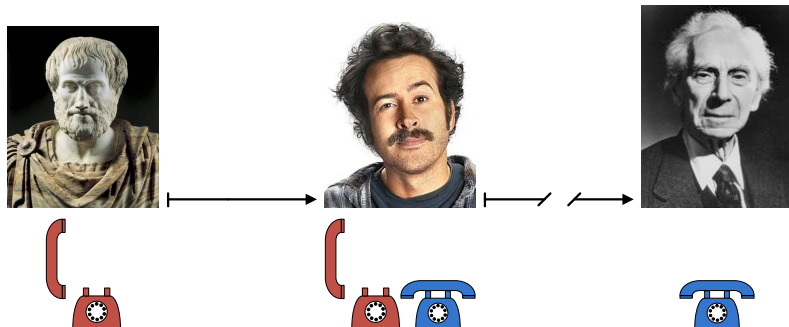
# Visible Process



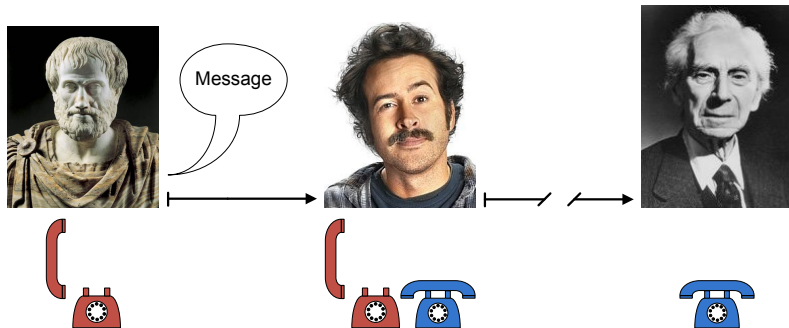
# Visible Process



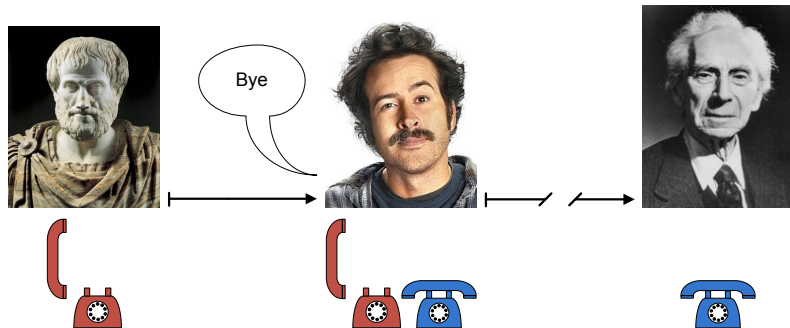
# Visible Process



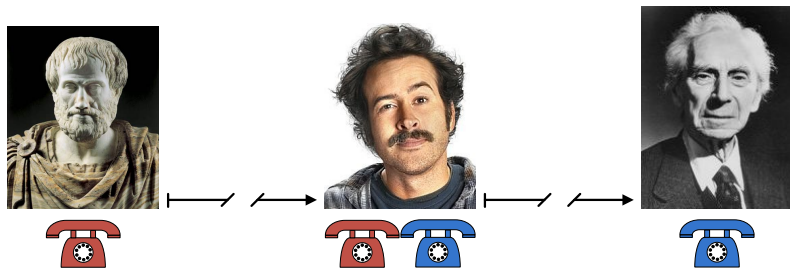
# Visible Process



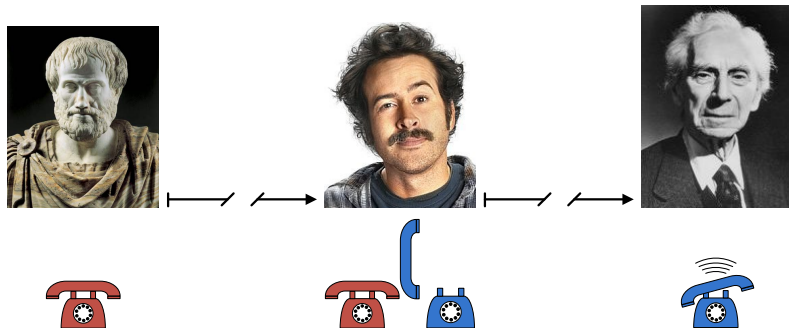
# Visible Process



# Visible Process



# Visible Process



2008-10-19

## How to Make Your Process Invisible

└ Introduction

└ Visible Process

Visible Process



At this point, we have become visible again; Aristotle thinks Bertrand has received his message, but in fact he has not. At this point if Aristotle contacted Bertrand by other means, he could discover this. In fact, Bertrand may never take the message from Earl, but Aristotle thinks he has done so.

# Extended Read

A phone conversation consists of:

- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
- The speaker speaking (sending data)
- The listener saying goodbye
- Both hanging up

# Extended Read

A phone conversation consists of:

- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
- The speaker speaking (sending data)
  - **Extended Read Action**
- The listener saying goodbye
- Both hanging up

# How to Make Your Process Invisible

└ Introduction

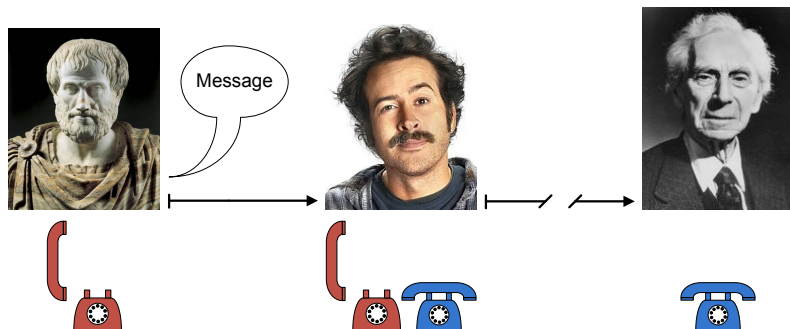
└ Extended Read

A phone conversation consists of:

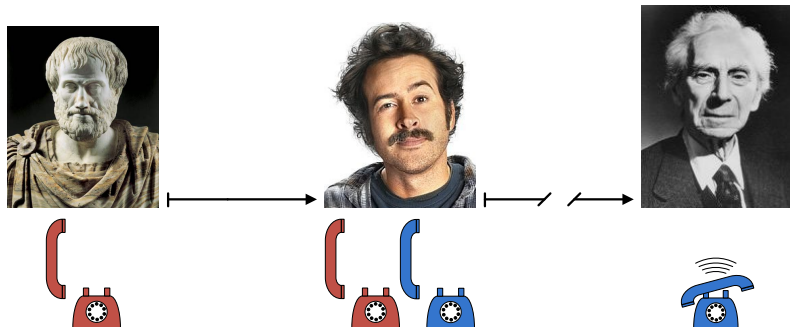
- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
- The speaker speaking (sending data)
  - **Extended Read Action**
- The listener saying goodbye
- Both hanging up

We can now make Earl invisible when choice is only offered over inputs, by forwarding the message to Bertrand during an extended read action in the communication from Aristotle.

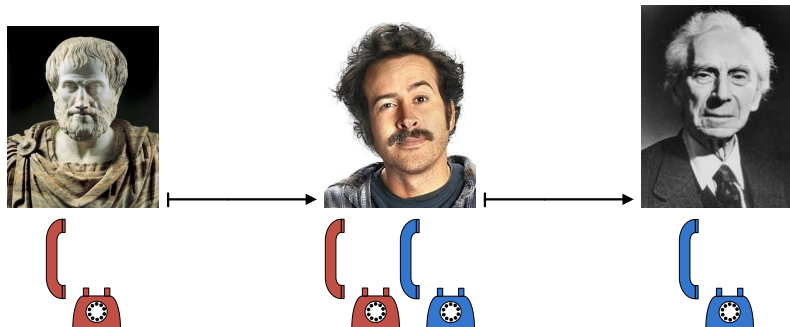
# Going Invisible in $\text{occam-}\pi$ and similar



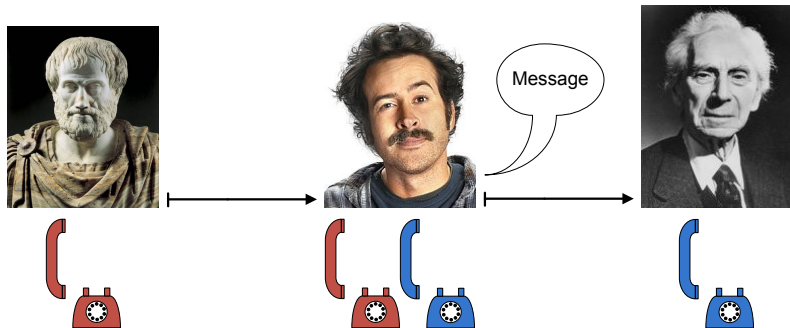
# Going Invisible in $\text{occam-}\pi$ and similar



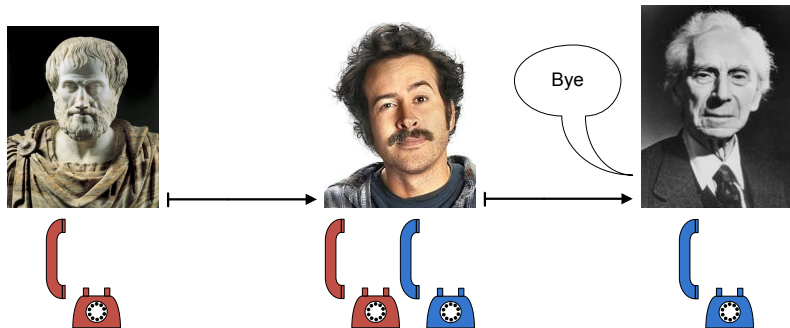
# Going Invisible in $\text{occam-}\pi$ and similar



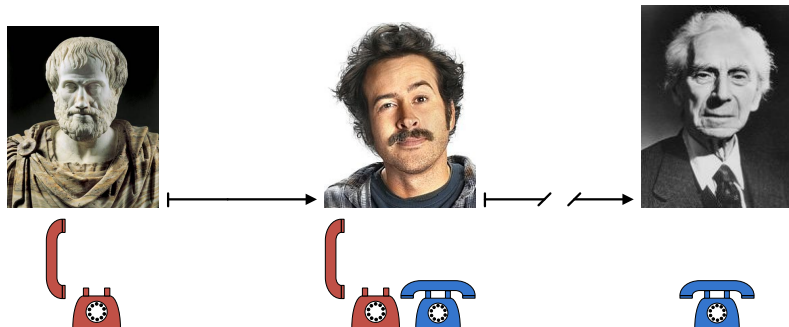
# Going Invisible in $\text{occam-}\pi$ and similar



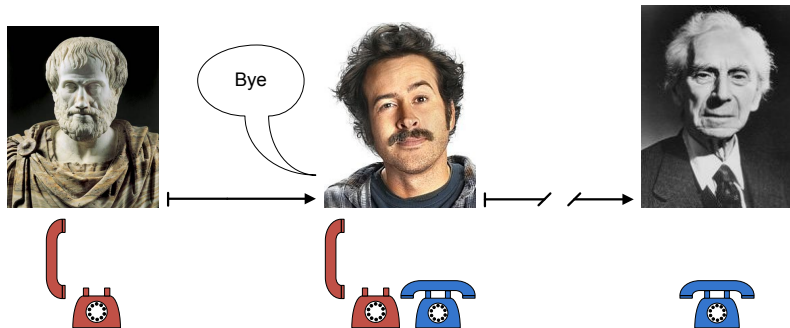
# Going Invisible in $\text{occam-}\pi$ and similar



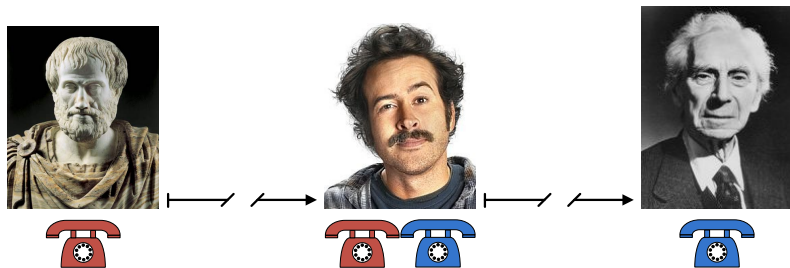
# Going Invisible in $\text{occam-}\pi$ and similar



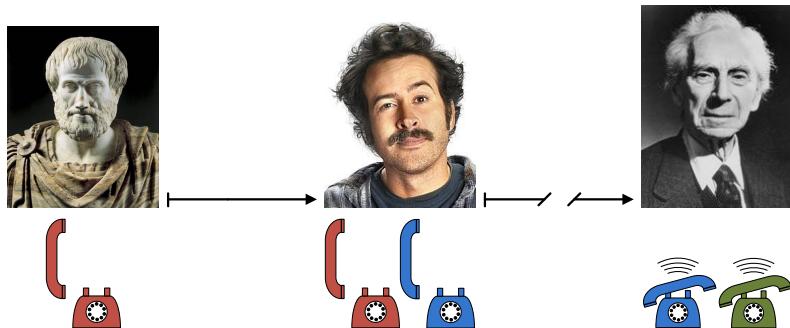
# Going Invisible in $\text{occam-}\pi$ and similar



# Going Invisible in $\text{occam-}\pi$ and similar



# Invisible, even with choice on inputs



# Job Done!

- Our process is now invisible by using extended reads, even when choice over inputs is considered.

# Job Done!

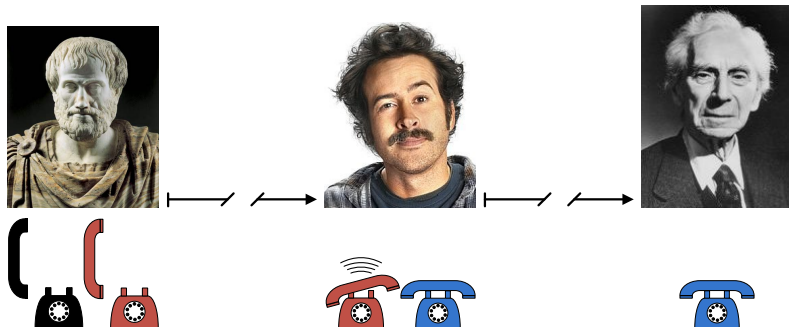
- Our process is now invisible by using extended reads, even when choice over inputs is considered.
- But what if your implementation supports choice over outputs instead?

# Visible with choice on outputs

## The New Rules:

- 1 A philosopher (sequential code) may either:
  - do some work
  - wait for one of several calls where he will speak (write with choice)
  - commit to making one specific call where he will listen (read)

# Visible with choice on outputs



2008-10-19

## How to Make Your Process Invisible

└ Introduction

└ Visible with choice on outputs

Visible with choice on outputs



We are already visible if there is choice on outputs. Earl is always prepared to accept input from Aristotle, and will do so even if Bertrand never takes the message. So Aristotle will choose to deliver the message to Bertrand rather than another philosopher, even though Bertrand will never take it!

# Extended Write

A phone conversation consists of:

- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
- The speaker speaking (sending data)
- The listener saying goodbye
- Both hanging up

# Extended Write

A phone conversation consists of:

- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
  - **Extended Write Action**
- The speaker speaking (sending data)
- The listener saying goodbye
- Both hanging up

A phone conversation consists of:

- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
  - Extended Write Action
- The speaker speaking (sending data)
- The listener saying goodbye
- Both hanging up

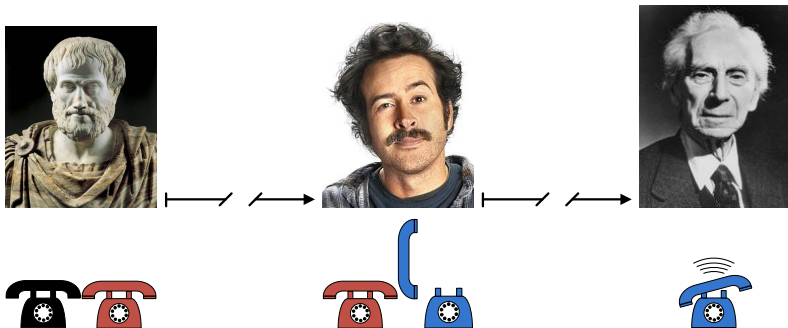
To be invisible with choice on outputs, we will reverse our pattern for Earl. Earl will start by offering to output to Bertrand, but once Bertrand is on the line, he will use his extended write action to offer to read data from Aristotle. This works if there is only choice on outputs (and not inputs).

# Extended Write and Read

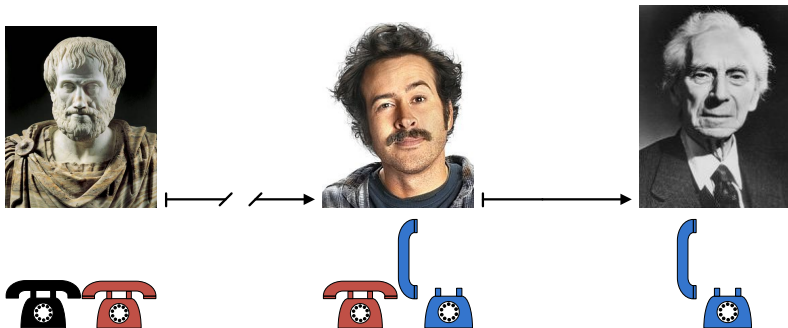
A phone conversation consists of:

- One philosopher picking up the phone, causing the other end to ring
- The other end being picked up
  - **Extended Write Action**
- The speaker speaking (sending data)
  - **Extended Read Action**
- The listener saying goodbye
- Both hanging up

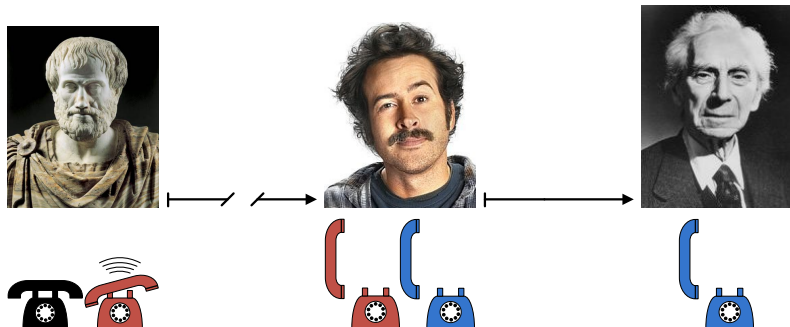
# Invisible with choice on outputs



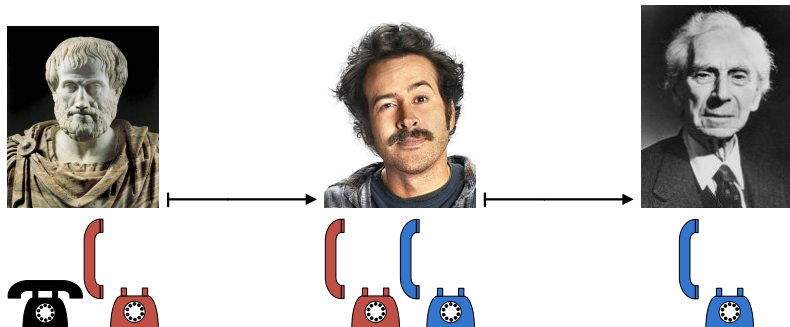
# Invisible with choice on outputs



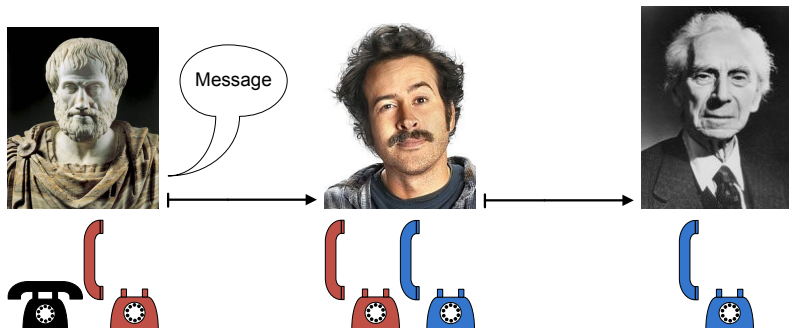
# Invisible with choice on outputs



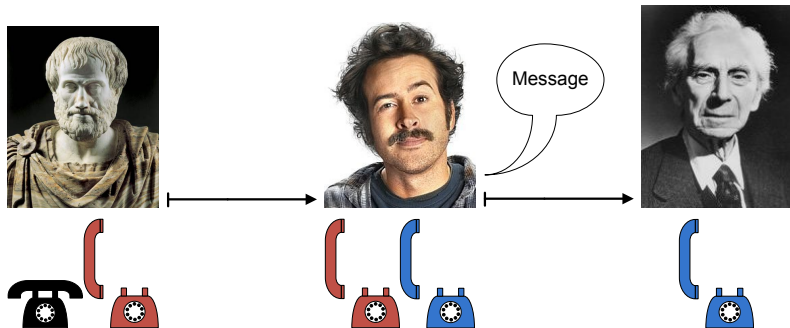
# Invisible with choice on outputs



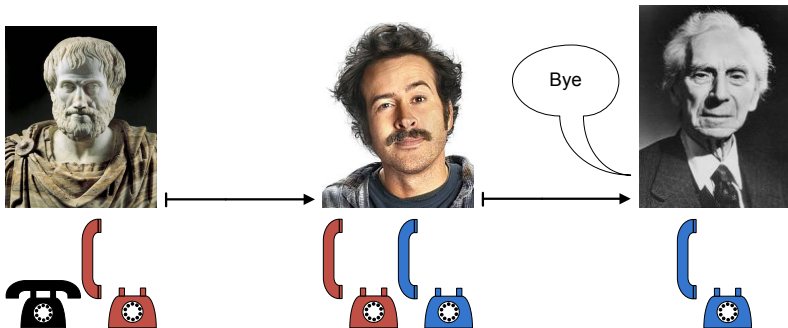
# Invisible with choice on outputs



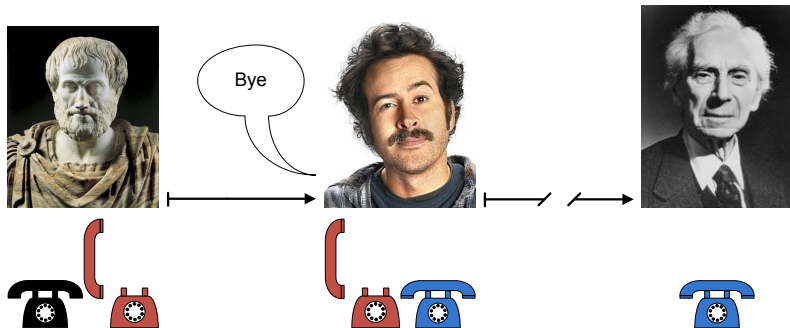
# Invisible with choice on outputs



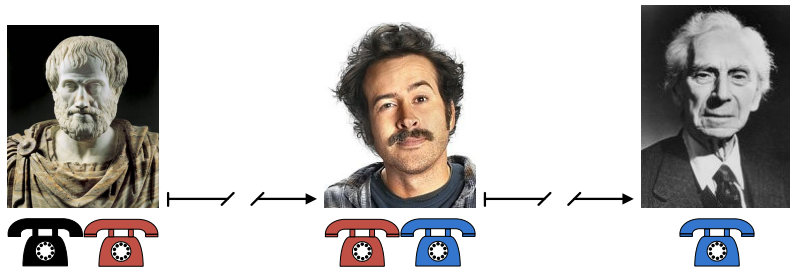
# Invisible with choice on outputs



# Invisible with choice on outputs



# Invisible with choice on outputs

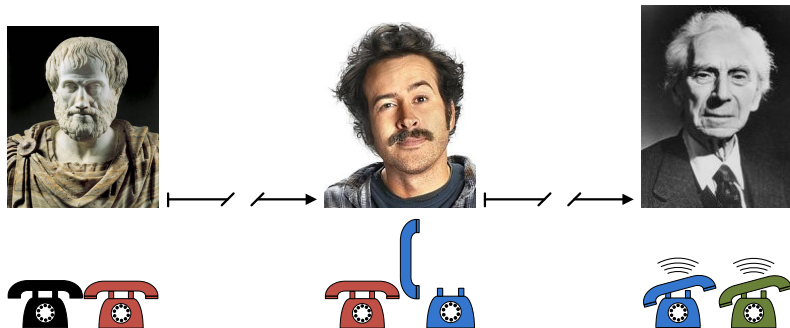


# Choice on both

## The Even Newer Rules:

- 1 A philosopher (sequential code) may either:
  - do some work
  - wait for one of several calls where he may either speak or listen (choice over reads and/or writes)

# Visible with choice on both



2008-10-19

## How to Make Your Process Invisible

└ Introduction

└ Visible with choice on both

Visible with choice on both



With choice on both ends, we are stuck. Earl does not want to offer to contact Aristotle until Bertrand is ready, but does not want to contact Bertrand until Aristotle is ready.

# Classic Disjunctive Choice (a la CSP)

Wait for: single event *a* **or** single event *b* (**or** single event *c* **or** ...)

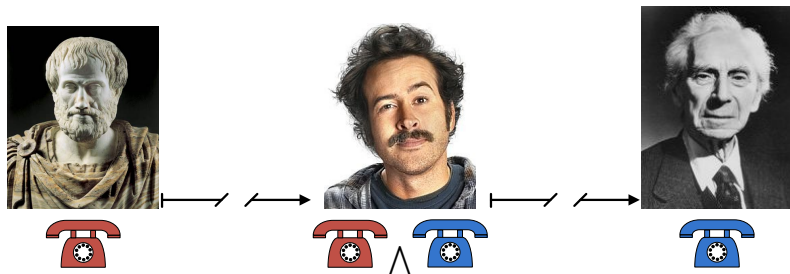
Earl can only wait for: red? **or** blue!

# New! Conjunctive Choice

Wait for: events  $a$  **and**  $b$  (**or** single event  $c$  or ...)

Earl can wait for: red? **and** blue! (but will always engage in both, or neither)

# Wait for Both Phones



## How to Make Your Process Invisible

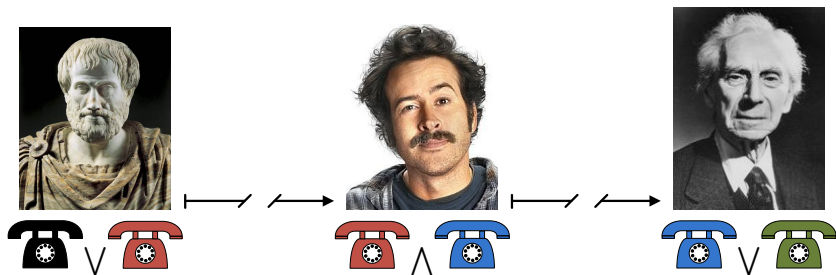
└ Introduction

└ Wait for Both Phones

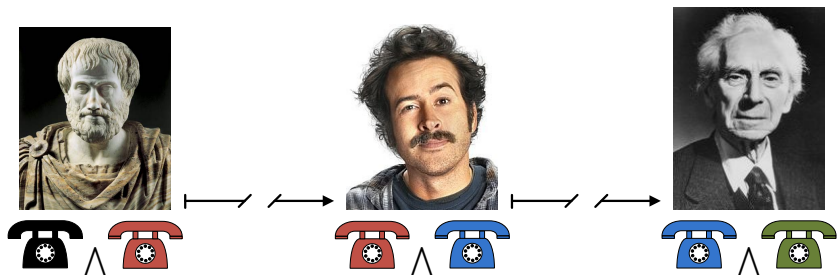


Earl now waits for both phones to be ready. He will wait until both are ready and then will engage in both conversations; he will never engage in just one. This works even if Aristotle and Bertrand are both offering other choices (next slide) or if they are similarly waiting for multiple options to be ready (slide after next).

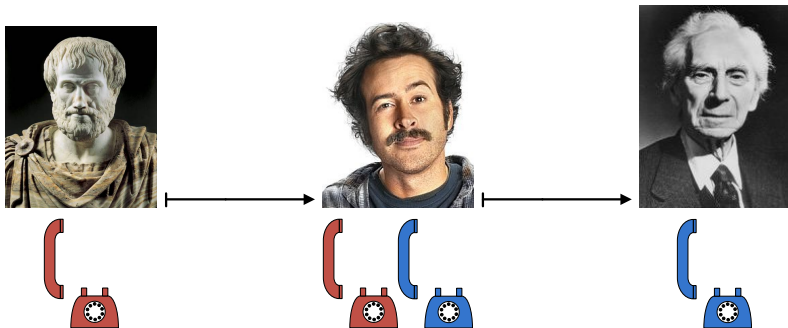
# Wait for Both Phones



# Wait for Both Phones



# Invisible with choice on both



2008-10-19

## How to Make Your Process Invisible

- └ Introduction

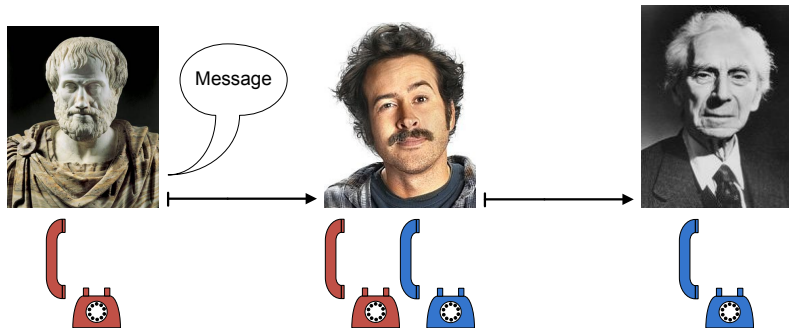
- └ Invisible with choice on both

Invisible with choice on both

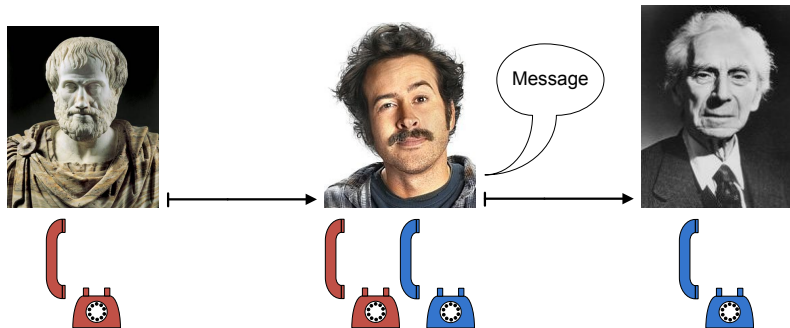


The final solution is that Earl must wait for both phones, and then perform an extended read from Aristotle, where the message is passed to an extended write to Bertrand. In programming terms, Earl must have internal concurrency and an internal channel.

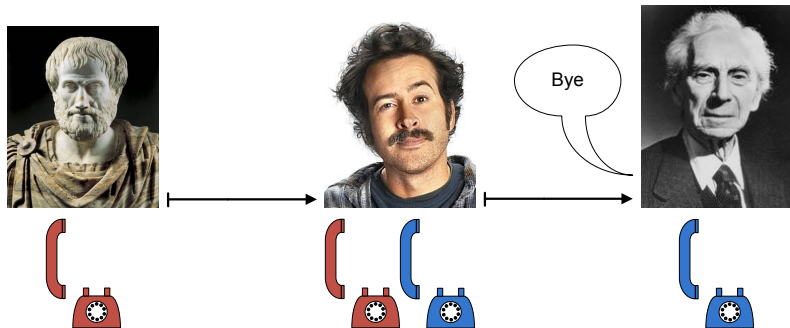
# Invisible with choice on both



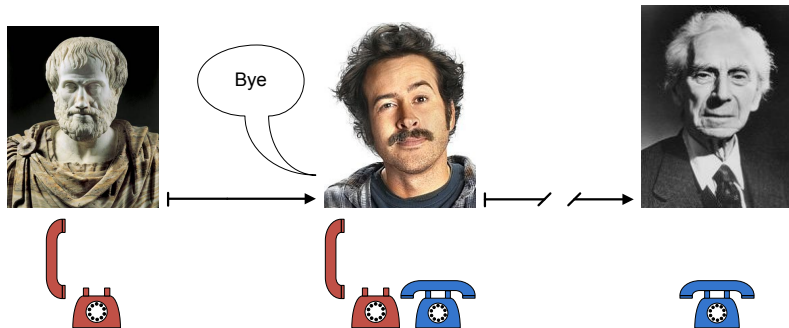
# Invisible with choice on both



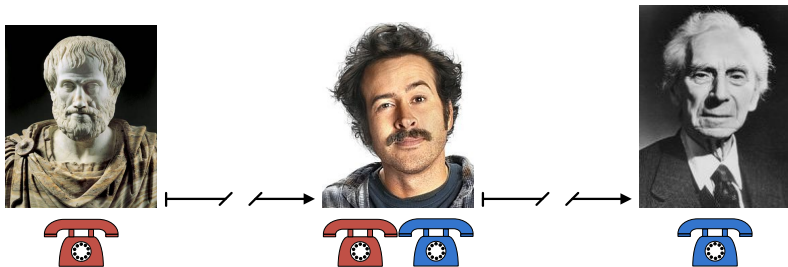
# Invisible with choice on both



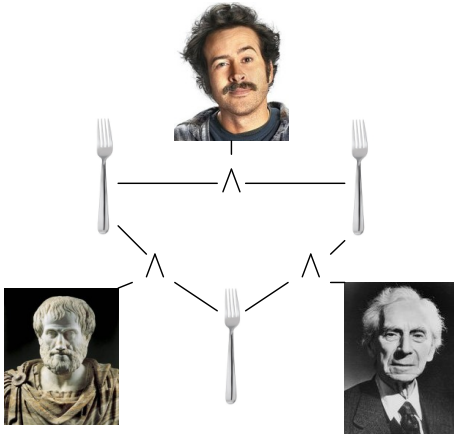
# Invisible with choice on both



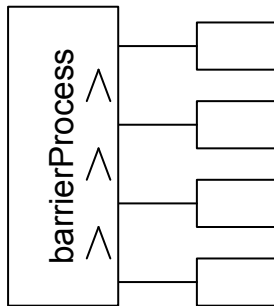
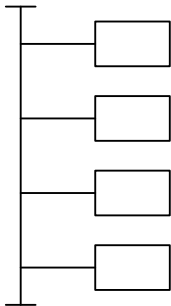
# Invisible with choice on both

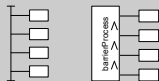


# Dining Philosophers



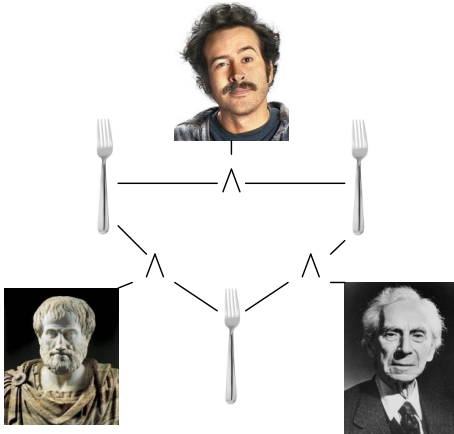
# Barriers and Conjunction



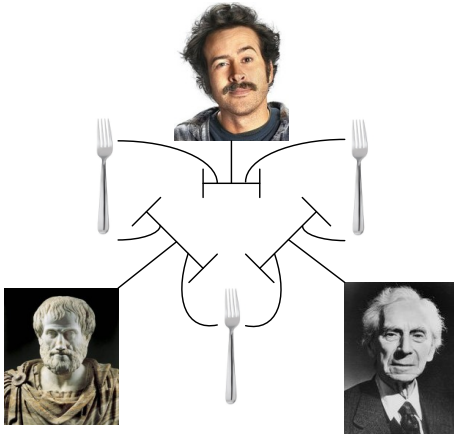


There is a duality between barriers (multiway synchronisations) and conjunction. A barrier can be transformed into an active process that waits for a conjunction of two-party synchronisations (one with each member of the barrier). So we can use this to map our conjunctive dining philosophers example from the previous slide into a solution that uses standard disjunctive choice (a pure CSP solution).

# Dining Philosophers



# Dining Philosophers



# The End

We have a working implementation in CHP (using an oracle-like system) and a draft paper