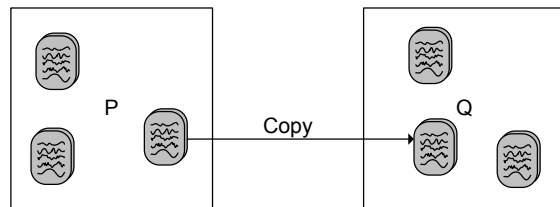


# Implicitly Mobile Data

A way to lighten the load in occam-pi

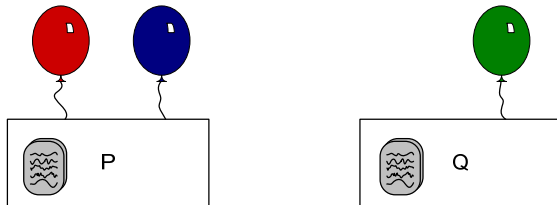
## 1. Original Model

Processes sharing data (and writing to it concurrently) are a source of programming bugs, because multiple conflicting concurrent writes can leave data in an incoherent state. The occam programming language solved this in the past by prohibiting shared data, leaving encapsulated processes with only private data. Private data (the stone tablets) can only be sent to other processes (from P to Q) by copying, but this is very slow.

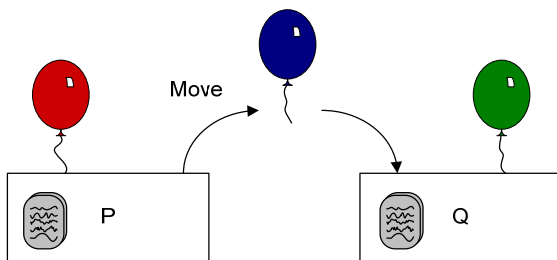


## 2. Explicit Mobility

To address this inefficiency without the hazards of shared data, occam-pi (the successor to occam) introduced explicit mobility. Data can be explicitly tagged by the programmer as mobile (the balloons):



When mobile data is sent between processes it is transferred, which is very fast. This means that the sending process P loses the reference to the data (the blue balloon's string), and cannot access the data again afterwards.



To support explicit mobility, the compiler performs *usage analysis* on the program and generates an error if a process that no longer holds a reference to data tries to read it, for example in this pseudo-code for P:

```

Declare mobile data blue
Write a value to blue
Send blue from P to Q ← Reference lost
Read from blue ← Compiler generates error here
    
```

This new movement behaviour requires the programmer to write their programs differently. To avoid this, and the burden of explicit tagging, we introduce implicit mobility.

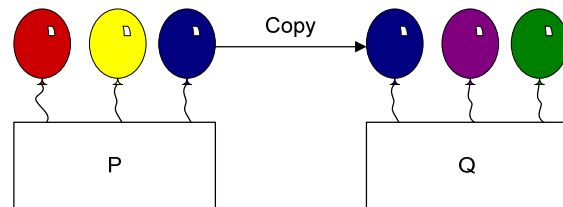
## 3. Implicit Mobility

Implicit mobility allows programmers to write as they did for the original occam, dealing only with copying behaviour. But behind the scenes, all data is actually mobile.

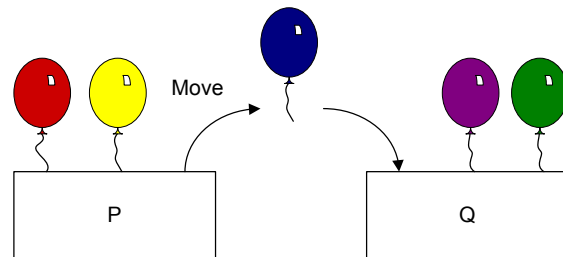
The compiler uses its analysis to make the decision to either move or copy data instead of issuing an error. If the data is not read from again by the same process, it can safely be moved, otherwise it must be copied. So we have:

```

Declare blue
Write a value to blue
Send blue from P to Q ← Copies due to following line
Read from blue
    
```



If the final line of the previous code was removed, the compiler would instead decide to move the value:



**Implicit mobility makes programs almost as efficient as using explicit mobility. It retains the simple copy semantics of occam and does not introduce any new features to the language itself.**

Neil Christopher Charles Brown  
 nccb2@kent.ac.uk  
<http://www.cs.kent.ac.uk/people/rpg/nccb2>