An occam Model of XCHANs

Peter Welch (phw@kent.ac.uk)

CPA 2013 Fringe, Napier University, 26 August, 2013
An XCHAN is a finitely (possibly zero) buffered channel that is **asynchronous** in the sense that it never blocks.

If a writer writes to an XCHAN that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write **fails**. The writer gets the success status of each write.

An XCHAN also signals on a **feedback** channel when a write will be successful …
An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** also signals on a feedback channel when a write is successful …

---

An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** also signals on a *feedback* channel when a write will be successful …

We model an **XCHAN** with an *occam*-π process and channels.
PROTOCOL XCHAN
CASE
   ready
:

PROC x.write (VAL DATA d, BOOL status,
   CHAN XCHAN out.x?,
   CHAN DATA out!)

PRI ALT
   out.x ? ready
   SEQ
      out ! d
      status := TRUE
   SKIP
      status := FALSE
:

Non-blocking write to XCHAN
(with success status result)
or

PROC x.write.sync (VAL DATA d, 
CHAN XCHAN out.x?, 
CHAN DATA out!)

SEQ

out.x ? ready
out ! d

::

Synchronous write to XCHAN (blocks until taken)
PROTOCOL XCHAN

CASE
  ready
:

ALT
  out.x ? ready
  out ! d
  ... other guarded processes

Response to XCHAN signal (will not block)
PROC xchan.1 (CHAN DATA in?, out!,
CHAN XCHAN in.x!)

WHILE TRUE
  DATA d:
  SEQ
    in.x ! ready
    in ? d
    out ! d

1-buffered XCHAN
PROTOCOL XCHAN

CASE
  ready :

n-buffered XCHAN

implementation
PROTOCOL XCHAN
CASE ready:

n-buffered XCHAN

implementation

Standard blocking buffer
PROTOCOL XCHAN

CASE

  ready

:

PROC xchan.0 (CHAN DATA in?, out!,
  CHAN XCHAN in.x!)

:

0-buffered XCHAN
PROTOCOL XCHAN
CASE
ready:
::

When and only when a reader commits, this code executes.

Finally, the message is written.

0-buffered XCHAN

implementation
PROC xchan.0 (CHAN DATA in?, out!, CHAN XCHAN in.x!)

WHILE TRUE
  DATA d:
  out !! -- fish for reader
  SEQ
  in.x ! ready
  in ? d
  !! d

When and only when a reader commits, this code executes.

Finally, the message is written.

0-buffered XCHAN
PROC xchan.0 (CHAN DATA in?, out!,
CHAN XCHAN in.x!)

WHILE TRUE
SEQ
  in.x ! fish   -- fish for writer
  DATA d:
  out !!       -- fish for reader
SEQ
  in.x ! ready
  in ? d
  !! d

When and only when a reader commits, this code executes.

Finally, the message is written.

0-buffered XCHAN (better)
Non-blocking write to XCHAN (with success status result)
PROC x.write (VAL DATA d,  
    CHAN XCHAN out.x?,  
    CHAN DATA out!)  
INITIAL BOOL writing IS TRUE:  
WHILE writing  
    ALT  
        out.x ? fish  
        SKIP  
        out.x ? ready  
        SEQ  
            out ! d  
            writing := FALSE  
:  

Synchronous write to XCHAN (blocks until taken)
PROTOCOL XCHAN

CASE

fish
ready :

ALT

out.x ? fish
SKIP
out.x ? ready
out ! d
... other guarded processes

Response to XCHAN signal (will not block)
An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** also signals on a *feedback* channel when a write will be successful …

**XCHANs**

Any questions?