7. Concurrent Program Development

Jean-Raymond Abrial

2009

Purpose of Lecture

- To present a methodology for developing concurrent programs

- To clearly define the kind of concurrent program we are interested in

- To present an example of such a development

- To show the difficulty of defining a clear specification

- To cover the development of the example

Outline

- Comparing distributed and concurrent computations

- Example: 4-slot fully asynchronous communication mechanism

- Non-concurrent and concurrent behaviors: atomicity

- Studying interleaving

- Specifying: traces

- Purpose of refinement

- Formal development

Various Development Examples

- Distributed programs
  - simple file transfer
  - bounded retransmission protocol
  - leader election on a ring
  - process synchronization on a tree
  - Mobile routing algorithm
  - leader election on a connected graph
- Sequential programs
  - many examples
- Concurrent programs
  - today’s example

Characterizing Distributed Programs

- the same sequential program executed on different computers

- they all together cooperate to achieve a common goal

- no centralized control

- they communicate in a well defined way

- typical examples: the leader election distributed program

Characterizing Concurrent Programs

- different sequential program executed on the same computer

- they compete to make some individual usage of a shared resource

- no centralized protection mechanism around the shared resource

- competing programs can freely interrupt each other

- interruption can occur around well defined atomic actions

- atomic actions are determined by the hardware of the computer

H.R. Simpson *Four-slot Fully Asynchronous Communication Mechanism* Computer and Digital Techniques. IEE Proceedings. Vol 137 (1) (Jan 1990)

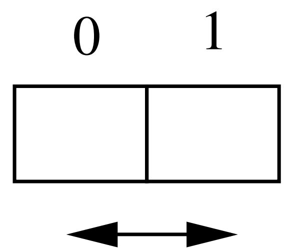
N. Henderson and S.E. Paynter *The Formal Classification and Verification of Simpson's 4-slot Asynchronous Communication Mechanism* Proceedings of FM'02 LNCS Springer (2002)

J. Rushby *Model-Checking Simpson's Four-Slot Fully Asynchronous Communication Mechanism* SRI International (2002)

N. Henderson *Proving the Correctness of Simpson's 4-slot ACM Using an Assertional Rely-Guarantee Proof Method* Proceedings of FM'03 LNCS Springer (2003)

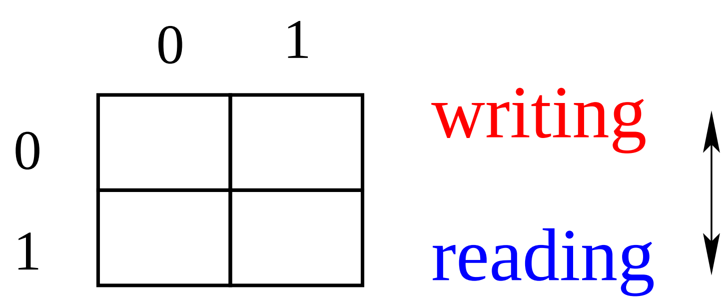
Simpson's 4-slot Algorithm

- A writer writes data items on a pair of slots (0 and 1) alternatively



- A reader concurrently tries to read the last written data item

- To avoid writing and reading the same slot, there is a second pair



Simpson's 4-slot Algorithm Constraints

- The two pairs of slots are not protected by any centralized device

- Hence, no critical section between the reader and the writer

- The communication is purely asynchronous

Shared Data Structure

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$

- D is a generic set

- Variable $data$ defines the 2 pairs of 2 slots. Here is $data(1)(0)$:

	0	1
0		
1	**X**	

Shared Data Structure (cont'd)

$$reading \in \{0, 1\}$$

$$latest \in \{0, 1\}$$

$$slot \in \{0, 1\} \longrightarrow \{0, 1\}$$

- Variables $reading$ denotes the pair used by the reader.

- Variables $latest$ denotes the last pair used by the writer

- Variable $slot$ indicates the slot in which the writer or the reader are currently writing or reading.

Pseudo-code: the Writer

Pseudo-code: the Writer

global variables:

Pseudo-code: the Writer

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$

global variables:

Pseudo-code: the Writer

global variables:

$$
\begin{aligned}
&\boldsymbol{data} \in \{0,1\} \rightarrow (\{0,1\} \rightarrow \boldsymbol{D}) \\
&\boldsymbol{slot} \in \{0,1\} \rightarrow \{0,1\}
\end{aligned}
$$

global variables:

$$\begin{array}{l} \boldsymbol{data} \in \{0,1\} \rightarrow (\{0,1\} \rightarrow \boldsymbol{D}) \\ \boldsymbol{slot} \in \{0,1\} \rightarrow \{0,1\} \\ \boldsymbol{reading} \in \{0,1\} \end{array}$$

Pseudo-code: the Writer

global variables:

$$\begin{aligned} &data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D) \\ &slot \in \{0,1\} \rightarrow \{0,1\} \\ &reading \in \{0,1\} \\ &latest \in \{0,1\} \end{aligned}$$

global variables:

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$
$$slot \in \{0,1\} \rightarrow \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

Pseudo-code: the Writer

global variables:

$$data \in \{0,1\} \to (\{0,1\} \to D)$$
$$slot \in \{0,1\} \to \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

local variable: $pair\_w$
local variable: $indx\_w$

Pseudo-code: the Writer

global variables:

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$
$$slot \in \{0,1\} \rightarrow \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

$pair\_w := 1 - reading;$ /* chosing a pair different from reading */

local variable: $pair\_w$

local variable: $indx\_w$

Pseudo-code: the Writer

global variables:

$$data \in \{0,1\} \to (\{0,1\} \to D)$$
$$slot \in \{0,1\} \to \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)
$pair\_w := 1 - reading;$ /* chosing a pair different from reading */
$indx\_w := 1 - slot(pair\_w);$ /* chosing a different slot */

local variable: $pair\_w$
local variable: $indx\_w$

Pseudo-code: the Writer

global variables:

$$data \in \{0,1\} \to (\{0,1\} \to D)$$
$$slot \in \{0,1\} \to \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

$pair\_w := 1 - reading;$ /* chosing a pair different from reading */

$indx\_w := 1 - slot(pair\_w);$ /* chosing a different slot */

$data(pair\_w)(indx\_w) := d;$ /* pair_w */

local variable: $pair\_w$

local variable: $indx\_w$

Pseudo-code: the Writer

global variables:

$$data \in \{0,1\} \to (\{0,1\} \to D)$$
$$slot \in \{0,1\} \to \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

$pair\_w := 1 - reading;$ /* chosing a pair different from reading */

$indx\_w := 1 - slot(pair\_w);$ /* chosing a different slot */

$data(pair\_w)(indx\_w) := d;$ /* pair_w */

$slot(pair\_w) := indx\_w;$ /* storing the last written slot */

local variable: $pair\_w$

local variable: $indx\_w$

Pseudo-code: the Writer

global variables:

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$
$$slot \in \{0,1\} \rightarrow \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

$pair\_w := 1 - reading;$ /* chosing a pair different from reading */

$indx\_w := 1 - slot(pair\_w);$ /* chosing a different slot */

$data(pair\_w)(indx\_w) := d;$ /* pair_w */

$slot(pair\_w) := indx\_w;$ /* storing the last written slot */

$latest := pair\_w$ /* storing the last written pair */

local variable: $pair\_w$

local variable: $indx\_w$

Pseudo-code: the Reader

global variables:

$$\begin{aligned}
&data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)\\
&slot \in \{0,1\} \rightarrow \{0,1\}\\
&reading \in \{0,1\}\\
&latest \in \{0,1\}
\end{aligned}$$

global variables:

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$
$$slot \in \{0,1\} \rightarrow \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Reader

global variables:

$$\begin{aligned} &data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D) \\ &slot \in \{0,1\} \rightarrow \{0,1\} \\ &reading \in \{0,1\} \\ &latest \in \{0,1\} \end{aligned}$$

Reader

local variable: $indx\_r$

Pseudo-code: the Reader

global variables:

$$\begin{array}{l} data \in \{0,1\} \to (\{0,1\} \to D) \\ slot \in \{0,1\} \to \{0,1\} \\ reading \in \{0,1\} \\ latest \in \{0,1\} \end{array}$$

Reader

$reading := latest;$ /* Chosing the last written pair */

local variable: $indx\_r$

Pseudo-code: the Reader

global variables:

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$
$$slot \in \{0,1\} \rightarrow \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Reader

$reading := latest;$ /* Chosing the last written pair */

$indx\_r := slot(reading);$ /* Chosing the last written slot */

local variable: $indx\_r$

Pseudo-code: the Reader

global variables:

$$data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$$
$$slot \in \{0,1\} \rightarrow \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Reader

$reading := latest;$ /* Chosing the last written pair */

$indx\_r := slot(reading);$ /* Chosing the last written slot */

$y := data(reading)(indx\_r)$ /* Reading */

local variable: $indx\_r$

Pseudo-code of Simpson's 4-slot Algorithm

global variables:

$$data \in \{0,1\} \to (\{0,1\} \to D)$$
$$slot \in \{0,1\} \to \{0,1\}$$
$$reading \in \{0,1\}$$
$$latest \in \{0,1\}$$

Writer(x)

$pair\_w := 1 - reading;$
$indx\_w := 1 - slot(pair\_w);$
$data(pair\_w)(indx\_w) := d;$
$slot(pair\_w) := indx\_w;$
$latest := pair\_w$

local variable: $pair\_w$
local variable: $indx\_w$

Reader

$reading := latest;$
$indx\_r := slot(reading);$
$y := data(reading)(indx\_r)$

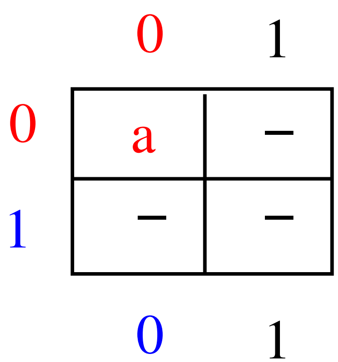
local variable: $indx\_r$

Writer(x)

$pair\_w := 1 - reading;$
$indx\_w := 1 - slot(pair\_w);$
$data(pair\_w)(indx\_w) := d;$
$slot(pair\_w) := indx\_w;$
$latest := pair\_w$

Initially:

$reading = 1$
$slot = \{0 \mapsto 1, 1 \mapsto 1\}$



writing(a)

$pair\_w = \mathbf{0}$
$indx\_w = \mathbf{0}$
$slot(pair\_w) = 0$
$latest = 0$

Writing Alone Animation

Writer(x)
$pair\_w := 1 - reading;$
$indx\_w := 1 - slot(pair\_w);$
$data(pair\_w)(indx\_w) := d;$
$slot(pair\_w) := indx\_w;$
$latest := pair\_w$

Initially:
$reading = 1$
$slot = \{0 \mapsto 1, 1 \mapsto 1\}$

	0	1
0	a	−
1	−	−
	0	1

writing(a)

	0	1
0	a	b
1	−	−
	0	1

writing(b)

$pair\_w = \mathbf{0}$
$indx\_w = \mathbf{0}$
$slot(pair\_w) = 0$
$latest = 0$

$pair\_w = \mathbf{0}$
$indx\_w = \mathbf{1}$
$slot(pair\_w) = 1$
$latest = 0$

Writer(x)

$pair\_w := 1 - reading;$
$indx\_w := 1 - slot(pair\_w);$
$data(pair\_w)(indx\_w) := d;$
$slot(pair\_w) := indx\_w;$
$latest := pair\_w$

Initially:

$reading = 1$
$slot = \{0 \mapsto 1, 1 \mapsto 1\}$

	0	1
0	a	–
1	–	–
	0	1

writing(a)

$pair\_w = 0$
$indx\_w = 0$
$slot(pair\_w) = 0$
$latest = 0$

	0	1
0	a	b
1	–	–
	0	1

writing(b)

$pair\_w = 0$
$indx\_w = 1$
$slot(pair\_w) = 1$
$latest = 0$

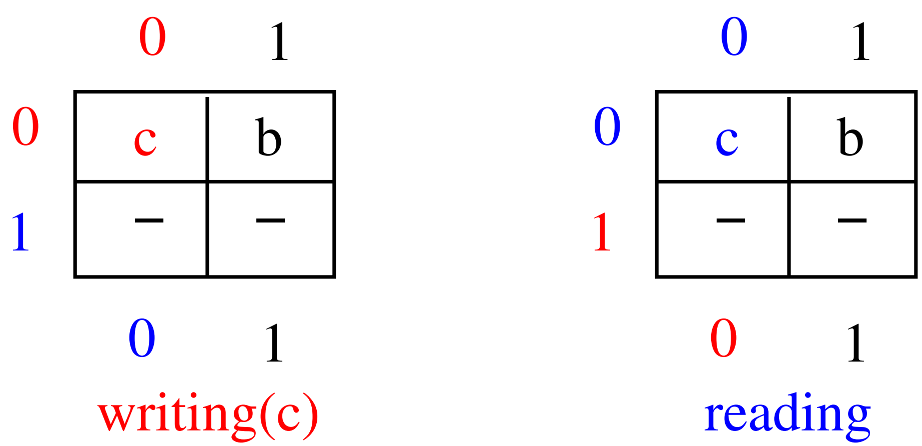
	0	1
0	c	b
1	–	–
	0	1

writing(c)

$pair\_w = 0$
$indx\_w = 0$
$slot(pair\_w) = 0$
$latest = 0$

Reader

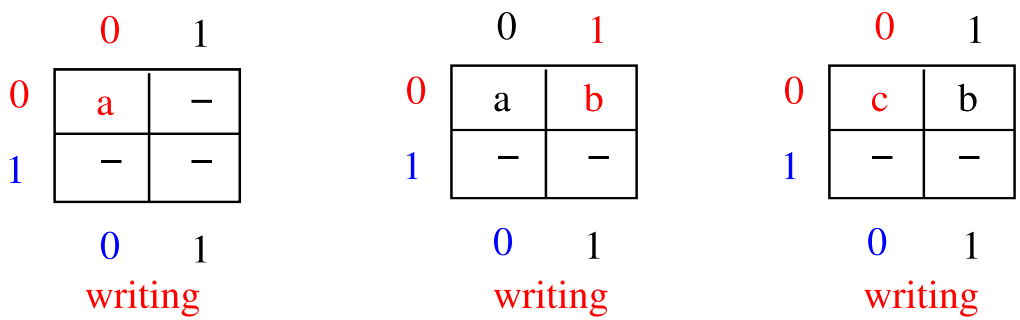
$$reading := latest;$$
$$indx\_r := slot(reading);$$
$$y := data(reading)(indx\_r)$$



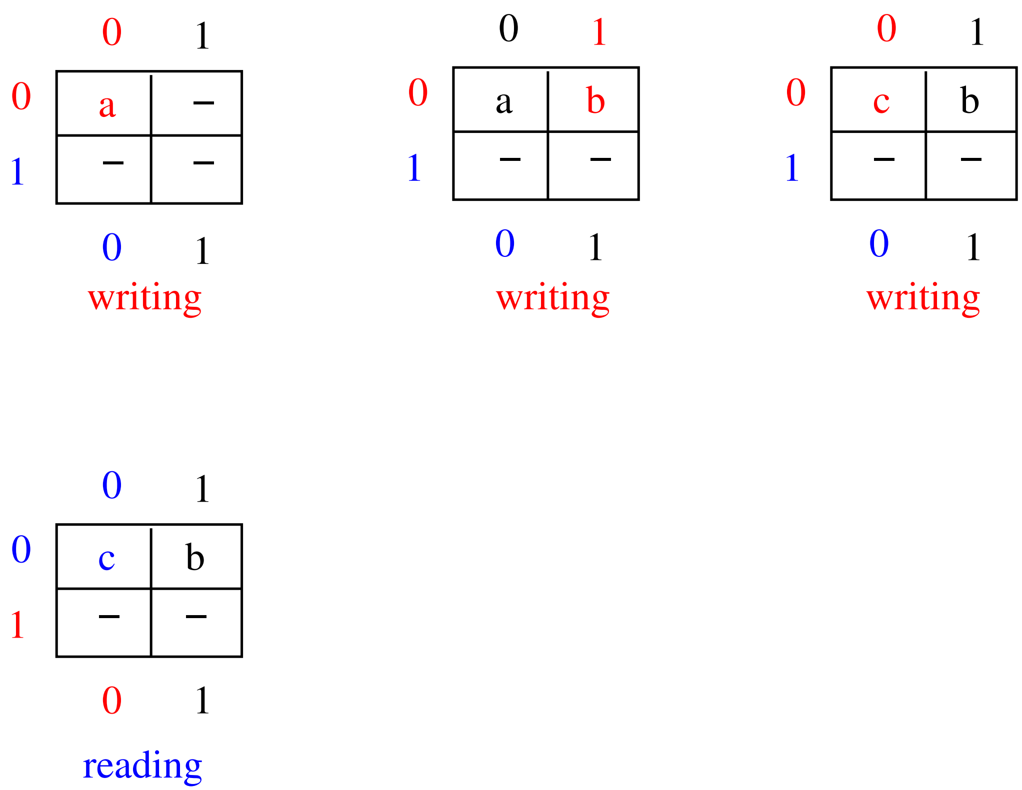
$$slot(0) = 0$$
$$latest = 0$$
$$reading = 1$$

$$reading = 0$$
$$indx\_r = 0$$

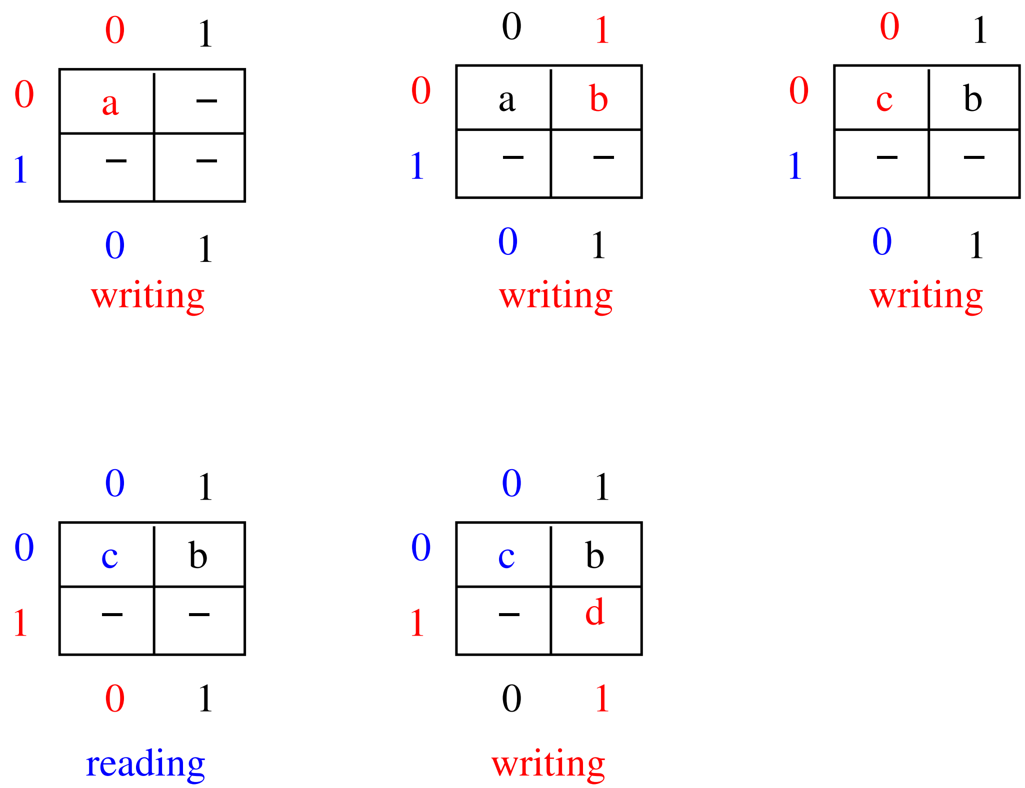
A Simple Writing and Reading Animation



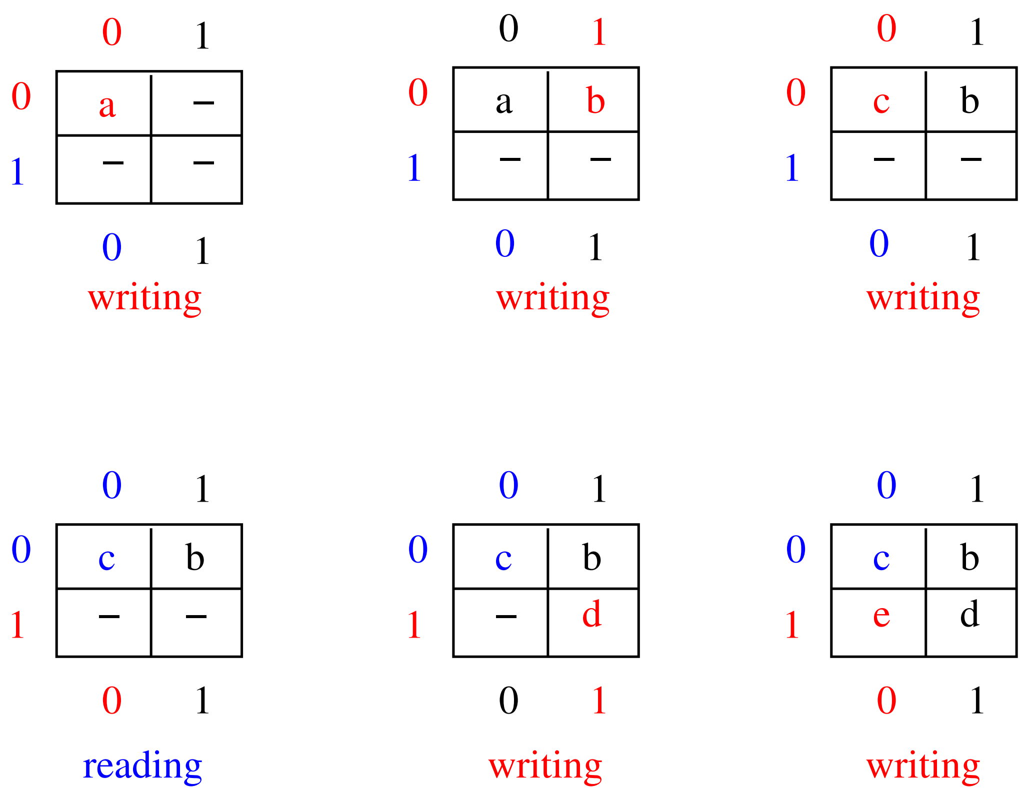
A Simple Writing and Reading Animation



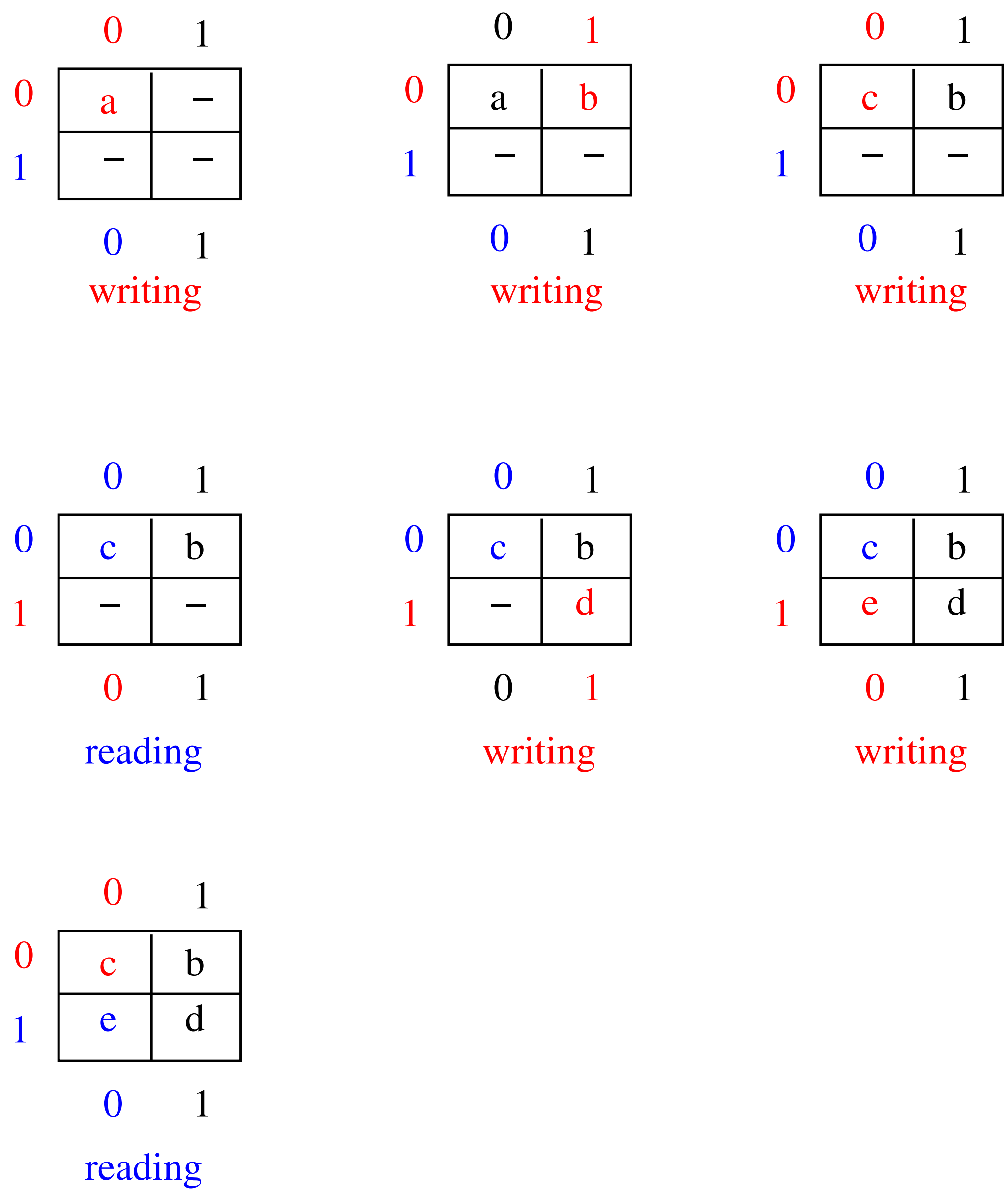
A Simple Writing and Reading Animation



A Simple Writing and Reading Animation



A Simple Writing and Reading Animation



A Simple Writing and Reading Animation

	0	1
0	a	–
1	–	–

writing

	0	1
0	a	b
1	–	–

writing

	0	1
0	c	b
1	–	–

writing

	0	1
0	c	b
1	–	–

reading

	0	1
0	c	b
1	–	d

writing

	0	1
0	c	b
1	e	d

writing

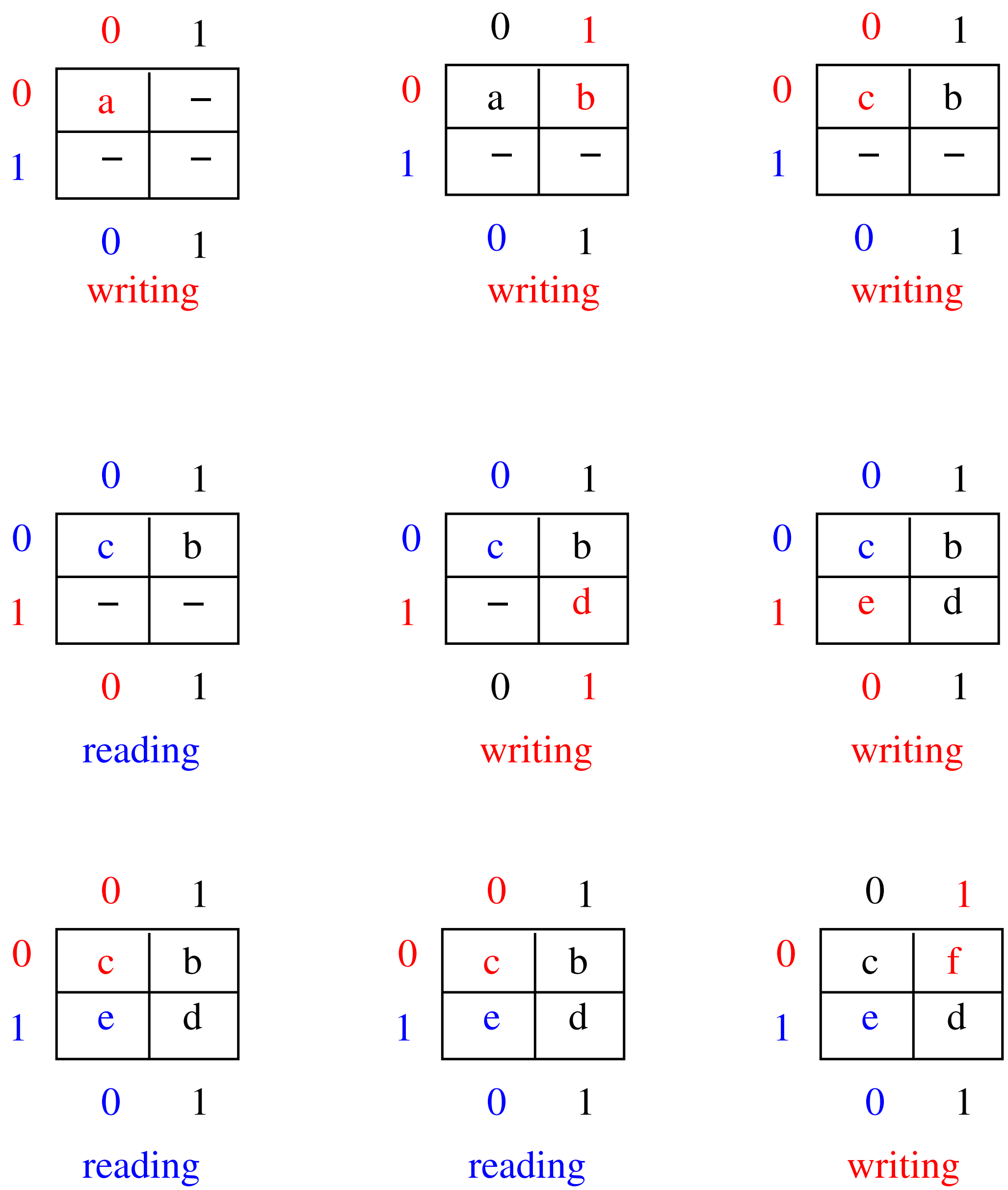
	0	1
0	c	b
1	e	d

reading

	0	1
0	c	b
1	e	d

reading

A Simple Writing and Reading Animation



- Each instruction of the Writer and Reader is an atomic action

- Moreover the writing and reading must be disjoint:

$$pair\_w = reading \;\;\Rightarrow\;\; indx\_w \neq indx\_r$$

Interleaving the Rr and the Wr: Far More Complex

	$\ldots$
Begin Writing	$pair\_w := 1 - reading$
Begin Reading	$reading := latest$
	$indx\_w := 1 - slot(pair\_w)$
	$indx\_r := slot(reading)$
End Reading	$y := data(reading)(indx\_r)$
	$data(pair\_w)(indx\_w) := d$
Begin Reading	$reading := latest$
	$slot(pair\_w) := indx\_w$
End Writing	$latest := pair\_w$
	$indx\_r := slot(reading)$
Begin Writing	$pair\_w := 1 - reading$
End Reading	$y := data(reading)(indx\_r)$
	$indx\_w := 1 - slot(pair\_w)$
	$data(pair\_w)(indx\_w) := d$
Begin Reading	$reading := latest$
	$\ldots$

An Aside: Computing the Number of Interleaving

- Given 2 programs with m and n instructions (including 0)

- Let $U(m, n)$ be the number of interleaving

$$U(m, 0) = 1$$

$$U(0, n) = 1$$

- When m and n are positive:

$$U(m, n) \ = \ U(m-1, n) + U(m, n-1)$$

A Recursive Program: Extremely Inefficient


```
int U(int m, int n)
   {
    if (m==0 || n==0) return 1;
    return U(m-1,n)+U(m,n-1);
   }
```

- Problem with overflow

A Dynamic Programming Attempt

```
int U(int m, int n)
   {
    int M[m+1][n+1],i,j;
    for (i=0; i<=m; ++i) M[i][0]=1;
    for (j=0; j<=n; ++j) M[0][j]=1;
    for (i=1; i<=m; ++i)
       for (j=1; j<=n; ++j)
          M[i][j]=M[i-1][j]+M[i][j-1];
    return M[m][n];
   }
```

- Problem with overflow

A Dynamic Programming approach: Overflow

```
int U(int m, int n)
    {
     int M[m+1][n+1],i,j,a,b;
     for (i=0; i<=m; ++i) M[i][0]=1;
     for (j=0; j<=n; ++j) M[0][j]=1;
     for (i=1; i<=m; ++i)
         for (j=1; j<=n; ++j)
             {
              a=M[i-1][j];
              b=M[i][j-1];
              if (a>INT_MAX-b) return 0;
              M[i][j]=a+b;
             }
     return M[m][n];
    }
```

- Returning value 0 when overflow

- Calculating $U(0,0), U(5,3), U(10,6), U(15,9), \ldots$

```
main(void)
    {
     int i,a,b,r,ok=1;
     for (i=0; ok; ++i)
         {
          a=5*i;
          b=3*i;
          r=U(a,b);
          if (r==0)
             {
              printf("   U(%d,%d)    = OVERFLOW\n",a,b);
              ok=0;
             }
          else printf("   U(%d,%d)    = %d\n",a,b,r);
         }
    }
```

Observing the Result

```
U(0,0)    = 1
U(5,3)    = 56
U(10,6)    = 8008
U(15,9)    = 1307504
U(20,12)    = 225792840
U(25,15)    = OVERFLOW
```

- Overflow with 5 writings together with 5 readings: $\boldsymbol{U(25,15)}$

- Remember: INT_MAX is equal to 2,147,483,647

- Conclusion: Checking all possible interleaving is impossible

Problem: the Specification

- We understand the concurrent behavior

- But we do not know how to specify it

- We need a global synthetic view of our system

- We suppose that we have a partial view of our programs

- This partial view corresponds to their termination

Problem: the Specification (more)

- Reasoning on the termination is done by:
    - storing the history of what is written
    - storing the history of what is read

- Exploiting the relationship between these histories (called traces)

carrier set: D

variables: w, r, wt, rd

inv0_1: $w \in \mathbb{N}_1$

inv0_2: $r \in \mathbb{N}_1$

inv0_3: $wt \in 1 \,..\, w \rightarrow D$

inv0_4: $rd \in 1 \,..\, r \rightarrow D$

- The specification is the relationship between the traces

First Relationship

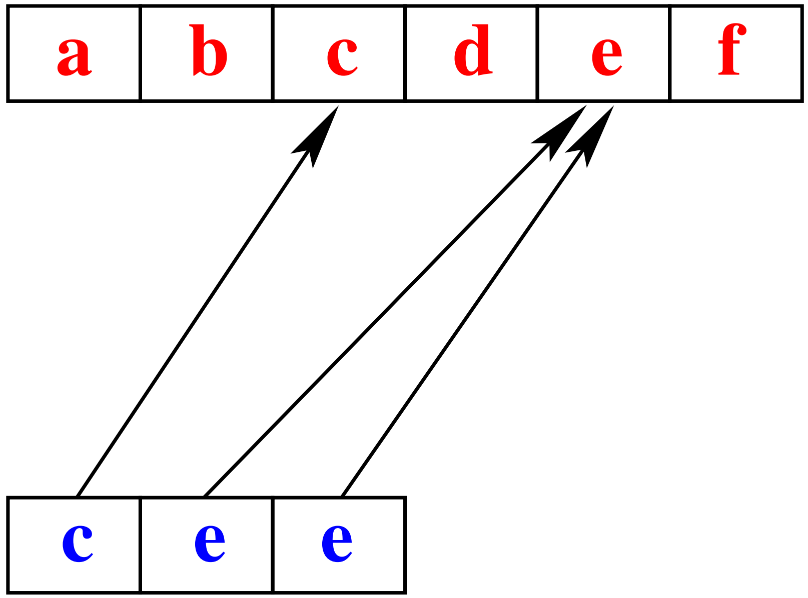
- What is read has been written before

- The reading order follows that of writing

- Some writing might be missing in the reading trace

- Some reading might be repeated in the reading trace

Writing trace



Reading trace

Formalizing the First Relationship

variables: f

inv0_5: $f \in 1 \,..\, r \rightarrow 1 \,..\, w$

inv0_6: $rd = (f \,;\, wt)$

inv0_7: $\forall i, j \cdot \left(\begin{array}{l} i \in 1 \,..\, r \\ j \in i + 1 \,..\, r \\ \Rightarrow \\ f(i) \leq f(j) \end{array} \right)$

- Function f links the reading trace to the writing trace

- Function f is non-decreasing

- We must express that the Reader makes some progress

variables: g

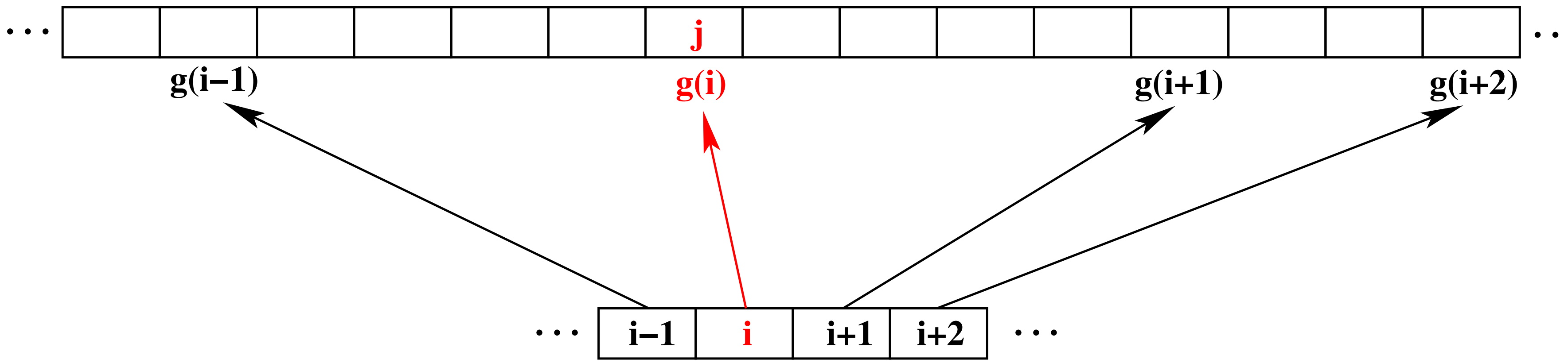
inv0_8: $g \in 1\,..\,r \rightarrow 1\,..\,w$

inv0_9: $\forall\, i, j \cdot \begin{pmatrix} i \in 1\,..\,r \\ j \in i+1\,..\,r \\ \Rightarrow \\ g(i) \leq g(j) \end{pmatrix}$

inv0_10: $\forall\, i \cdot \begin{pmatrix} i \in 1\,..\,r \\ \Rightarrow \\ f(i) \leq g(i) \end{pmatrix}$

- $g(i)$ denotes the place where the writer has previously written when the reader reads

Illustration



- The ith reading just follows the jth writing

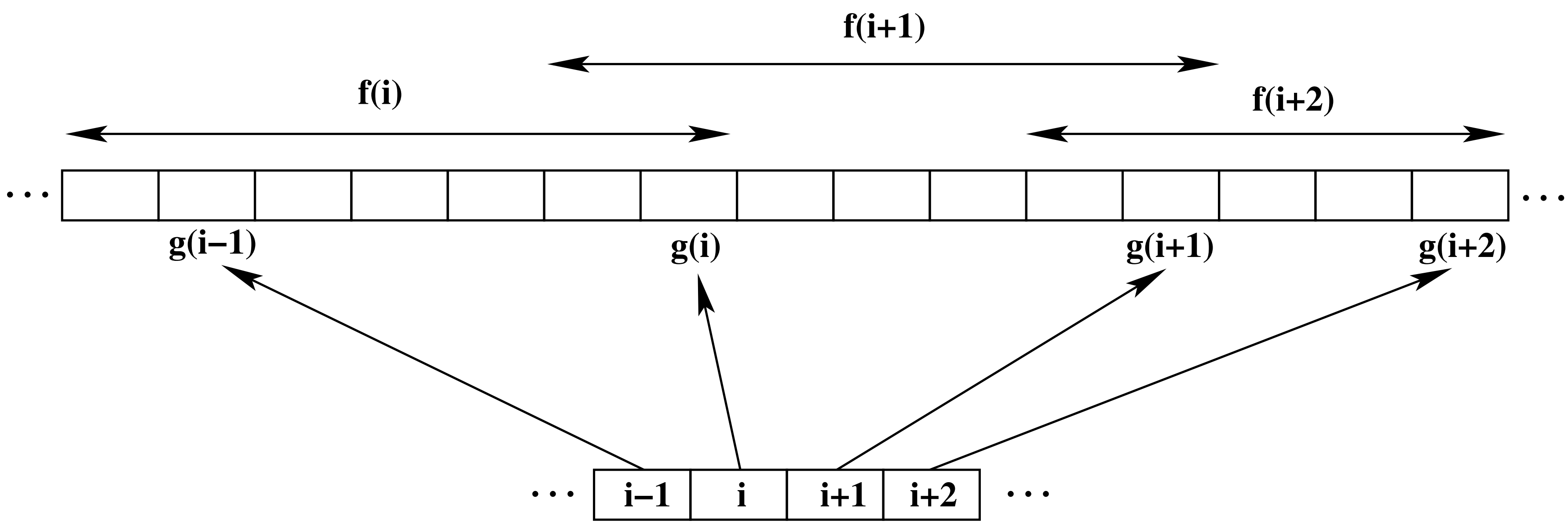
inv0_11: $\forall i \cdot \begin{pmatrix} i \in 1 \,..\, r-1 \\ \Rightarrow \\ g(i) - 1 \leq f(i+1) \end{pmatrix}$

Given an index i in $1 \,..\, r-1$, we have thus the following.

$$f(i+1) \;\in\; g(i) - 1 \,..\, g(i+1)$$

Illustration

This can be illustrated in the following diagram:



- The Reader makes progress

Initial Values

- To simplify, we introduce an initial value (written and then read)

constants: $d0$

prp0_1: $d0 \in D$

init
$w := 1$
$r := 1$
$wt := \{1 \mapsto d0\}$
$rd := \{1 \mapsto d0\}$
$f := \{1 \mapsto 1\}$
$g := \{1 \mapsto 1\}$

```
write
   w := w + 1
   wt(w + 1) :∈ D
```

$$\begin{array}{l} \text{write} \\ \quad w := w + 1 \\ \quad wt(w+1) :\in D \end{array}$$

$$\begin{array}{l} \text{read} \\ \quad \textbf{any } v \textbf{ where} \\ \quad\quad v \in \max(\{g(r)-1, f(r)\}) \,..\, w \\ \quad \textbf{then} \\ \quad\quad r := r + 1 \\ \quad\quad f(r+1) := v \\ \quad\quad g(r+1) := w \\ \quad\quad rd(r+1) := wt(v) \\ \quad \textbf{end} \end{array}$$

Refinement???

- How are we going to refine this initial model?

- What is our goal?

- What is the shape of the final model?

- We have to look at the two concurrent sequential programs

Remember the two Concurrent Programs

Writer(x)

$1: \quad pair\_w := 1 - reading;$

$2: \quad indx\_w := 1 - slot(pair\_w);$

$3: \quad data(pair\_w)(indx\_w) := x;$

$4: \quad slot(pair\_w) := indx\_w;$

$5: \quad latest := pair\_w$

Reader

$1: \quad reading := latest;$

$2: \quad indx\_r := slot(reading);$

$3: \quad y := data(reading)(indx\_r)$

- Introducing address counters

inv0_12: $adr\_w \in 1\,..\,5$

inv0_13: $adr\_r \in 1\,..\,3$

Sketch of Final Refinement: Writer

- Each instruction corresponds to a separate event

Writer_1
when
$adr\_w = 1$
then
$x :\in D$
$pair\_w := 1 - reading$
$adr\_w := 2$
end

Writer_2
when
$adr\_w = 2$
then
$indx\_w := 1 - slot(pair\_w)$
$adr\_w := 3$
end

Writer_3
when
$adr\_w = 3$
then
$data(pair\_w)(index\_w) := x$
$adr\_w := 4$
end

Sketch of Final Refinement: Writer (cont'd)

Writer_4
when
$adr\_w = 4$
then
$slot(pair\_w) := indx\_w$
$adr\_w := 5$
end

Writer_5
when
$adr\_w = 5$
then
$latest := pair\_w$
$adr\_w := 1$
end

Sketch of Final Refinement: Reader

Reader_1
when
$adr\_r = 1$
then
$reading := latest$
$adr\_r := 2$
end

Reader_2
when
$adr\_r = 2$
then
$indx\_r := slot(reading)$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$y := data(reading)(indx\_r)$
$adr\_r := 1$
end

Writer_1
when
$adr\_w = 1$
then
$adr\_w := 2$
end

Writer_2
when
$adr\_w = 2$
then
$adr\_w := 3$
end

Writer_3
when
$adr\_w = 3$
then
$adr\_w := 4$
end

Writer_4
when
$adr\_w = 4$
then
$w := w + 1$
$wt(w + 1) :\in D$
$adr\_w := 5$
end

Writer_5
when
$adr\_w = 5$
then
$adr\_w := 1$
end

Reader_1
when
$adr\_r = 1$
then
$adr\_r := 2$
end

Reader_2
any v **where**
$adr\_r = 2$
$v \in \max(\{g(r) - 1, f(r)\}) \,..\, w$
then
$r := r + 1$
$f(r+1) := v$
$g(r+1) := w$
$rd(r+1) := wt(v)$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$adr\_r := 1$
end

Back to the two Concurrent Programs

- The initial events do not correspond to the final refined events. Why?
- The effective writing ends at Writer address 4 (writing is done)
- The effective reading ends at Reader address 2 (the choice is done)

Writer(x)

```
1 :   pair_w := 1 − reading;
2 :   indx_w := 1 − slot(pair_w);
3 :   data(pair_w)(indx_w) := x;
4 :   slot(pair_w) := indx_w;
5 :   latest := pair_w
```

Reader

```
1 :   reading := latest;
2 :   indx_r := slot(reading);
3 :   y := data(reading)(indx_r)
```

Proofs of Initial Model

Total: 25

Interactive: 3

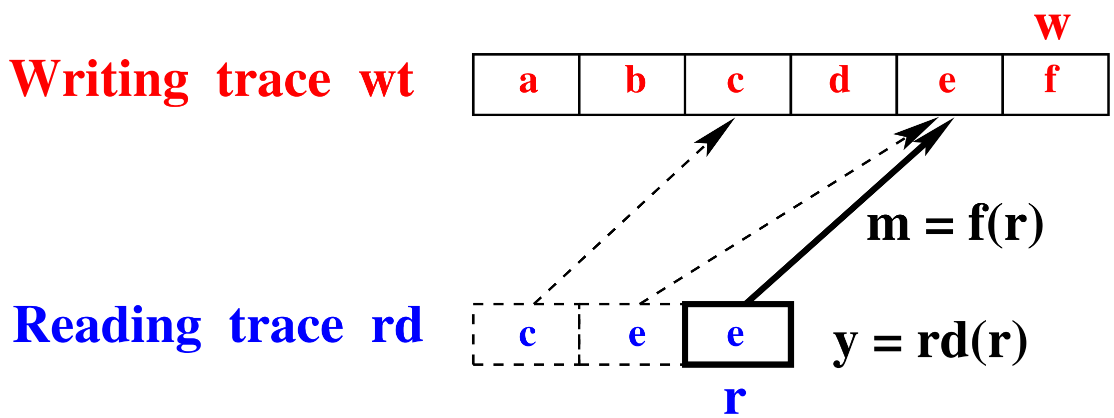
Purpose:

- Splitting the writing and reading actions,
- Removing the reading and writing traces,
- Introducing the data structure of concurrent programs.

Refinements:

1. Splitting the writer and reader. Removing reading trace.
2. Introducing Simpson's algorithm data structure.
3. Removing the writing trace.
4. Splitting the writer into two more parts.

- Removing the Reading Trace



- The reading trace rd is replaced by variable y
- The connecting function f is replaced by variable m
- The connecting function g is replaced by variable u
- Moving the writing trace to the first writing event

variables: $adr\_w, adr\_r, w,$
wtp, u, m, y

inv1_1: $wtp \in \mathbb{N}_1 \nrightarrow D$

inv1_2: $u = g(r)$

inv1_3: $m = f(r)$

inv1_4: $y \in D$

inv1_5: $adr\_w \in \{1, 5\} \Rightarrow wt = wtp$

inv1_6: $adr\_w \in \{2, 3, 4\} \Rightarrow wt = (1 \,..\, w) \lhd wtp$

inv1_7: $adr\_w \in \{2, 3, 4\} \Rightarrow \mathrm{dom}(wtp) = 1 \,..\, w + 1$

inv1_8: $adr\_r = 1 \Rightarrow y = rd(r)$

inv1_9: $adr\_r = 3 \Rightarrow wt(m) = rd(r)$

Writer_1
when
$adr\_w = 1$
then
$wtp(w+1) :\in D$
$adr\_w := 2$
end

Writer_4
when
$adr\_w = 4$
then
$w := w+1$
$adr\_w := 5$
end

(abstract-)Writer_4
when
$adr\_w = 4$
then
$w := w+1$
$wt(w+1) :\in D$
$adr\_w := 5$
end

inv1_5: $adr\_w \in \{1,5\} \;\Rightarrow\; wt = wtp$

inv1_6: $adr\_w \in \{2,3,4\} \;\Rightarrow\; wt = (1\,..\,w) \lhd wtp$

inv1_7: $adr\_w \in \{2,3,4\} \;\Rightarrow\; \mathrm{dom}(wtp) = 1\,..\,w+1$

Reader_1
when
$adr\_r = 1$
then
$adr\_r := 2$
end

Reader_2
when
$adr\_r = 2$
then
$m :\in \max(\{u - 1, m\}) \,..\, w$
$u := w$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$y := wtp(m)$
$adr\_r := 1$
end

inv1_2: $u = g(r)$

inv1_3: $m = f(r)$

inv1_4: $y \in D$

inv1_8: $adr\_r = 1 \Rightarrow y = rd(r)$

inv1_9: $adr\_r = 3 \Rightarrow wt(m) = rd(r)$

(abstract-)Reader_2
any v **where**
$adr\_r = 2$
$v \in \max(\{g(r) - 1, f(r)\}) \,..\, w$
then
$r := r + 1$
$f(r + 1) := v$
$g(r + 1) := w$
$rd(r + 1) := wt(v)$
$adr\_r := 3$
end

Total: 57

Interactive: 1

Introducing Simpson's algorithm data structure

variables: $w, y, wtp, adr\_w, adr\_r, m,$
$reading,$
$pair\_w,$
$latest,$
$indx\_r,$
$indx\_wp,$
$slot$
$idata$

Refinement 2: the State

inv2_1: $reading \in \{0,1\}$

inv2_2: $pair\_w \in \{0,1\}$

inv2_3: $latest \in \{0,1\}$

inv2_4: $indx\_r \in \{0,1\}$

inv2_5: $indx\_wp \in \{0,1\}$

inv2_6: $slot \in \{0,1\} \rightarrow \{0,1\}$

inv2_7: $idata \in \{0,1\} \rightarrow (\{0,1\} \rightarrow w+1)$

inv2_8: $adr\_w \in \{1,5\} \Rightarrow idata \in \{0,1\} \rightarrow (\{0,1\} \rightarrow w)$

- Variable $idata$ contains an index on the writing trace

Writer_1
when
$adr\_w = 1$
then
$pair\_w := 1 - reading$
$indx\_wp := 1 - slot(1 - reading)$
$idata(1 - reading)(1 - slot(1 - reading)) := w + 1$
$wtp(w + 1) :\in D$
$adr\_w := 2$
end

(abstract-)Writer_1
when
$adr\_w = 1$
then
$wtp(w + 1) :\in D$
$adr\_w := 2$
end

Refinement 2: the Events (cont'd)

Writer_2
when
$adr\_w = 2$
then
$adr\_w := 3$
end

Writer_3
when
$adr\_w = 3$
then
$adr\_w := 4$
end

Writer_4
when
$adr\_w = 4$
then
$slot(pair\_w) := indx\_wp$
$w := w + 1$
$adr\_w := 5$
end

Writer_5
when
$adr\_w = 5$
then
$latest := pair\_w$
$adr\_w := 1$
end

Reader_1
when
$adr\_r = 1$
then
$reading := latest$
$adr\_r := 2$
end

Reader_3
when
$adr\_r = 3$
then
$y := wtp(m)$
$adr\_r := 1$
end

Reader_2
when
$adr\_r = 2$
then
$m := idata(reading)(slot(reading))$
$indx\_r := slot(reading)$
$adr\_r := 3$
end

Refinement 2: How to Discover Invariants

inv2_9: $adr\_r = 2 \;\Rightarrow\; idata(reading)(slot(reading)) \in m \mathrel{..} w$

inv2_10: $adr\_r \in \{1,3\} \;\Rightarrow\; idata(latest)(slot(latest)) \in m \mathrel{..} w$

inv2_11:
$$\begin{array}{l} adr\_r = 2 \\ \Rightarrow \\ idata(latest)(slot(latest)) \in idata(reading)(slot(reading)) \mathrel{..} w \end{array}$$

Reader_1
when
$adr\_r = 1$
then
$reading := latest$
$adr\_r := 2$
end

Reader_2
when
$adr\_r = 2$
then
$m := idata(reading)(slot(reading))$
$indx\_r := slot(reading)$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$y := wtp(m)$
$adr\_r := 1$
end

(abstract-)Reader_2
when
$adr\_r = 2$
then
$m :\in \max(\{u - 1, m\}) \mathrel{..} w$
$u := w$
$adr\_r := 3$
end

Refinement 2: How to Discover Invariants (cont'd)

inv2_12: $adr\_r = 2 \;\Rightarrow\; u - 1 \leq idata(reading)(slot(reading))$

inv2_13: $adr\_r \in \{1, 3\} \;\Rightarrow\; u - 1 \leq idata(latest)(slot(latest))$

inv2_14: $w - 1 \leq idata(latest)(slot(latest))$

Reader_1
when
$adr\_r = 1$
then
$reading := latest$
$adr\_r := 2$
end

Reader_2
when
$adr\_r = 2$
then
$m := idata(reading)(slot(reading))$
$indx\_r := slot(reading)$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$y := wtp(m)$
$adr\_r := 1$
end

(abstract-)Reader_2
when
$adr\_r = 2$
then
$m :\in \max(\{u - 1, m\}) \,..\, w$
$u := w$
$adr\_r := 3$
end

Refinement 2: Additional Invariants

inv2_15: $adr\_w \in \{1, 5\} \Rightarrow idata(pair\_w)(indx\_wp) = w$

inv2_16: $adr\_w \in \{2, 3, 4\} \Rightarrow idata(pair\_w)(indx\_wp) = w + 1$

inv2_17: $adr\_w = 1 \Rightarrow pair\_w = latest$

inv2_18: $reading = pair\_w \Rightarrow latest = reading$

inv2_19: $adr\_w \in \{1, 5\} \Rightarrow indx\_wp = slot(pair\_w)$

inv2_20: $adr\_w \in \{2, 3, 4\} \Rightarrow indx\_wp = 1 - slot(pair\_w)$

inv2_21: $adr\_w \in \{2, 3, 4\} \Rightarrow idata(latest)(slot(latest)) = w$

Total: 103

Interactive: 3

- Removing the writing trace

variables: $y, adr\_w, adr\_r,$
$pair\_w, indx\_r,$
$reading, latest, slot,$
$Data, indx\_wp$

inv3_1: $Data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$

inv3_2: $$\forall x, y \cdot \left(\begin{array}{l} x \in \{0,1\} \\ y \in \{0,1\} \\ \Rightarrow \\ wtp(idata(x)(y)) = Data(x)(y) \end{array} \right)$$

inv3_2: $adr\_r = 3 \Rightarrow m = idata(reading)(indx\_r)$

Writer_1

when

$adr\_w = 1$

then

$pair\_w := 1 - reading$

$indx\_wp := 1 - slot(1 - reading)$

$Data(1 - reading)(1 - slot(1 - reading)) :\in D$

$adr\_w := 2$

end

Writer_2
when
$adr\_w = 2$
then
$adr\_w := 3$
end

Writer_3
when
$adr\_w = 3$
then
$adr\_w := 4$
end

Writer_4
when
$adr\_w = 4$
then
$slot(pair\_w) := indx\_wp$
$adr\_w := 5$
end

Writer_5
when
$adr\_w = 5$
then
$latest := pair\_w$
$adr\_w := 1$
end

Reader_1
when
$adr\_r = 1$
then
$reading := latest$
$adr\_r := 2$
end

Reader_2
when
$adr\_r = 2$
then
$indx\_r := slot(reading)$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$y := Data(reading)(indx\_r)$
$adr\_r := 1$
end

Refinement 3: Proofs

Total: 13

Interactive: 0

- The final Touch

variables: $y, adr\_w, adr\_r,$
$pair\_w, indx\_r,$
$reading, latest, slot,$
$data, indx\_w, x$

inv4_1: $data \in \{0,1\} \rightarrow (\{0,1\} \rightarrow D)$

inv4_2: $indx\_w \in \{0,1\}$

inv4_2: $x \in D$

inv4_3: $adr\_w \in \{1, 4, 5\} \;\Rightarrow\; Data = data$

inv4_4:
$$
\begin{array}{l}
adr\_w \in \{2, 3\} \\
\Rightarrow \\
Data = data \mathbin{\lhd\!\!\!-} \{pair\_w \mapsto (data(pair\_w) \mathbin{\lhd\!\!\!-} \{indx\_w \mapsto x\})\}
\end{array}
$$

inv4_2: $adr\_w \in \{3, 4\} \;\Rightarrow\; indx\_w = indx\_wp$

thm4_1:
$$
\begin{array}{l}
adr\_w = 3 \\
adr\_r = 3 \\
pair\_w = reading \\
\Rightarrow \\
indx\_r \neq indx\_w
\end{array}
$$

Sketch of Final Refinement: Writer

Writer_1
when
$adr\_w = 1$
then
$x :\in D$
$pair\_w := 1 - reading$
$adr\_w := 2$
end

Writer_2
when
$adr\_w = 2$
then
$indx\_w := 1 - slot(pair\_w)$
$adr\_w := 3$
end

Writer_3
when
$adr\_w = 3$
then
$data(pair\_w)(index\_w) := x$
$adr\_w := 4$
end

Sketch of Final Refinement: Writer (cont'd)

Writer_4
when
$adr\_w = 4$
then
$slot(pair\_w) := indx\_w$
$adr\_w := 5$
end

Writer_5
when
$adr\_w = 5$
then
$latest := pair\_w$
$adr\_w := 1$
end

Sketch of Final Refinement: Reader

Reader_1
when
$adr\_r = 1$
then
$reading := latest$
$adr\_r := 2$
end

Reader_2
when
$adr\_r = 2$
then
$indx\_r := slot(reading)$
$adr\_r := 3$
end

Reader_3
when
$adr\_r = 3$
then
$y := data(reading)(indx\_r)$
$adr\_r := 1$
end

Total: 50

Interactive: 2

Summary of Proofs (total, interactive)

Initial Model	**25**	**3**
Refinement 1	**57**	**1**
Refinement 2	**103**	**3**
Refinement 3	**13**	**0**
Refinement 4	**50**	**2**
TOTAL	**248**	**9**

Conclusion

- The development of concurrent programs is not easy,
- The number of different interleaving is enormous,
- Defining the specification requires involving traces,
- We use the technique of cutting the initial (one shot) programs,
- The proofs are not difficult,
- But the proper decomposition is not trivial.