

*Simpson's and Harris's algorithm(s) are
linearisable
(or: Five slides that shook the world)*

Richard Bornat
School of EIS, Middlesex University (retired)
joint work with Hasan Amjad (retired)

November 2009

Three outrageous claims and a provocation

In the last six months or so, I think I have discovered

Three outrageous claims and a provocation

In the last six months or so, I think I have discovered

- ▶ An explanation by data and atomicity refinement of Simpson's four-slot algorithm – i.e. **why/how** it works;

Three outrageous claims and a provocation

In the last six months or so, I think I have discovered

- ▶ An explanation by data and atomicity refinement of Simpson's four-slot algorithm – i.e. **why/how** it works;
- ▶ An explanation by (ditto) of Harris's three-slot algorithm – also why/how it works;

Three outrageous claims and a provocation

In the last six months or so, I think I have discovered

- ▶ An explanation by data and atomicity refinement of Simpson's four-slot algorithm – i.e. **why/how** it works;
- ▶ An explanation by (ditto) of Harris's three-slot algorithm – also why/how it works;
- ▶ An understandable, adequate and **sequential** specification of both.

Three outrageous claims and a provocation

In the last six months or so, I think I have discovered

- ▶ An explanation by data and atomicity refinement of Simpson's four-slot algorithm – i.e. **why/how** it works;
- ▶ An explanation by (ditto) of Harris's three-slot algorithm – also why/how it works;
- ▶ An understandable, adequate and **sequential** specification of both.

“You can read a proof any day. Fireworks are a special occasion.”

(M—— P——)

An obvious specification

An obvious specification

It is obvious that ...

An obvious specification

It is obvious that ...

An obvious specification

data $b = \text{null}$ in

$$\left(\begin{array}{l} \dots \\ \text{write}(\text{data } w) \hat{=} \langle b := w \rangle \\ \text{ni} \end{array} \parallel \begin{array}{l} \dots \\ \text{data } \text{read}() \hat{=} \\ \text{data } y \text{ in } \langle y := b \rangle; \text{return } y \text{ ni} \end{array} \right)$$

An obvious specification

data $b = null$ in

$$\left(\begin{array}{l} \dots \\ write(\text{data } w) \hat{=} \langle b := w \rangle \\ ni \end{array} \parallel \begin{array}{l} \dots \\ \text{data } read() \hat{=} \\ \text{data } y \text{ in } \langle y := b \rangle; \text{return } y \text{ ni} \end{array} \right)$$

- ▶ This is **the** single-place buffer algorithm.

An obvious specification

data $b = \text{null}$ in

$$\left(\begin{array}{l} \dots \\ \text{write}(\text{data } w) \hat{=} \langle b := w \rangle \\ \text{ni} \end{array} \parallel \begin{array}{l} \dots \\ \text{data } \text{read}() \hat{=} \\ \text{data } y \text{ in } \langle y := b \rangle; \text{return } y \text{ ni} \end{array} \right)$$

- ▶ This is **the** single-place buffer algorithm.
- ▶ Other SPBAs do what this does (or they're not SPBAs).

An obvious specification

data $b = null$ in

$$\left(\begin{array}{l} \dots \\ write(\text{data } w) \hat{=} \langle b := w \rangle \\ \text{ni} \end{array} \parallel \begin{array}{l} \dots \\ \text{data } read() \hat{=} \\ \text{data } y \text{ in } \langle y := b \rangle; \text{return } y \text{ ni} \end{array} \right)$$

- ▶ This is **the** single-place buffer algorithm.
- ▶ Other SPBAs do what this does (or they're not SPBAs).
- ▶ So this **has** to be the specification of Simpson's (and Harris's) algorithm.

An obvious specification

data $b = null$ in data* $ws = .null, rs = .$ in

$$\left(\begin{array}{l} \dots \\ write(w) \hat{=} \langle b := w; \\ \quad ws := ws.w \rangle \end{array} \parallel \begin{array}{l} \dots \\ data read() \hat{=} \\ data y in \langle y := b; \\ \quad rs := rs.y \rangle; return y ni \end{array} \right)$$

ni ni

- ▶ This is **the** single-place buffer algorithm.
- ▶ Other SPBAs do what this does (or they're not SPBAs).
- ▶ So this **has** to be the specification of Simpson's (and Harris's) algorithm.
- ▶ Add some auxiliaries to aid description of communication.

An obvious specification

data $b = null$ in data* $ws = .null, rs = .$ in

$$\left(\begin{array}{l} \dots \\ write(w) \hat{=} \langle b := w; \\ \quad ws := ws.w \rangle \\ \dots \end{array} \right) \parallel \left(\begin{array}{l} \dots \\ data read() \hat{=} \\ data y in \langle y := b; \\ \quad rs := rs.y \rangle; return y ni \\ \dots \end{array} \right)$$

ni ni

- ▶ This is **the** single-place buffer algorithm.
- ▶ Other SPBAs do what this does (or they're not SPBAs).
- ▶ So this **has** to be the specification of Simpson's (and Harris's) algorithm.
- ▶ Add some auxiliaries to aid description of communication.
- ▶ **Lossy** and **stuttering** but **coherent**: $[rs] \preceq ws \wedge ws_{\Omega} = b$.

An obvious specification

data $b = null$ in data* $ws = .null, rs = .$ in

$$\left(\begin{array}{l} \dots \\ write(w) \hat{=} \langle b := w; \\ \quad ws := ws.w \rangle \end{array} \right) \parallel \left(\begin{array}{l} \dots \\ data read() \hat{=} \\ data y in \langle y := b; \\ \quad rs := rs.y \rangle; return y ni \end{array} \right)$$

ni ni

- ▶ This is **the** single-place buffer algorithm.
- ▶ Other SPBAs do what this does (or they're not SPBAs).
- ▶ So this **has** to be the specification of Simpson's (and Harris's) algorithm.
- ▶ Add some auxiliaries to aid description of communication.
- ▶ **Lossy** and **stuttering** but **coherent**: $[rs] \preceq ws \wedge ws_{\Omega} = b$.
- ▶ **Freshness**: initially $b = B$; eventually $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$.

*Atomicity refinement: aid to explanation; problem
for specification*

*Atomicity refinement: aid to explanation; problem
for specification*

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

Atomicity refinement: aid to explanation; problem for specification

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

We split the atom: $\{P \star \boxed{sP}\} \langle A \rangle \{P' \star \boxed{sP'}\} \langle B \rangle \{Q' \star \boxed{sQ'}\}$.

Atomicity refinement: aid to explanation; problem for specification

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

We split the atom: $\{P \star \boxed{sP}\} \langle A \rangle \{P' \star \boxed{sP'}\} \langle B \rangle \{Q' \star \boxed{sQ'}\}$.

Actions $sP \rightsquigarrow sP'$ and $sP' \rightsquigarrow sQ'$ must not affect environment's stability.

Atomicity refinement: aid to explanation; problem for specification

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

We split the atom: $\{P \star \boxed{sP}\} \langle A \rangle \{P' \star \boxed{sP'}\} \langle B \rangle \{Q' \star \boxed{sQ'}\}$.

Actions $sP \rightsquigarrow sP'$ and $sP' \rightsquigarrow sQ'$ must not affect environment's stability.

$Q \star \boxed{sQ}$ and $Q' \star \boxed{sQ'}$ must have the same sequential force (stabilise to the same assertion).

Atomicity refinement: aid to explanation; problem for specification

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

We split the atom: $\{P \star \boxed{sP}\} \langle A \rangle \{P' \star \boxed{sP'}\} \langle B \rangle \{Q' \star \boxed{sQ'}\}$.

Actions $sP \rightsquigarrow sP'$ and $sP' \rightsquigarrow sQ'$ must not affect environment's stability.

$Q \star \boxed{sQ}$ and $Q' \star \boxed{sQ'}$ must have the same sequential force (stabilise to the same assertion).

We (may) gain understanding of the result, but we (may also) introduce new behaviours.

Atomicity refinement: aid to explanation; problem for specification

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

We split the atom: $\{P \star \boxed{sP}\} \langle A \rangle \{P' \star \boxed{sP'}\} \langle B \rangle \{Q' \star \boxed{sQ'}\}$.

Actions $sP \rightsquigarrow sP'$ and $sP' \rightsquigarrow sQ'$ must not affect environment's stability.

$Q \star \boxed{sQ}$ and $Q' \star \boxed{sQ'}$ must have the same sequential force (stabilise to the same assertion).

We (may) gain understanding of the result, but we (may also) introduce new behaviours.

The specification must constrain those new behaviours.

Atomicity refinement: aid to explanation; problem for specification

We have $\{P \star \boxed{sP}\} \langle A; B \rangle \{Q \star \boxed{sQ}\}$, with action $sP \rightsquigarrow sQ$ against which the environment is stable.

We split the atom: $\{P \star \boxed{sP}\} \langle A \rangle \{P' \star \boxed{sP'}\} \langle B \rangle \{Q' \star \boxed{sQ'}\}$.

Actions $sP \rightsquigarrow sP'$ and $sP' \rightsquigarrow sQ'$ must not affect environment's stability.

$Q \star \boxed{sQ}$ and $Q' \star \boxed{sQ'}$ must have the same sequential force (stabilise to the same assertion).

We (may) gain understanding of the result, but we (may also) introduce new behaviours.

The specification must constrain those new behaviours.

The sequential specification is inadequate unless **there are no new behaviours**.

There are no new behaviours!

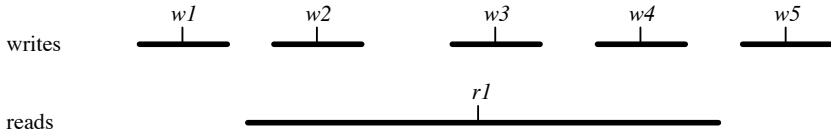
Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.

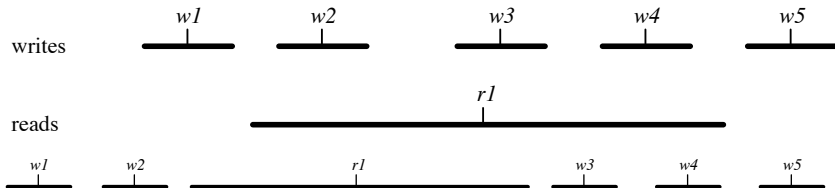
There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.



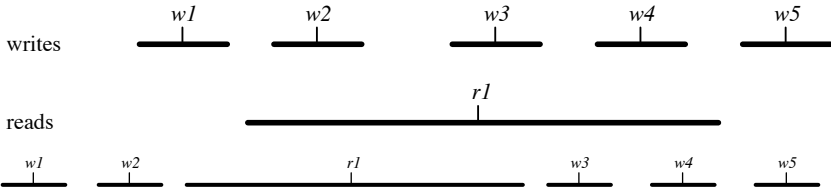
There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.



There are no new behaviours!

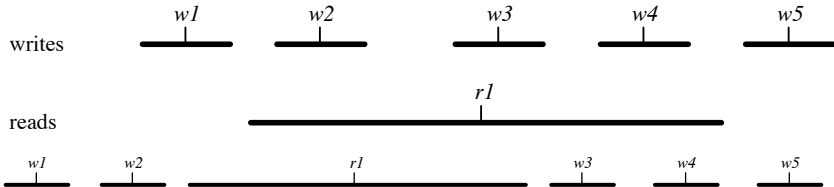
Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.



All our refinements have this characteristic: procedures which contain an instant dividing past from future.

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.

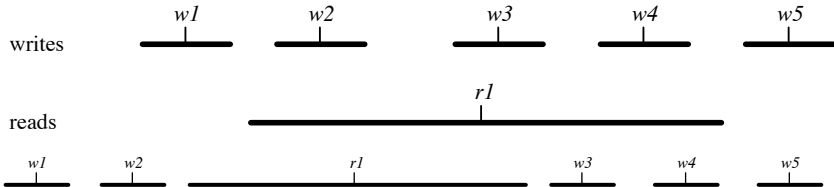


All our refinements have this characteristic: procedures which contain an instant dividing past from future.

It's **obvious** from freshness $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$:

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.

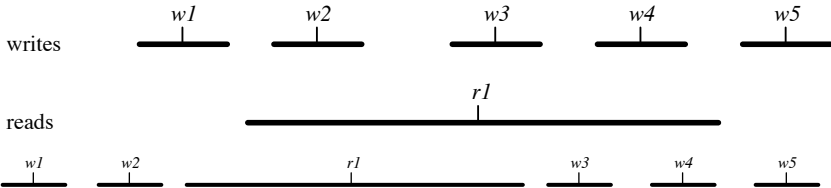


All our refinements have this characteristic: procedures which contain an instant dividing past from future.

It's **obvious** from freshness $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$: The one that wrote B' effectively occurred before the read;

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.

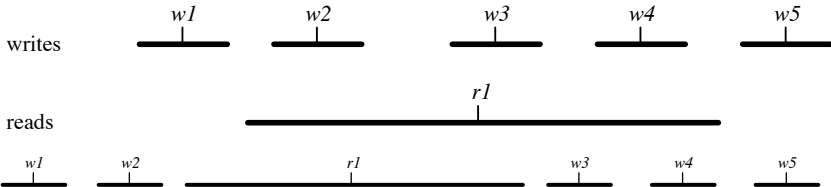


All our refinements have this characteristic: procedures which contain an instant dividing past from future.

It's **obvious** from freshness $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$: The one that wrote B' effectively occurred before the read; those following it occurred effectively after the read.

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.



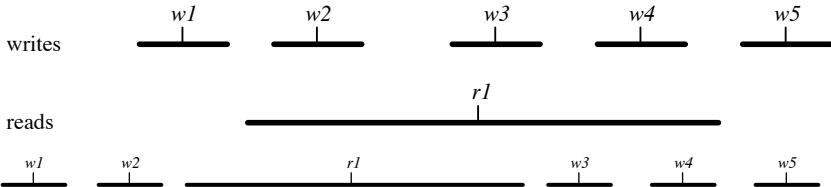
All our refinements have this characteristic: procedures which contain an instant dividing past from future.

It's **obvious** from freshness $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$: The one that wrote B' effectively occurred before the read; those following it occurred effectively after the read.

Every concurrent execution is equivalent to a sequential execution.

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.



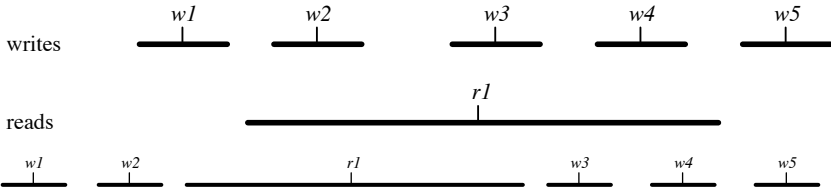
All our refinements have this characteristic: procedures which contain an instant dividing past from future.

It's **obvious** from freshness $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$: The one that wrote B' effectively occurred before the read; those following it occurred effectively after the read.

Every concurrent execution is equivalent to a sequential execution.
That's linearisability.

There are no new behaviours!

Specification a sequential interleaving of $\langle b := \dots \rangle$ s and $\langle y := \dots \rangle$ s.
But we wrapped them in time-taking procedure calls.



All our refinements have this characteristic: procedures which contain an instant dividing past from future.

It's **obvious** from freshness $\exists B'(y = B' \wedge [rs.B.B'] \preceq ws)$: The one that wrote B' effectively occurred before the read; those following it occurred effectively after the read.

Every concurrent execution is equivalent to a sequential execution.
That's linearisability.

I could show you the linearisation points in each refinement, but you would be bored, so I won't.

Discussion?