From event structures theory to weak memory models

Simon Castellan

Imperial College London, UK

January 16th, 2017
QMUL Seminar

Message-passing on my computer

Consider this program `mp`:

$$\begin{array}{c}
\texttt{data} = \texttt{flag} = 0 \\
\begin{array}{l|l}
\texttt{data} := 17; & r \leftarrow \texttt{flag}; \\
\texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\}
\end{array}
\end{array}$$

Message-passing on my computer

Consider this program `mp`:

$$\begin{array}{c} \texttt{data} = \texttt{flag} = 0 \\ \begin{array}{l||l} \texttt{data} := 17; & r \leftarrow \texttt{flag}; \\ \texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array} \end{array}$$

Possible execution traces on my computer:

Message-passing on my computer

Consider this program mp:

$$\begin{array}{c} \texttt{data} = \texttt{flag} = 0 \\ \begin{array}{l|l} \texttt{data} := 17; & r \leftarrow \texttt{flag}; \\ \texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array} \end{array}$$

Possible execution traces on my computer:

- $\mathtt{W}_{\texttt{data}:=17}$

Message-passing on my computer

Consider this program mp:

$$
\begin{array}{l|l}
\multicolumn{2}{c}{\texttt{data} = \texttt{flag} = 0} \\
\texttt{data} := 17; & r \leftarrow \texttt{flag}; \\
\texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\}
\end{array}
$$

Possible execution traces on my computer:

- $\texttt{W}_{\texttt{data}:=17} \cdot \texttt{W}_{\texttt{flag}:=1}$

Message-passing on my computer

Consider this program mp:

$$\begin{array}{c} \texttt{data} = \texttt{flag} = 0 \\ \begin{array}{l||l} \texttt{data} := 17; & r \leftarrow \texttt{flag}; \\ \texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array} \end{array}$$

Possible execution traces on my computer:

- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1}$

Message-passing on my computer

Consider this program mp:

$$\begin{array}{c} \texttt{data} = \texttt{flag} = 0 \\ \begin{array}{l|l} \texttt{data} := 17; & r \leftarrow \texttt{flag}; \\ \texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array} \end{array}$$

Possible execution traces on my computer:

- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17}$

Message-passing on my computer

Consider this program $\mathtt{mp}$:

$$\begin{array}{c} \mathtt{data} = \mathtt{flag} = 0 \\ \begin{array}{l} \mathtt{data} := 17; \\ \mathtt{flag} := 1 \end{array} \;\Big\|\; \begin{array}{l} r \leftarrow \mathtt{flag}; \\ \mathtt{if}(r == 1)\{v \leftarrow \mathtt{data}\} \end{array} \end{array}$$

Possible execution traces on my computer:

- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17}$
- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1}$
- $\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1}$

Message-passing on my phone

$$\begin{array}{c} \texttt{data} = \texttt{flag} = 0 \\ \begin{array}{l||l} \texttt{data} := 17; & r \leftarrow \texttt{flag}; \\ \texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array} \end{array}$$

Possible execution traces on my **phone**:

- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{R}_{\texttt{data}=17}$
- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{flag}:=1}$
- $\mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1}$

Message-passing on my phone

$$\mathtt{data} = \mathtt{flag} = 0$$

$$\begin{array}{l|l} \mathtt{data} := 17; & r \leftarrow \mathtt{flag}; \\ \mathtt{flag} := 1 & \mathtt{if}(r == 1)\{v \leftarrow \mathtt{data}\} \end{array}$$

Possible execution traces on my **phone**:

- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17}$
- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1}$
- $\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1}$
- $\mathtt{W}_{\mathtt{flag}:=1}$

Message-passing on my phone

$$\texttt{data} = \texttt{flag} = 0$$
$$\begin{array}{l} \texttt{data} := 17; \\ \texttt{flag} := 1 \end{array} \;\Bigg\|\; \begin{array}{l} r \leftarrow \texttt{flag}; \\ \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array}$$

Possible execution traces on my **phone**:

- $\texttt{W}_{\texttt{data}:=17} \cdot \texttt{W}_{\texttt{flag}:=1} \cdot \texttt{R}_{\texttt{flag}=1} \cdot \texttt{R}_{\texttt{data}=17}$
- $\texttt{W}_{\texttt{data}:=17} \cdot \texttt{R}_{\texttt{flag}=0} \cdot \texttt{W}_{\texttt{flag}:=1}$
- $\texttt{R}_{\texttt{flag}=0} \cdot \texttt{W}_{\texttt{data}:=17} \cdot \texttt{W}_{\texttt{flag}:=1}$
- $\texttt{W}_{\texttt{flag}:=1} \cdot \texttt{R}_{\texttt{flag}=1}$

Message-passing on my phone

$$\mathtt{data} = \mathtt{flag} = 0$$
$$\begin{array}{l} \mathtt{data} := 17; \\ \mathtt{flag} := 1 \end{array} \;\Big\|\; \begin{array}{l} r \leftarrow \mathtt{flag}; \\ \mathtt{if}(r == 1)\{v \leftarrow \mathtt{data}\} \end{array}$$

Possible execution traces on my **phone**:

- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17}$
- $\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1}$
- $\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1}$
- $\mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=0}$

Message-passing on my phone

$$
\begin{array}{c}
\texttt{data} = \texttt{flag} = 0 \\
\begin{array}{l||l}
\texttt{data} := 17; & r \leftarrow \texttt{flag}; \\
\texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\}
\end{array}
\end{array}
$$

Possible execution traces on my **phone**:

- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{R}_{\texttt{data}=17}$
- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{flag}:=1}$
- $\mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1}$
- $\mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{R}_{\texttt{data}=0} \cdot \mathtt{W}_{\texttt{data}:=17}$

Message-passing on my phone

$$\texttt{data} = \texttt{flag} = 0$$
$$\begin{array}{l|l} \texttt{data} := 17; & r \leftarrow \texttt{flag}; \\ \texttt{flag} := 1 & \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array}$$

Possible execution traces on my **phone**:

- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{R}_{\texttt{data}=17}$
- $\mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{flag}:=1}$
- $\mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{W}_{\texttt{flag}:=1}$
- $\mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{R}_{\texttt{data}=0} \cdot \mathtt{W}_{\texttt{data}:=17}$
- $\mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{R}_{\texttt{data}=17}$
- $\mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{W}_{\texttt{data}:=17} \cdot \mathtt{R}_{\texttt{flag}=1} \cdot \mathtt{R}_{\texttt{data}=17}$
- $\mathtt{R}_{\texttt{flag}=0} \cdot \mathtt{W}_{\texttt{flag}:=1} \cdot \mathtt{W}_{\texttt{data}:=17}$

A different **architecture**, much harder to reason about . . .

Structure behind traces

$$\begin{cases} \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \end{cases}$$

$$\left\{ \ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \right.$$

$$\begin{cases} \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \end{cases}$$

Structure behind traces

$$\begin{cases} \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \end{cases}$$

$$\begin{tikzcd} \mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{W}_{\mathtt{data}:=17} \arrow[d] \\ \mathtt{R}_{\mathtt{flag}=1} \arrow[r] & \mathtt{R}_{\mathtt{data}=17} \end{tikzcd}$$

$$\left\{ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \right.$$

$$\begin{cases} \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \end{cases}$$

Structure behind traces

$$\begin{cases} \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \end{cases}$$

$$\begin{tikzcd} \mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{W}_{\mathtt{data}:=17} \arrow[d] \\ \mathtt{R}_{\mathtt{flag}=1} \arrow[r] & \mathtt{R}_{\mathtt{data}=17} \end{tikzcd}$$

$$\left\{ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \right.$$

$$\begin{tikzcd} \mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{R}_{\mathtt{data}=0} \arrow[d] \\ \mathtt{R}_{\mathtt{flag}=1} \arrow[ur] & \mathtt{W}_{\mathtt{data}:=17} \end{tikzcd}$$

$$\begin{cases} \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \end{cases}$$

Structure behind traces

$$\left\{\begin{array}{l}
\mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \\
\mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{data}=17} \\
\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17}
\end{array}\right.$$

$$\begin{tikzcd}
\mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{W}_{\mathtt{data}:=17} \arrow[d] \\
\mathtt{R}_{\mathtt{flag}=1} \arrow[r] & \mathtt{R}_{\mathtt{data}=17}
\end{tikzcd}$$

$$\left\{\ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=0} \cdot \mathtt{W}_{\mathtt{data}:=17}\right.$$

$$\begin{tikzcd}
\mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{R}_{\mathtt{data}=0} \arrow[d] \\
\mathtt{R}_{\mathtt{flag}=1} \arrow[ur] & \mathtt{W}_{\mathtt{data}:=17}
\end{tikzcd}$$

$$\left\{\begin{array}{l}
\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\
\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\
\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17}
\end{array}\right.$$

$$\begin{tikzcd}
\mathtt{R}_{\mathtt{flag}=0} \arrow[d] & \mathtt{W}_{\mathtt{data}:=17} \\
\mathtt{W}_{\mathtt{flag}:=1} &
\end{tikzcd}$$

Structure behind traces

$$\left\{\begin{array}{l}
\mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \\
\mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{data}=17} \\
\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17}
\end{array}\right.$$

$$\begin{tikzcd}
\mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{W}_{\mathtt{data}:=17} \arrow[d] \\
\mathtt{R}_{\mathtt{flag}=1} \arrow[r] & \mathtt{R}_{\mathtt{data}=17}
\end{tikzcd}$$

$$\left\{\ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=0} \cdot \mathtt{W}_{\mathtt{data}:=17}\right.$$

$$\begin{tikzcd}
\mathtt{W}_{\mathtt{flag}:=1} \arrow[d] & \mathtt{R}_{\mathtt{data}=0} \arrow[d] \\
\mathtt{R}_{\mathtt{flag}=1} \arrow[ur] & \mathtt{W}_{\mathtt{data}:=17}
\end{tikzcd}$$

$$\left\{\begin{array}{l}
\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\
\mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\
\mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17}
\end{array}\right.$$

$$\begin{tikzcd}
\mathtt{R}_{\mathtt{flag}=0} \arrow[d] & \mathtt{W}_{\mathtt{data}:=17} \\
\mathtt{W}_{\mathtt{flag}:=1} &
\end{tikzcd}$$

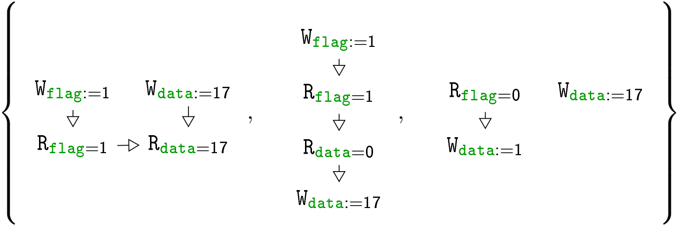
Sets of partial orders and event structures

The **set of partial orders** describes the semantics of `mp`:

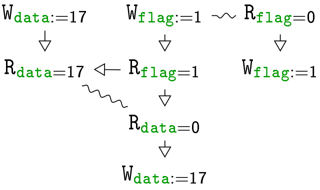
$$
\left\{
\begin{tikzcd}[ampersand replacement=\&]
\mathtt{W}_{\mathtt{flag}:=1} \arrow[d] \& \mathtt{W}_{\mathtt{data}:=17} \arrow[d] \\
\mathtt{R}_{\mathtt{flag}=1} \arrow[r] \& \mathtt{R}_{\mathtt{data}=17}
\end{tikzcd}
,\quad
\begin{tikzcd}[ampersand replacement=\&]
\mathtt{W}_{\mathtt{flag}:=1} \arrow[d] \\
\mathtt{R}_{\mathtt{flag}=1} \arrow[d] \\
\mathtt{R}_{\mathtt{data}=0} \arrow[d] \\
\mathtt{W}_{\mathtt{data}:=17}
\end{tikzcd}
,\quad
\begin{tikzcd}[ampersand replacement=\&]
\mathtt{R}_{\mathtt{flag}=0} \arrow[d] \& \mathtt{W}_{\mathtt{data}:=17} \\
\mathtt{W}_{\mathtt{data}:=1} \&
\end{tikzcd}
\right\}
$$

Sets of partial orders and event structures

The **set of partial orders** describes the semantics of `mp`:

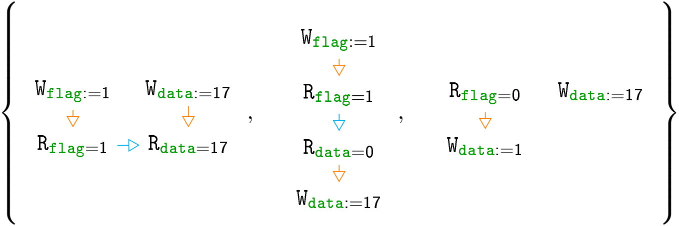


This set of partial orders can be summed by an **event structure**:

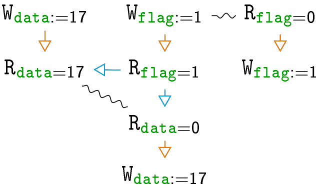


Sets of partial orders and event structures

The **set of partial orders** describes the semantics of `mp`:



This set of partial orders can be summed by an **event structure**:



This talk

1. **A semantics for threads, and a semantics for memory**

2. **Using theory to mix them.**

3. **Applications to theory and practice.**

I. A SEMANTICS FOR THREADS, AND A SEMANTICS FOR MEMORY

Modelling MiniARM

MiniARM: An assembly language with relaxed semantics

Syntax. Idents split in thread-local registers and global `variables`.

$$
\begin{aligned}
e &::= r \mid e + e \mid \dots \\
t &::= \texttt{fence}; t \mid \texttt{x} := e; t \mid r \leftarrow \texttt{x}; t \\
p &::= t \parallel \dots \parallel t
\end{aligned}
$$

Actions. We observe the following actions from the programs:

$$\Sigma ::= \texttt{W}_{\texttt{x}:=k} \mid \texttt{R}_{\texttt{x}=k} \mid \texttt{fence}.$$

Semantics. Described by a labeled transition system on states p, μ:

$$\langle p \mathbin{@} \mu \rangle \xrightarrow{\ell \in \Sigma} \langle p' \mathbin{@} \mu' \rangle. \qquad (\mu, \mu' : \texttt{Var} \to \mathbb{N})$$

It is **relaxed**: operations on independent variables can be reordered.

A few rules

Thread rules: $\langle t \,@\, \mu\rangle \xrightarrow{\ell} \langle t' \,@\, \mu'\rangle$:

$$\frac{}{\langle \mathtt{x} := k; t@\mu\rangle \xrightarrow{\mathtt{W}_{\mathtt{x}:=k}} \langle t@\mu[\mathtt{x} := k]\rangle}$$

A few rules

Thread rules: $\langle t \,@\, \mu \rangle \xrightarrow{\ell} \langle t' \,@\, \mu' \rangle$:

$$\frac{}{\langle \mathtt{x} := k; t @ \mu \rangle \xrightarrow{\mathtt{W}_{\mathtt{x}:=k}} \langle t @ \mu[\mathtt{x} := k] \rangle} \qquad \frac{}{\langle \mathtt{fence}; t @ \mu \rangle \xrightarrow{\mathtt{fence}} \langle t @ \mu \rangle}$$

A few rules

Thread rules: $\langle t \mathbin{@} \mu\rangle \xrightarrow{\ell} \langle t' \mathbin{@} \mu'\rangle$:

$$\frac{}{\langle \mathtt{x} := k; t@\mu\rangle \xrightarrow{\mathtt{W}_{\mathtt{x}:=k}} \langle t@\mu[\mathtt{x} := k]\rangle} \qquad \frac{}{\langle \mathtt{fence}; t@\mu\rangle \xrightarrow{\mathtt{fence}} \langle t@\mu\rangle}$$

$$\frac{\langle t@\mu\rangle \xrightarrow{\ell} \langle t'@\mu'\rangle \qquad \ell \neq \mathtt{fence} \qquad \mathtt{var}(\ell) \neq \mathtt{x}}{\langle \mathtt{x} := k; t@\mu\rangle \xrightarrow{\ell} \langle \mathtt{x} := k; t'@\mu'\rangle}$$

A few rules

Thread rules: $\langle t \mathbin{@} \mu\rangle \xrightarrow{\ell} \langle t' \mathbin{@} \mu'\rangle$:

$$\frac{}{\langle \mathtt{x} := k; t@\mu\rangle \xrightarrow{\mathtt{W}_{\mathtt{x}:=k}} \langle t@\mu[\mathtt{x} := k]\rangle} \qquad \frac{}{\langle \mathtt{fence}; t@\mu\rangle \xrightarrow{\mathtt{fence}} \langle t@\mu\rangle}$$

$$\frac{\langle t@\mu\rangle \xrightarrow{\ell} \langle t'@\mu'\rangle \qquad \ell \neq \mathtt{fence} \qquad \mathtt{var}(\ell) \neq \mathtt{x}}{\langle \mathtt{x} := k; t@\mu\rangle \xrightarrow{\ell} \langle \mathtt{x} := k; t'@\mu'\rangle}$$

And then:

$$\frac{\langle t_i \mathbin{@} \mu\rangle \xrightarrow{\ell} \langle t_i' \mathbin{@} \mu'\rangle}{\langle t_1 \parallel \ldots \parallel t_i \parallel \ldots \parallel t_n \mathbin{@} \mu\rangle \xrightarrow{\ell} \langle t_1 \parallel \ldots \parallel t_i' \parallel \ldots \parallel t_n \mathbin{@} \mu'\rangle}$$

Operational traces and memory states

These rules generates the **operational (partial) traces**:

$$\mathrm{Tr}(p, \mu) = \{\ell_1 \dots \ell_n \mid \langle p \mathbin{@} \mu \rangle \xrightarrow{\ell_1} \dots \xrightarrow{\ell_n} \langle p' \mathbin{@} \mu' \rangle\}.$$

From, there we can compute the final memory states:

$$\begin{aligned} \mathsf{MemStates}(p) &= \{\mu(t) \mid t \in \mathrm{Tr}(p, \mu) \text{ is a maximal trace}\}. \\ \text{where} \quad \mu(t) &= \mathtt{x} \mapsto \text{last value written to } \mathtt{x} \text{ or zero.} \\ &: \mathcal{V} \to \mathbb{N} \end{aligned}$$

Labeled event structures

Definition

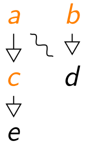
A (Σ-labeled) **event structure** is a tuple $(E, \leq_E, \sharp_E, \ell : E \to \Sigma)$ where $(E, \leq_E)$ is a partial order and $\sharp_E$ is a symmetric relation on E, satisfying *finite causes* and *conflict inheritance*.

$$
\begin{tikzcd}
a \arrow[d] \arrow[r, no head, squiggly] & b \arrow[d] \\
c \arrow[d] & d \\
e &
\end{tikzcd}
$$

Labeled event structures

Definition

A (Σ-labeled) **event structure** is a tuple $(E, \leq_E, \sharp_E, \ell : E \to \Sigma)$ where $(E, \leq_E)$ is a partial order and $\sharp_E$ is a symmetric relation on E, satisfying *finite causes* and *conflict inheritance*.

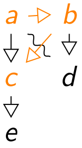


- **Configurations** are downclosed, conflict-free subsets of E. $\mathscr{C}(E)$ is the set of configurations of E.

Labeled event structures

Definition

A (Σ-labeled) **event structure** is a tuple $(E, \leq_E, \sharp_E, \ell : E \to \Sigma)$ where $(E, \leq_E)$ is a partial order and $\sharp_E$ is a symmetric relation on E, satisfying *finite causes* and *conflict inheritance*.



- **Configurations** are downclosed, conflict-free subsets of E. $\mathscr{C}(E)$ is the set of configurations of E.

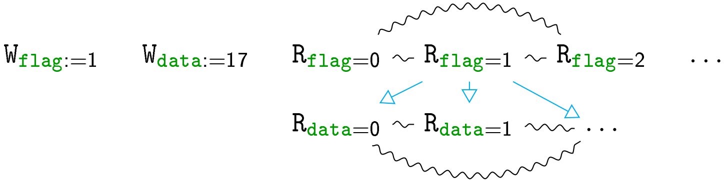
- A **trace** of E is a linearisation of a configuration of E. $\mathrm{Tr}(E)$ is the set of traces of E (can be seen as a subset of Σ^*).

Our goal: a mapping $[\![\cdot]\!]$ from states to event structures s.t.:

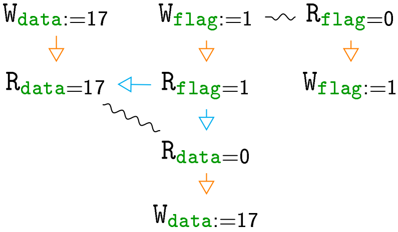
$$\mathrm{Tr}(p, \mu) = \mathrm{Tr}[\![p, \mu]\!].$$

An overview of the semantics

1. **Thread semantics**: context is left open (and unknown)



2. **Final semantics**: context is assumed empty
   Compute interactions with memory:



Thread semantics

Fences. $[\![\texttt{fence}; t]\!] = \texttt{fence} \cdot [\![t]\!]$

$(\leq_{\ell \cdot E} = \leq_E \cup \{(\ell, \ell')\})$

$$\begin{tikzcd} \texttt{fence} \arrow[d] \\ [\![t]\!] \end{tikzcd}$$

Thread semantics

Fences. $[\![\texttt{fence}; t]\!] = \texttt{fence} \cdot [\![t]\!]$

$(\leq_{\ell \cdot E} = \leq_E \cup \{(\ell, \ell')\})$

$$\begin{tikzcd} \texttt{fence} \arrow[d] \\ {[\![t]\!]} \end{tikzcd}$$

Writes. $[\![\texttt{x} := k; t]\!] = \texttt{W}_{\texttt{x}:=k}; [\![t]\!]$

$(\leq_{\ell ; E} = \leq_E \cup \{(\ell, \texttt{fence}), (\ell, \ell') \mid \texttt{var}(\ell) = \texttt{var}(\ell')\}).$

$$\begin{tikzcd} \texttt{W}_{\texttt{x}:=k} \arrow[d, bend right] \arrow[d, bend left] \\ {[\![t]\!]} \end{tikzcd}$$

Thread semantics

Fences. $[\![\mathtt{fence}; t]\!] = \mathtt{fence} \cdot [\![t]\!]$

$(\leq_{\ell \cdot E} = \leq_E \cup \{(\ell, \ell')\})$

$$\begin{tikzcd} \mathtt{fence} \arrow[d] \\ [\![t]\!] \end{tikzcd}$$

Writes. $[\![\mathtt{x} := k; t]\!] = \mathtt{W}_{\mathtt{x}:=k}; [\![t]\!]$

$(\leq_{\ell ; E} = \leq_E \cup \{(\ell, \mathtt{fence}), (\ell, \ell') \mid \mathtt{var}(\ell) = \mathtt{var}(\ell')\}).$

$$\begin{tikzcd} \mathtt{W}_{\mathtt{x}:=k} \arrow[d, bend right] \arrow[d, bend left] \\ [\![t]\!] \end{tikzcd}$$

Reads. $[\![r \leftarrow \mathtt{x}; t]\!] = \sum_{n \in \mathbb{N}} \mathtt{R}_{\mathtt{x}=n}; [\![t[n/r]]\!]$

$$\begin{tikzcd} \mathtt{R}_{\mathtt{x}=0} \arrow[r, squiggly, no head] \arrow[d, bend right] \arrow[d, bend left] & \mathtt{R}_{\mathtt{x}=1} \arrow[r, squiggly, no head] \arrow[d, bend right] \arrow[d, bend left] & \ldots \\ [\![t[0/r]]\!] & [\![t[1/r]]\!] & \ldots \end{tikzcd}$$

Thread semantics

Fences. $[\![\texttt{fence}; t]\!] = \texttt{fence} \cdot [\![t]\!]$

$(\leq_{\ell \cdot E} = \leq_E \cup \{(\ell, \ell')\})$

$$\begin{tikzcd} \texttt{fence} \arrow[d] \\ {[\![t]\!]} \end{tikzcd}$$

Writes. $[\![\texttt{x} := k; t]\!] = \texttt{W}_{\texttt{x}:=k}; [\![t]\!]$

$(\leq_{\ell ; E} = \leq_E \cup \{(\ell, \texttt{fence}), (\ell, \ell') \mid \texttt{var}(\ell) = \texttt{var}(\ell')\})$.

$$\begin{tikzcd} \texttt{W}_{\texttt{x}:=k} \arrow[d, bend right] \arrow[d, bend left] \\ {[\![t]\!]} \end{tikzcd}$$

Reads. $[\![r \leftarrow \texttt{x}; t]\!] = \sum_{n \in \mathbb{N}} \texttt{R}_{\texttt{x}=n}; [\![t[n/r]]\!]$

$$\begin{tikzcd} \texttt{R}_{\texttt{x}=0} \arrow[r, squiggly, no head] \arrow[d, bend right] \arrow[d, bend left] & \texttt{R}_{\texttt{x}=1} \arrow[r, squiggly, no head] \arrow[d, bend right] \arrow[d, bend left] & \ldots \\ {[\![t[0/r]]\!]} & {[\![t[1/r]]\!]} & \ldots \end{tikzcd}$$

Program. No interaction: $[\![t_1 \parallel \ldots \parallel t_n]\!] = [\![t_1]\!] \parallel \ldots [\![t_n]\!]$.

Wiring memory behaviour

The memory behaviour is specified through **consistent traces**:

$$C_\mu ::= \mathtt{W}_{\mathtt{x}:=k} \cdot C_{\mu[\mathtt{x}:=k]} \mid \mathtt{fence} \cdot C_\mu \mid \mathtt{R}_{\mathtt{x}=\mu(\mathtt{x})} \cdot C_\mu$$

Theorem

For a machine state (p, μ), $\mathit{Tr}(p, \mu) = \mathit{Tr}[\![p]\!] \cap C_\mu$.

Wiring memory behaviour

The memory behaviour is specified through **consistent traces**:

$$C_\mu ::= \mathtt{W}_{\mathtt{x}:=k} \cdot C_{\mu[\mathtt{x}:=k]} \mid \mathtt{fence} \cdot C_\mu \mid \mathtt{R}_{\mathtt{x}=\mu(\mathtt{x})} \cdot C_\mu$$

Theorem

For a machine state (p, μ)*,* $\mathit{Tr}(p, \mu) = \mathit{Tr}[\![p]\!] \cap C_\mu$.

But I promised an e.s. $[\![p, \mu]\!]$! (**causally** account for memory)

Causal account for the memory

$\rightsquigarrow$ Instead of a set of *traces* C_μ, a set of *partial orders* $\mathscr{C}_\mu$.

No canonical notions, but several compromises:

Definition

A partial order $\mathbf{q}$ is:

- **semantically consistent** when $\mathrm{Tr}(\mathbf{q}) \subseteq C_\mu$.
- **syntactically consistent** when for each variable `x`, actions in $\mathbf{q}$ on `x` are linearly ordered.

Causal account for the memory

$\rightsquigarrow$ Instead of a set of *traces* C_μ, a set of *partial orders* $\mathscr{C}_\mu$.

No canonical notions, but several compromises:

Definition

A partial order $\mathbf{q}$ is:

- **semantically consistent** when $\mathrm{Tr}(\mathbf{q}) \subseteq C_\mu$.
- **syntactically consistent** when for each variable `x`, actions in $\mathbf{q}$ on `x` are linearly ordered.

$$\mathscr{C}_\mu^{\mathsf{sem}} = \{\mathbf{q} \mid \mathbf{q} \text{ semantically consistent}\}.$$
$$\mathscr{C}_\mu^{\mathsf{syn}} = \{\mathbf{q} \mid \mathbf{q} \text{ syntactically consistent}\}.$$

We have $\mathrm{Tr}(\mathscr{C}_\mu^{\mathsf{sem}}) = \mathrm{Tr}(\mathscr{C}_\mu^{\mathsf{syn}}) = C_\mu$ but $\mathscr{C}_\mu^{\mathsf{sem}} \subsetneq \mathscr{C}_\mu^{\mathsf{syn}}$.

II. Event structure theory

How to merge $[\![p]\!]$ and $\mathscr{C}_\mu$

Briding a gap: event-based and execution-based models

ES

$[\![p]\!]$

Briding a gap: event-based and execution-based models

$\mathbf{ES}$ $\mathrm{Set}(\mathbf{PO})$

$[\![p]\!]$ $\mathscr{C}_\mu$

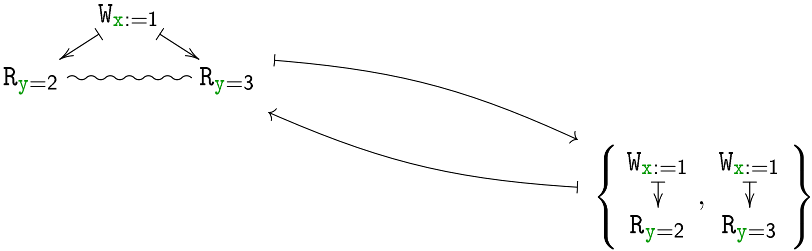
Briding a gap: event-based and execution-based models

$$\mathbf{ES} \xrightarrow{\mathscr{C}(\cdot)} \mathrm{Set}(\mathbf{PO})$$

$$\begin{tikzcd} & \mathtt{W}_{\mathtt{x}:=1} \arrow[dl, |->] \arrow[dr, |->] & \\ \mathtt{R}_{\mathtt{y}=2} \arrow[rr, no head, squiggly] & & \mathtt{R}_{\mathtt{y}=3} \end{tikzcd} \longmapsto \left\{ \begin{tikzcd} \mathtt{W}_{\mathtt{x}:=1} \arrow[d, |->] \\ \mathtt{R}_{\mathtt{y}=2} \end{tikzcd} , \begin{tikzcd} \mathtt{W}_{\mathtt{x}:=1} \arrow[d, |->] \\ \mathtt{R}_{\mathtt{y}=3} \end{tikzcd} \right\}$$

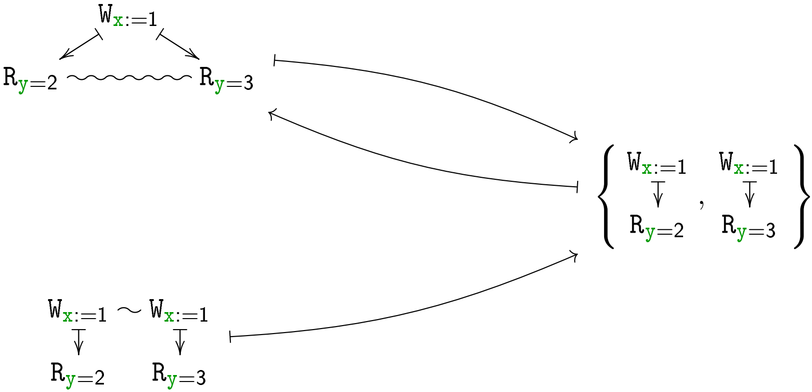
Briding a gap: event-based and execution-based models

$$\begin{tikzcd} \mathbf{ES} \arrow[rr, bend left, "\mathscr{C}(\cdot)"] & & \mathrm{Set}(\mathbf{PO}) \arrow[ll, bend left, "\mathrm{Pr}(\cdot)"] \end{tikzcd}$$



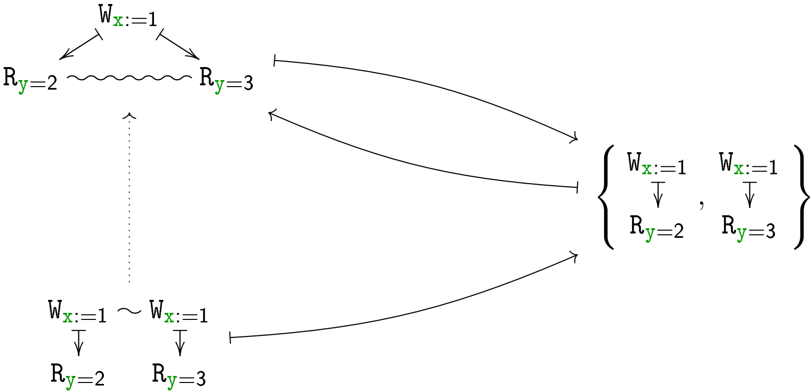
Briding a gap: event-based and execution-based models

$$\begin{tikzcd} \mathbf{ES} \arrow[rr, bend left, "\mathscr{C}(\cdot)"] & & \mathrm{Set}(\mathbf{PO}) \arrow[ll, bend left, "\mathrm{Pr}(\cdot)"] \end{tikzcd}$$



Briding a gap: event-based and execution-based models

$$\begin{tikzcd} \mathbf{ES} \arrow[rr, bend left, "\mathscr{C}(\cdot)"] \arrow[rr, phantom, "\top"] & & \mathrm{Set}(\mathbf{PO}) \arrow[ll, bend left, "\mathrm{Pr}(\cdot)"] \end{tikzcd}$$



The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{P}r(\mathscr{Q})$:

- **Events** Those $\mathbf{q} \in \mathscr{Q}$ with a top element
- **Causality** $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$
- **Conflict** $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{P}r(\mathscr{Q})$:

Events	Those $\mathbf{q} \in \mathscr{Q}$ with a top element
Causality	$\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$
Conflict	$\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{array}{c} a \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ d \end{array} \right\}, \left\{ \begin{array}{ccc} & a & \\ \swarrow & & \searrow \\ c & & d \end{array} \right\} \right\}.$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{Pr}(\mathscr{Q})$:

Events Those $\mathbf{q} \in \mathscr{Q}$ with a top element

Causality $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$

Conflict $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{array}{c} a \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ d \end{array} \right\}, \left\{ \begin{array}{ccc} & a & \\ \swarrow & & \searrow \\ c & & d \end{array} \right\} \right\}.$$

$$\mathrm{Pr}(\mathscr{Q}) = \begin{array}{cccc} a & & b & \\ c & c & & d \end{array}$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{Pr}(\mathscr{Q})$:

- **Events** Those $\mathbf{q} \in \mathscr{Q}$ with a top element
- **Causality** $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$
- **Conflict** $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{array}{c} a \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ d \end{array} \right\}, \left\{ \begin{array}{ccc} & a & \\ \swarrow & & \searrow \\ c & & d \end{array} \right\} \right\}.$$

$$\mathrm{Pr}(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] & & b \\ c & c & & d \end{tikzcd}$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{Pr}(\mathscr{Q})$:

- **Events** Those $\mathbf{q} \in \mathscr{Q}$ with a top element
- **Causality** $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$
- **Conflict** $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{tikzcd} a \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ d \end{tikzcd} \right\}, \left\{ \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ c & & d \end{tikzcd} \right\} \right\}.$$

$$\mathrm{Pr}(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] & & b \arrow[dl] & \\ c & c & & d \end{tikzcd}$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathrm{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{Pr}(\mathscr{Q})$:

- **Events** Those $\mathbf{q} \in \mathscr{Q}$ with a top element
- **Causality** $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$
- **Conflict** $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{array}{c} a \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ d \end{array} \right\}, \left\{ \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ c & & d \end{tikzcd} \right\} \right\}.$$

$$\mathrm{Pr}(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] & & b \arrow[dl] \arrow[dr] & \\ c & c & & d \end{tikzcd}$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{P}r(\mathscr{Q})$:

Events Those $\mathbf{q} \in \mathscr{Q}$ with a top element

Causality $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$

Conflict $\mathbf{q} \sim\!\sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$
\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{tikzcd} a \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ d \end{tikzcd} \right\}, \left\{ \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ c & & d \end{tikzcd} \right\} \right\}.
$$

$$
\mathrm{Pr}(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] \arrow[rr, no head, squiggly] & & b \arrow[dl] \arrow[dr] & \\ c & c & & d \end{tikzcd}
$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{Pr}(\mathscr{Q})$:

- **Events** Those $\mathbf{q} \in \mathscr{Q}$ with a top element
- **Causality** $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$
- **Conflict** $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{tikzcd} a \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ d \end{tikzcd} \right\}, \left\{ \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ c & & d \end{tikzcd} \right\} \right\}.$$

$$\mathrm{Pr}(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] \arrow[rr, no head, squiggly] & & b \arrow[dl] \arrow[dr] & \\ c & c & & d \end{tikzcd}$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{Pr}(\mathscr{Q})$:

Events Those $\mathbf{q} \in \mathscr{Q}$ with a top element

Causality $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$

Conflict $\mathbf{q} \sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{array}{c} a \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ c \end{array} \right\}, \left\{ \begin{array}{c} b \\ \downarrow \\ d \end{array} \right\}, \left\{ \begin{tikzcd}[column sep=small] & a \arrow[dl] \arrow[dr] & \\ c & & d \end{tikzcd} \right\} \right\}.$$

$$\mathrm{Pr}(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] \arrow[r, no head, squiggly] & b \arrow[dl] \arrow[dr] & \\ c & c & d \end{tikzcd}$$

The prime construction: details

Consider a set $\mathscr{Q} \in \mathsf{Set}(\mathbf{PO})$ *closed under prefix*. We define $\mathrm{P}r(\mathscr{Q})$:

Events Those $\mathbf{q} \in \mathscr{Q}$ with a top element

Causality $\mathbf{q} \leq \mathbf{q}'$ when $\mathbf{q}$ is a prefix of $\mathbf{q}'$

Conflict $\mathbf{q} \sim\!\sim \mathbf{q}'$ when they have no upper bound in $\mathscr{Q}$.

$$\mathscr{Q} = \left\{ \{a\}, \{b\}, \left\{ \begin{tikzcd} a \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ c \end{tikzcd} \right\}, \left\{ \begin{tikzcd} b \arrow[d] \\ d \end{tikzcd} \right\}, \left\{ \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ c & & d \end{tikzcd} \right\} \right\}.$$

$$\Pr(\mathscr{Q}) = \begin{tikzcd} a \arrow[d] \arrow[r, no head, squiggly] & b \arrow[dl] \arrow[dr] & \\ c \quad c & & d \end{tikzcd}$$

NB: $\mathscr{C}(\mathrm{Pr}(\mathscr{Q})) \cong \mathscr{Q}$.

A partial product on partial orders

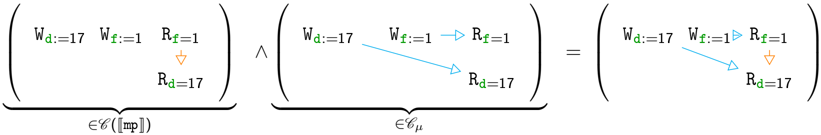
Given two partial orders $\leq_{\mathbf{q}}, \leq_{\mathbf{q}'}$ on the same carrier set, write:

$$\mathbf{q} \wedge \mathbf{q}' = \begin{cases} (\mathbf{q} \cup \mathbf{q}')^* & \text{if a partial order} \\ \text{undefined} & \text{otherwise} \end{cases} .$$

A partial product on partial orders

Given two partial orders $\leq_{\mathbf{q}}, \leq_{\mathbf{q}'}$ on the same carrier set, write:

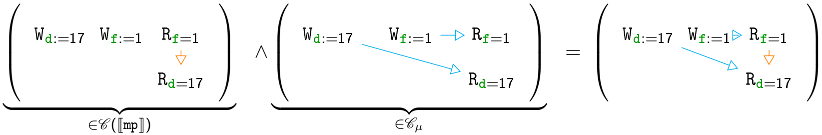
$$\mathbf{q} \wedge \mathbf{q}' = \begin{cases} (\mathbf{q} \cup \mathbf{q}')^* & \text{if a partial order} \\ \text{undefined} & \text{otherwise} \end{cases} .$$

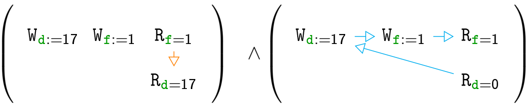


A partial product on partial orders

Given two partial orders $\leq_{\mathbf{q}}, \leq_{\mathbf{q}'}$ on the same carrier set, write:

$$\mathbf{q} \wedge \mathbf{q}' = \begin{cases} (\mathbf{q} \cup \mathbf{q}')^* & \text{if a partial order} \\ \text{undefined} & \text{otherwise} \end{cases} .$$





= undefined

. . . generating a product on event structures

For $P, Q \in \mathrm{Sets}(\mathbf{PO})$, let:

$$P * Q = \{p \wedge q \mid p \in P, q \in Q\}.$$

...generating a product on event structures

For $P, Q \in \mathrm{Sets}(\mathbf{PO})$, let:

$$P{*}Q = \{p \wedge q \mid p \in P, q \in Q\}.$$

For $E, E' \in \mathbf{ES}$, let:

$$E{*}E' = \mathrm{Pr}(\mathscr{C}(E){*}\mathscr{C}(E')).$$

...generating a product on event structures

For $P, Q \in \mathrm{Sets}(\mathbf{PO})$, let:

$$P * Q = \{p \wedge q \mid p \in P, q \in Q\}.$$

For $E, E' \in \mathbf{ES}$, let:

$$E * E' = \mathrm{Pr}(\mathscr{C}(E) * \mathscr{C}(E')).$$

Theorem

Both operations are categorical products.

Note:

$$\mathrm{Tr}(E * E') = \mathrm{Tr}(E) \cap \mathrm{Tr}(E')$$

Illustrations of this construction

- Conflicts are merged:

$$\left(\; a \sim b \quad c \; \right) * \left(\; a \quad b \sim c \; \right) = \left(\; a \sim b \sim c \; \right)$$

Illustrations of this construction

- Conflicts are merged:

$$\left(\; a \sim b \quad c \; \right) * \left(\; a \quad b \sim c \; \right) = \left(\; a \sim b \sim c \; \right)$$

- Events with incompatible histories are duplicated:

$$\left(\; a \sim a \quad b \; \right) * \left(\begin{array}{c} a \\ \downarrow \\ b \end{array} \right) = \left(\begin{array}{ccc} a & \sim & a \\ \downarrow & & \downarrow \\ b & & b \end{array} \right)$$

A final model

Define $[\![p, \mu]\!]^{\mathsf{mem}} = \mathrm{Pr}(\mathscr{C}([\![p]\!]) * \mathscr{C}_\mu^{\mathsf{mem}})$. We have:

$$\mathrm{Tr}[\![p, \mu]\!]^{\mathsf{mem}} = \mathrm{Tr}[\![p]\!] \cap \mathrm{Tr}[\![\mathscr{C}_\mu^{\mathsf{mem}}]\!] = \mathrm{Tr}[\![p]\!] \cap C_\mu = \mathrm{Tr}(p, \mu).$$

Yields the desired result:

$$[\![\mathtt{mp}, (\mathtt{x} \mapsto 0)]\!]^{\mathsf{syn}} = \begin{tikzcd} \mathtt{W}_{\mathtt{data}:=17} \arrow[d] & \mathtt{W}_{\mathtt{flag}:=1} \arrow[d] \arrow[r, no head, squiggly] & \mathtt{R}_{\mathtt{flag}=0} \arrow[d] \\ \mathtt{R}_{\mathtt{data}=17} \arrow[dr, no head, squiggly] & \mathtt{R}_{\mathtt{flag}=1} \arrow[l] \arrow[d] & \mathtt{W}_{\mathtt{flag}:=1} \\ & \mathtt{R}_{\mathtt{data}=0} \arrow[d] & \\ & \mathtt{W}_{\mathtt{data}:=17} & \end{tikzcd}$$

III. Applications

(1) Theory: Data racefreedom
(Joint work with Jade Alglave and Jean-Marie Madiot)

Races and sizes

A race: two co-located concurrent accesses (among which a write).

$$\texttt{data} := \texttt{0xdeadbeef} \;\Big\|\; \begin{array}{l} r \leftarrow \texttt{data} \\ \texttt{assert}\ (\texttt{data} \in \{0, \texttt{0xdeadbeef}\}) \end{array}$$

If `data` is two words, we might see: $\texttt{data} = \texttt{0xdead0000}$.

Races and sizes

A race: two co-located concurrent accesses (among which a write).

$$\texttt{data} := \texttt{0xdeadbeef} \;\Big\|\; \begin{array}{l} r \leftarrow \texttt{data} \\ \texttt{assert}\ (\texttt{data} \in \{0, \texttt{0xdeadbeef}\}) \end{array}$$

If $\texttt{data}$ is two words, we might see: $\texttt{data} = \texttt{0xdead0000}$.

But, $\texttt{mp}$ should be ok:

$$\begin{array}{l} \texttt{data} := 17; \\ \texttt{flag} := 1 \end{array} \;\Big\|\; \begin{array}{l} r \leftarrow \texttt{flag}; \\ \texttt{if}(r == 1)\{v \leftarrow \texttt{data}\} \end{array}$$

The race on $\texttt{flag}$ does not matter since $\texttt{flag}$ is “small”.

“Small locations” and the notion of race

To model this, we split variables into two groups:

`multi word variables` `single word variables`.

Races on single words variables are ok (necessary to implement eg. locks).

Definition

A **race** of a program p is a a trace $w \in (\mathbb{N} \times \Sigma)^*$ of the form:

$$w = \ldots \cdot (i, \mathtt{R}_{\mathtt{x}=k}) \cdot (j, \mathtt{W}_{\mathtt{x}:=k'})$$

with $i \neq j$ and `x` is a multi word variable.

Definition

A program is **well-synchronized** (or **race-free**) when none of its traces *on SC* are races.

Data-Racefreedom

A wanted property for most architectures:

Definition (Data Racefreedom (DRF))

An architecture $\mathcal{A}$ satisfies (DRF) when for all well-synchronized program p,

$$\mathrm{MemStates}_{SC}(p) = \mathrm{MemStates}_{\mathcal{A}}(p).$$

Data-Racefreedom

A wanted property for most architectures:

Definition (Data Racefreedom (DRF))

An architecture $\mathcal{A}$ satisfies (DRF) when for all well-synchronized program p,

$$\mathrm{MemStates}_{\mathrm{SC}}(p) = \mathrm{MemStates}_{\mathcal{A}}(p).$$

Theorem

Our architecture MiniARM satisfies data racefreedom.

Proof.

In two steps:

1. Show that if p is race-free *on MiniARM*, then $\mathrm{MemStates}_{\mathrm{SC}}(p) = \mathrm{MemStates}_{\mathcal{A}}(p)$.
2. If p has a race on MiniARM, then it has a race on SC. □

III. Applications

(2) Practice: smaller structures
(Joint work with Jade Alglave and Jean-Marie Madiot)

Two different compromises

- $\mathscr{C}_\mu^{\mathsf{sem}}$ is too hard to compute (check **all** the traces)
- $\mathscr{C}_\mu^{\mathsf{syn}}$ is too big (forces linearisation on each variable)

$\rightsquigarrow$ Can we do better? Not force all writes to be synchronized.

$$\mathtt{x} := 1 \parallel \mathtt{x} := 2$$

$$\begin{tikzcd}
 & & \mathtt{W}_{\mathtt{x}:=1} \arrow[r, phantom, "\sim"] \arrow[d] & \mathtt{W}_{\mathtt{x}:=2} \arrow[d] \\
\mathtt{W}_{\mathtt{x}:=1} & \mathtt{W}_{\mathtt{x}:=2} & \mathtt{W}_{\mathtt{x}:=2} & \mathtt{W}_{\mathtt{x}:=1}
\end{tikzcd}$$

$[\![p]\!]^{\mathsf{sem}}$ $\qquad\qquad$ $[\![p]\!]^{\mathsf{syn}}$

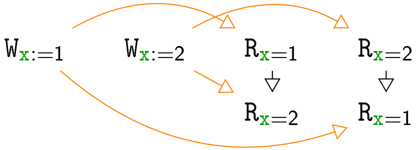
Observing order

Consider:

$$\begin{tikzcd} \mathtt{W}_{\mathtt{x}:=1} & \mathtt{W}_{\mathtt{x}:=2} & \mathtt{R}_{\mathtt{x}=1} \arrow[d] & \mathtt{R}_{\mathtt{x}=2} \arrow[d] \\ & & \mathtt{R}_{\mathtt{x}=2} & \mathtt{R}_{\mathtt{x}=1} \end{tikzcd}$$

Observing order

Consider:



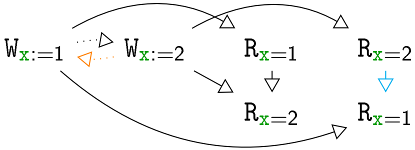
Observing order

Consider:

$$\begin{tikzcd}
\mathtt{W}_{\mathtt{x}:=1} \arrow[r, dotted] \arrow[rr, bend left] \arrow[drrr, bend right] & \mathtt{W}_{\mathtt{x}:=2} \arrow[rr, bend left] \arrow[dr] & \mathtt{R}_{\mathtt{x}=1} \arrow[d] & \mathtt{R}_{\mathtt{x}=2} \arrow[d] \\
& & \mathtt{R}_{\mathtt{x}=2} & \mathtt{R}_{\mathtt{x}=1}
\end{tikzcd}$$

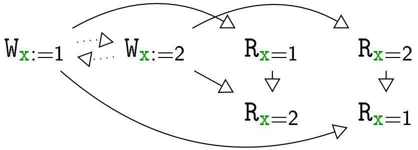
Observing order

Consider:



Observing order

Consider:



By investigation, we find a set of axioms...

Lazy consistency

Definition

A partial order **q** is **lazily consistent** when it satisfies:

- For every read $r_x \in \mathbf{q}$, there exists a (unique) maximal write just(w_x) below r_x, with the same value.
- If whenever $w_x : \mathtt{W}_{\mathtt{x}:=\_} \leq r_y : \mathtt{R}_{\mathtt{y}=\_}$ and $w_y : \mathtt{W}_{\mathtt{y}:=\_} \leq r_x : \mathtt{R}_{\mathtt{x}=\_}$ with $\mathrm{just}(r_x) \neq w_x, \mathrm{just}(r_y) \neq w_y$, then $w_x < \mathrm{just}(r_x)$ or $w_y < \mathrm{just}(r_y)$ (or possibily both).

We write $\mathscr{C}_\mu^{\mathtt{lazy}}$ for the set of lazily consistent po.

Lazy consistency

Definition

A partial order $\mathbf{q}$ is **lazily consistent** when it satisfies:

- For every read $r_x \in \mathbf{q}$, there exists a (unique) maximal write $\mathrm{just}(w_x)$ below r_x, with the same value.
- If whenever $w_x : \mathtt{W}_{\mathtt{x}:=\_} \leq r_y : \mathtt{R}_{\mathtt{y}=\_}$ and $w_y : \mathtt{W}_{\mathtt{y}:=\_} \leq r_x : \mathtt{R}_{\mathtt{x}=\_}$ with $\mathrm{just}(r_x) \neq w_x, \mathrm{just}(r_y) \neq w_y$, then $w_x < \mathrm{just}(r_x)$ or $w_y < \mathrm{just}(r_y)$ (or possibily both).

We write $\mathscr{C}_\mu^{\mathtt{lazy}}$ for the set of lazily consistent po.

Theorem (**Weaker correctness**)

Up to permutations of reads and independent writes, every trace of a lazy consistent po is consistent.

$\rightsquigarrow \mathrm{MemStates}(\llbracket p \rrbracket^{\mathrm{lazy}}) = \mathrm{MemStates}(p)$.

A demo

$$P = \quad \texttt{x} := 1 \;\Big\|\; \texttt{x} := 3$$

$$Q = \quad \begin{array}{l} \texttt{x} := 1 \\ \end{array} \left\| \begin{array}{l} \texttt{x} := 3 \\ \end{array} \right\| \begin{array}{l} r \leftarrow \texttt{x} \\ s \leftarrow \texttt{x} \end{array}$$

Related work / Extensions

Related work.

- Brookes et al.'s model of TSO with pomsets.
- Pichon et al.'s operational semantics on event structures
- Jeffrey and Riely's axiomatic model using event structures

Extensions. Extend this to:

- Real ARM, TSO, Linux-C, etc.
- More complicated C11 models.