Denotational event structure for relaxed memory

Jade Alglave[1], Simon Castellan[2], Jean-Marie Madiot[3]

[1]ARM, and University College London, UK
[2]Imperial College London, UK
[3]INRIA

7th July, 2018
LOLA 2018

Message-passing on my computer

Consider the program `mp`:

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

Message-passing on my computer

Consider the program `mp`:

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

Possible execution traces on my computer:

Message-passing on my computer

Consider the program mp:

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

Possible execution traces on my computer:

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

Message-passing on my computer

Consider the program `mp`:

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

Possible execution traces on my computer:

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

Message-passing on my computer

Consider the program mp:

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

Possible execution traces on my computer:

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

Message-passing on my computer

Consider the program mp:

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

Possible execution traces on my computer:

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

Message-passing on my computer

Consider the program mp:

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

Possible execution traces on my computer:

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

Message-passing on my phone

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

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

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

Possible execution traces on my **phone**:

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

Message-passing on my phone

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

Possible execution traces on my **phone**:

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

Message-passing on my phone

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

Possible execution traces on my **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

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

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

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

Possible execution traces on my **phone**:

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

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

Structure behind traces

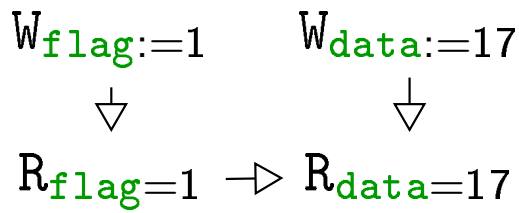
$$\left\{\begin{array}{l} \mathtt{W_{flag}{:=}1} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}17} \\ \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{data}{=}17} \\ \mathtt{W_{data}{:=}17} \cdot \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}17} \end{array}\right.$$

$$\left\{\ \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}0} \cdot \mathtt{W_{data}{:=}17}\right.$$

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

Structure behind traces

$$\left\{\begin{array}{l} \mathtt{W_{flag}{:=}1} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}17} \\ \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{data}{=}17} \\ \mathtt{W_{data}{:=}17} \cdot \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}17} \end{array}\right.$$

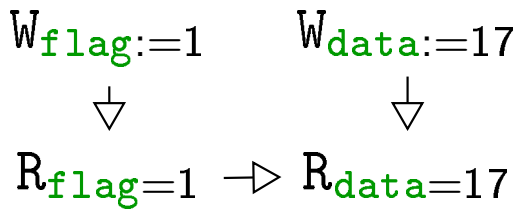


$$\left\{ \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}0} \cdot \mathtt{W_{data}{:=}17} \right.$$

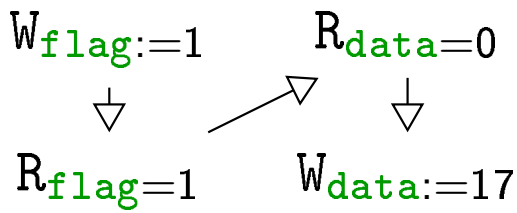
$$\left\{\begin{array}{l} \mathtt{R_{flag}{=}0} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{W_{flag}{:=}1} \\ \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{flag}{=}0} \cdot \mathtt{W_{flag}{:=}1} \\ \mathtt{R_{flag}{=}0} \cdot \mathtt{W_{flag}{:=}1} \cdot \mathtt{W_{data}{:=}17} \end{array}\right.$$

Structure behind traces

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



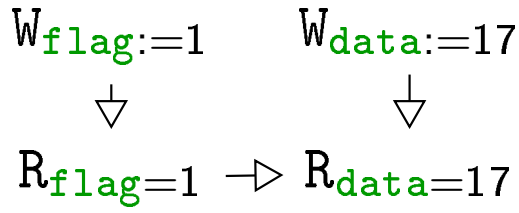
$$\{\ \mathtt{W}_{\mathtt{flag}}{:=}1 \cdot \mathtt{R}_{\mathtt{flag}}{=}1 \cdot \mathtt{R}_{\mathtt{data}}{=}0 \cdot \mathtt{W}_{\mathtt{data}}{:=}17$$



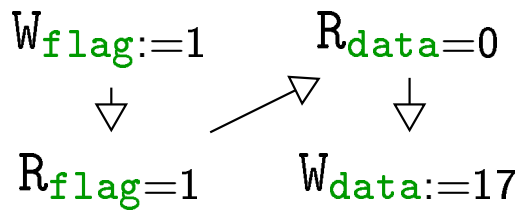
$$\begin{cases} \mathtt{R}_{\mathtt{flag}}{=}0 \cdot \mathtt{W}_{\mathtt{data}}{:=}17 \cdot \mathtt{W}_{\mathtt{flag}}{:=}1 \\ \mathtt{W}_{\mathtt{data}}{:=}17 \cdot \mathtt{R}_{\mathtt{flag}}{=}0 \cdot \mathtt{W}_{\mathtt{flag}}{:=}1 \\ \mathtt{R}_{\mathtt{flag}}{=}0 \cdot \mathtt{W}_{\mathtt{flag}}{:=}1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

Structure behind traces

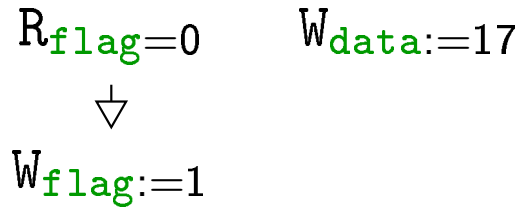
$$\begin{cases} \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{R}_{\mathtt{flag}=1} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{data}=17} \\ \mathtt{W}_{\mathtt{data}:=17}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



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

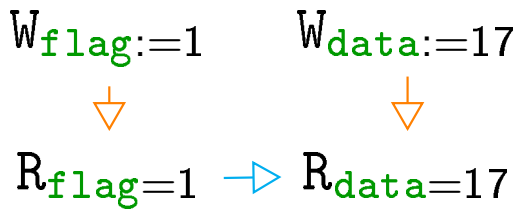


$$\begin{cases} \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{W}_{\mathtt{data}:=17} \cdot \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \\ \mathtt{R}_{\mathtt{flag}=0} \cdot \mathtt{W}_{\mathtt{flag}:=1} \cdot \mathtt{W}_{\mathtt{data}:=17}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

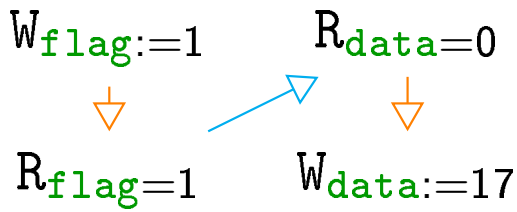


Structure behind traces

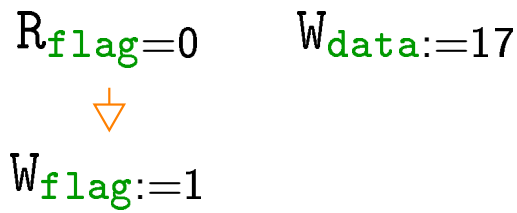
$$\begin{cases} \mathtt{W_{flag}{:=}1} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}17} \\ \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{data}{=}17} \\ \mathtt{W_{data}{:=}17} \cdot \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}17} \end{cases}$$



$$\left\{ \mathtt{W_{flag}{:=}1} \cdot \mathtt{R_{flag}{=}1} \cdot \mathtt{R_{data}{=}0} \cdot \mathtt{W_{data}{:=}17} \right.$$

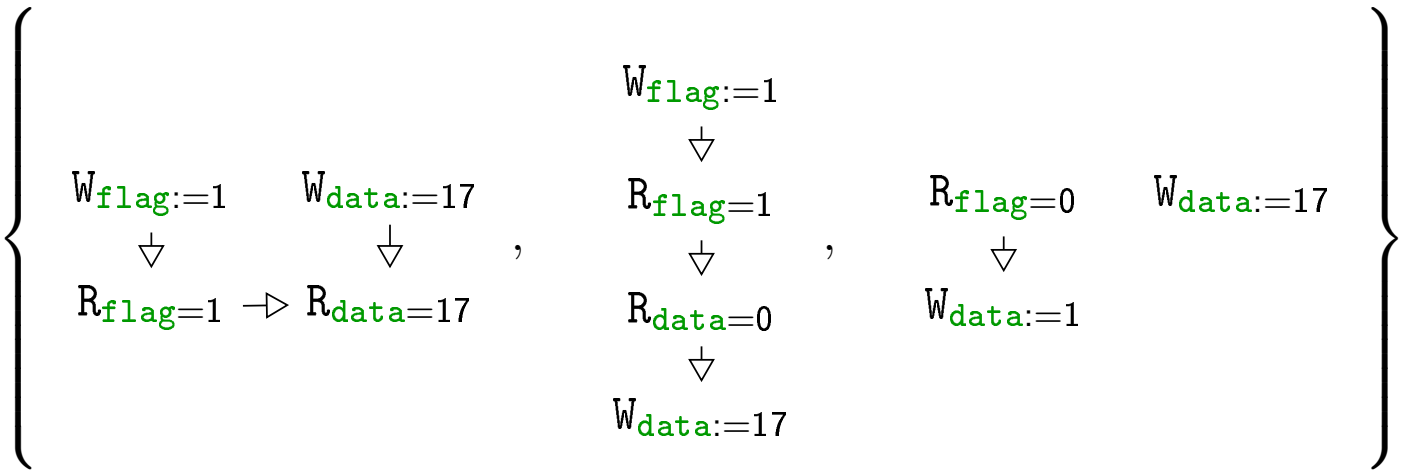


$$\begin{cases} \mathtt{R_{flag}{=}0} \cdot \mathtt{W_{data}{:=}17} \cdot \mathtt{W_{flag}{:=}1} \\ \mathtt{W_{data}{:=}17} \cdot \mathtt{R_{flag}{=}0} \cdot \mathtt{W_{flag}{:=}1} \\ \mathtt{R_{flag}{=}0} \cdot \mathtt{W_{flag}{:=}1} \cdot \mathtt{W_{data}{:=}17} \end{cases}$$



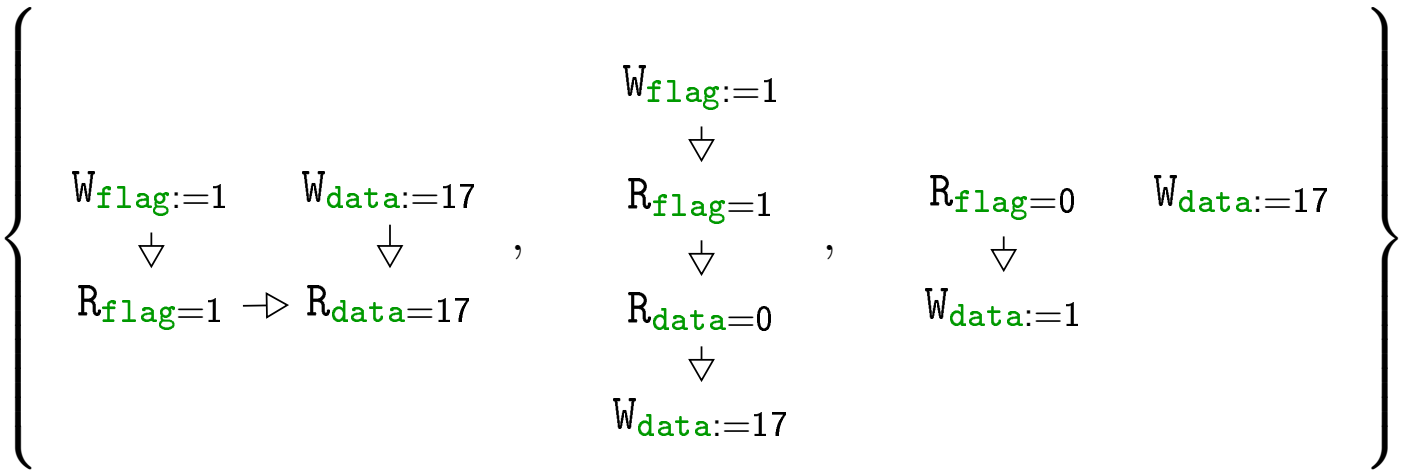
Sets of partial orders and event structures

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

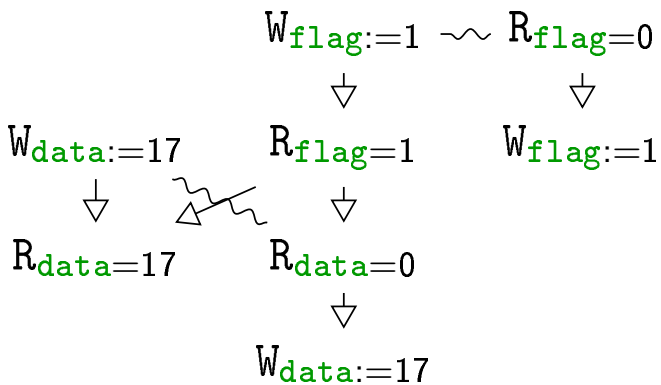


Sets of partial orders and event structures

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

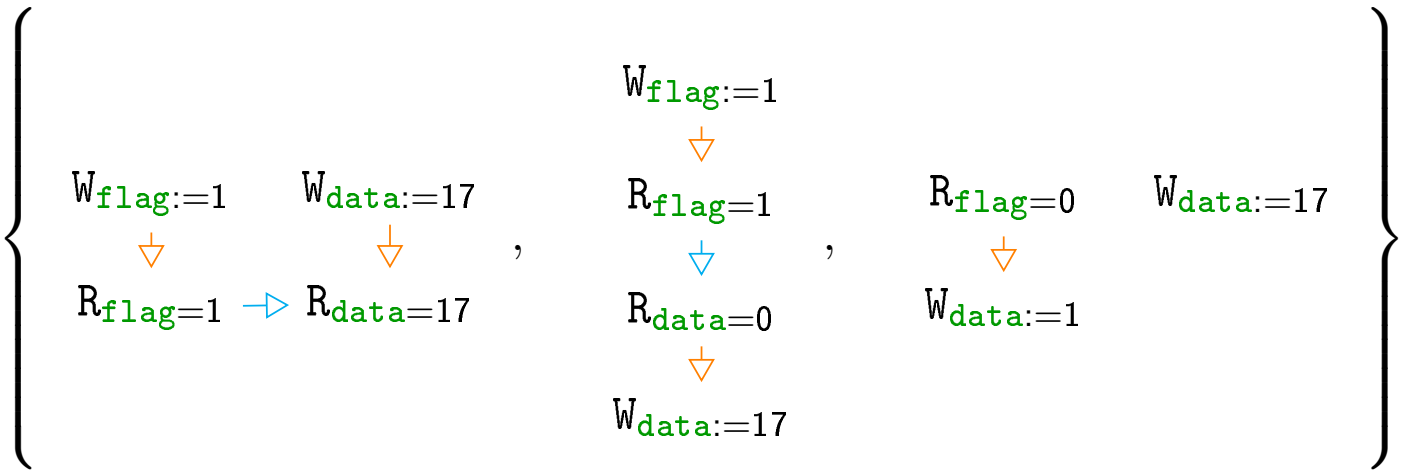


This set of partial orders can be summed by an **event structure**:

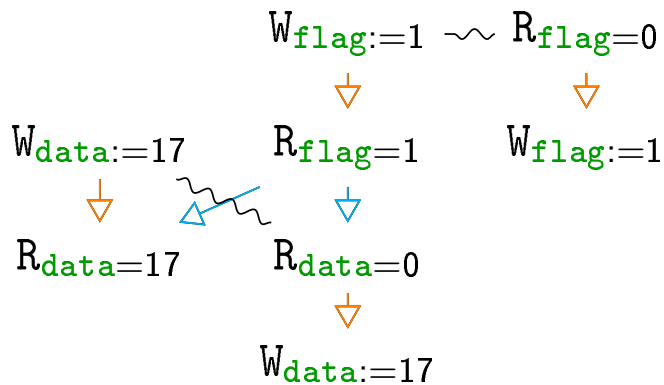


Sets of partial orders and event structures

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. **From programs to event structures**: in a denotational style.
   $\rightsquigarrow$ Combine a semantics for the *threads* and for the *memory*

2. **Using the model to reason about programs**
   $\rightsquigarrow$ For race-free programs, optimisations preserve behaviour.

3. **Using the model to explore the behaviour of programs**
   $\rightsquigarrow$ Herd investigations: How to avoid a linear history per-variable?

I. From programs to event structures

Modelling MiniRMO ($\sim$ MiniARM):

Syntax.

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

Two kinds of idents: thread-local registers and global `variables`.

Operational semantics

Operational semantics is formulated as a LTS over the labels

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

The states of the LST are pairs $(p, \mu : \mathcal{V} \to N)$.

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

Operational semantics

Operational semantics is formulated as a LTS over the labels

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

The states of the LST are pairs $(p, \mu : \mathcal{V} \to N)$.

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

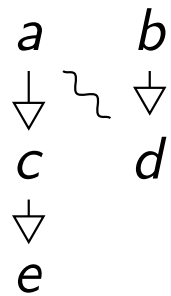
Our goal: a mapping $[\![\cdot]\!]$ from states to event structures s.t.:

$$(p, \mu) \qquad \text{bisimilar to} \qquad [\![(p, \mu)]\!].$$

Labeled event structures

Definition

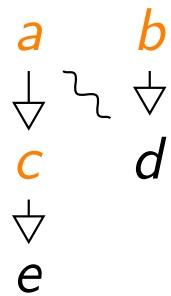
A (Σ-labeled) **event structure** is a tuple $(E, \leq_E, \sharp_E, \ell : E \to \Sigma)$ where $(E, \leq_E)$ is a partial order and $\sharp_E$ is a symmetric relation on E, satisfying *finite causes* and *conflict inheritance*.



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*.

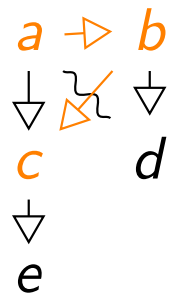


- **Configurations** are downclosed, conflict-free subsets of E. $\mathscr{C}(E)$ is the set of configurations of E.

Labeled event structures

Definition

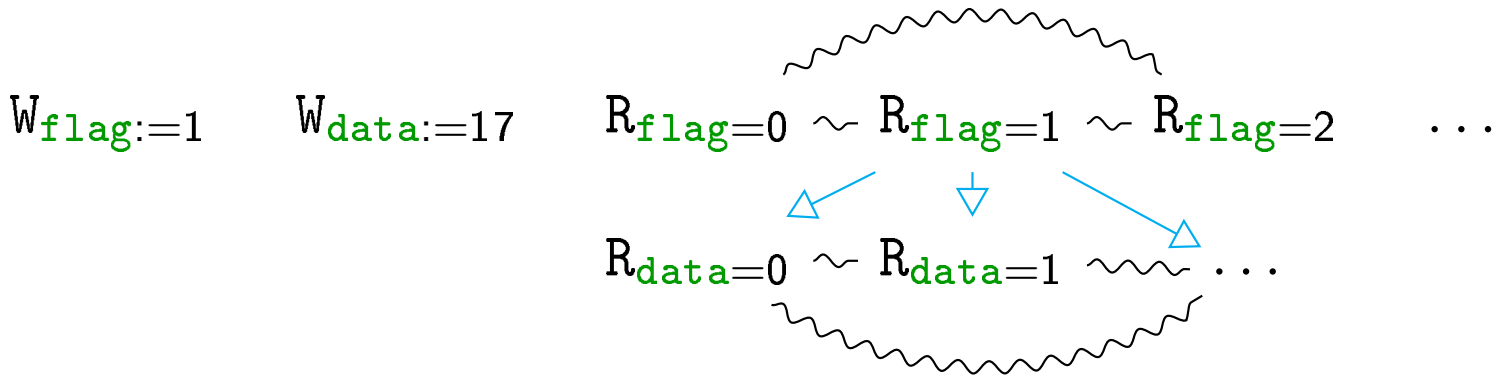
A (Σ-labeled) **event structure** is a tuple $(E, \leq_E, \sharp_E, \ell : E \to \Sigma)$ where $(E, \leq_E)$ is a partial order and $\sharp_E$ is a symmetric relation on E, satisfying *finite causes* and *conflict inheritance*.



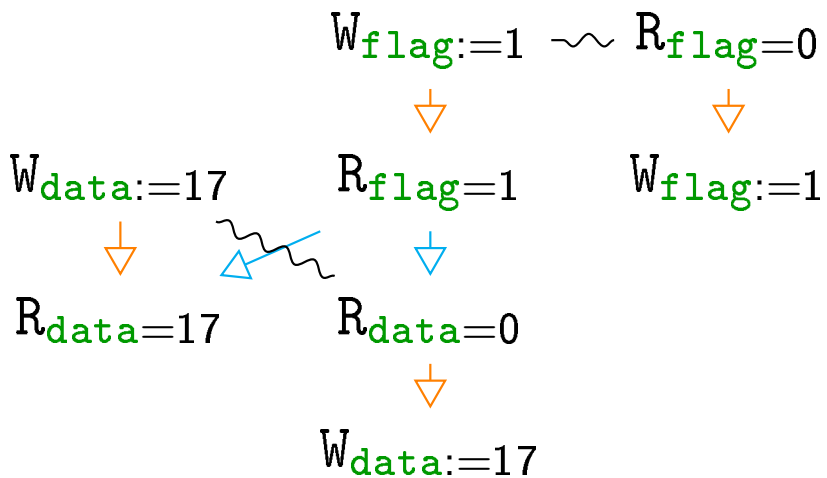
- **Configurations** are downclosed, conflict-free subsets of E. $\mathscr{C}(E)$ is the set of configurations of E.
- Configurations form naturally a LTS: $x \xrightarrow{a} y$ when $y = x \cup \{e\}$ & $\mathsf{lbl}(e) = a$

An overview of the semantics

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



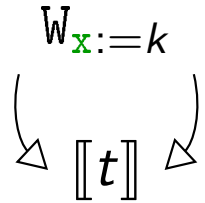
2. **Final semantics**: context is assumed empty
   Compute interactions with memory:



Thread semantics

By induction on threads. For instance

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



The partial order is given by:

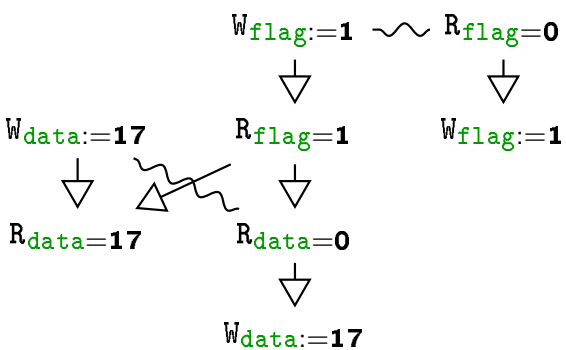
$$(\leq_{\ell;E} = \leq_E \cup \{(\ell, e) \mid e \in E \mid e \text{ is a fence or an operation on } x\}$$

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

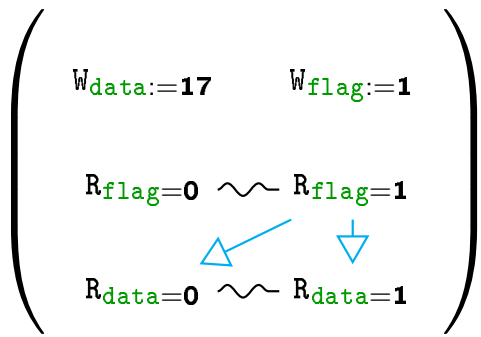
How to interpret memory?

Memory histories (eg. $[\mathtt{W}_{x:=2} \cdot \mathtt{R}_{x=2} \cdot \mathtt{W}_{x:=3}]$) form an *infinite* e.s. C_μ.

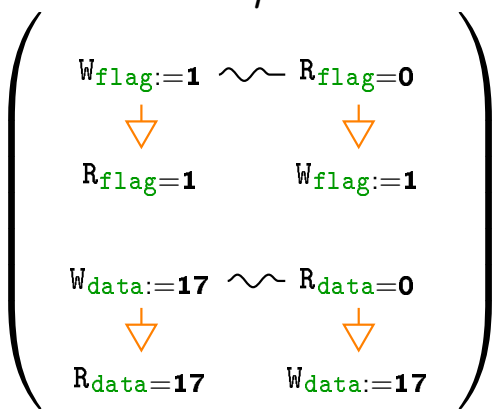
$$[\![(p,\mu)]\!] = [\![p]\!] * C_\mu$$



$=$



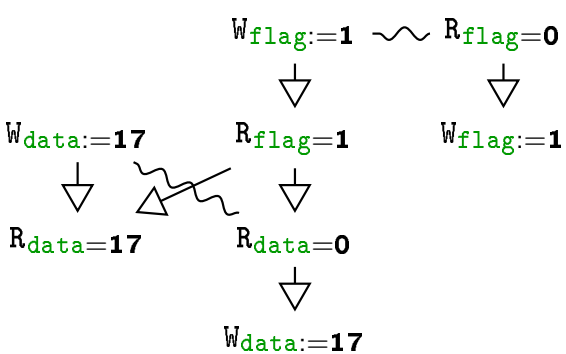
$*$



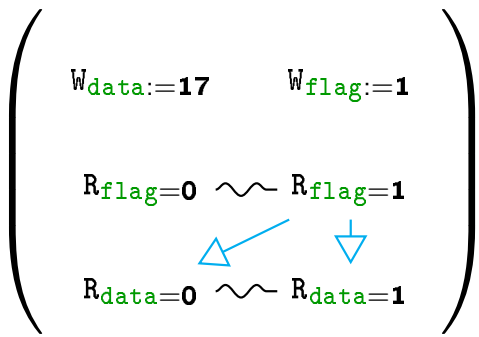
How to interpret memory?

Memory histories (eg. $[\mathtt{W}_{x:=2} \cdot \mathtt{R}_{x=2} \cdot \mathtt{W}_{x:=3}]$) form an *infinite* e.s. C_μ.

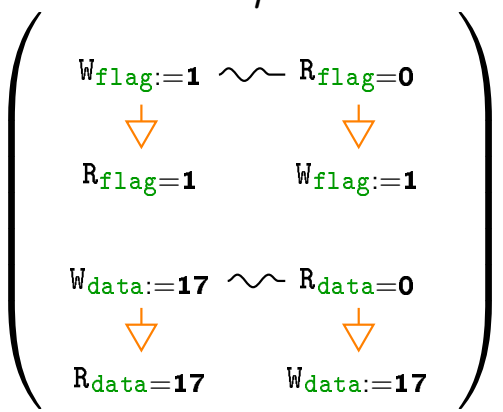
$$[\![(p,\mu)]\!] = [\![p]\!] * C_\mu$$



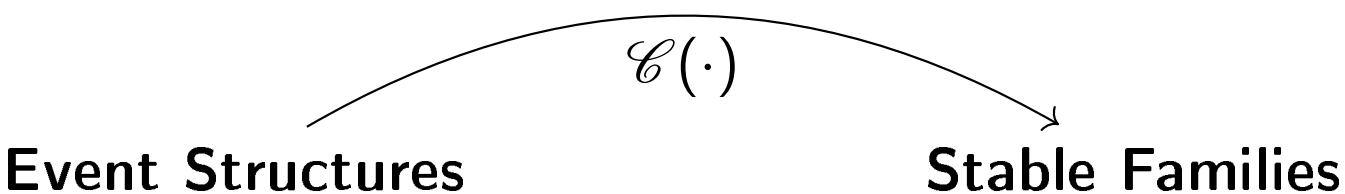
$=$



$*$



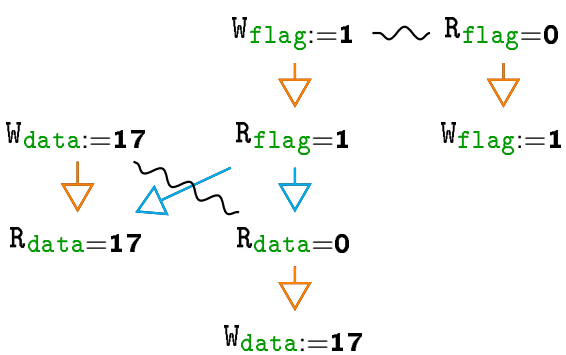
The product is obtained by a coreflection:



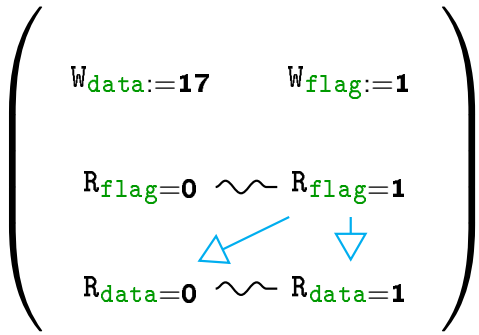
How to interpret memory?

Memory histories (eg. $[\mathtt{W_{x:=2}} \cdot \mathtt{R_{x=2}} \cdot \mathtt{W_{x:=3}}]$) form an *infinite* e.s. C_μ.

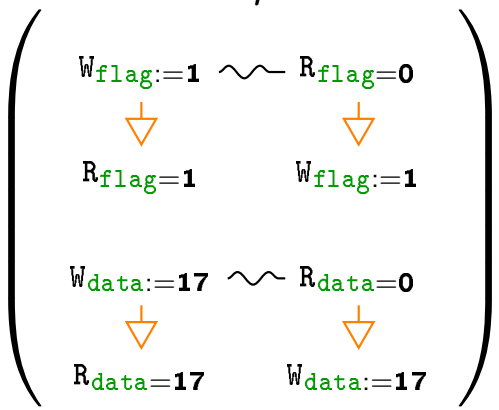
$$[\![(p,\mu)]\!] = [\![p]\!] * C_\mu$$



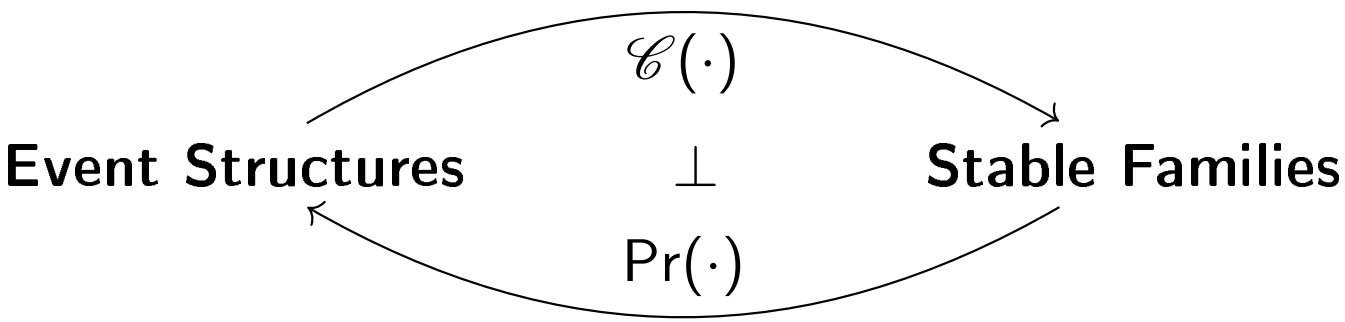
$=$



$*$



The product is obtained by a coreflection:



Reasoning with the model

Sequential consistency: the standard model for shared memory.

Via the same technique, we can build $[\![(p, \mu)]\!]_{\mathtt{SC}}$ (no reordering).

$\rightsquigarrow$ Can we relate $[\![(p, \mu)]\!]_{\mathtt{MiniRMO}}$ and $[\![(p, \mu)]\!]_{\mathtt{SC}}$ if p is well-bahved?

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

But, `mp` should be ok:

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

To model this, we split variables into two groups:

atomic and **non-atomic**.

$\rightsquigarrow$ Races on atomic variables are ok (necessary for eg. locks).
$\rightsquigarrow$ Atomic variables should have release/acquire semantics.

Race-free programs

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 $\mathtt{x}$ is non-atomic.

Definition

A program is **race-free** when none of its traces *on SC* are races.

Race-free programs

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 $\mathtt{x}$ is non-atomic.

Definition

A program is **race-free** when none of its traces *on SC* are races.

Theorem (Strong Data Racefreedom (DRF))

For a race-free program p,

$$[\![p]\!]_{SC} \textit{ weakly bisimilar to } [\![p]\!]_{\mathtt{MiniRMO}}.$$

(Where operations on small variables are considered internal.)

Proof.

By studying properties of the simulation $[\![p]\!]_{\mathtt{SC}} \to [\![p]\!]_{\mathtt{MiniRMO}}$. □

Smaller memory models?

Not seen, not caught!

We choose a *particular* memory implementation, C_μ.

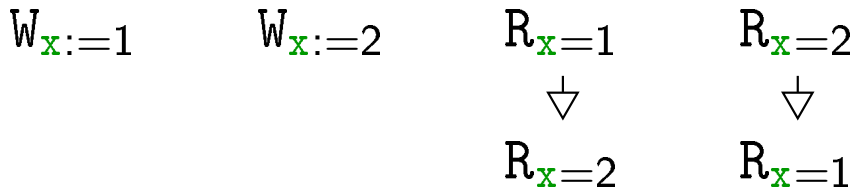
$$\mathtt{x} := 1 \parallel \mathtt{x} := 2$$

$$\begin{array}{ccc} \mathtt{W}_{\mathtt{x}:=1} & \sim & \mathtt{W}_{\mathtt{x}:=2} \\ \downarrow & & \downarrow \\ \mathtt{W}_{\mathtt{x}:=2} & & \mathtt{W}_{\mathtt{x}:=1} \end{array}$$

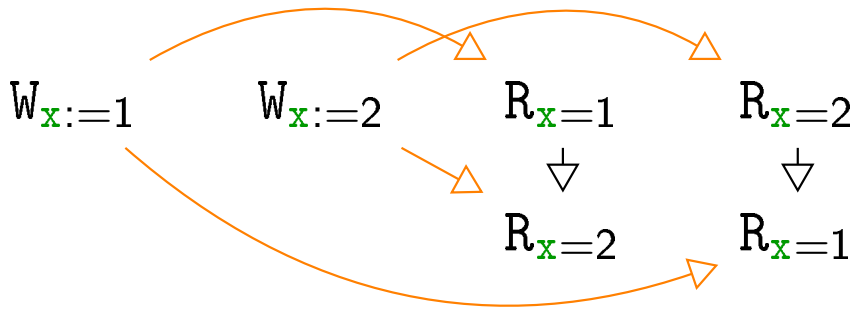
$$[\![p]\!]$$

However no one is observing the order between the writes...
$\rightsquigarrow$ Can we change to have a less sequentializing implementation?

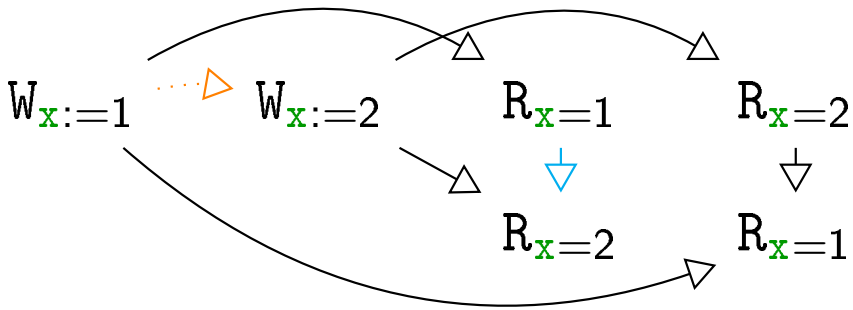
When do we need to sequentialize?



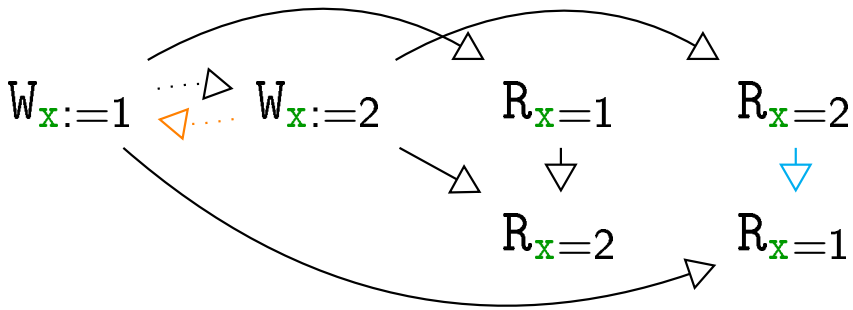
When do we need to sequentialize?



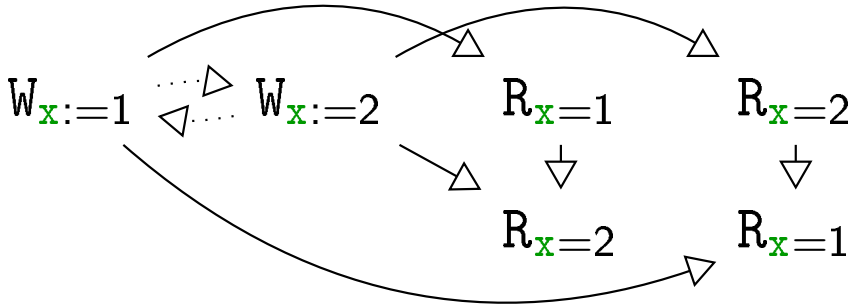
When do we need to sequentialize?



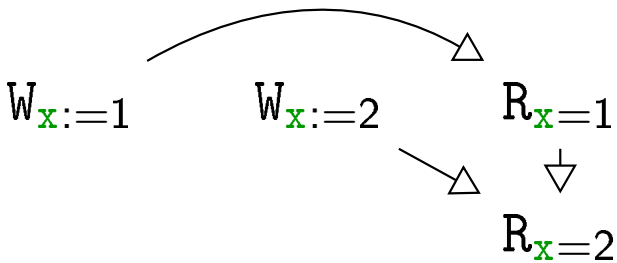
When do we need to sequentialize?



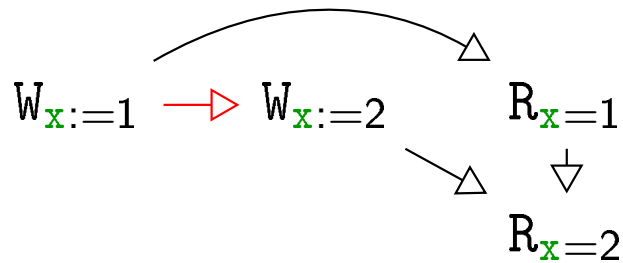
When do we need to sequentialize?



We find a notion of **lazily consistent partial-ordered** history:



not consistent



consistent

Theorem (**Weaker correctness**)

Any trace of a lazy consistent history can be reordered without permuting writes on the same variable to a consistent trace.

$\rightsquigarrow \mathsf{MemStates}([\![p]\!]^{\mathsf{lazy}}) = \mathsf{MemStates}(p)$.

A demo

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

$$Q = \quad \begin{array}{c|c|c} \texttt{x} := 1 & \texttt{x} := 3 & r \leftarrow \texttt{x} \\ & & s \leftarrow \texttt{x} \end{array}$$

Related work / Extensions

Related work.

- Brookes & Kavanagh's model of TSO with pomsets.
- Pichon & Sewell's operational semantics on event structures
- Jeffrey & Riely's axiomatic model using event structures

Extensions. Extend this to:

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