Mechanised Owicki-Gries Proofs for C11

Brijesh Dongol
University of Surrey

Joint work with

Sadegh Dalvandi (University of Surrey)

Simon Doherty (University of Sheffield)

Heike Wehrheim (University of Paderborn)

John Derrick (University of Sheffield)

A weak memory talk

$$\{\texttt{talk} = \texttt{weak\_memory}\}$$
$$\texttt{reaction} := \texttt{listen}(\texttt{talk})$$
$$\{\texttt{reaction} = \smiley \vee \texttt{reaction} = \text{[illegible]}\}$$

A weak memory talk

$\{\texttt{talk} = \texttt{weak\_memory}\}$

$\quad \texttt{reaction} := \texttt{listen}(\texttt{talk}) \qquad \Big\| \qquad \texttt{this}(\texttt{talk})$

$\{\texttt{reaction} = \smiley \vee \texttt{reaction} =$  $\}$

$\{\ \texttt{reaction} = \smiley\ \}$

A weak memory talk

$$
\begin{array}{l|l}
\{\texttt{talk} = \texttt{weak\_memory}\} & \\
\quad \texttt{reaction} := \texttt{listen}(\texttt{talk}) & \quad \texttt{this}(\texttt{talk}) \\
\{\texttt{reaction} = ☺ \vee \texttt{reaction} = \ \} &
\end{array}
$$

$$\{\ \texttt{reaction} = ☺\ \}$$

Turning into ☺ — relate weak memory semantics to Hoare logic and Owicki-Gries style proof rules

Outline

C11 Axiomatic Semantics

Axiomatic C11 semantics

Example (Message Passing).

Init: `f` $:=$ `0`; `d` $:=$ `0`;

thread 1	**thread 2**
`d` $:=$ `5`;	**do** `r1` $\leftarrow$ `f`
`f` $:=$ `1`;	**until** `r1` $=$ `1`;
	`r2` $\leftarrow$ `d`;

Axiomatic C11 semantics

Example (Message Passing).

Init: f := 0; d := 0;	

thread 1
d := 5;
f := 1;
	thread 2
	do r1 ← f
	until r1 = 1;
	r2 ← d;

In C11, `r2` can have a final value `0` — the execution below is allowed

$$\begin{tikzcd} & wr(f,0), wr(d,0) \arrow[dl, "\mathsf{sb}"'] \arrow[dr, "\mathsf{sb}"] & \\ wr_1(d,5) \arrow[d, "\mathsf{sb}"'] & & rd_2(f,1) \arrow[d, "\mathsf{sb}"] \\ wr_1(f,1) & & rd_2(d,0) \end{tikzcd}$$

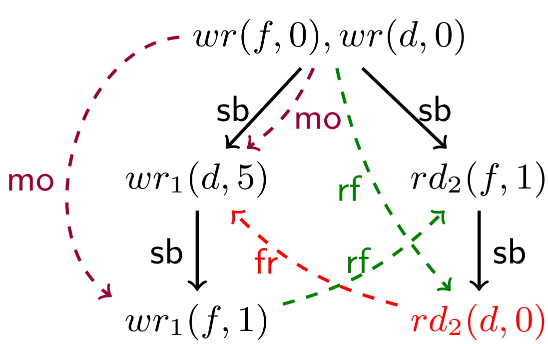
Axiomatic C11 semantics

Example (Message Passing).

Init: `f := 0;` `d := 0;`

thread 1	**thread 2**
`d := 5;`	**do** `r1 ← f`
`f := 1;`	**until** `r1 = 1;`
	`r2 ← d;`

In C11, `r2` can have a final value `0` — the execution below is allowed



Axiomatic C11 semantics

Example (Message Passing).

Init: `f := 0;  d := 0;`

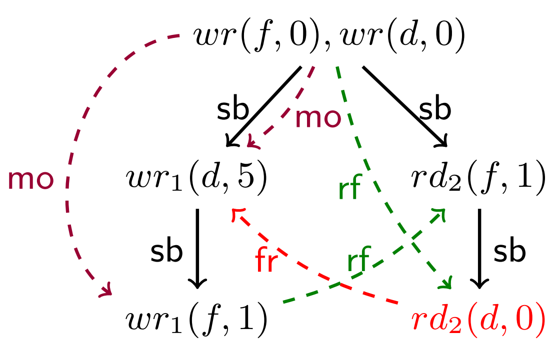
thread 1	**thread 2**
`d := 5;`	**do** `r1 ← f`
`f := 1;`	**until** `r1 = 1;`
	`r2 ← d;`

Corrected Message Passing.

Init: `f := 0;  d := 0;`

thread 1	**thread 2**
`d := 5;`	**do** $\texttt{r1} \leftarrow^{\texttt{A}} \texttt{f}$
$\texttt{f} :=^{\texttt{R}} \texttt{1};$	**until** `r1 = 1;`
	`r2 ← d;`

In C11, `r2` can have a final value `0` — the execution below is allowed



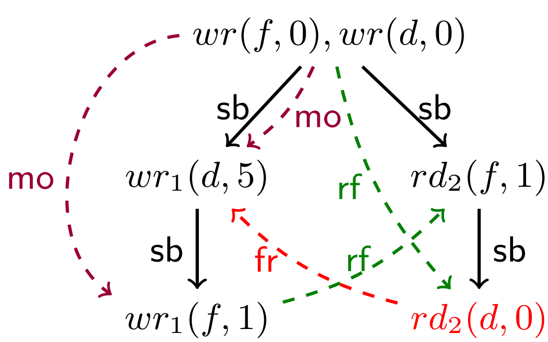
Axiomatic C11 semantics

Example (Message Passing).

Init: `f := 0;  d := 0;`

thread 1	**thread 2**
`d := 5;`	**do** `r1 ← f`
`f := 1;`	**until** `r1 = 1;`
	`r2 ← d;`

In C11, `r2` can have a final value `0` — the execution below is allowed

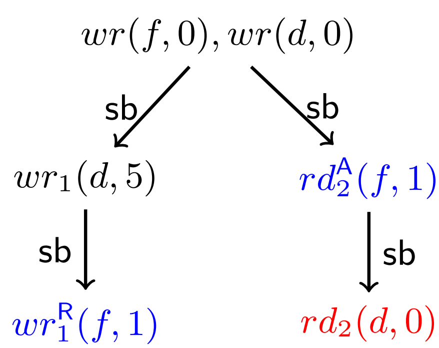


Corrected Message Passing.

Init: `f := 0;  d := 0;`

thread 1	**thread 2**
`d := 5;`	**do** `r1` $\leftarrow^{\mathtt{A}}$ `f`
`f` $:=^{\mathtt{R}}$ `1;`	**until** `r1 = 1;`
	`r2 ← d;`

The following execution is now disallowed



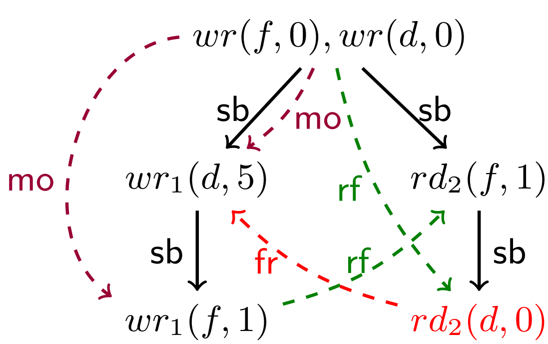
Axiomatic C11 semantics

Example (Message Passing).

Init: `f := 0;  d := 0;`

thread 1	**thread 2**
`d := 5;`	**do** `r1 ← f`
`f := 1;`	**until** `r1 = 1;`
	`r2 ← d;`

In C11, `r2` can have a final value `0` — the execution below is allowed

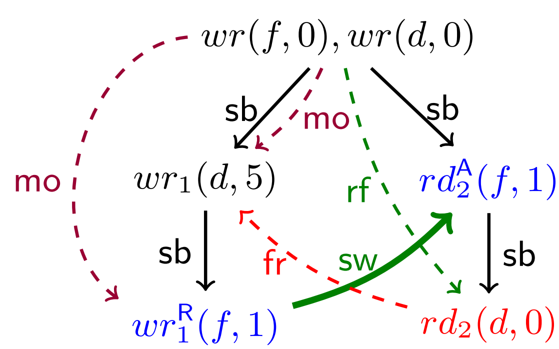


Corrected Message Passing.

Init: `f := 0;  d := 0;`

thread 1	**thread 2**
`d := 5;`	**do** `r1` $\xleftarrow{\mathtt{A}}$ `f`
`f` $:=^{\mathtt{R}}$ `1;`	**until** `r1 = 1;`
	`r2 ← d;`

The following execution is now disallowed



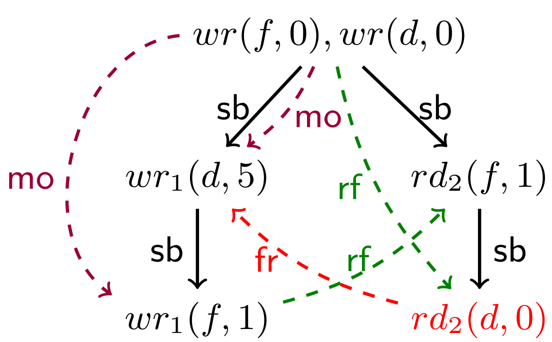
Axiomatic C11 semantics

Example (Message Passing).

Init: $\mathtt{f} := 0;\ \ \mathtt{d} := 0;$

thread 1	||	**thread 2**
$\mathtt{d} := 5;$		**do** $\mathtt{r1} \leftarrow \mathtt{f}$
$\mathtt{f} := 1;$		**until** $\mathtt{r1} = 1;$
		$\mathtt{r2} \leftarrow \mathtt{d};$

In C11, `r2` can have a final value `0` — the execution below is allowed

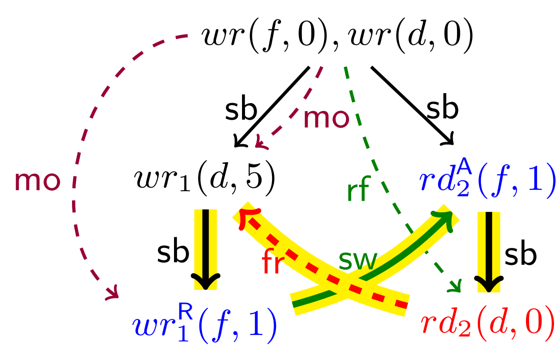


Corrected Message Passing.

Init: $\mathtt{f} := 0;\ \ \mathtt{d} := 0;$

thread 1	||	**thread 2**
$\mathtt{d} := 5;$		**do** $\mathtt{r1} \leftarrow^{\mathtt{A}} \mathtt{f}$
$\mathtt{f} :=^{\mathtt{R}} 1;$		**until** $\mathtt{r1} = 1;$
		$\mathtt{r2} \leftarrow \mathtt{d};$

The following execution is now disallowed



What about verification?

- Axiomatic semantics useful for certain forms of verification, e.g., SMT, BMC, ...
- But how can we link with existing works
  — Hoare Logic, Owicki/Gries, Rely/Guarantee ?

We need an operational semantics for C11

C11 Operational Semantics

Point of departure

- Start with operational semantics by Doherty et al (2019)
  — proved sound and complete with respect to RC11
- For the experts: restrict attention to a fragment of C11
  - All operations are either *relaxed*, *write-releasing*, or *read-acquiring*
  - Do **not** model *fences* or *release-sequences*
  - Assume *no-thin-air*, i.e., $\mathsf{sb} \cup \mathsf{rf}$ acyclic
- Strategy: construct valid C11 graphs by stepping through program in thread order (without consulting axioms)
- Brings us back to well understood (programmer friendly) notion

  Concurrency = Interleaving of threads
- What's different?
  - More non-determinism in choosing the next C11 state
  - Both reads **and** writes may change state configuration

Observing a C11 state

Key point.

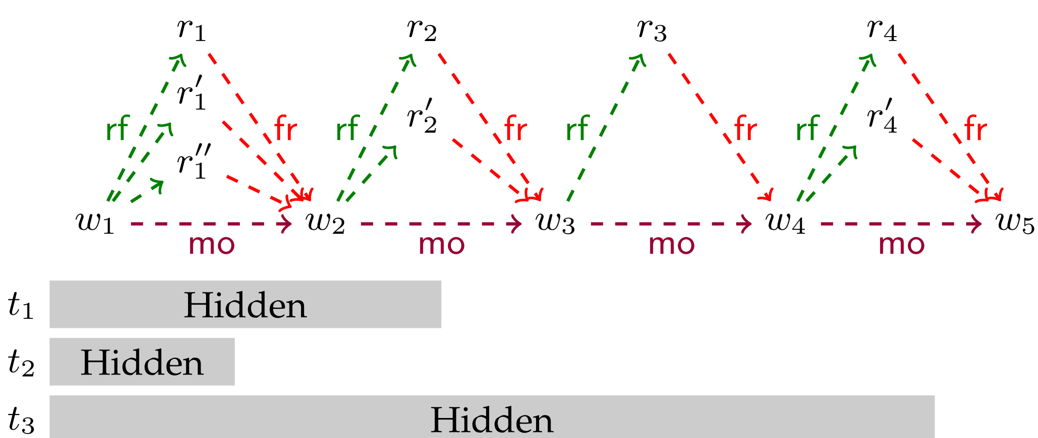
- Each thread has its own observable set of writes
- Observable writes can be determined from the current C11 state

Observing a C11 state

Key point.

- Each thread has its own observable set of writes
- Observable writes can be determined from the current C11 state

Example. Restricting $\mathsf{mo} \cup \mathsf{rf} \cup \mathsf{fr}$ to a single variable, we have:



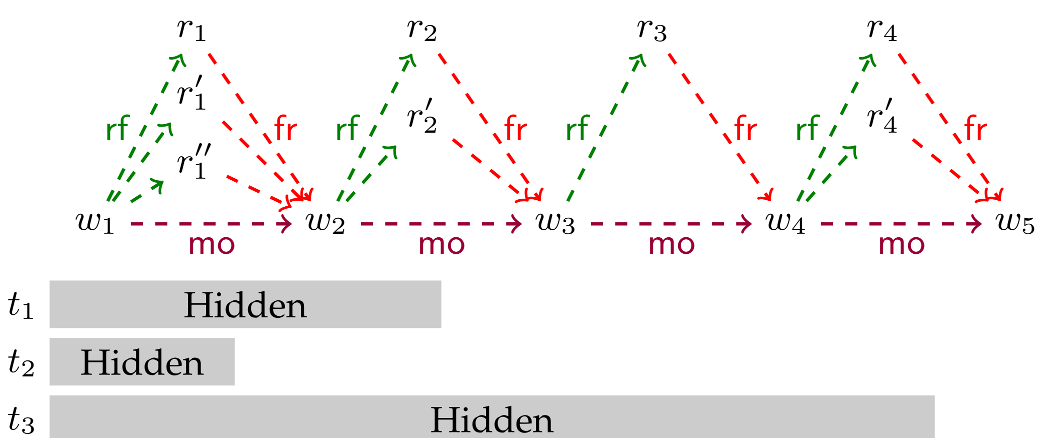
- Thread t_1 can observe w_3, w_4, w_5
- Thread t_2 can observe w_2, w_3, w_4, w_5
- Thread t_3 can observe w_5

Observing a C11 state

Key point.

- Each thread has its own observable set of writes
- Observable writes can be determined from the current C11 state

Example. Restricting $\mathsf{mo} \cup \mathsf{rf} \cup \mathsf{fr}$ to a single variable, we have:



- Thread t_1 can observe w_3, w_4, w_5
- Thread t_2 can observe w_2, w_3, w_4, w_5
- Thread t_3 can observe w_5

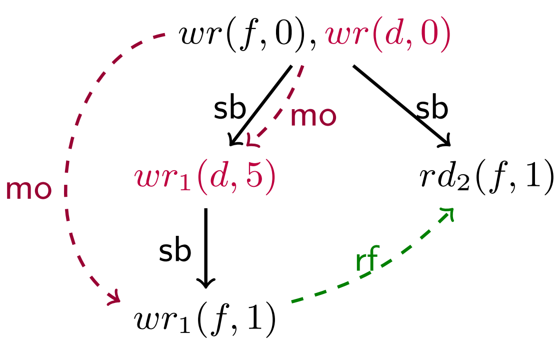
Observable set changes as threads interact with the C11 state

Message passing with “bad” transition

Init: `f` $:=$ `0`; `d` $:=$ `0`

thread 1	**thread 2**
`d` $:=$ `5`;	**do** `r1` $\leftarrow$ `f` **until** `r1` $=$ `1`;
`f` $:=$ `1`;	`r2` $\leftarrow$ `d`;

Pre-state

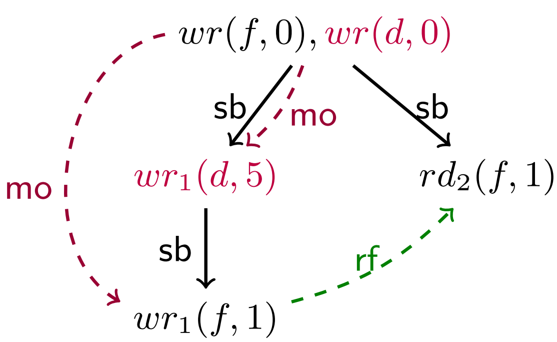


Message passing with “bad” transition

Init: $\texttt{f} := 0; \texttt{d} := 0$

thread 1	**thread 2**
$\texttt{d} := 5;$	**do** $\texttt{r1} \leftarrow \texttt{f}$ **until** $\texttt{r1} = 1;$
$\texttt{f} := 1;$	$\texttt{r2} \leftarrow \texttt{d};$

Pre-state



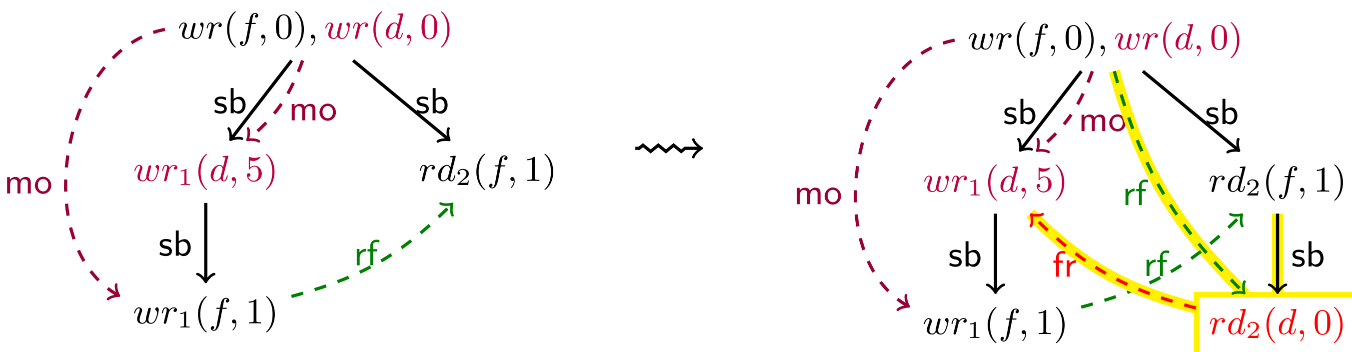
Thread 2 can observe both writes to d

Message passing with “bad” transition

Init: $\texttt{f} := 0; \texttt{d} := 0$

thread 1	**thread 2**
`d := 5;`	**do** `r1 ← f` **until** `r1 = 1;`
`f := 1;`	`r2 ← d;`

Pre-state $\rightsquigarrow$ Possible post-state



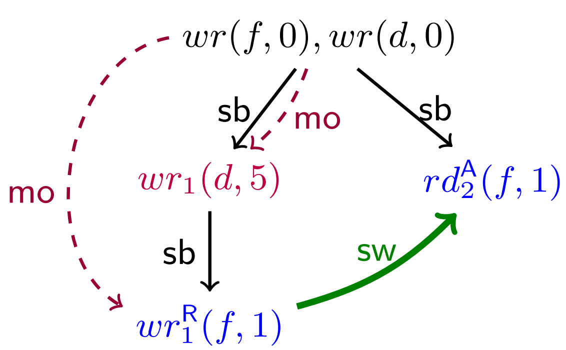
Thread 2 can observe both writes to d

“Bad” transition with read from $wr(d,0)$ is possible

Message passing with release/acquire annotations

Init: `f` $:=$ `0`; `d` $:=$ `0`		
thread 1	$\|$	**thread 2**
`d` $:=$ `5`;		**do** `r1` $\xleftarrow{\mathtt{A}}$ `1` **until** `r1` $=$ `1`;
`f` $:=^{\mathtt{R}}$ `1`;		`r2` $:=$ `d`;

Pre-state

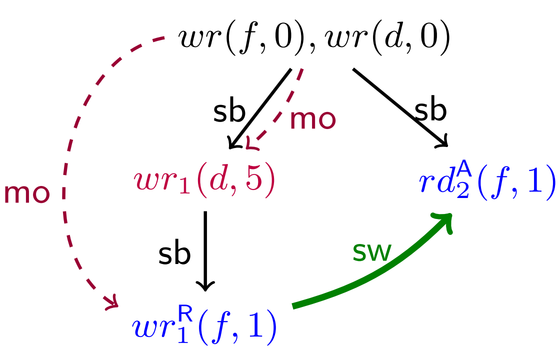


Message passing with release/acquire annotations

Init: $\mathtt{f} := 0; \mathtt{d} := 0$

thread 1	**thread 2**
$\mathtt{d} := 5;$	**do** $\mathtt{r1} \xleftarrow{\mathtt{A}} 1$ **until** $\mathtt{r1} = 1;$
$\mathtt{f} :=^{\mathtt{R}} 1;$	$\mathtt{r2} := \mathtt{d};$

Pre-state



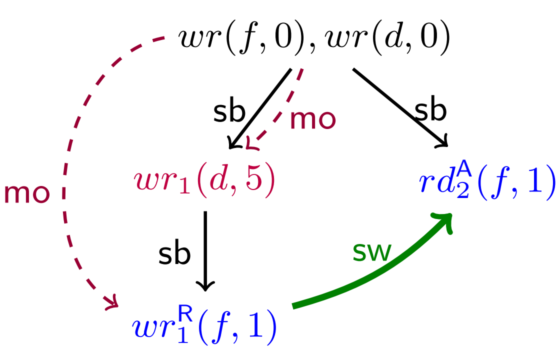
Thread 2 can only observe $wr_1(d,5)$

Message passing with release/acquire annotations

Init: $\texttt{f} := 0; \texttt{d} := 0$

thread 1	‖	thread 2
$\texttt{d} := 5;$		**do** $\texttt{r1} \xleftarrow{\texttt{A}} 1$ **until** $\texttt{r1} = 1;$
$\texttt{f} :=^{\texttt{R}} 1;$		$\texttt{r2} := \texttt{d};$

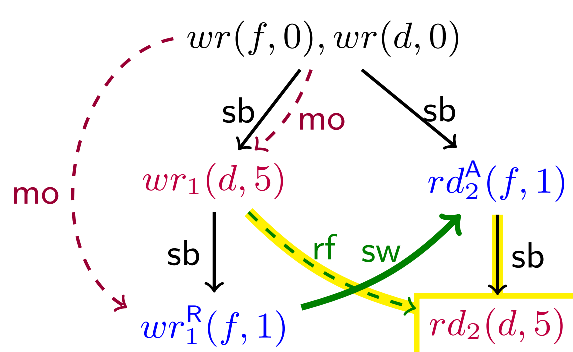
Pre-state



Thread 2 can only observe $wr_1(d,5)$

$\rightsquigarrow$

Only possible post-state



Only the "good" transition is available

C11 Owicki-Gries Proofs in Isabelle

Proof outline for message passing

Init: $\texttt{d} := 0; \texttt{f} := 0;$

$\texttt{d} := 5;$	**do** $\texttt{r1} \xleftarrow{\texttt{A}} \texttt{f}$ **until** $\texttt{r1} = 1;$
$\texttt{f} :=^{\texttt{R}} 1;$	$\texttt{r2} \leftarrow \texttt{d};$

Proof outline for message passing

Init: `d` $:=$ `0`; `f` $:=$ `0`;

$$\{d =_1 0 \wedge d =_2 0 \wedge f =_1 0 \wedge f =_2 0\}$$

$\{\neg(f \approx_2 1) \wedge d =_1 0\}$	$\|$	$\{[f = 1]_2(d = 5)\}$
`d` $:=$ `5`;	$\|$	**do** `r1` $\overset{\mathtt{A}}{\leftarrow}$ `f` **until** `r1` $=$ `1`;
$\{\neg(f \approx_2 1) \wedge d =_1 5\}$	$\|$	$\{d =_2 5\}$
`f` $:=^{\mathtt{R}}$ `1`;	$\|$	`r2` $\leftarrow$ `d`;
$\{true\}$	$\|$	$\{r2 = 5\}$

$$\{r2 = 5\}$$

Proof outline for message passing

Init: `d` := `0`; `f` := `0`;

$\{d =_1 0 \wedge d =_2 0 \wedge f =_1 0 \wedge f =_2 0\}$

$\{\neg(f \approx_2 1) \wedge d =_1 0\}$	$\{[f = 1]_2(d = 5)\}$
`d` := `5`;	**do** `r1` $\xleftarrow{\texttt{A}}$ `f` **until** `r1` = `1`;
$\{\neg(f \approx_2 1) \wedge d =_1 5\}$	$\{d =_2 5\}$
`f` $:=^{\texttt{R}}$ `1`;	`r2` $\leftarrow$ `d`;
$\{true\}$	$\{r2 = 5\}$

$\{r2 = 5\}$

Recall the Owicki-Gries technique:

$$\frac{\vdash \{P_1\}C_1\{Q_1\} \| \{P_2\}C_2\{Q_2\} \qquad P \Rightarrow P_1 \wedge P_2 \qquad Q_1 \wedge Q_2 \Rightarrow Q}{\vdash \{P\}\ (\{P_1\}C_1\{Q_1\} \| \{P_2\}C_2\{Q_2\})\ \{Q\}}$$

$$\frac{\begin{array}{ll} \vdash \{P_1\}C_1\{Q_1\} & \{P_1\}C_1\{Q_1\} \text{ is interference free wrt } C_2 \\ \vdash \{P_2\}C_2\{Q_2\} & \{P_2\}C_2\{Q_2\} \text{ is interference free wrt } C_1 \end{array}}{\vdash \{P_1\}C_1\{Q_1\} \| \{P_2\}C_2\{Q_2\}}$$

Proof outline for message passing

Init: $\texttt{d} := 0; \texttt{f} := 0;$

$$\{d =_1 0 \wedge d =_2 0 \wedge f =_1 0 \wedge f =_2 0\}$$

$$\begin{array}{l||l}
\{\neg(f \approx_2 1) \wedge d =_1 0\} & \{[f = 1]_2(d = 5)\} \\
\texttt{d} := 5; & \textbf{do}\ \texttt{r1} \leftarrow^{\texttt{A}} \texttt{f}\ \textbf{until}\ \texttt{r1} = 1; \\
\{\neg(f \approx_2 1) \wedge d =_1 5\} & \{d =_2 5\} \\
\texttt{f} :=^{\texttt{R}} 1; & \texttt{r2} \leftarrow \texttt{d}; \\
\{true\} & \{r2 = 5\}
\end{array}$$

$$\{r2 = 5\}$$

- The C11 state is a special implicit variable in the program
- Assertions are predicates over program states (including the C11 states)

Proof outline for message passing

$$\textbf{Init: } \texttt{d} := 0; \texttt{f} := 0;$$

$$\{d =_1 0 \wedge d =_2 0 \wedge f =_1 0 \wedge f =_2 0\}$$

$$\begin{array}{l||l}
\{\neg(f \approx_2 1) \wedge d =_1 0\} & \{[f = 1]_2(d = 5)\} \\
\texttt{d} := 5; & \textbf{do } \texttt{r1} \xleftarrow{\texttt{A}} \texttt{f} \textbf{ until } \texttt{r1} = 1; \\
\{\neg(f \approx_2 1) \wedge d =_1 5\} & \{d =_2 5\} \\
\texttt{f} :=^{\texttt{R}} 1; & \texttt{r2} \leftarrow \texttt{d}; \\
\{true\} & \{r2 = 5\}
\end{array}$$

$$\{r2 = 5\}$$

- The C11 state is a special implicit variable in the program
- Assertions are predicates over program states (including the C11 states)
- We define special assertions on C11 state:

$x \approx_t v$	$\leftrightarrow$	Thread t possibly observes value v for x
$x =_t v$	$\leftrightarrow$	Thread t definitely observes value v for x
$[x = u]_t(y = v)$	$\leftrightarrow$	If thread t observes $x = u$ then it will definitely observe $y = v$

Hoare-style axioms

- Rules for compound statements are exactly as in Hoare logic
- But have a new set of basic axioms for (atomic) reads and writes (76 at last count), e.g.,

$$\text{d\_obs\_WrX\_set}\ \frac{}{\{x =_t u\}\ [\mathtt{x} := \mathtt{v}]_{\mathtt{t}}\ \{x =_t v\}}$$

$$\text{not\_pobs\_RdA\_pres}\ \frac{}{\{\neg(x \approx_t u)\}\ [\mathtt{v} \leftarrow^{\mathtt{A}} \mathtt{y}]_{\mathtt{t'}}\ \{\neg(x \approx_t u)\}}$$

$$\text{c\_obs\_WrR\_pres}\ \frac{z \neq y \qquad z \neq x \qquad x \neq y}{\{[x = u]_t(y = v)\}\ [\mathtt{z} :=^{\mathtt{R}} \mathtt{w}]_{\mathtt{t}}\ \{[x = u]_t(y = v)\}}$$

Hoare-style axioms

- Rules for compound statements are exactly as in Hoare logic
- But have a new set of basic axioms for (atomic) reads and writes (76 at last count), e.g.,

$$\text{d\_obs\_WrX\_set}\ \frac{}{\{x =_t u\}\ [\mathtt{x} := \mathtt{v}]_{\mathtt{t}}\ \{x =_t v\}}$$

$$\text{not\_pobs\_RdA\_pres}\ \frac{}{\{\neg(x \approx_t u)\}\ [\mathtt{v} \leftarrow^{\mathtt{A}} \mathtt{y}]_{\mathtt{t}'}\ \{\neg(x \approx_t u)\}}$$

$$\text{c\_obs\_WrR\_pres}\ \frac{z \neq y \qquad z \neq x \qquad x \neq y}{\{[x = u]_t(y = v)\}\ [\mathtt{z} :=^{\mathtt{R}} \mathtt{w}]_{\mathtt{t}}\ \{[x = u]_t(y = v)\}}$$

- All basic axioms verified in Isabelle, e.g.,

corollary `d_obs_RdX_other:`
  `"wfs` $\sigma \Longrightarrow$ `x` $\neq$ `y` $\Longrightarrow$
    `[x` $=_t$ `u]` $\sigma \Longrightarrow \sigma$ `[v` $\leftarrow$ `y]`$_t$ σ`'` $\Longrightarrow$ `[x` $=_t$ `u]` σ`'"`
  by `(metis RdX_def avar.simps(1) d_obs_other)`

C11 Owicki-Gries in Isabelle

- Owicki-Gries theory is included in standard Isabelle distribution (Nieto and Nipkow, 2002)

C11 Owicki-Gries in Isabelle

- Owicki-Gries theory is included in standard Isabelle distribution (Nieto and Nipkow, 2002)
- We have extended Nieto-Nipkow's `WHILE` language with relaxed / release-acquire statements
- C11 state is embedded in the standard state, e.g., for message passing

```
record MP =
d :: V
f :: V
r1 :: V
r2 :: V
σ :: C11_state
```

- C11 states updated w.r.t. our operational semantics

Proof of message passing in Isabelle

```
lemma MessagePassing:
 "‖- ⦃ (wfs ´σ ´f ´d) ∧ [ ´d =₁ 0 ] ´σ ∧ [ ´d =₂ 0 ] ´σ
                      ∧ [ ´f =₁ 0 ] ´σ ∧ [ ´f =₂ 0 ] ´σ ⦄
COBEGIN ⦃ (wfs ´σ ´f ´d)  ∧ ¬[ ´f ≈₂ 1 ] ´σ ∧ [ ´d =₁ 0 ] ´σ ⦄
        < ´d [ ´σ] :=₁ 5> ;;
        ⦃ (wfs ´σ ´f ´d)  ∧ ¬[ ´f ≈₂ 1 ] ´σ ∧ [ ´d =₁ 5 ] ´σ ⦄
        < ´f [ ´σ]^R :=₁ 1>
        ⦃ [ ´d =₁ 5] ´σ ⦄
‖
        ⦃ (wfs ´σ ´f ´d)  ∧ [ ´f = 1 ]₂⦇ ´d = 5 ⦈ ´σ ⦄
        DO ⦃ (wfs ´σ ´f ´d)  ∧ [ ´f = 1 ]₂⦇ ´d = 5 ⦈ ´σ ⦄
           < ´r1 [ ´σ]^A ←₂ ´f>
        UNTIL ´r1 = 1
        INV ⦃ (wfs ´σ ´f ´d) ∧ [ ´f = 1 ]₂⦇ ´d = 5 ⦈ ´σ
                             ∧ ( ´r1 = 1 ⟶ [ ´d =₂ 5 ] ´σ) ⦄
        OD;; ⦃ (wfs ´σ ´f ´d)  ∧ [ ´d =₂ 5 ] ´σ ⦄
        < ´r2 [ ´σ] ←₂ ´d>
        ⦃ ´r2 = 5 ⦄
COEND
⦃ ´r2 = 5 ⦄"
  apply oghoare
  apply auto
  using d_obs_diff_false zero_neq_numeral
  by blast+
```

Case study 2: Peterson's mutual exclusion

Init: $\texttt{flag}_1 := \texttt{false}; \texttt{flag}_2 := \texttt{false}; \texttt{turn} = 1$

thread 1	**thread 2**
$\texttt{flag}_1 := \texttt{true};$	$\texttt{flag}_2 := \texttt{true};$
$\textbf{swap}^{\texttt{RA}}(\texttt{turn}, 2);$	$\textbf{swap}^{\texttt{RA}}(\texttt{turn}, 1);$
do	**do**
$\quad \texttt{r1} \leftarrow^{\texttt{A}} \texttt{flag}_2;$	$\quad \texttt{r3} \leftarrow^{\texttt{A}} \texttt{flag}_1;$
$\quad \texttt{r2} \leftarrow \texttt{turn};$	$\quad \texttt{r4} \leftarrow \texttt{turn};$
until $\neg\texttt{r1} \vee \texttt{r2} = 1;$	**until** $\neg\texttt{r3} \vee \texttt{r4} = 2;$
$//\texttt{CS1};$	$//\texttt{CS2};$
$\texttt{flag}_1 :=^{\texttt{R}} \texttt{false};$	$\texttt{flag}_2 :=^{\texttt{R}} \texttt{false};$

- Encoded and verified in Isabelle
- Requires new types of assertions describing the C11 state
- Same auxiliary variable as proof in sequentially consistent setting (Apt and Olderog, 2009)
- However, proof requires more work beyond `oghoare` and `auto`

  — currently investigating ways to speed this up

Conclusions

- Operational semantics by Doherty et al (2019) makes deductive verification possible for (a realistic fragment of) C11
- Verification based on well-understood Owicki-Gries theory
- Straightforward extension of Nieto and Nipkow's mechanisations of Owicki-Gries in Isabelle
- Paper describing these works is forthcoming

- Currently investigating links with distributed correctness (with Philippa Gardner)

- Any questions, please e-mail: b.dongol@surrey.ac.uk