Modular Termination Verification for Non-blocking Concurrency

Julian Sutherland

Joint work with: Pedro da Rocha Pinto, Thomas Dinsdale-Young, Philippa Gardner

July, 2015

Module Abstractions

Given the following modules.

Counter   Stack   Queue

- What is the right specification?
  - Sufficiently strong for clients to be able to use it constructively.
  - Sufficiently weak for any “reasonable” implementations of the module to satisfy it.
- How much can we abstract?
- Can we prove termination?

Example of a Client of a Counter Module

$$
\begin{array}{c}
\texttt{x} := \texttt{makeCounter}(); \\
\begin{array}{l||l}
\texttt{n} := \texttt{random}(); & \texttt{m} := \texttt{random}(); \\
\texttt{i} := 0; & \texttt{j} := 0; \\
\texttt{while}\ (\texttt{i} < \texttt{n})\ \{ & \texttt{while}\ (\texttt{j} < \texttt{m})\ \{ \\
\quad \texttt{incr}(\texttt{x}); & \quad \texttt{incr}(\texttt{x}); \\
\quad \texttt{i} := \texttt{i} + 1; & \quad \texttt{j} := \texttt{j} + 1; \\
\} & \}
\end{array}
\end{array}
$$

Counter Module Operations: Partial Correctness

$$\vdash \left\{\mathsf{emp}\right\}\ \texttt{makeCounter}()\ \left\{\mathsf{C}(\mathsf{ret}, 0)\right\}$$

$$\vdash \mathbb{V} n \in \mathbb{N}.\ \left\langle \mathsf{C}(\texttt{x}, n) \right\rangle\ \texttt{read}(\texttt{x})\ \left\langle \mathsf{C}(\texttt{x}, n) \wedge \mathsf{ret} = n \right\rangle$$

$$\vdash \mathbb{V} n \in \mathbb{N}.\ \left\langle \mathsf{C}(\texttt{x}, n) \right\rangle\ \texttt{incr}(\texttt{x})\ \left\langle \mathsf{C}(\texttt{x}, n+1) \right\rangle$$

Spin Counter: Increment

$$\vdash \forall n \in \mathbb{N}.\, \big\langle \mathsf{C}(\mathtt{x}, n) \big\rangle\ \mathtt{incr}(\mathtt{x})\ \big\langle \mathsf{C}(\mathtt{x}, n+1) \big\rangle$$

```
function incr(x) {
   b := 0;
   while (b = 0) {
      v := [x];
      b := CAS(x, v, v + 1);
   }
}
```

Counter Module Operations : Total Correctness

$$\forall \alpha. \vdash_\tau \{\mathsf{emp}\}\ \mathtt{makeCounter}()\ \{\mathsf{C}(\mathsf{ret}, 0, \alpha)\}$$

$$\vdash_\tau \mathbb{V} n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \right\rangle\ \mathtt{read}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \land \mathsf{ret} = n \right\rangle$$

$$\forall \beta. \vdash_\tau \mathbb{V} n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \land \alpha > \beta(\alpha) \right\rangle\ \mathtt{incr}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n+1, \beta(\alpha)) \right\rangle$$

Counter Module Operations : Total Correctness

$$\forall \alpha. \vdash_\tau \{\mathsf{emp}\}\ \texttt{makeCounter}()\ \{\mathsf{C}(\mathsf{ret}, 0, \alpha)\}$$

$$\vdash_\tau \mathbb{\forall} n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \right\rangle\ \texttt{read}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \wedge \mathsf{ret} = n \right\rangle$$

$$\forall \beta. \vdash_\tau \mathbb{\forall} n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \wedge \alpha > \beta(\alpha) \right\rangle\ \texttt{incr}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n+1, \beta(\alpha)) \right\rangle$$

$$\forall \alpha > \beta. \mathsf{C}(x, n, \alpha) \implies \mathsf{C}(x, n, \beta)$$

Counter Module Operations : Total Correctness

$$\forall \alpha. \vdash_\tau \{\mathsf{emp}\}\ \texttt{makeCounter}()\ \{\mathsf{C}(\mathsf{ret}, 0, \alpha)\}$$

$$\vdash_\tau \forall\!\!\!\forall n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \right\rangle\ \texttt{read}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \wedge \mathsf{ret} = n \right\rangle$$

$$\forall \beta. \vdash_\tau \forall\!\!\!\forall n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \wedge \alpha > \beta(\alpha) \right\rangle\ \texttt{incr}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n+1, \beta(\alpha)) \right\rangle$$

$$\forall \alpha > \beta.\, \mathsf{C}(x, n, \alpha) \implies \mathsf{C}(x, n, \beta)$$

Non-impedance relationship in the counter module:

Counter Module Operations : Total Correctness

$$\forall \alpha. \vdash_\tau \{\mathsf{emp}\}\ \texttt{makeCounter}()\ \{\mathsf{C}(\mathsf{ret}, 0, \alpha)\}$$

$$\vdash_\tau \forall\!\!\!\forall n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \right\rangle\ \texttt{read}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \land \mathsf{ret} = n \right\rangle$$

$$\forall \beta. \vdash_\tau \forall\!\!\!\forall n \in \mathbb{N}, \alpha. \left\langle \mathsf{C}(\mathtt{x}, n, \alpha) \land \alpha > \beta(\alpha) \right\rangle\ \texttt{incr}(\mathtt{x})\ \left\langle \mathsf{C}(\mathtt{x}, n+1, \beta(\alpha)) \right\rangle$$

$$\forall \alpha > \beta. \mathsf{C}(x, n, \alpha) \implies \mathsf{C}(x, n, \beta)$$

Non-impedance relationship in the counter module:

Total Correctness for Loops

$$\frac{\forall \gamma \leq \alpha. \vdash_\tau \{p(\gamma) \land \mathbb{B}\}\ \mathbb{C}\ \{\exists \beta. p(\beta) \land \beta < \gamma\}}{\vdash_\tau \{p(\alpha)\}\ \texttt{while}\ (\mathbb{B})\ \mathbb{C}\ \{\exists \beta. p(\beta) \land \neg \mathbb{B} \land \beta \leq \alpha\}}$$

Example of a Client of a Counter Module

$$\texttt{x} := \texttt{makeCounter}();$$

$\texttt{n} := \texttt{random}();$	$\texttt{m} := \texttt{random}();$
$\texttt{i} := 0;$	$\texttt{j} := 0;$
$\texttt{while}\ (\texttt{i} < \texttt{n})\ \{$	$\texttt{while}\ (\texttt{j} < \texttt{m})\ \{$
$\quad \texttt{incr}(\texttt{x});$	$\quad \texttt{incr}(\texttt{x});$
$\quad \texttt{i} := \texttt{i} + 1;$	$\quad \texttt{j} := \texttt{j} + 1;$
$\}$	$\}$

Example of a Client of a Counter Module

$$\{ \ \mathsf{emp} \ \}$$

```
x := makeCounter();
n := random();        ||  m := random();
i := 0;               ||  j := 0;
while (i < n) {       ||  while (j < m) {
  incr(x);            ||    incr(x);
  i := i + 1;         ||    j := j + 1;
}                     ||  }
```

$$\{ \ \mathsf{C}(\mathtt{x}, \mathtt{n} + \mathtt{m}, 0) \ \}$$

Building abstraction

$$I(\mathbf{CClient}_r(x, n)) \triangleq \exists \alpha.\, \mathsf{C}(x, n, \alpha) * [\textsc{Total}(n, \alpha)]_r$$

$$I(\mathbf{CClient}_r(x, \circ)) \triangleq \mathsf{True}$$

Building abstraction

$$I(\mathbf{CClient}_r(x, n)) \triangleq \exists \alpha.\, \mathsf{C}(x, n, \alpha) * [\textsc{Total}(n, \alpha)]_r$$
$$I(\mathbf{CClient}_r(x, \circ)) \triangleq \mathsf{True}$$

$$\textsc{Inc}(x, n + m, \alpha \oplus \beta, \pi_1 + \pi_2) = \textsc{Inc}(x, n, \alpha, \pi_1) \bullet \textsc{Inc}(x, m, \beta, \pi_2)$$
$$\textsc{Total}(n, \alpha) \bullet \textsc{Inc}(m, \beta, 1) \text{ defined } \implies n = m \wedge \alpha = \beta$$

Building abstraction

$$I(\mathbf{CClient}_r(x, n)) \triangleq \exists \alpha.\, \mathsf{C}(x, n, \alpha) * [\textsc{Total}(n, \alpha)]_r$$
$$I(\mathbf{CClient}_r(x, \circ)) \triangleq \mathsf{True}$$

$$\textsc{Inc}(x, n+m, \alpha \oplus \beta, \pi_1 + \pi_2) = \textsc{Inc}(x, n, \alpha, \pi_1) \bullet \textsc{Inc}(x, m, \beta, \pi_2)$$
$$\textsc{Total}(n, \alpha) \bullet \textsc{Inc}(m, \beta, 1) \text{ defined} \implies n = m \wedge \alpha = \beta$$

$$\textsc{Inc}(m, \gamma, \pi) : n \rightsquigarrow n+1 \qquad \textsc{Inc}(m, \gamma, 1) : n \rightsquigarrow \circ$$

Building abstraction

$$I(\mathbf{CClient}_r(x, n)) \triangleq \exists\alpha.\, \mathsf{C}(x, n, \alpha) * [\textsc{Total}(n, \alpha)]_r$$

$$I(\mathbf{CClient}_r(x, \circ)) \triangleq \mathsf{True}$$

$$\textsc{Inc}(x, n + m, \alpha \oplus \beta, \pi_1 + \pi_2) = \textsc{Inc}(x, n, \alpha, \pi_1) \bullet \textsc{Inc}(x, m, \beta, \pi_2)$$

$$\textsc{Total}(n, \alpha) \bullet \textsc{Inc}(m, \beta, 1) \text{ defined } \implies n = m \wedge \alpha = \beta$$

$$\textsc{Inc}(m, \gamma, \pi) : n \rightsquigarrow n + 1 \qquad \textsc{Inc}(m, \gamma, 1) : n \rightsquigarrow \circ$$

Proving the Client

$$\{\ \mathsf{emp}\ \}$$
$$\mathtt{x} := \mathtt{makeCounter}();$$
$$\{\ \mathsf{C}(\mathrm{x}, 0, \omega \oplus \omega)\ \}$$

$$\vdots$$

Proving the Client

$$\{\ \mathsf{emp}\ \}$$
$$\texttt{x} := \texttt{makeCounter}();$$
$$\{\ \mathsf{C}(\mathrm{x}, 0, \omega \oplus \omega)\ \}$$
$$\{\ \mathbf{CClient}(\mathrm{x}, 0) * [\textsc{Inc}(0, \omega \oplus \omega, 1)]\ \}$$
$$\{\ \exists v.\, \mathbf{CClient}(\mathrm{x}, v) * [\textsc{Inc}(0, \omega, \tfrac{1}{2})] \wedge 0 \leq v\ \} \parallel \ldots$$
$$\vdots$$

Proving the client

$\{\ \exists v.\, \mathbf{CClient}(\mathbf{x}, v) * [\textsc{Inc}(0, \omega, \frac{1}{2})] \wedge 0 \leq v\ \}$

```
n := random();
i := 0;

while (i < n) {



  incr(x);
  i := i + 1;


}
```

$\ldots$

Proving the client

$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(0, \omega, \tfrac{1}{2})] \wedge 0 \le v\ \}$

```
n := random();
i := 0;

while (i < n) {



   incr(x);
   i := i + 1;


}
```

$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n}, 0, \tfrac{1}{2})]\ \}$

…

Proving the client

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(0, \omega, \tfrac{1}{2})] \wedge 0 \leq v\ \}$$

$\mathtt{n} := \mathtt{random}();$

$\mathtt{i} := 0;$

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{i}, \mathtt{n}, \tfrac{1}{2})] \wedge 0 \leq v \wedge \mathtt{i} = 0\ \}$$

$\mathtt{while}\ (\mathtt{i} < \mathtt{n})\ \{$

$\quad \mathtt{incr}(\mathtt{x});$

$\quad \mathtt{i} := \mathtt{i} + 1;$

$\}$

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n}, 0, \tfrac{1}{2})]\ \}$$

$\parallel$ $\ldots$

Proving the client

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(0, \omega, \tfrac{1}{2})] \land 0 \leq v\ \}$$

```
n := random();
i := 0;
```

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{i}, \mathtt{n}, \tfrac{1}{2})] \land 0 \leq v \land \mathtt{i} = 0\ \}$$

```
while (i < n) {
```

$\forall \beta.$

$$\left\{ \begin{array}{l} \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{i}, \beta, \tfrac{1}{2})] \land \mathtt{i} \leq v \land \mathtt{i} \leq \mathtt{n} \\ \land \beta = \mathtt{n} - \mathtt{i} \end{array} \right\}$$

```
  incr(x);
  i := i + 1;
}
```

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n}, 0, \tfrac{1}{2})]\ \}$$

$\|$ …

Proving the client

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(0, \omega, \tfrac{1}{2})] \land 0 \le v\ \}$$

$\mathtt{n} := \mathtt{random}();$

$\mathtt{i} := 0;$

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{i}, \mathtt{n}, \tfrac{1}{2})] \land 0 \le v \land \mathtt{i} = 0\ \}$$

$\mathtt{while}\ (\mathtt{i} < \mathtt{n})\ \{$

$\quad \forall \beta.$

$$\left\{ \begin{array}{l} \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{i}, \beta, \tfrac{1}{2})] \land \mathtt{i} \le v \land \mathtt{i} \le \mathtt{n} \\ \land\, \beta = \mathtt{n} - \mathtt{i} \end{array} \right\}$$

$\quad \mathtt{incr}(\mathtt{x});$

$\quad \mathtt{i} := \mathtt{i} + 1;$

$$\left\{ \begin{array}{l} \exists \delta, v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{i}, \delta, \tfrac{1}{2})] \land \mathtt{i} \le v \land \mathtt{i} \le \mathtt{n} \\ \land\, \delta = \mathtt{n} - \mathtt{i} \land \delta < \beta \end{array} \right\}$$

$\}$

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n}, 0, \tfrac{1}{2})]\ \}$$

$\Big\|\ \ldots$

Proving the client

$$\{\ \mathsf{emp}\ \}$$
$$\mathtt{x} := \mathtt{makeCounter}();$$
$$\{\ \mathsf{C}(\mathrm{x}, 0, \omega \oplus \omega)\ \}$$
$$\{\ \mathbf{CClient}(\mathrm{x}, 0) * [\textsc{Inc}(0, \omega \oplus \omega, 1)]\ \}$$
$$\left.\begin{array}{l} \{\ \exists v.\, \mathbf{CClient}(\mathrm{x}, v) * [\textsc{Inc}(0, \omega, \tfrac{1}{2})] \land 0 \leq v\ \} \\ \dots \\ \{\ \exists v.\, \mathbf{CClient}(\mathrm{x}, v) * [\textsc{Inc}(\mathrm{n}, 0, \tfrac{1}{2})]\ \} \end{array}\right\| \dots$$

Proving the client

$$\{\ \mathsf{emp}\ \}$$
$$\mathtt{x} := \mathtt{makeCounter}();$$
$$\{\ \mathsf{C}(\mathtt{x}, 0, \omega \oplus \omega)\ \}$$
$$\{\ \mathbf{CClient}(\mathtt{x}, 0) * [\textsc{Inc}(0, \omega \oplus \omega, 1)]\ \}$$

$$\begin{array}{l} \{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(0, \omega, \frac{1}{2})] \wedge 0 \leq v\ \} \\ \dots \\ \{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n}, 0, \frac{1}{2})]\ \} \end{array} \Bigg\| \dots$$

$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n}, 0, \tfrac{1}{2})] * [\textsc{Inc}(\mathtt{m}, 0, \tfrac{1}{2})]\ \}$$
$$\{\ \exists v.\, \mathbf{CClient}(\mathtt{x}, v) * [\textsc{Inc}(\mathtt{n} + \mathtt{m}, 0, 1)]\ \}$$
$$\{\ \mathsf{C}(\mathtt{x}, \mathtt{n} + \mathtt{m}, 0)\ \}$$

What to take home

- Ordinals can be used to bound interference in a module.
- Generally, termination is not guaranteed unless we restrict the environment.
- Atomic triples allow us to restrict the environment.
- The client can choose how to decrease the ordinals.
- Non-impedance seems to be a useful way of specifying blocking within a module.

Conclusions

- Introduced atomic triples with total correctness interpretation.
- Introduced Total-TaDA, that extends TaDA for total correctness.
- Modular approach: clients and implementations are verified independently.
- Examples: Counters, Stacks, Queues, Sets, Graphs

Current/Future work

- Extend logic (and specifications?) to blocking algorithms
- Non-terminating behaviour