Verifying TSO Programs

Bart Jacobs

DistriNet Research Group, Department of Computer Science,
KU Leuven, Belgium

Microsoft Research Cambridge Workshop on Verified Concurrent Programs: Theory, Tools, and Experiments
12 June 2013

A Racy Java Program

```
class Foo { static Object x; }

fork {
    Foo.x = new Bar();
}
println(Foo.x.getClass());
```

A Java Interpreter

```
struct class *c = /* Bar */;
struct object *o = allocate_object(c->size);
o->class = c;
// ...
struct object **field = /* &Foo.x */;
*field = o;

struct object **field = /* &Foo.x */;
struct object *v = *field;
struct class *vc = v->class;
printf("%s\n", vc->name);
```

Programming Language

$$
\begin{array}{rcl}
 & & g \in G \\
\mathit{Heaps} & = & G \to \mathbb{Z} \\
\delta \in \Delta & = & \mathit{Heaps} \to \mathit{Heaps} \\
c \in \mathit{Cmds} & ::= & \delta; c \mid c(g) \mid \textbf{done} \mid \textbf{fail} \\
\mathit{prog} & ::= & \textbf{cobegin}\ c \parallel \cdots \parallel c\ \textbf{coend}
\end{array}
$$

Small-Step Semantics

$$
\begin{array}{rcl}
\mathit{Buffers} & = & \Delta^* \\
\theta \in \mathit{ThreadConfigs} & = & \mathit{Buffers} \times \mathit{Cmds} \\
\gamma \in \mathit{Configs} & = & \mathit{Heaps} \times (\mathit{ThreadConfigs} \to_{\mathrm{fin}} \mathbb{N}) \\
\rightsquigarrow & \subseteq & \mathit{Configs} \times \mathit{Configs}
\end{array}
$$

Small-Step Semantics

Enqueue

$$\frac{}{(h, (\overline{\delta}, \delta; c) \uplus \Theta) \rightsquigarrow (h, (\overline{\delta} \cdot \delta, c) \uplus \Theta)}$$

Dequeue

$$\frac{}{(h, (\delta \cdot \overline{\delta}, c) \uplus \Theta) \rightsquigarrow (\delta(h), (\overline{\delta}, c) \uplus \Theta)}$$

Read

$$\frac{}{(h, (\overline{\delta}, c(g)) \uplus \Theta) \rightsquigarrow (h, (\overline{\delta}, c(\overline{\delta}(h)(g))) \uplus \Theta)}$$

Proof System

Assume $A \subseteq \wp(\mathit{Heap})$. Assume $\preceq$ a pre-order on A. α ranges over A.

$$
\begin{array}{l}
\mathsf{valid}(\alpha, \delta; c) = \\
\quad \exists f. \\
\qquad (\forall \alpha' \succeq \alpha.\ f(\alpha') \succeq \alpha') \wedge \\
\qquad (\forall \alpha' \succeq \alpha, \alpha'' \succeq \alpha'.\ f(\alpha'') \succeq f(\alpha')) \wedge \\
\qquad (\forall \alpha' \succeq \alpha, h \in \alpha'.\ \delta(h) \in f(\alpha')) \wedge \\
\qquad \mathsf{valid}(f(\alpha), c) \\
\mathsf{valid}(\alpha, c(g)) = \\
\quad \forall v.\ \exists \alpha'. \\
\qquad (\forall \alpha'' \succeq \alpha, h \in \alpha''.\ h(g) = v \Rightarrow \alpha'' \succeq \alpha') \wedge \\
\qquad \mathsf{valid}(\alpha', c(v)) \\
\mathsf{valid}(\alpha, \textbf{done}) = \mathsf{True} \\
\mathsf{valid}(\alpha, \textbf{fail}) = \mathsf{False}
\end{array}
$$

Proof System, with Local Knowledge

Assume all $L \subseteq \mathit{Heaps}$ preserved by updates of other threads.

$$
\begin{array}{l}
\mathsf{valid}(\alpha, L, \delta; c) = \\
\quad \exists f, L'. \\
\qquad (\forall \alpha' \succeq \alpha.\ f(\alpha') \succeq \alpha') \wedge \\
\qquad (\forall \alpha' \succeq \alpha, \alpha'' \succeq \alpha'.\ f(\alpha'') \succeq f(\alpha')) \wedge \\
\qquad (\forall \alpha' \succeq \alpha, h \in \alpha' \cap L.\ \delta(h) \in f(\alpha') \cap L') \wedge \\
\qquad \mathsf{valid}(f(\alpha), L', c) \\
\mathsf{valid}(\alpha, L, c(g)) = \\
\quad \forall v.\ \exists \alpha'. \\
\qquad (\forall \alpha'' \succeq \alpha, h \in \alpha'' \cap L.\ h(g) = v \Rightarrow \alpha'' \succeq \alpha') \wedge \\
\qquad \mathsf{valid}(\alpha', L, c(v)) \\
\mathsf{valid}(\alpha, L, \textbf{done}) = \mathsf{True} \\
\mathsf{valid}(\alpha, L, \textbf{fail}) = \mathsf{False}
\end{array}
$$

Soundness Theorem

Theorem (Soundness)

If $h \in \alpha \cap \bigcap_i L_i$ *and* $\forall i.\ \mathsf{valid}(h, L_i, c_i)$, *then*

$$(h, (\epsilon, c_1) \uplus \cdots \uplus (\epsilon, c_n)) \not\rightsquigarrow^* (h', (\overline{\delta}, \textbf{fail}) \uplus \Theta)$$

Soundness Proof

$$\mathsf{valid\_tcfg}(\alpha, \delta_1 \cdot \ldots \cdot \delta_n, c) = \mathsf{valid}(\alpha, L, \delta_1; \ldots; \delta_n; c)$$
$$\mathsf{valid\_cfg}((h, \Theta)) = \exists \alpha.\ \forall \theta \in \Theta.\ \exists L.\ h \in \alpha \cap L \wedge \mathsf{valid\_tcfg}(\alpha, L, \theta)$$

Lemma (Monotonicity)

If $\mathsf{valid}(\alpha, L, c)$ *and* $\alpha' \succeq \alpha$ *then* $\mathsf{valid}(\alpha', L, c)$.

Theorem (Preservation)

If $\mathsf{valid\_cfg}(\gamma)$ *and* $\gamma \rightsquigarrow \gamma'$ *then* $\mathsf{valid\_cfg}(\gamma')$.

Proof of Example Program

$\mathsf{heap}(O) = \forall^* o \in O.\ \exists c, O_0.\ o.\mathsf{class} \xrightarrow{-} c * \mathsf{class\_inv}_c(o, O_0) \wedge O_0 \subseteq O$
$\mathsf{class\_inv}_{\mathrm{Cell}}(o, O) = \exists v.\ o.\mathsf{contents} \mapsto v \wedge v \in O$

$A = \{\mathsf{heap}(O) \mid O \subseteq \mathit{Addresses}\}$
$\mathit{heap}(O) \preceq \mathit{heap}(O') \Leftrightarrow O \subseteq O'$

```
c := alloc_cell();
cobegin
   c' := alloc_cell();
   c.contents := c'
||
   x := c.contents;
   x := x.contents
coend
```

Proof of Example Program

c := alloc_cell();
$\{\alpha_0 = \{c\}\}$
cobegin
  c′ := alloc_cell();
  noop $f(\alpha) = \alpha \cup \{c'\}$
  $\{\alpha_0 = \{c, c'\}\}$
  c.contents := c′ $f(\alpha) = \alpha$
||
  $\{\alpha_0 = \{c\}\}$
  x := c.contents; $\alpha'_0 = \{c, x\}$
  $\{\alpha_0 = \{c, x\}\}$
  y := x.contents $\alpha'_0 = \{c, x, y\}$
coend