Formal Specification of Java Concurrency to Assist Software Verification

Brad Long and Ben Long

Software Verification Research Centre

School of Into Tech and Elec Eng

The University of Queensland

Presented by: Paul Strooper

Overview

- Motivation
- Formal (Z) model of Java Concurrency
  - state and operation schemas: request a lock, wait, notify, ...
  - safety and liveness properties: deadlock, starvation, ...
- Discussion

Motivation

- Verification of concurrent programs is difficult
  - inherent non-determinism
  - safety and liveness properties: contention, deadlock, etc.
- Models of Java Concurrency
  - textual models insufficient
  - formal model to support static analysis, run-time checks, test case generation, etc.
- Formal Z model
  - includes specification of general correctness properties

Z Specification – State Schema

System

$$
\begin{aligned}
&active : \mathbb{P}\, THREAD \\
&locked : OBJECT \rightarrow THREAD \\
&candidates : THREAD \rightarrow OBJECT \\
&blocked : THREAD \rightarrow OBJECT \\
&waiting : THREAD \rightarrow OBJECT
\end{aligned}
$$

$$
\begin{aligned}
&\operatorname{ran} locked \cup \operatorname{dom} blocked \cup \operatorname{dom} candidates \cup \operatorname{dom} waiting \subseteq active \\
&\forall (o, t) : locked \bullet (t, o) \notin (blocked \cup candidates \cup waiting) \\
&\operatorname{ran} blocked \subseteq \operatorname{dom} locked \\
&blocked \subseteq candidates \\
&\operatorname{dom} blocked \cap \operatorname{dom} waiting = \varnothing \\
&\operatorname{dom} candidates \cap \operatorname{dom} waiting = \varnothing
\end{aligned}
$$

Z Specification – Requesting a Lock

requestLock

$\Delta System$

$t? : THREAD;\ o? : OBJECT$

$t? \notin \operatorname{dom} candidates$

$((o?, t?) \notin locked \Rightarrow candidates' = candidates \oplus \{(t?, o?)\})$

$((o?, t?) \in locked \Rightarrow candidates' = candidates)$

$active' = active \land locked' = locked$

$blocked' = blocked \land waiting' = waiting$

Z Specification – Serving a Lock

serveLock

$\Delta System$

$o? : OBJECT$

$active' = active \land waiting' = waiting$

$(o? \notin \operatorname{ran} candidates \Rightarrow$ [1]

$locked' = locked \land$

$candidates' = candidates \land blocked' = blocked)$

$(o? \in \operatorname{ran} candidates \land o? \notin \operatorname{dom} locked \Rightarrow$ [2]

$\exists t : \operatorname{dom} candidates \mid (t, o?) \in candidates \bullet$

$(locked' = locked \oplus \{(o?, t)\} \land$

$candidates' = candidates \setminus \{(t, o?)\} \land$

$blocked' = (blocked \setminus \{(t, o?)\}) \oplus (candidates' \rhd \{o?\})))$

$(o? \in \operatorname{ran} candidates \land o? \in \operatorname{dom} locked \Rightarrow$ [3]

$locked' = locked \land$

$candidates' = candidates \land$

$blocked' = blocked \oplus (candidates \rhd \{o?\}))$

Z Specification – Releasing a Lock

$$
\begin{array}{|l}
\textit{releaseLock} \\
\Delta \textit{System} \\
o? : \textit{OBJECT} \\
\hline
\textit{locked}' = \{o?\} \lhd \textit{locked} \\
\textit{active}' = \textit{active} \land \textit{blocked}' = \textit{blocked} \\
\textit{candidates}' = \textit{candidates} \land \textit{waiting}' = \textit{waiting} \\
\hline
\end{array}
$$

Z Specification – wait()

wait

$\Delta System$

$o? : OBJECT;\ t? : THREAD$

$(o?, t?) \in locked$

$waiting' = waiting \oplus \{(t?, o?)\}$

$locked' = locked \setminus \{(o?, t?)\}$

$active' = active \wedge blocked' = blocked$

$candidates' = candidates$

Z Specification – notify()

notify

$\Delta System$

$o? : OBJECT$

$active' = active \land locked' = locked \land blocked' = blocked$

$o? \in \text{dom}\, locked$

$(o? \in \text{ran}\, waiting \Rightarrow$

$\quad \exists t : \text{dom}\, waiting \mid (t, o?) \in waiting \bullet$

$\quad (waiting' = waiting \setminus \{(t, o?)\} \land$

$\quad candidates' = candidates \oplus \{(t, o?)\}))$

$(o? \notin \text{ran}\, waiting \Rightarrow$

$\quad (waiting' = waiting \land candidates' = candidates))$

Z Specification – notifyAll()

notifyAll

$\Delta System$

$o? : OBJECT$

$o? \in \mathrm{dom}\, locked$

$waiting' = waiting \rhd \{o?\}$

$candidates' = candidates \oplus (waiting \rhd \{o?\})$

$active' = active \land locked' = locked \land blocked' = blocked$

Z Specification – System

$$sysRequestLock \mathrel{\widehat=} requestLock \fatsemi serveLock$$
$$sysReleaseLock \mathrel{\widehat=} releaseLock \fatsemi serveLock$$
$$sysWait \mathrel{\widehat=} wait \fatsemi serveLock$$
$$sysNotify \mathrel{\widehat=} notify \fatsemi serveLock$$
$$sysNotifyAll \mathrel{\widehat=} notifyAll \fatsemi serveLock$$

Z Specification – Properties 1

- Absence from permanent waiting (*dormancy*)

$$\forall t : \mathrm{dom}\, waiting \bullet \Diamond (t \in \mathrm{dom}\, candidates)$$

- Absence from permanent blocking

$$\forall (t, o) : blocked \bullet \Diamond ((o, t) \in locked)$$

- Absence from permanent locking

$$\forall (o, t) : locked \bullet \Diamond ((o, t) \notin locked)$$

Z Specification – Properties 2

- Absence from deadlock

$$DeadlockCycle == \mathrm{dom}((blocked \fatsemi locked)^{+} \cap id_{THREAD})$$

$$DeadlockCycle = \varnothing$$

- Absence from starvation

$$Deadlocked == \mathrm{dom}((blocked \fatsemi locked) \rhd DeadlockCycle)$$

$$\forall\, (t, o) : blocked \setminus Deadlocked \bullet \Diamond\, ((o, t) \in locked)$$

Discussion

- Properties can serve as basis for software verification tools
  - static analysis
  - model checking
  - dynamic analysis
- Example: deadlock detection
  - manage sets of threads as specified in Z model
  - can be implemented using standard Java collection classes
  - check properties in JVM

Summary

- Formal Z model for Java concurrency
  - state schema and operations
  - safety and liveness properties
- Can serve as a basis for software verification tools