Concurrency Idioms and their Effect on Program Analysis
Guy Katz and David Harel

Facilitating analysis by tailoring the programming model to the problem at hand

Concurrent Programming
- Concurrency is pervasive and useful
- But it adds complexity: deadlocks, race conditions, starvation
- A tradeoff...

Retaining “Just Enough” Concurrency
- Tailor the model to the task at hand
- Only pick the required concurrency idioms
- Solve the problem efficiently, while keeping the program simple

The Request / Wait / Block (RWB) Model

Interweaving parallel behavior threads

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Threads request, wait-for and block events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Selection</td>
<td>An event that is requested and not blocked is triggered</td>
</tr>
<tr>
<td>Notification</td>
<td>Resume threads that requested or waited-for the event</td>
</tr>
</tbody>
</table>

The RWB Execution Cycle

The Blocking Idiom Facilitates Program Repair

- Safety violation: a bad state is reachable
- A patch blocks bad transitions, without introducing deadlocks
- Incremental, non-intrusive repair

Each Idiom Affords Unique Descriptive Succinctness and Makes Programs Smaller

- Smaller programs are easier to maintain and verify
- Each of requesting, waiting-for and blocking render some programs exponentially smaller
- Example: $L_n = (0^{n-1} \cdot (0 + 1))^\omega$
- $RWB$ implementation size: $O(\log^2 n \cdot \log \log n)$
- Size without blocking: $\Omega(n)$

$RWB$ implementation for $(0^5 \cdot (0 + 1))^\omega$