Concurrency Idioms and their Effect on Program Analysis

Guy Katz and David Harel

Weizmann Institute of Science

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

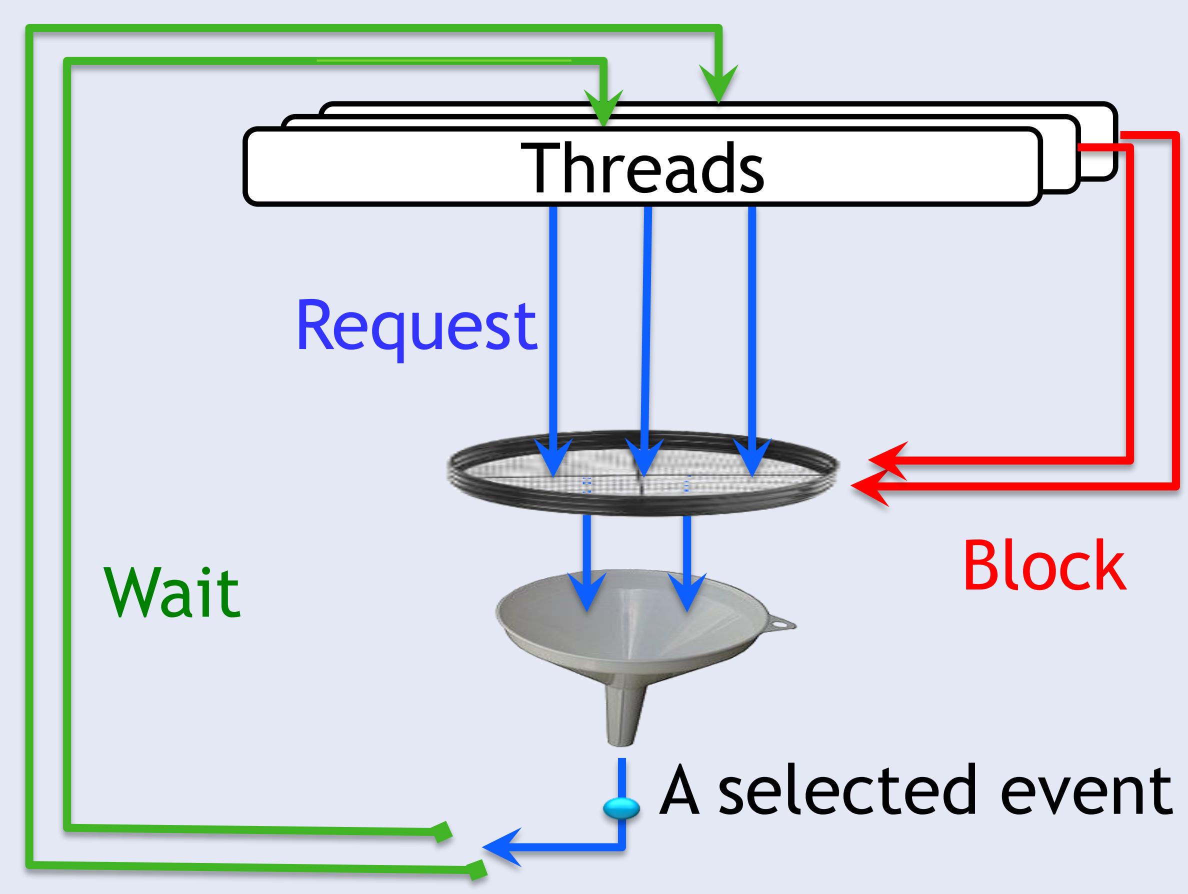
- Tailor the model to the task at hand
- Only pick the required concurrency idioms
- Solve the problem efficiently, while keeping the program simple

The Request / Wait / Block (*RWB*) Model

Interweaving parallel behavior threads

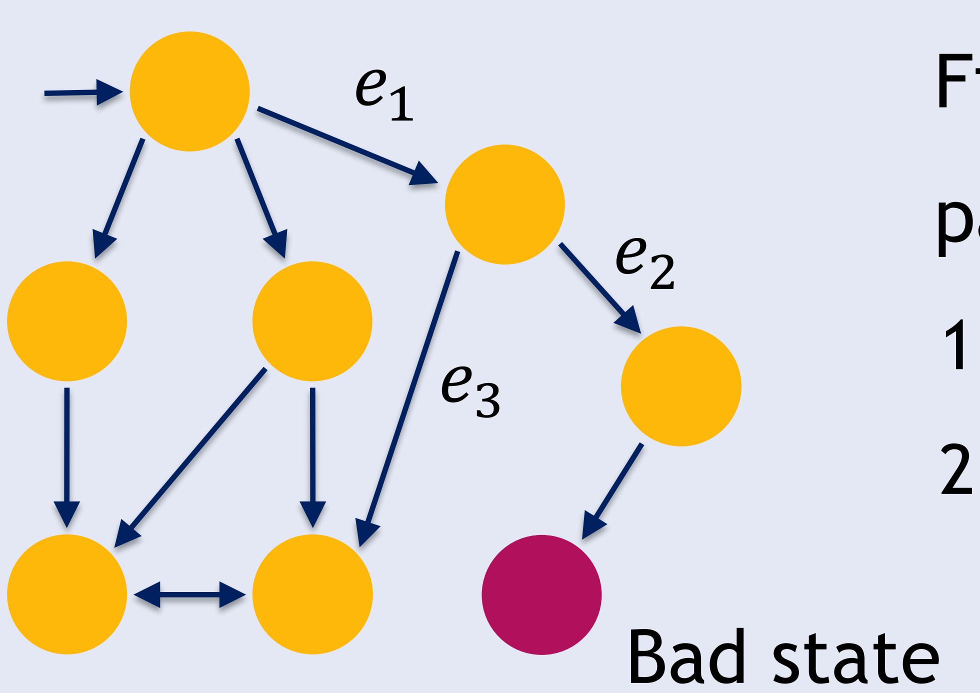
Declaration	Threads request, wait-for and block events
Event Selection	An event that is requested and not blocked is triggered
Notification	Resume threads that requested or waited-for the event

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



Fix violation by adding the patch thread:

1. wait-for e_1
2. wait-for e_3 while blocking e_2

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: $L_n = \left(0^{n-1} \cdot (0+1)\right)^{\omega}$
- *RWB* implementation size: O$(\log^2 n \cdot \log\log n)$
- Size without blocking: $\Omega(n)$

RWB implementation for $\left(0^5 \cdot (0+1)\right)^{\omega}$