typestate for concurrent objects

Silvia Crafa & Luca Padovani

typestate-oriented programming

(Aldrich *et al.*, OOPSLA 2009)

Aims

- **static** enforcement of object protocols

Mechanisms

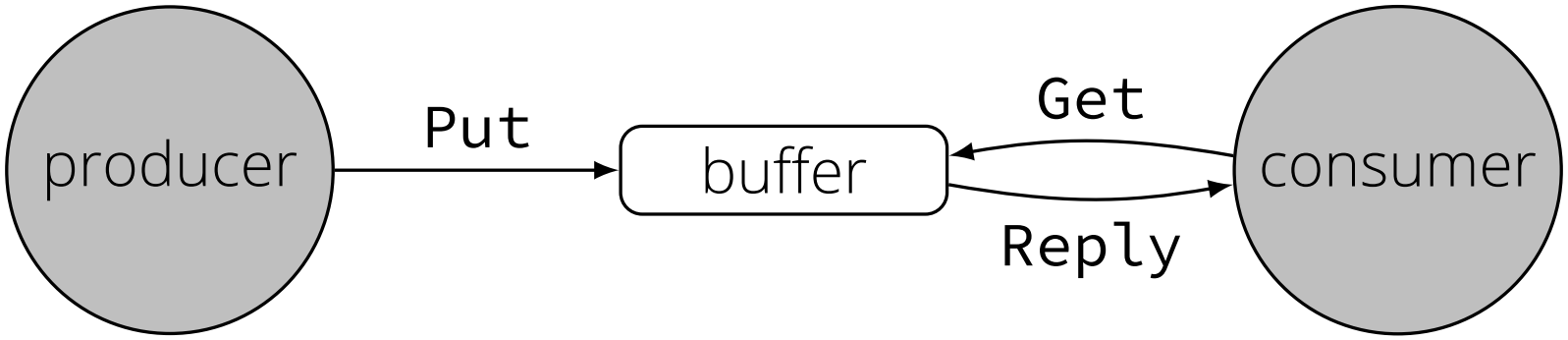
- **state** annotations in object types `Empty, Full`
- **flow-sensitive** type system `[Empty » Full]`
- **aliasing control**

What about **concurrent** objects?

- concurrent objects are aliased by definition
- state transitions aren't always statically trackable

concurrent linear buffer

a simple example of stateful concurrent object



- producer **knows** that buffer is initially empty
- consumer **does not** (and **cannot**) **know** when buffer is full

our proposal for concurrent TSOP

(Crafa & Padovani, OOPSLA 2015)

Hybrid approach

- strict protocol — if possible/desirable
- lax protocol + runtime support — otherwise

Objective Join Calculus (Fournet *et al.* 2003)

- concurrent objects
- synchronization patterns
- state and operations unified into messages

EXAMPLE

types for concurrent objects

commutative Kleene algebra (Conway 1971)

type	usage
1	discard
$\mathtt{m}(\bar{t})$	send $\mathtt{m}$
$t + s$	either t or s

types for concurrent objects

commutative Kleene algebra

(Conway 1971)

type	usage	
1	discard	
$\mathtt{m}(\bar{t})$	send $\mathtt{m}$	
$t + s$	either t or s	
$t \cdot s$	both t and s	**concurrently**
$*t$	t *ad libitum*	**concurrently**

types for concurrent objects

commutative Kleene algebra (Conway 1971)

type	usage	
0	stay away	
1	discard	
$\mathtt{m}(\bar{t})$	send $\mathtt{m}$	
$t + s$	either t or s	
$t \cdot s$	both t and s	**concurrently**
$*t$	t *ad libitum*	**concurrently**

types for concurrent objects

commutative Kleene algebra (Conway 1971)

type	usage	
0	stay away	
1	discard	
$\mathtt{m}(\bar{t})$	send $\mathtt{m}$	
$t + s$	either t or s	
$t \cdot s$	both t and s	**concurrently**
$*t$	t *ad libitum*	**concurrently**

$$\mathtt{EMPTY} \cdot \mathtt{Put} \cdot \mathtt{Get} \;+\; \mathtt{FULL} \cdot \mathtt{Get} \;+\; 1$$

TYPED EXAMPLES

wrap-up

what types tell us about concurrent objects

Properties

- communication safety
- protocol fidelity
- boundedness $\qquad\qquad$ $*$-free type $\Rightarrow$ bounded message queue

More information

- check for orphan messages
- check for impossible reactions
- type-driven safe object deallocation

more interesting examples to play with

available in the source distribution (online) or on my laptop

- read/write locks
- Michael & Scott concurrent queues
- future variables with cancellation and timeouts
- approximation of π from Akka tutorial
- master with unbounded workers
- sessions
- sequential non-uniform objects (files, stacks, iterators, ...)
- **got one not listed here? let's code it together!**

`http://www.di.unito.it/~padovani/Software/CobaltBlue`