Brief announcement: On the impossibility of detecting concurrency

Éric Goubault **Jérémy Ledent** Samuel Mimram

École Polytechnique, Paris

DISC 2018, New Orleans
October 16, 2018

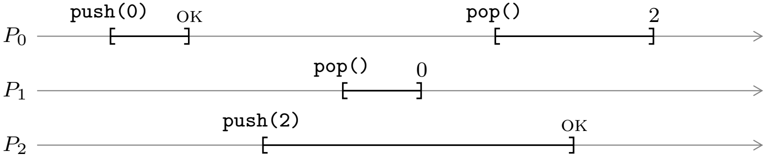
Concurrent specifications

Concurrent specifications

Idea: the specification of an object is the set of all the correct execution traces (Lamport, 1986).

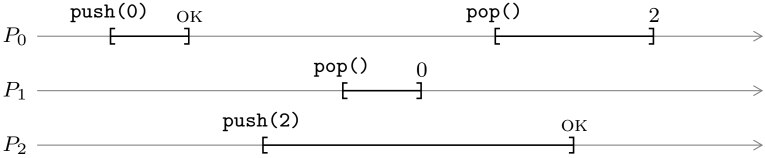
Concurrent specifications

Idea: the specification of an object is the set of all the correct execution traces (Lamport, 1986).



Concurrent specifications

Idea: the specification of an object is the set of all the correct execution traces (Lamport, 1986).

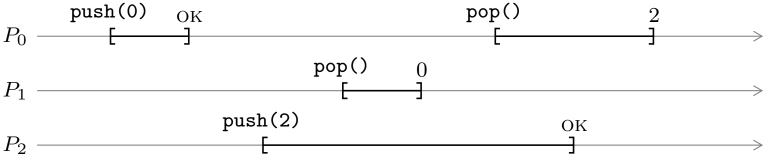


Write $\mathcal{T}$ for the set of all execution traces.

- A *concurrent specification* is a subset $\sigma \subseteq \mathcal{T}$.

Concurrent specifications

Idea: the specification of an object is the set of all the correct execution traces (Lamport, 1986).



Write $\mathcal{T}$ for the set of all execution traces.

- A *concurrent specification* is a subset $\sigma \subseteq \mathcal{T}$.
- A program implements a specification σ if all the traces that it can produce belong to σ.

Linearizability (Herlihy & Wing, 1990)

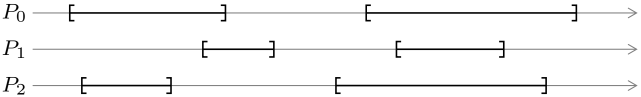
$$\text{SeqSpec} \xrightarrow{\text{Lin}} \text{ConcSpec}$$

- **Input:** a sequential specification σ (e.g. list, queue, ...).
- **Output:** a concurrent specification $\text{Lin}(\sigma)$.

Linearizability (Herlihy & Wing, 1990)

$$\text{SeqSpec} \xrightarrow{\text{Lin}} \text{ConcSpec}$$

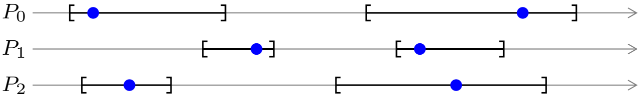
- **Input:** a sequential specification σ (e.g. list, queue, ...).
- **Output:** a concurrent specification $\text{Lin}(\sigma)$.



Linearizability (Herlihy & Wing, 1990)

$$\text{SeqSpec} \xrightarrow{\text{Lin}} \text{ConcSpec}$$

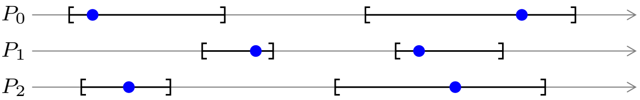
- **Input:** a sequential specification σ (e.g. list, queue, ...).
- **Output:** a concurrent specification $\text{Lin}(\sigma)$.



Linearizability (Herlihy & Wing, 1990)

$$\text{SeqSpec} \xrightarrow{\text{Lin}} \text{ConcSpec}$$

- **Input:** a sequential specification σ (e.g. list, queue, ...).
- **Output:** a concurrent specification $\text{Lin}(\sigma)$.

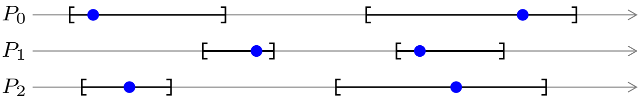


$$\text{Lin}(\sigma) = \{T \text{ concurrent trace} \mid T \text{ is linearizable w.r.t. } \sigma\}$$

Linearizability (Herlihy & Wing, 1990)

$$\text{SeqSpec} \xrightarrow{\text{Lin}} \text{ConcSpec}$$

- **Input:** a sequential specification σ (e.g. list, queue, ...).
- **Output:** a concurrent specification $\text{Lin}(\sigma)$.

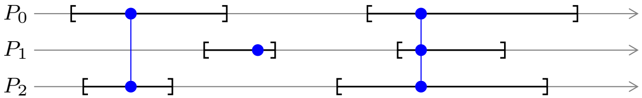


$$\text{Lin}(\sigma) = \{T \text{ concurrent trace} \mid T \text{ is linearizable w.r.t. } \sigma\}$$

Some objects are not linearizable!
Their specification cannot be expressed as $\text{Lin}(\sigma)$, for any σ.

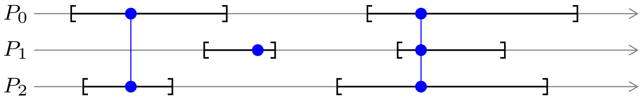
Concurrent variants of linearizability

Set-linearizability (Neiger)



Concurrent variants of linearizability

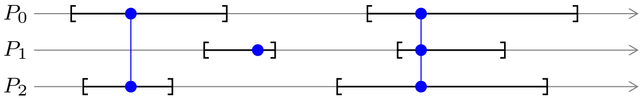
Set-linearizability (Neiger)



- Can specify: exchanger, immediate snapshot, set agreement.

Concurrent variants of linearizability

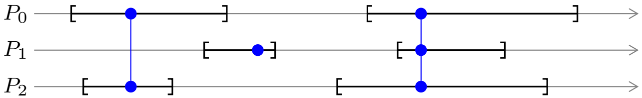
Set-linearizability (Neiger)



- Can specify: exchanger, immediate snapshot, set agreement.
- Cannot specify: validity, write-snapshot.

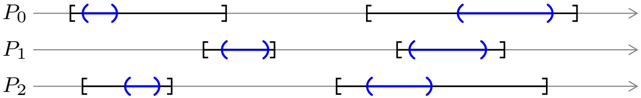
Concurrent variants of linearizability

Set-linearizability (Neiger)



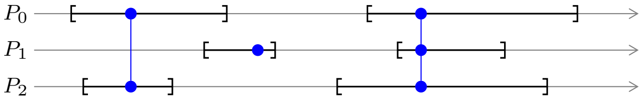
- Can specify: exchanger, immediate snapshot, set agreement.
- Cannot specify: validity, write-snapshot.

Interval-linearizability (Rajsbaum, Castañeda, Raynal)



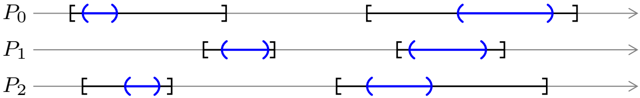
Concurrent variants of linearizability

Set-linearizability (Neiger)



- Can specify: exchanger, immediate snapshot, set agreement.
- Cannot specify: validity, write-snapshot.

Interval-linearizability (Rajsbaum, Castañeda, Raynal)



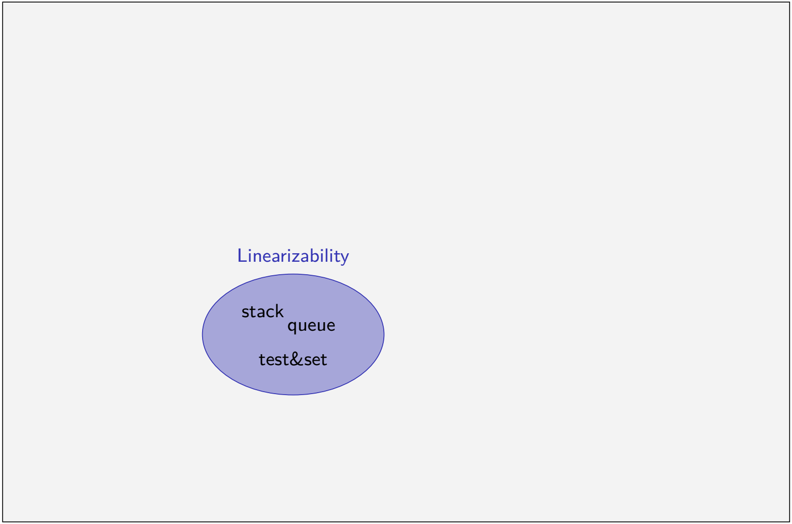
- Can specify every task!

Overview

Concurrent specifications

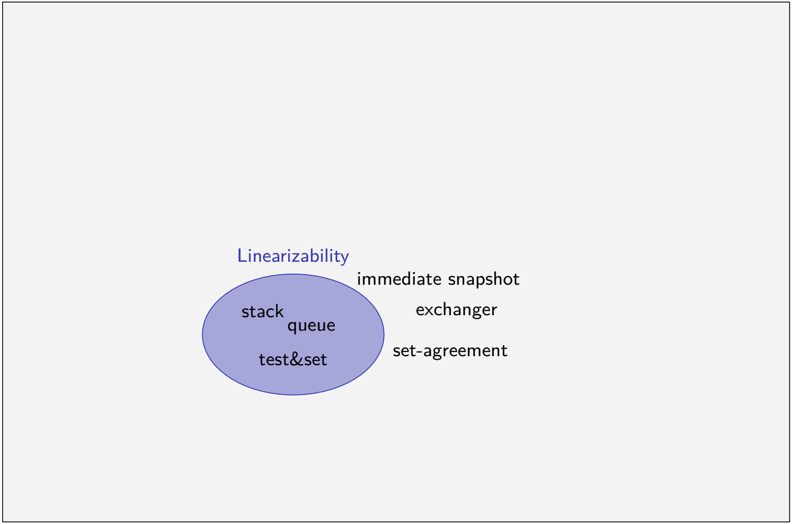
Overview

Concurrent specifications



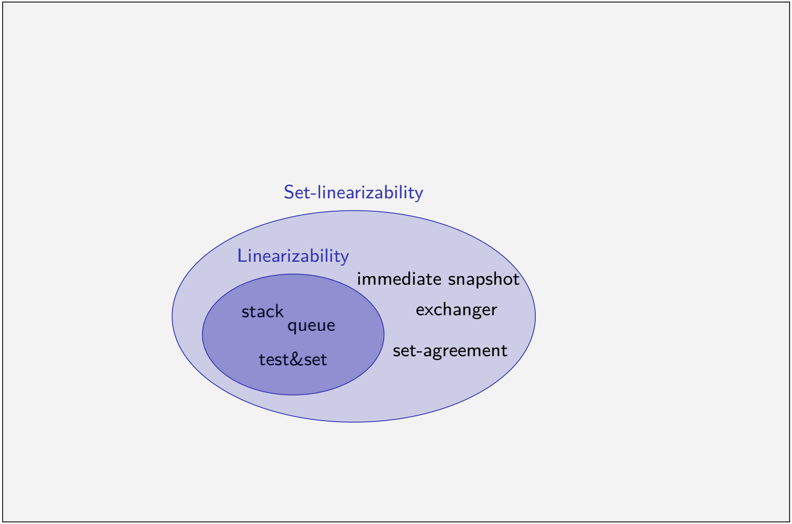
Overview

Concurrent specifications



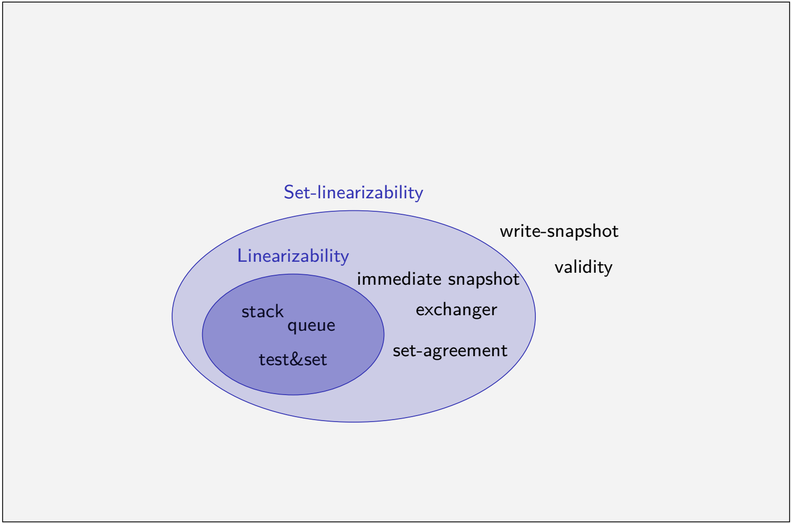
Overview

Concurrent specifications



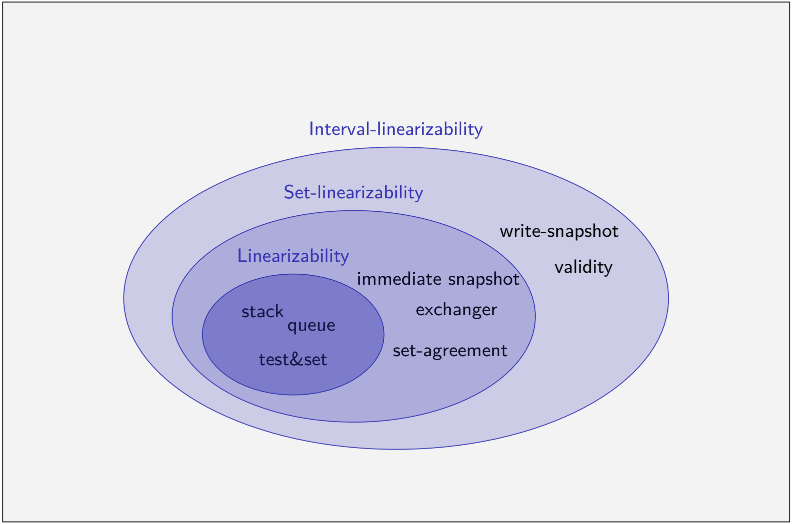
Overview

Concurrent specifications

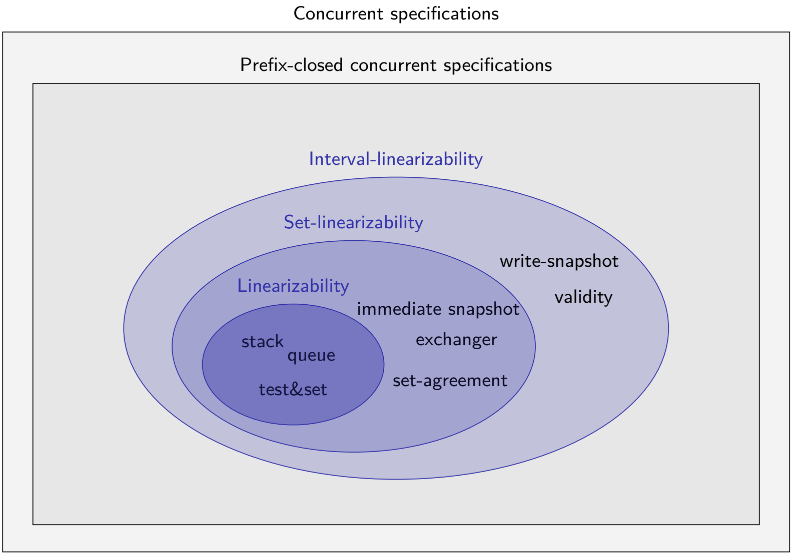


Overview

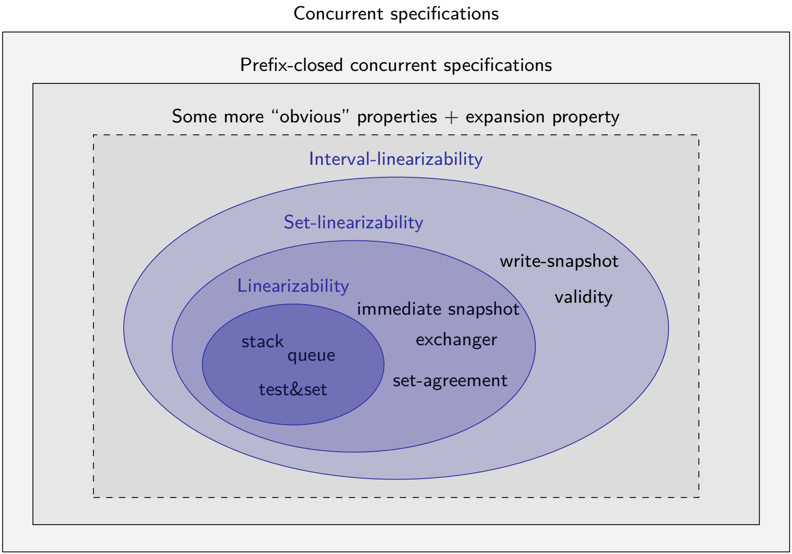
Concurrent specifications



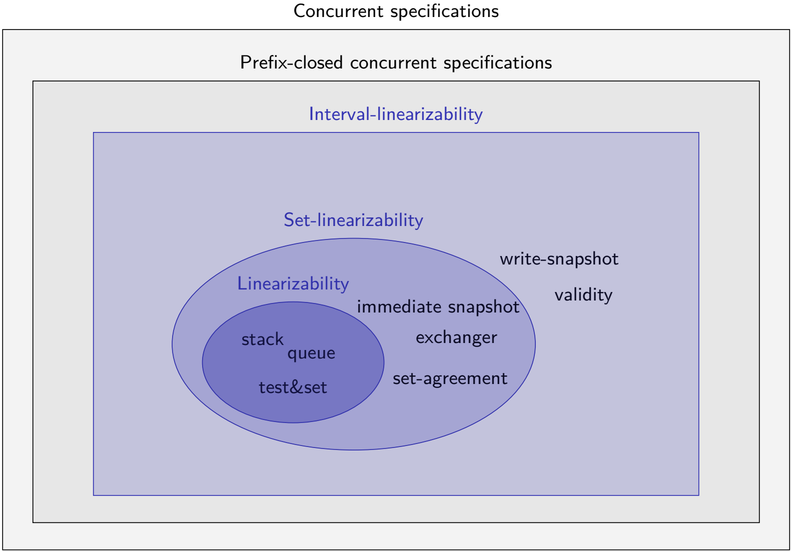
Overview


Concurrent specifications
Prefix-closed concurrent specifications
Interval-linearizability
Set-linearizability
write-snapshot
validity
Linearizability
immediate snapshot
exchanger
set-agreement
stack
queue
test&set

Overview


Concurrent specifications
Prefix-closed concurrent specifications
Some more “obvious” properties + expansion property
Interval-linearizability
Set-linearizability
Linearizability
stack
queue
test&set
immediate snapshot
exchanger
set-agreement
write-snapshot
validity

Overview



Concurrent specifications
Prefix-closed concurrent specifications
Interval-linearizability
Set-linearizability
write-snapshot
validity
Linearizability
immediate snapshot
stack
queue
exchanger
test&set
set-agreement

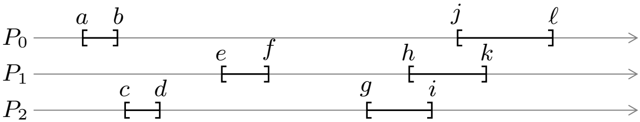
Expansion of intervals

A concurrent specification satisfies the expansion property if:

Expansion of intervals

A concurrent specification satisfies the expansion property if:

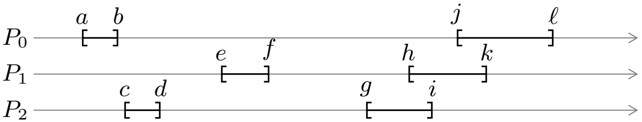
For any correct execution trace,



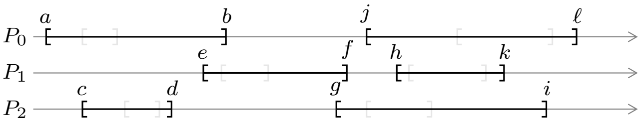
Expansion of intervals

A concurrent specification satisfies the expansion property if:

For any correct execution trace,



if we *expand* the intervals,



then the resulting trace is still correct.

Example: the Exchanger object

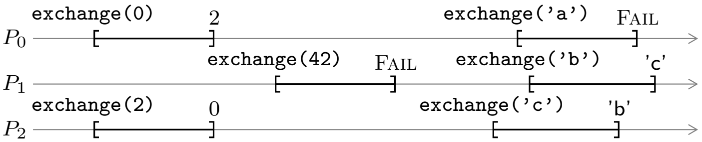
Similar to the one available in Java[1]: *"A synchronization point at which threads can pair and swap elements within pairs".*
Here, we consider a wait-free variant.

[1] `java.util.concurrent.Exchanger<V>`

Example: the Exchanger object

Similar to the one available in Java[1]: *"A synchronization point at which threads can pair and swap elements within pairs".*
Here, we consider a wait-free variant.

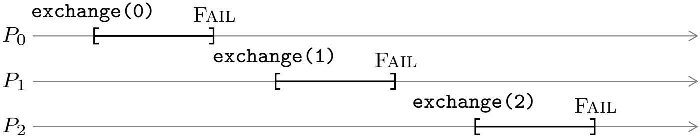
A typical execution of the exchanger looks like this:



[1] java.util.concurrent.Exchanger<V>

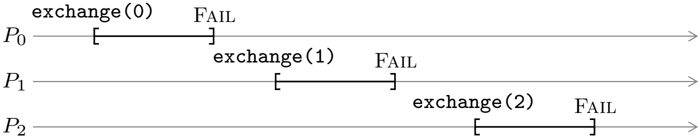
Example: the Exchanger object

The following execution is correct:

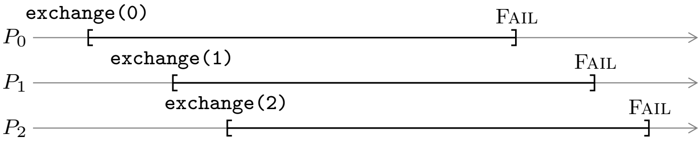


Example: the Exchanger object

The following execution is correct:



Hence, according to the expansion property,



should be considered correct too!

Results

- In a reasonable computational model:

Theorem

The semantics $[\![P]\!]$ of any program P has the expansion property.

Results

- In a reasonable computational model:

Theorem

The semantics $[\![P]\!]$ of any program P has the expansion property.

- Linearizability-based techniques can only produce specifications which satisfy the expansion property.

Theorem

For every sequential specification σ, $\mathsf{Lin}(\sigma)$ *has the expansion property.*

Results

- In a reasonable computational model:

Theorem

The semantics $[\![P]\!]$ of any program P has the expansion property.

- Linearizability-based techniques can only produce specifications which satisfy the expansion property.

Theorem

For every sequential specification σ, Lin(σ) *has the expansion property.*

We write ConcSpec for the set of concurrent specifications satisfying the expansion property (and prefix-closure, etc).

Results

$$\begin{tikzcd}[column sep=huge] \mathsf{SeqSpec} \arrow[rr, bend right, "\mathsf{Lin}"'] \arrow[rr, phantom, "\top"] & & \mathsf{ConcSpec} \arrow[ll, bend right, "\mathsf{U}"'] \end{tikzcd}$$

Theorem

The maps Lin *and* U *form a Galois connection: for every* $\sigma \in \mathsf{SeqSpec}$ *and* $\tau \in \mathsf{ConcSpec}$,

$$\mathsf{Lin}(\sigma) \subseteq \tau \qquad \iff \qquad \sigma \subseteq \mathsf{U}(\tau).$$

Results

Applications:

- By the properties of Galois connections,

$$\mathsf{Lin}(\mathsf{U}(\mathsf{Lin}(\sigma))) = \mathsf{Lin}(\sigma)$$

  This yields a simple criterion to check whether a given specification τ is linearizable: check whether $\mathsf{Lin}(\mathsf{U}(\tau)) = \tau$.

Results

Applications:

- By the properties of Galois connections,

$$\mathsf{Lin}(\mathsf{U}(\mathsf{Lin}(\sigma))) = \mathsf{Lin}(\sigma)$$

  This yields a simple criterion to check whether a given specification τ is linearizable: check whether $\mathsf{Lin}(\mathsf{U}(\tau)) = \tau$.

- The Galois connection for interval linearizability has the following corollary:

Theorem

ConcSpec *is the set of interval-linearizable specifications.*

Thanks!