Behavioural APIs & choreographic development

Emilio Tuosto @ GSSI & UoL
emilio.tuosto@gssi.it

NOVA LINCS FCT-UNL

26 February, 2020, Portugal

Research supported by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement No 778233.

BEHAPI Behavioural Application Program Interfaces

Plan

- Distributed APIs
- Why should we add “behavioural” specs?
- Exercises in choreographic design
  - Correctness-by-design
  - Debugging with choreographies
  - Choreographic-driven testing

Before we start...

Meta stuff

- Visiting on the BehAPI project (`https://www.um.edu.mt/projects/behapi`)
  - API-economy
  - Behavioural specs
  - Formal support
- Recently @ Gran Sasso Science Institute (`https://www.gssi.it`)
  - Advanced School of Study
  - CS: efficient and rigorous approaches to complex systems
    - Formal methods
    - Software Engineering
    - Algorithms

Meta stuff

- Visiting on the BehAPI project (`https://www.um.edu.mt/projects/behapi`)
  - API-economy
  - Behavioural specs
  - Formal support
- Recently @ Gran Sasso Science Institute (`https://www.gssi.it`)
  - Advanced School of Study
  - CS: efficient and rigorous approaches to complex systems
    - Formal methods
    - Software Engineering
    - Algorithms
  - CS is expanding and (more or less always) recruiting
  - open to collaborations with other institutions (e.g., co-supervisions)
  - E.G., post-doc position at INRIA Lille

Postcards from the city of the number 99

Carbon economy

Green economy

Circular economy

API economy

...

API Economy: what is it?

In IBM's words [https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=APW12357USEN&]

"A business API[a] is a public persona for an enterprise that exposes defined assets, data or services for consumption by a selected audience of developers, either inside or outside your organization. [...] application developers can easily leverage, publicize and aggregate a company's assets for broad-based consumption. "

---

[a] No worries if you've "a strange feeling of deja-vu" 😉
(see https://developer.ibm.com/apiconnect/documentation/api-101/evolution-growth-apis/)

API Economy: what is it?

In IBM's words [https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=APW12357USEN&]

“

exposes

data or services consumption

aggregate

”

---

[a] No worries if you've “a strange feeling of deja-vu” 😉
(see https://developer.ibm.com/apiconnect/documentation/api-101/evolution-growth-apis/)

API Economy: what is it?

In IBM's words [https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=APW12357USEN&]

“ exposes

data or services consumption

aggregate ”

[a] No worries if you've “a strange feeling of deja-vu” 😉
(see https://developer.ibm.com/apiconnect/documentation/api-101/evolution-growth-apis/)

‘Business’ jargon aside

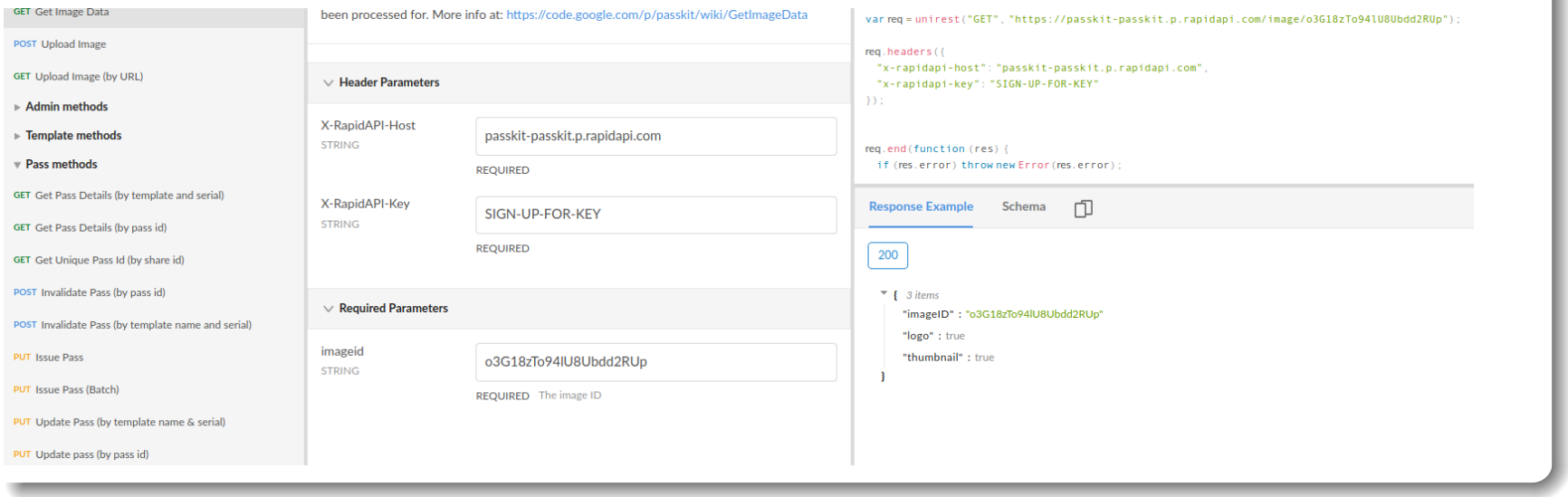
- Composition of distributed software
- (design-by-)contract
- Strong link to message-passing (GoLang, Erlang, Elixir, Scala, JMS, Akka, ...)

“An API is useless unless you document it”

`http://www.computerworld.com/article/2593623/app-development/application-programming-interface.`

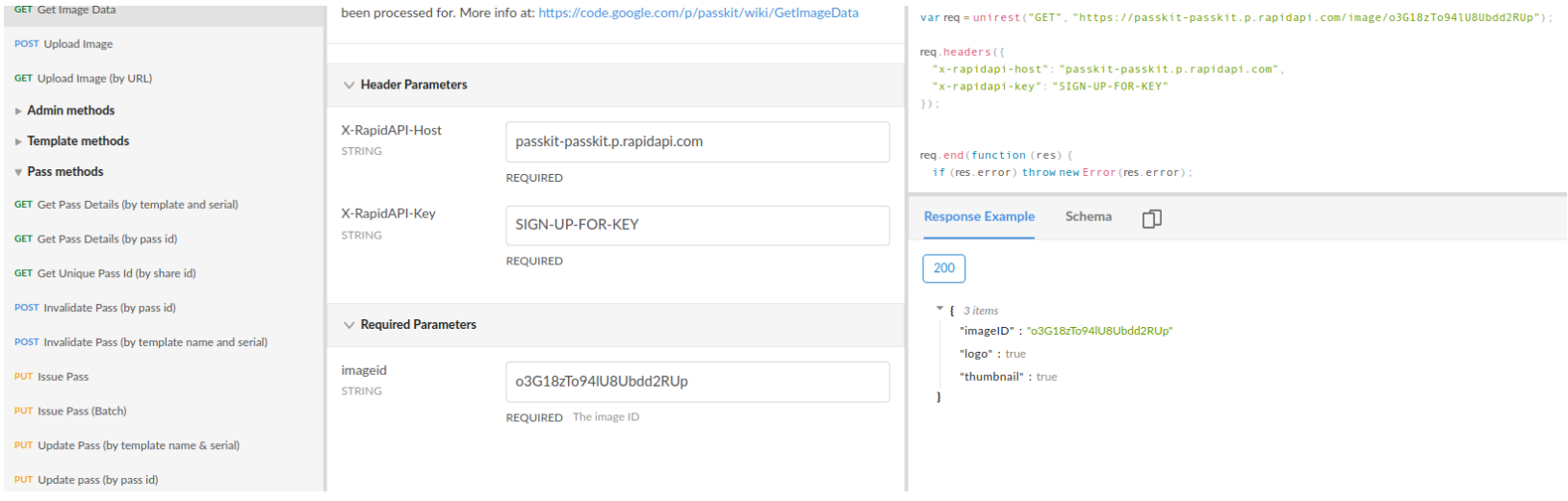
Missing bits

From the status quo...



Missing bits

From the status quo...

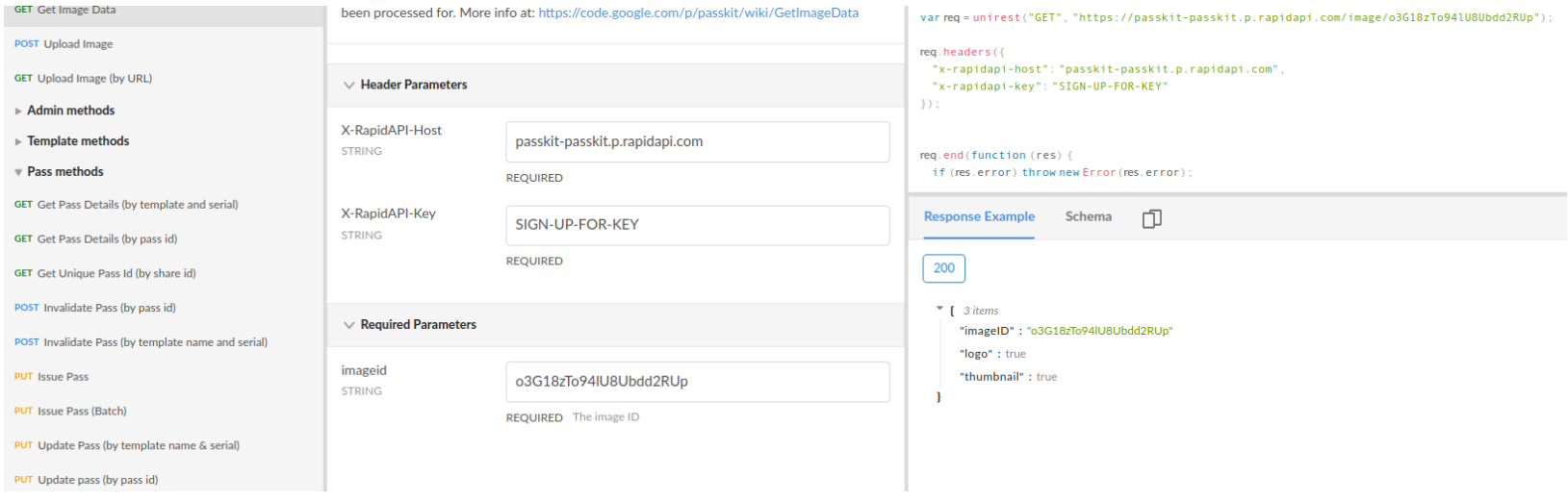


...to “smarter” descriptions

To overcome some limitations of current practices

Missing bits

From the status quo...



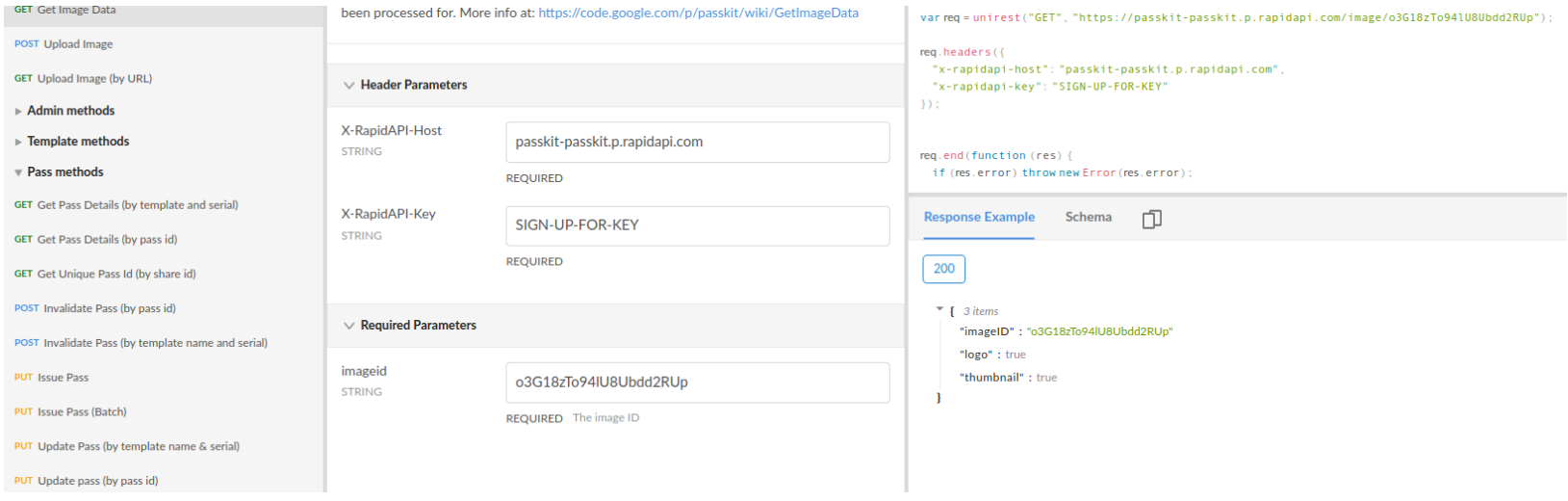
...to “smarter” descriptions

To overcome some limitations of current practices

- Higher abstractions: descriptions are very basic / low level

Missing bits

From the status quo...



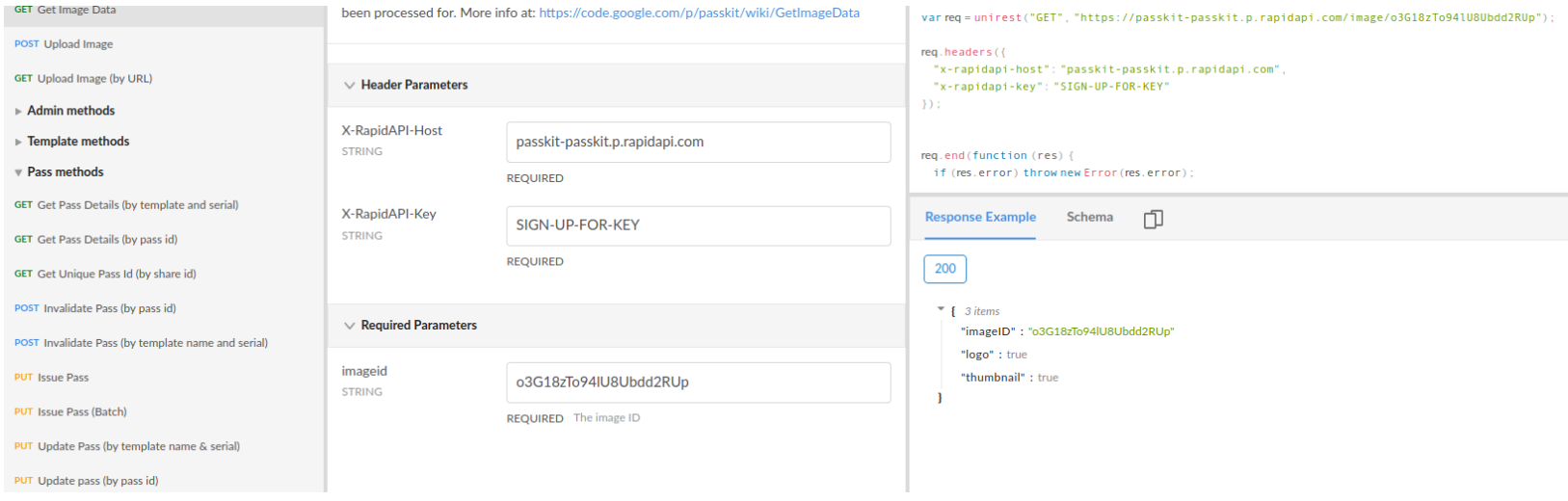
...to “smarter” descriptions

To overcome some limitations of current practices

- Higher abstractions: descriptions are very basic / low level
- Support multi-party: “manual” RESTful arch. are problematic

Missing bits

From the status quo...



...to "smarter" descriptions

To overcome some limitations of current practices

- Higher abstractions: descriptions are very basic / low level
- Support multi-party: "manual" RESTful arch. are problematic
- Software quality: very limited and not well-supported
- ...

Behaviour matters!

Jedis causes OutOfMemoryException after SocketTimeoutException #1747

Open **ragabar** opened this issue on Jan 16 · 0 comments

ragabar commented on Jan 16 • edited

Description

The problem happens when the client is waiting for 1 response, and after parsing the first bytes of a response, then get
SocketTimeoutException on read. The exception causes `jedis.close()` to send QUIT but keep processing the partially-read response from ZREVRANGE response.
Then the partially response might land on * by luck if the response contains a star.
With this low-probability incident, jedis will try to allocate an array with the size that he gets from the partial stream.
It might be lucky that the rest of the response line is a small enough numb*__*er to allocate in the heap, and might be unlucky to
allocate a very large array, or allocate with a negative number of elements.
This causes the JVM to crash with OOM Java heap space, suppresed in `java.net.SocketTimeoutException: Read timed out` Exception.

A similar bug has been discoverd when using pipelined, that at a SocketTimeout, `pipeline.close()` calls `sync()` in order to complete parsing
pending responses, but it end up reading partial-read response. causing the same problem as above.

Behave!

Why do behavioural descriptions matter?

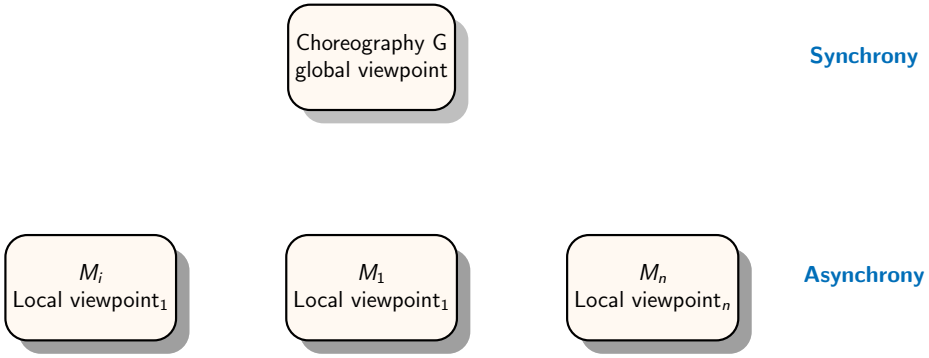
- Stateful computations
  - since availability of services/operations depends on the state
  - hence some "combination" of calls may leed to errors/exceptions
  - ex. in an API managing a distributed file system, a read-operation in a file must be preceded by an open-operation of the file
- Messaging
  - since decisions have to be taken in a (distributedly) by components having only partial awareness of the "global state"
  - hence wrong flows of messages my lead to deadlocks, livelocks, inconsistencies, message loss
  - ex. loss of message sent to a component after its termination was not properly propagated
- ...

What is a choreography?

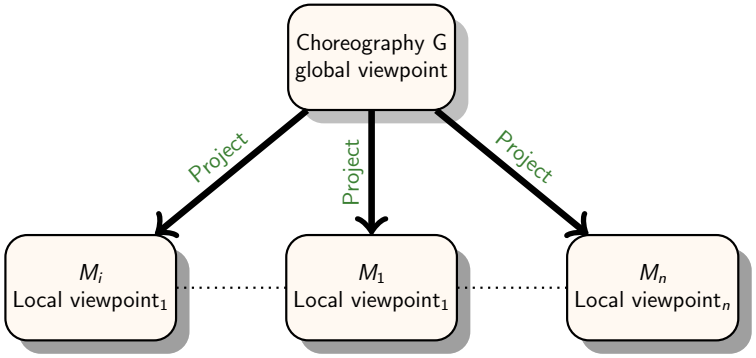
Quoting W3C...

"Using the Web Services Choreography specification, a contract containing a global definition of the common ordering conditions and constraints under which messages are exchanged, is produced that describes, from a global viewpoint [...] observable behaviour of all the parties involved. Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]"

“Top-down” & “Bottom-up” approach



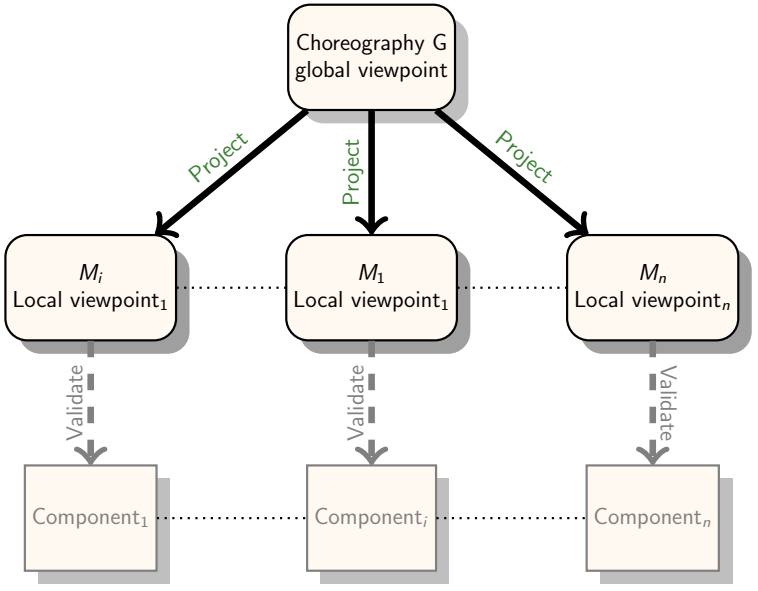
“Top-down” & “Bottom-up” approach



Synchrony

Asynchrony

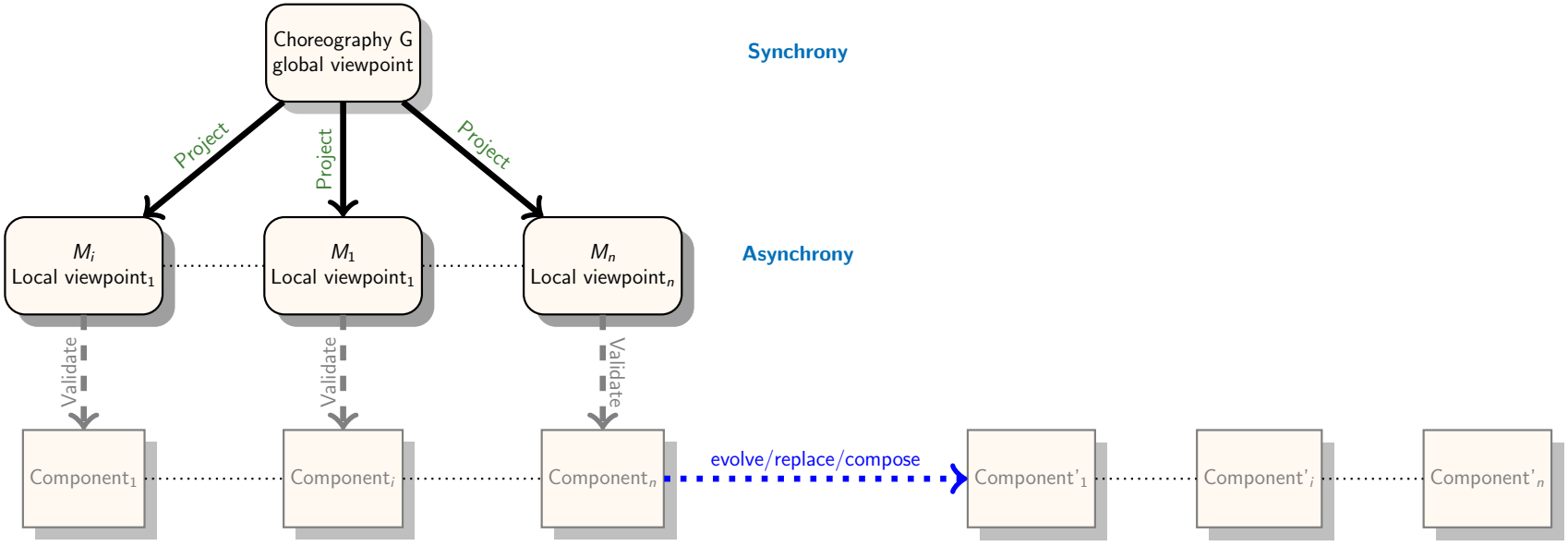
“Top-down” & “Bottom-up” approach



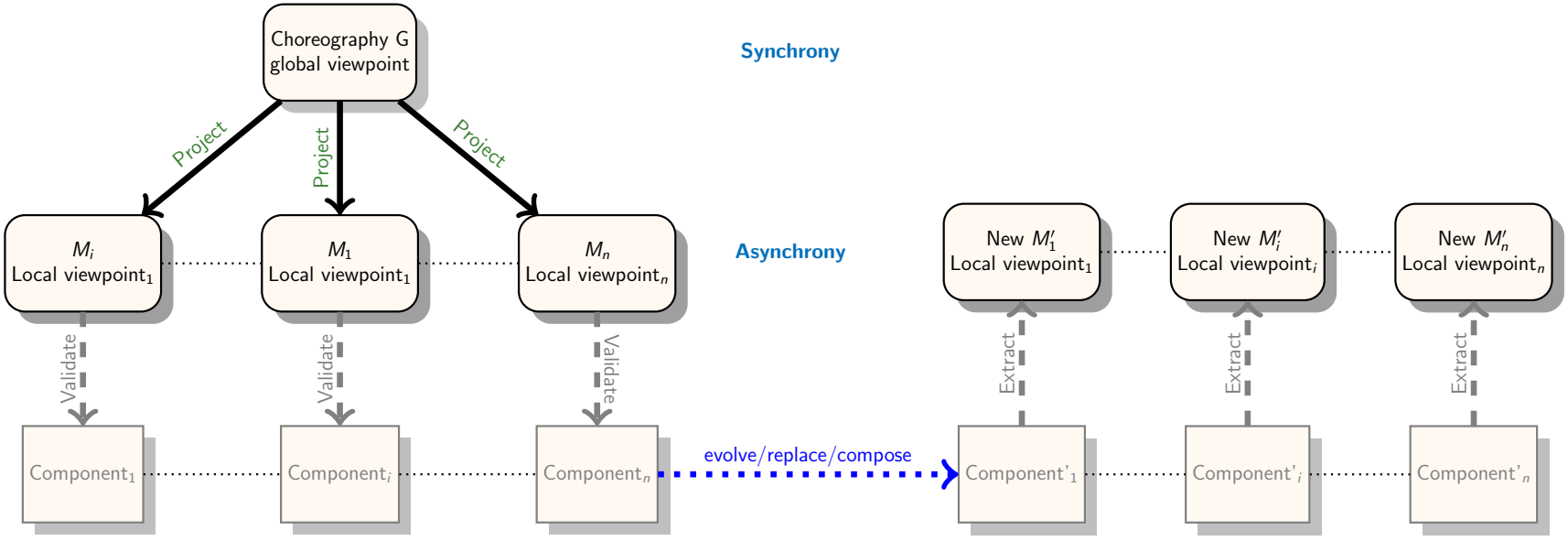
Synchrony

Asynchrony

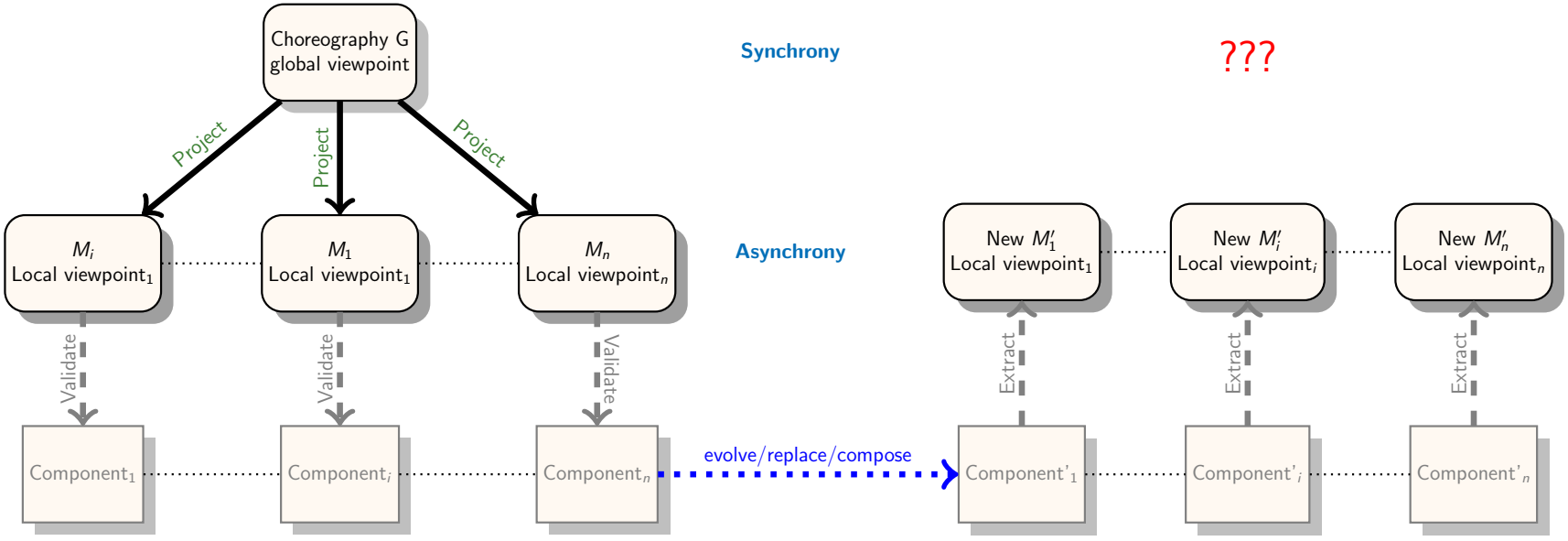
“Top-down” & “Bottom-up” approach



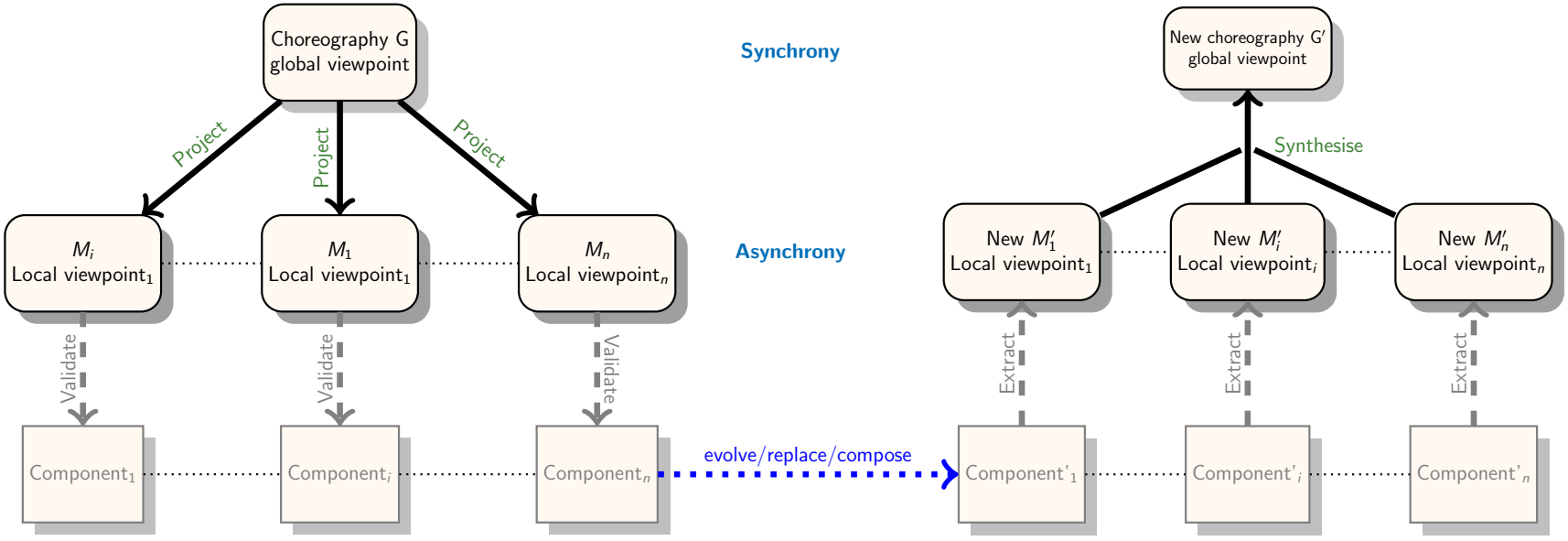
“Top-down” & “Bottom-up” approach



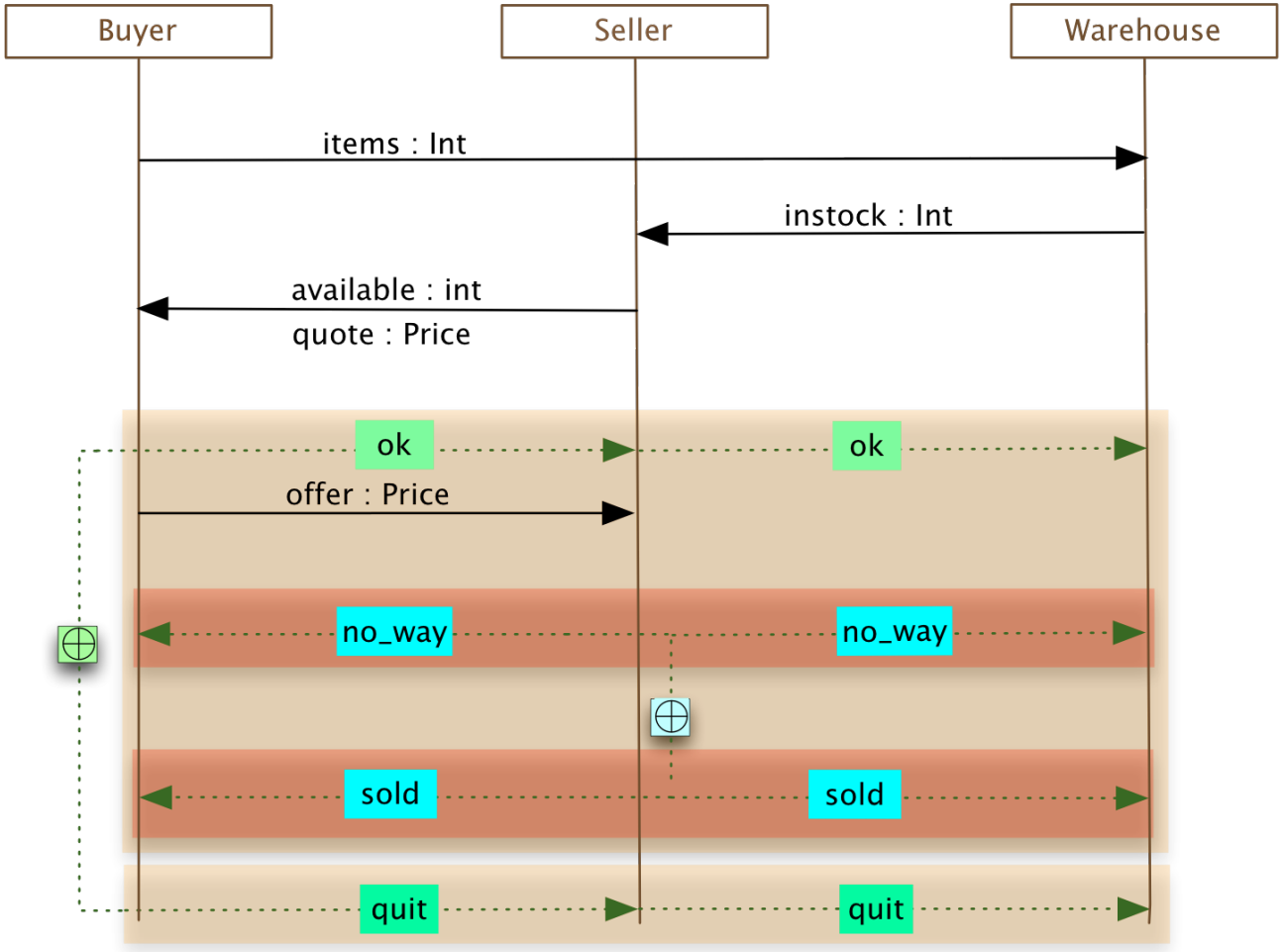
“Top-down” & “Bottom-up” approach



“Top-down” & “Bottom-up” approach

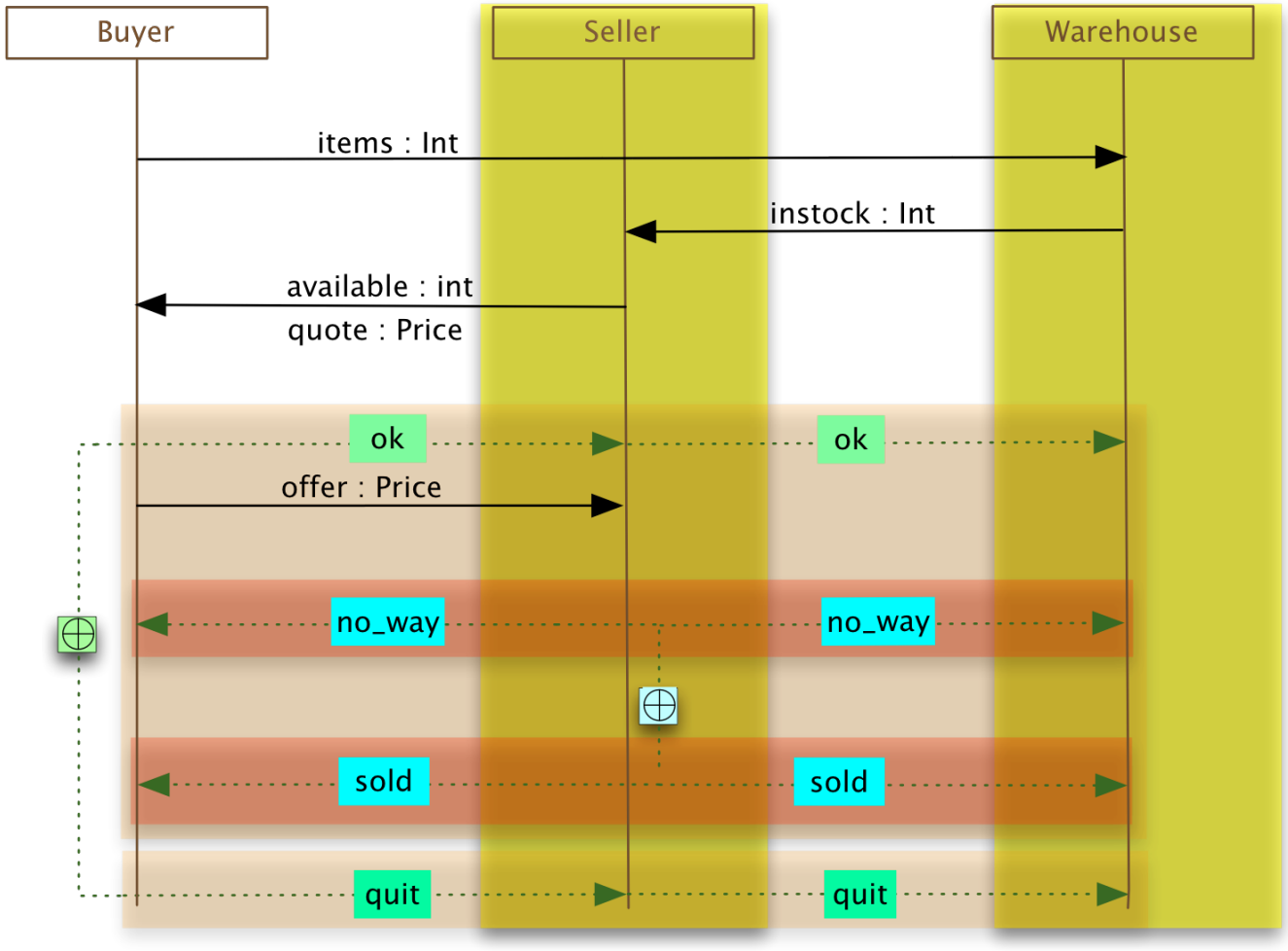


An intuition



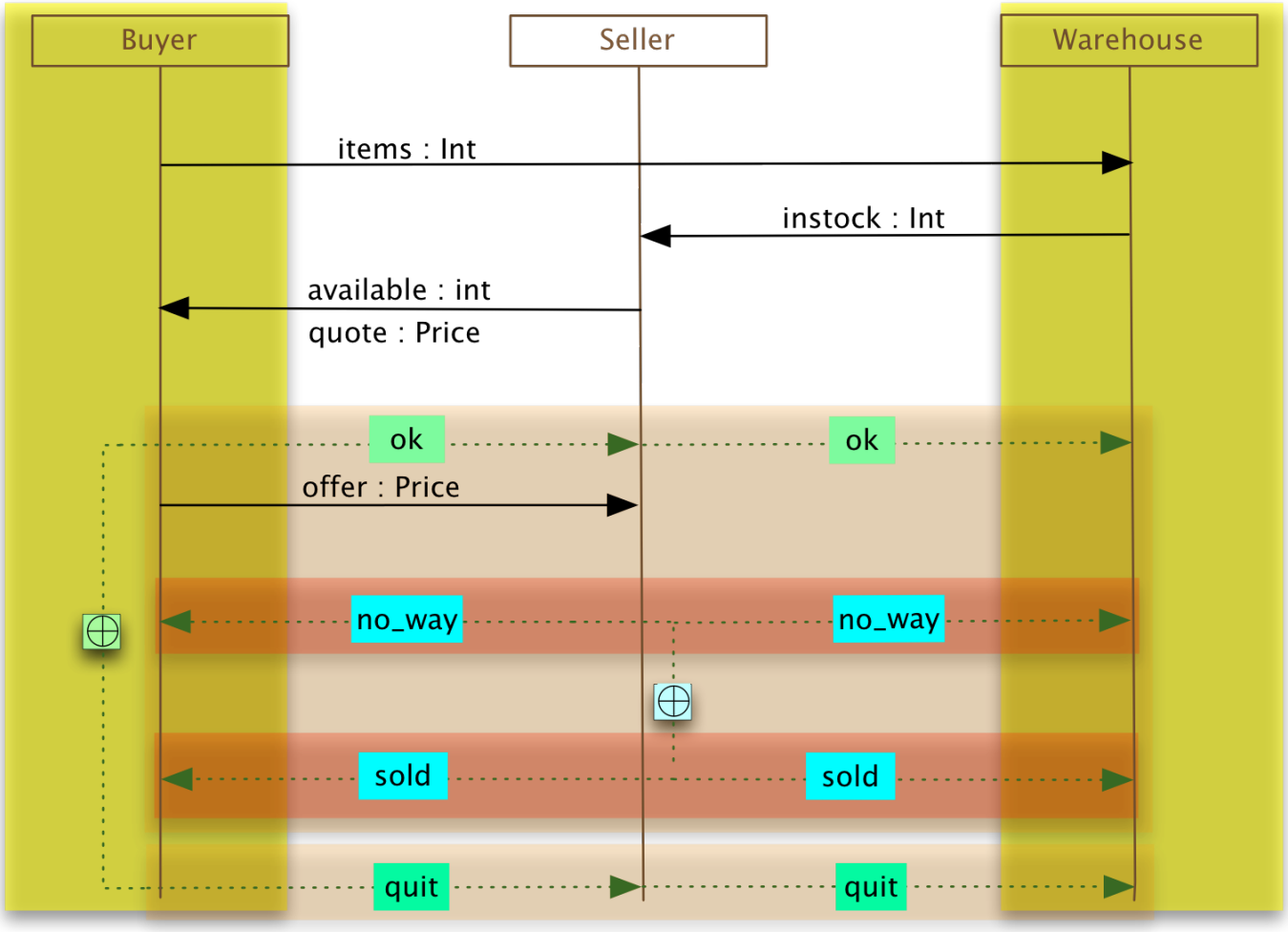
Global viewpoint

An intuition



Projecting on buyer

An intuition



Projecting on seller

Some considerations

Desiderata

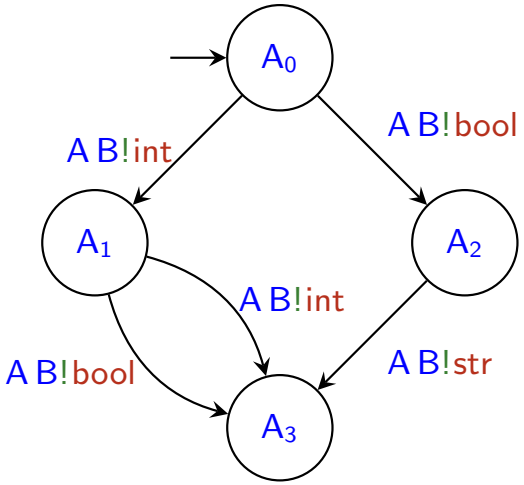
- progress (graceful termination or (dead)lock-freedom)
- no oblivious messages (aka orphan messages)
- no unspecified reception
- ...

More complexity

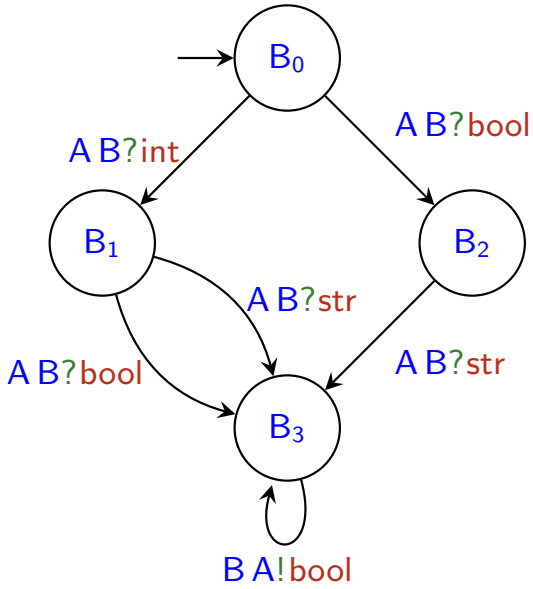
- several communication models
- not all global viewpoints "make sense"
  (constraints may be impossible to realise)
- data, time, ...

Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



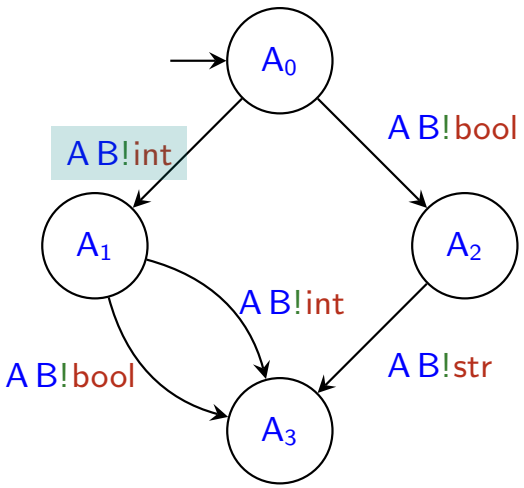
A



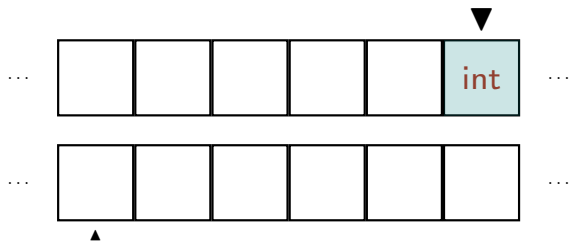
B

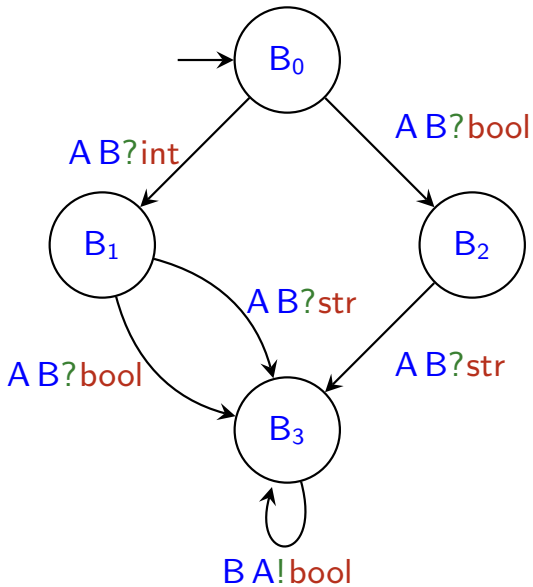
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A

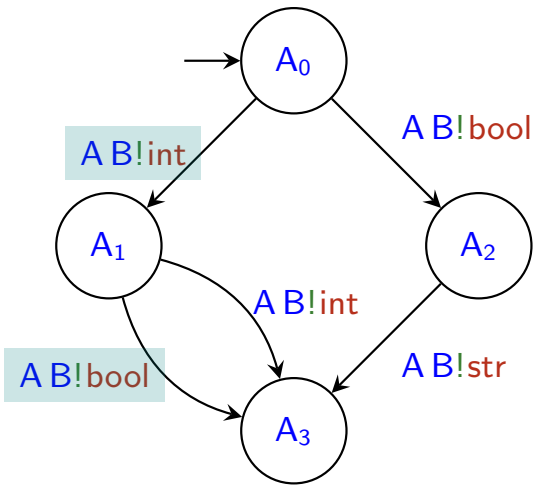




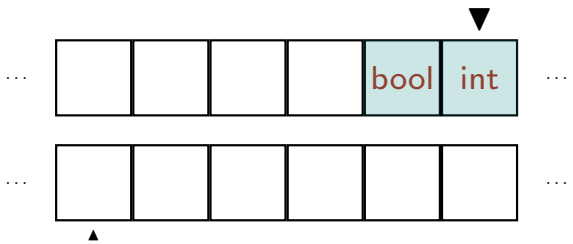
B

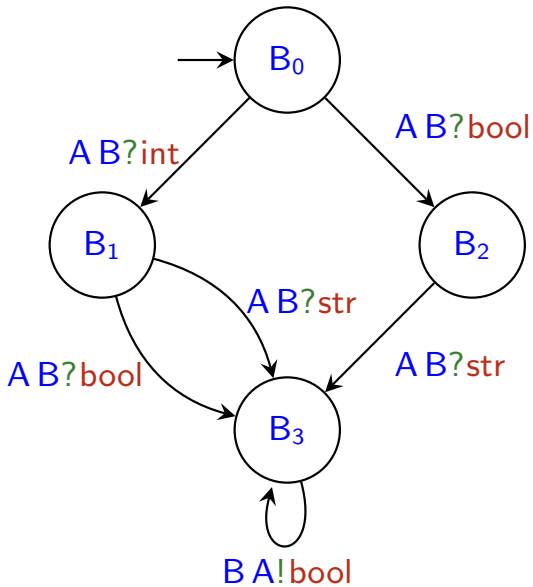
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A

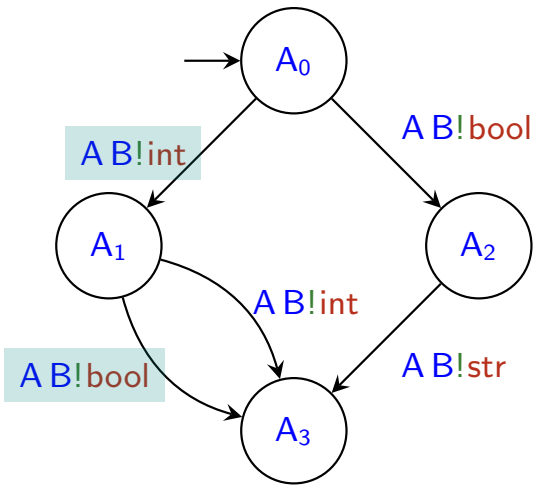




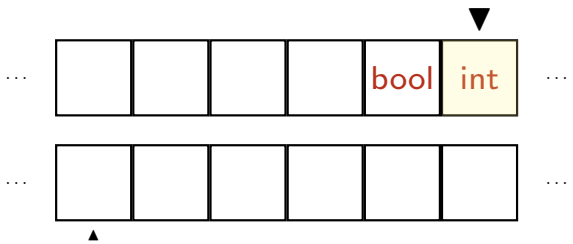
B

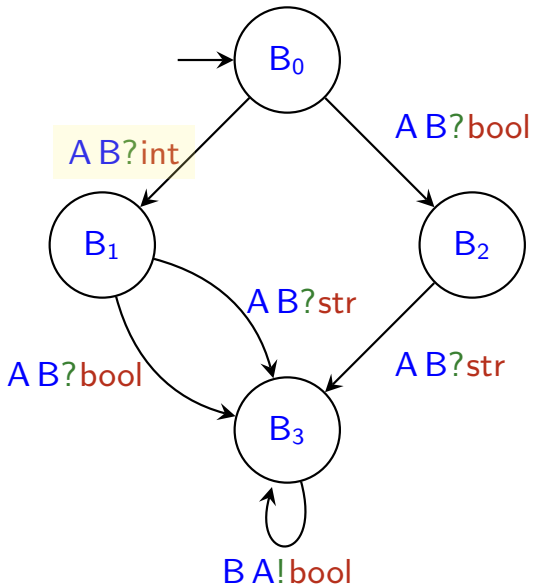
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A

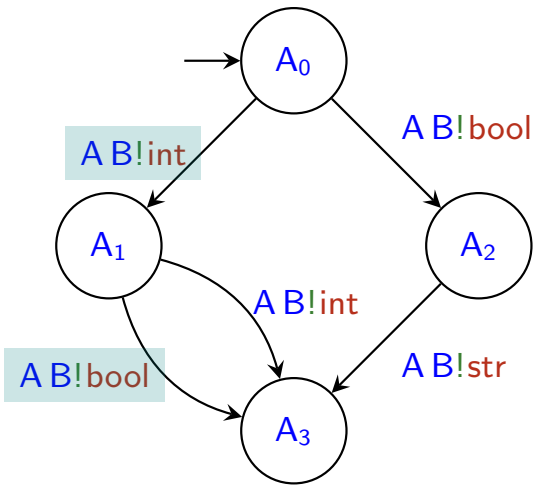




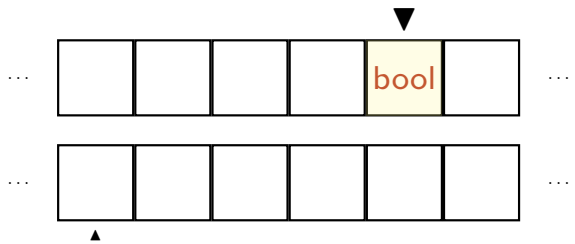
B

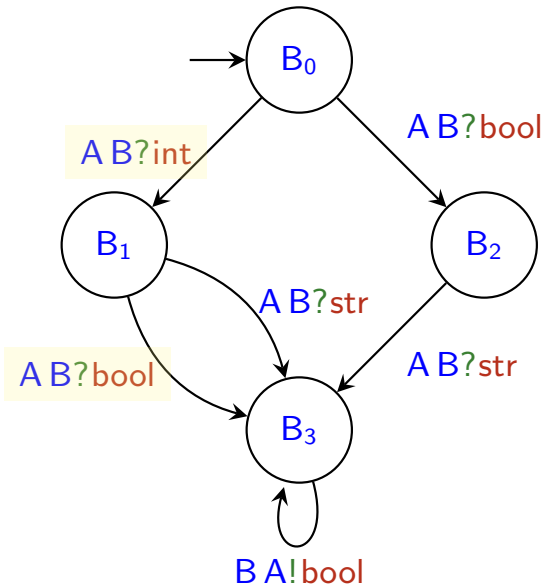
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A

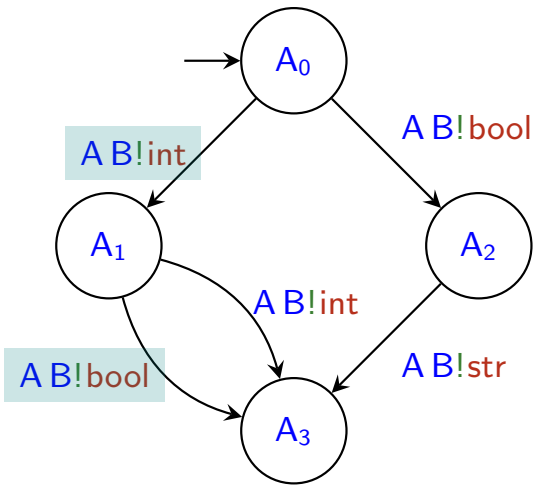




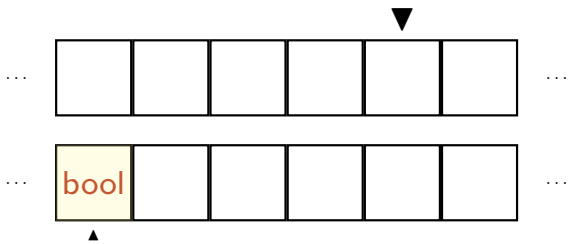
B

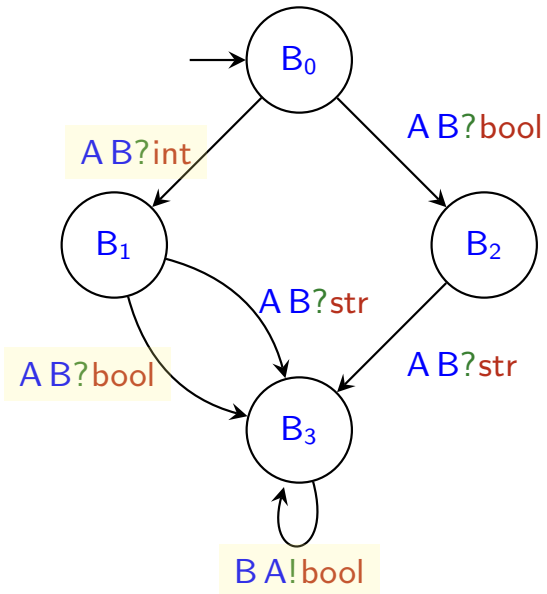
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A

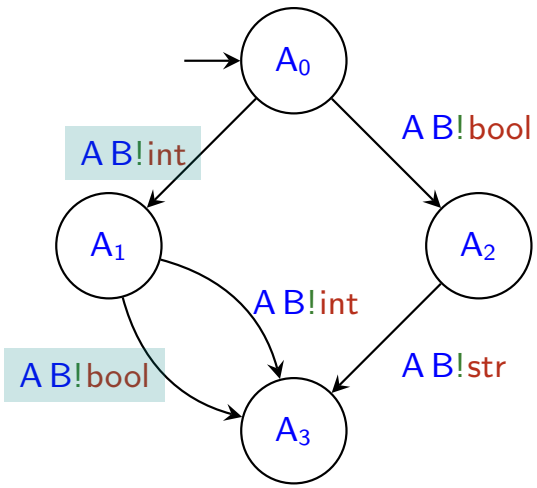




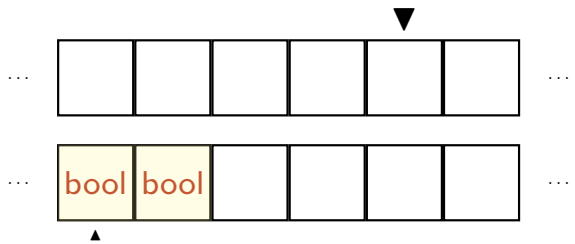
B

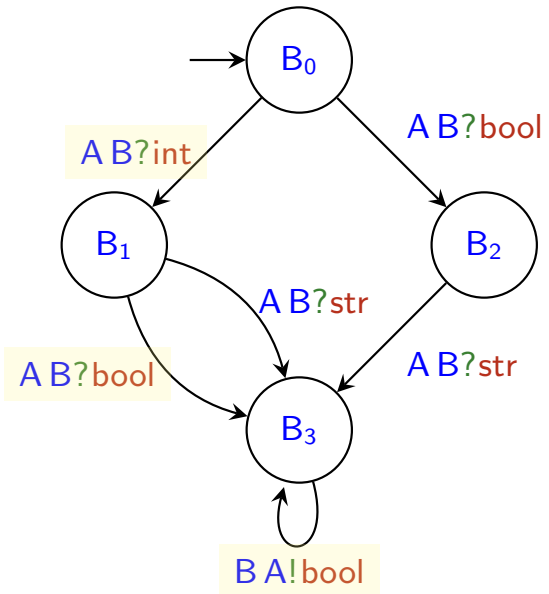
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A

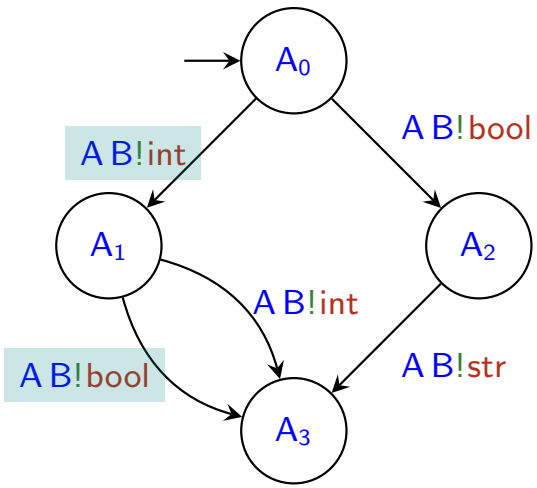




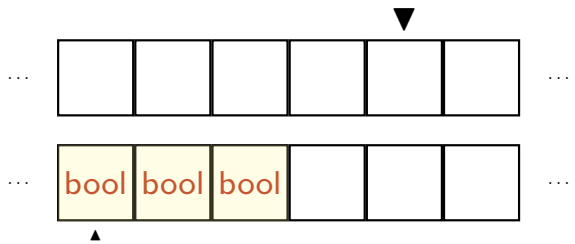
B

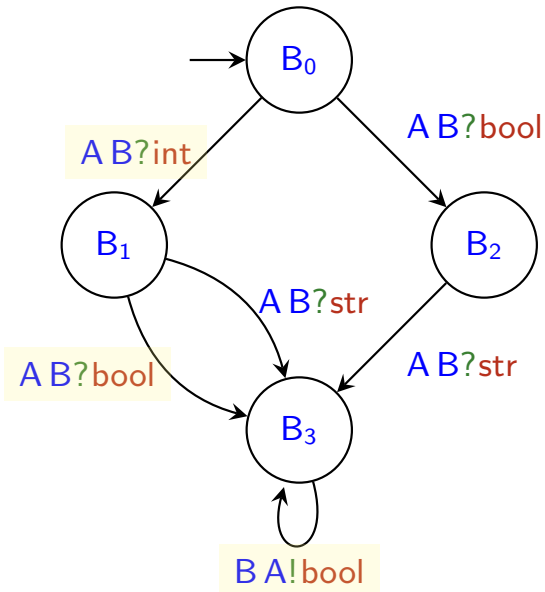
Local views, formally

Communicating finite state machines (Brand&Zafiropulo, 1983)



A





B

Global views, formally

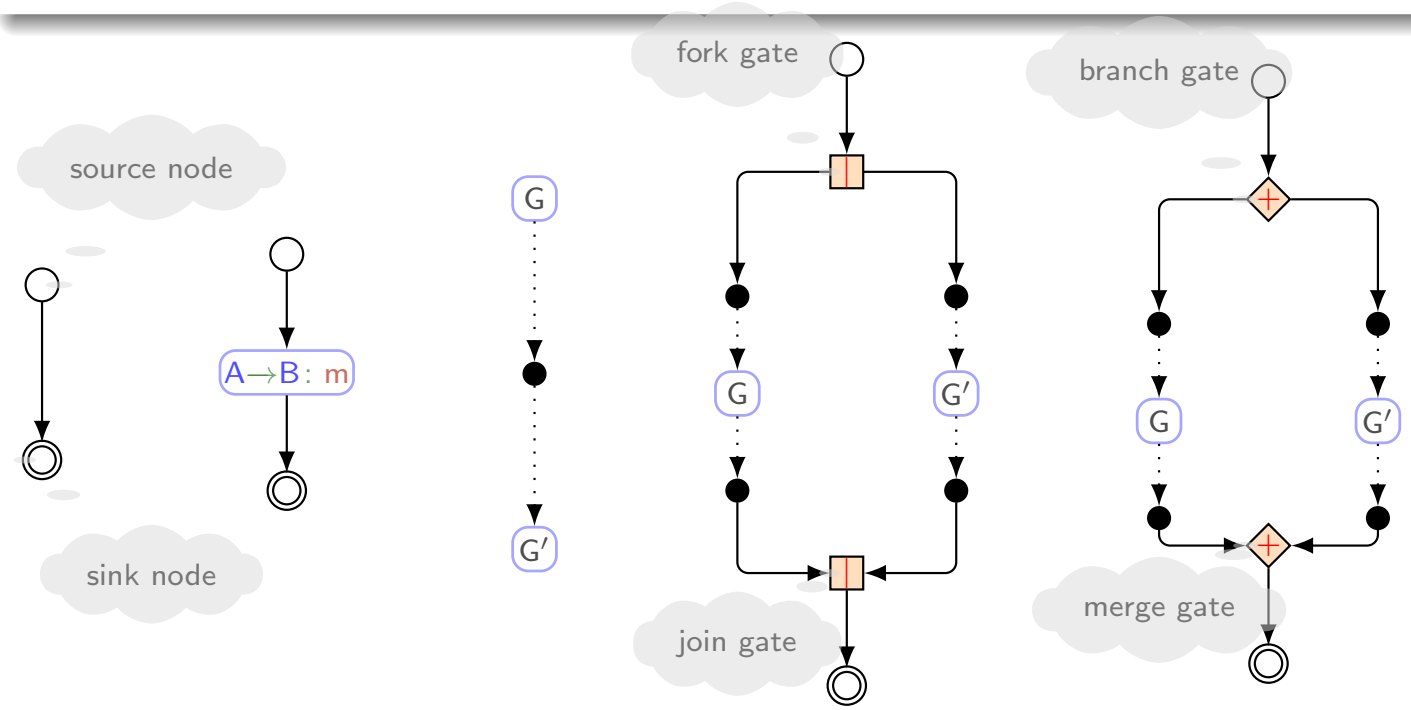
Global graphs

$$G ::= (o) \mid A \to B : m \mid G \mid G' \mid G; G' \mid G + G' \mid *G$$

Global views, formally

Global graphs

$$G ::= (o) \mid A \to B : m \mid G \mid G' \mid G; G' \mid G + G' \mid *G$$



Pomset semantics

Deciding on how to take decisions

In a branch $G_1 + G_2$

- there should be one active participant
- any non-active participant should be passive

Deciding on how to take decisions

In a branch $G_1 + G_2$

- there should be one active participant
- any non-active participant should be passive

Intuitively...

A is active when it locally decides which branch to take in a choice

B is passive when

- either B behaves uniformly in each branch
- or B "unambiguously understands" which branch A opted for through the information received on each branch

Deciding on how to take decisions

In a branch $G_1 + G_2$

- there should be one active participant
- any non-active participant should be passive

Intuitively...

A is active when it locally decides which branch to take in a choice

B is passive when

- either B behaves uniformly in each branch
- or B "unambiguously understands" which branch A opted for through the information received on each branch

Well-branchedness

When the above holds true for each choice, the choreography is well-branched. This enables correctness-by-design.

Class test

Which of the following global graphs is well-branched?

- $G_1 = A \to B : \mathsf{int} + A \to B : \mathsf{str}$
- $G_2 = A \to B : \mathsf{int} + (\mathsf{o})$
- $G_3 = A \to B : \mathsf{int} + A \to C : \mathsf{str}$
- $G_4 = \left(\begin{array}{l} A \to C : \mathsf{int}; A \to B : \mathsf{bool} \\ + \\ A \to C : \mathsf{str}; A \to C : \mathsf{bool}; A \to B : \mathsf{bool} \end{array} \right)$

Class test

Which of the following global graphs is well-branched?

- $G_1 = A \to B : \mathsf{int} + A \to B : \mathsf{str}$ 
- $G_2 = A \to B : \mathsf{int} + (\mathsf{o})$
- $G_3 = A \to B : \mathsf{int} + A \to C : \mathsf{str}$
- $G_4 = \left(\begin{array}{l} A \to C : \mathsf{int}; A \to B : \mathsf{bool} \\ \quad + \\ A \to C : \mathsf{str}; A \to C : \mathsf{bool}; A \to B : \mathsf{bool} \end{array} \right)$

Class test

Which of the following global graphs is well-branched?

- $G_1 = A \to B : \mathsf{int} + A \to B : \mathsf{str}$ 
- $G_2 = A \to B : \mathsf{int} + (\mathsf{o})$ 
- $G_3 = A \to B : \mathsf{int} + A \to C : \mathsf{str}$
- $G_4 = \left(\begin{array}{l} A \to C : \mathsf{int}; A \to B : \mathsf{bool} \\ \quad + \\ A \to C : \mathsf{str}; A \to C : \mathsf{bool}; A \to B : \mathsf{bool} \end{array} \right)$

Class test

Which of the following global graphs is well-branched?

- $G_1 = A \to B : \mathsf{int} + A \to B : \mathsf{str}$ 
- $G_2 = A \to B : \mathsf{int} + (\mathsf{o})$ 
- $G_3 = A \to B : \mathsf{int} + A \to C : \mathsf{str}$ 
- $G_4 = \left(\begin{array}{l} A \to C : \mathsf{int}; A \to B : \mathsf{bool} \\ \quad + \\ A \to C : \mathsf{str}; A \to C : \mathsf{bool}; A \to B : \mathsf{bool} \end{array} \right)$

Class test

Which of the following global graphs is well-branched?

- $G_1 = A \to B : \mathsf{int} + A \to B : \mathsf{str}$ 
- $G_2 = A \to B : \mathsf{int} + (\mathsf{o})$ 
- $G_3 = A \to B : \mathsf{int} + A \to C : \mathsf{str}$ 
- $G_4 = \left(\begin{array}{l} A \to C : \mathsf{int}; A \to B : \mathsf{bool} \\ \quad + \\ A \to C : \mathsf{str}; A \to C : \mathsf{bool}; A \to B : \mathsf{bool} \end{array} \right)$ 

Correctness-by-design via Choreographies

A Simple Exercise in BehAPI

Given B, a bank's API s.t.

- GET authReq :: authenticate; return authFail or granted
- GET authWithdrawal :: request cash; return allow or deny
- GET getBalance :: get balance; return balance

A Simple Exercise in BehAPI

Given B, a bank's API s.t.

- GET authReq :: authenticate; return authFail or granted
- GET authWithdrawal :: request cash; return allow or deny
- GET getBalance :: get balance; return balance

Develop A, the sw for an ATM machine

- GET auth :: authentication request; return authFail or granted
- GET withdraw :: request cash; return money or bye
- GET checkBalance :: check balance request; return balance
- ...

A Simple Exercise in BehAPI

Given B, a bank's API s.t.

- GET authReq :: authenticate; return authFail or granted
- GET authWithdrawal :: request cash; return allow or deny
- GET getBalance :: get balance; return balance

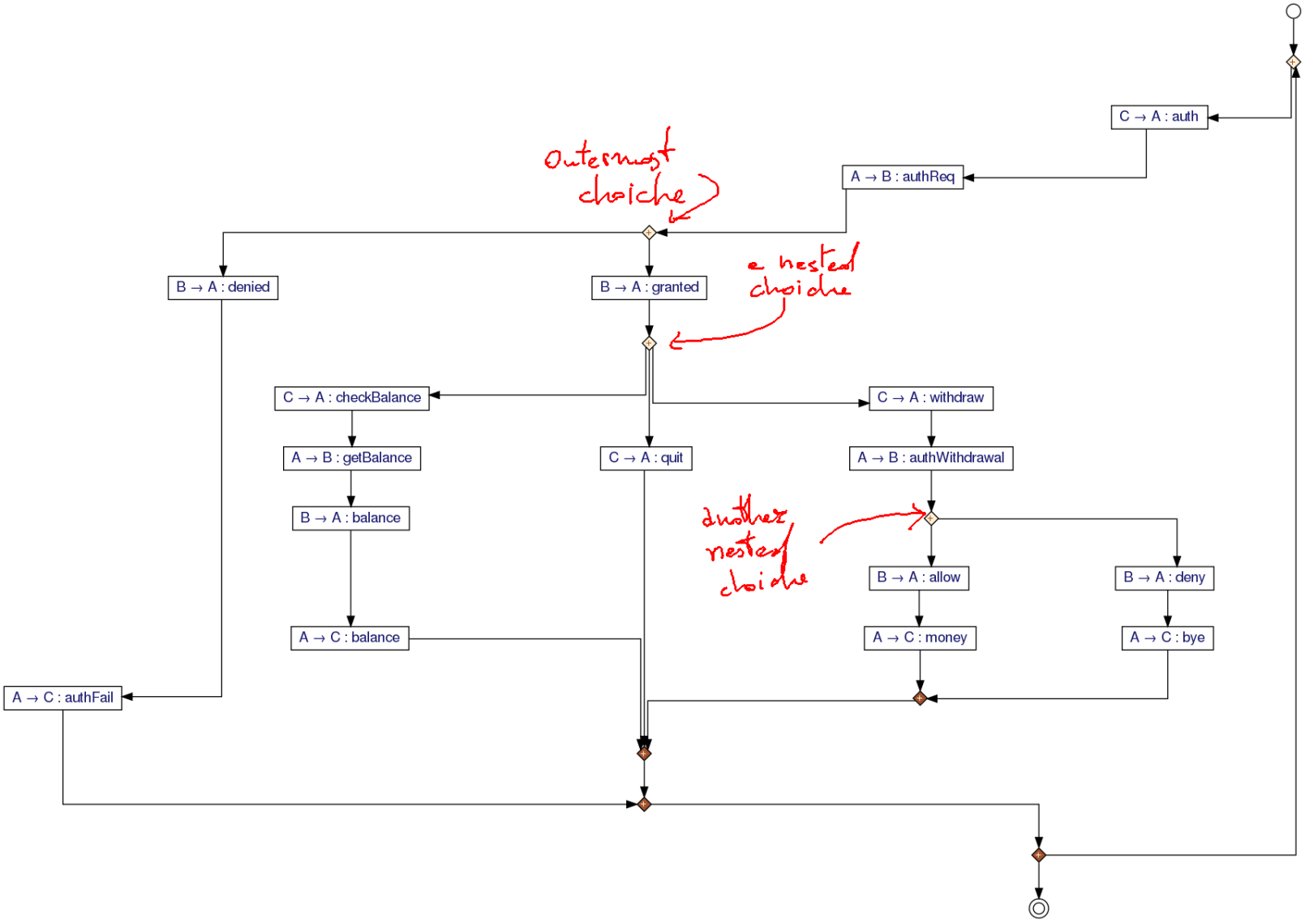
Develop A, the sw for an ATM machine

- GET auth :: authentication request; return authFail or granted
- GET withdraw :: request cash; return money or bye
- GET checkBalance :: check balance request; return balance
- ...

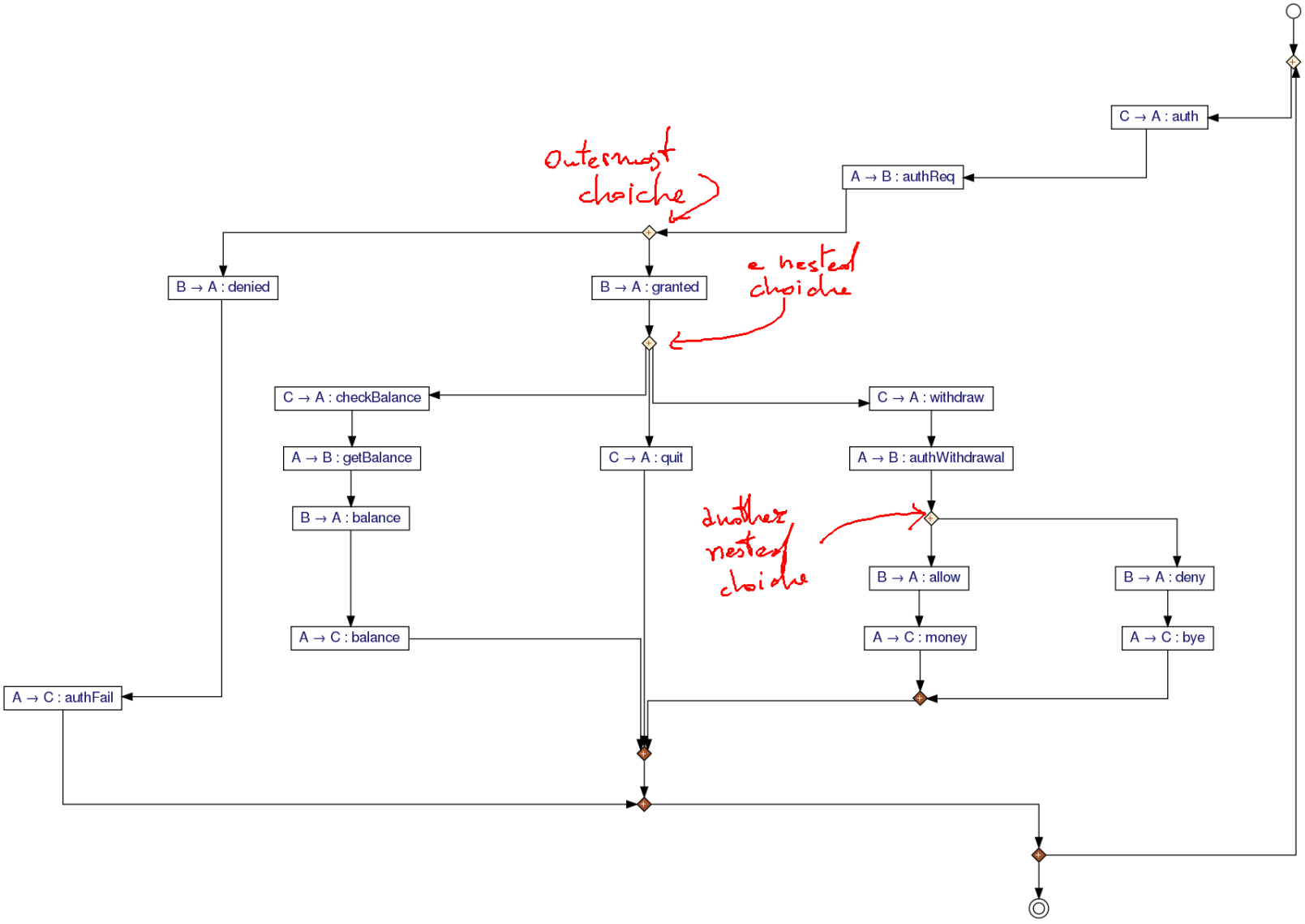
Modelling C, a fictional customer

- ...

Define the global view



Define the global view



hey eM: go to shell !

Debugging code ... “not for free”

Where do we start from?

- Is B wrong?
- Has C ventured on an unexpected communication pattern?
- Or is it A buggy?

Debugging code ... "not for free"

Where do we start from?

- Is B wrong?
- Has C ventured on an unexpected communication pattern?
- Or is it A buggy?

hey...wait a sec: the problem can't be in the code!

Debugging code ... "not for free"

Where do we start from?

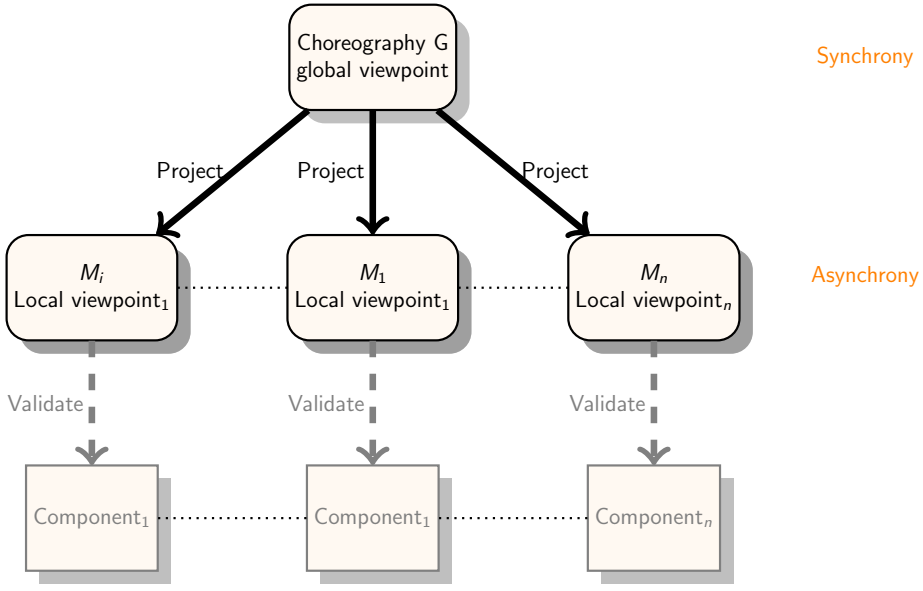
- Is B wrong?
- Has C ventured on an unexpected communication pattern?
- Or is it A buggy?

hey...wait a sec: the problem can't be in the code!

ehi eM: go to shell !

Choreography-driven testing

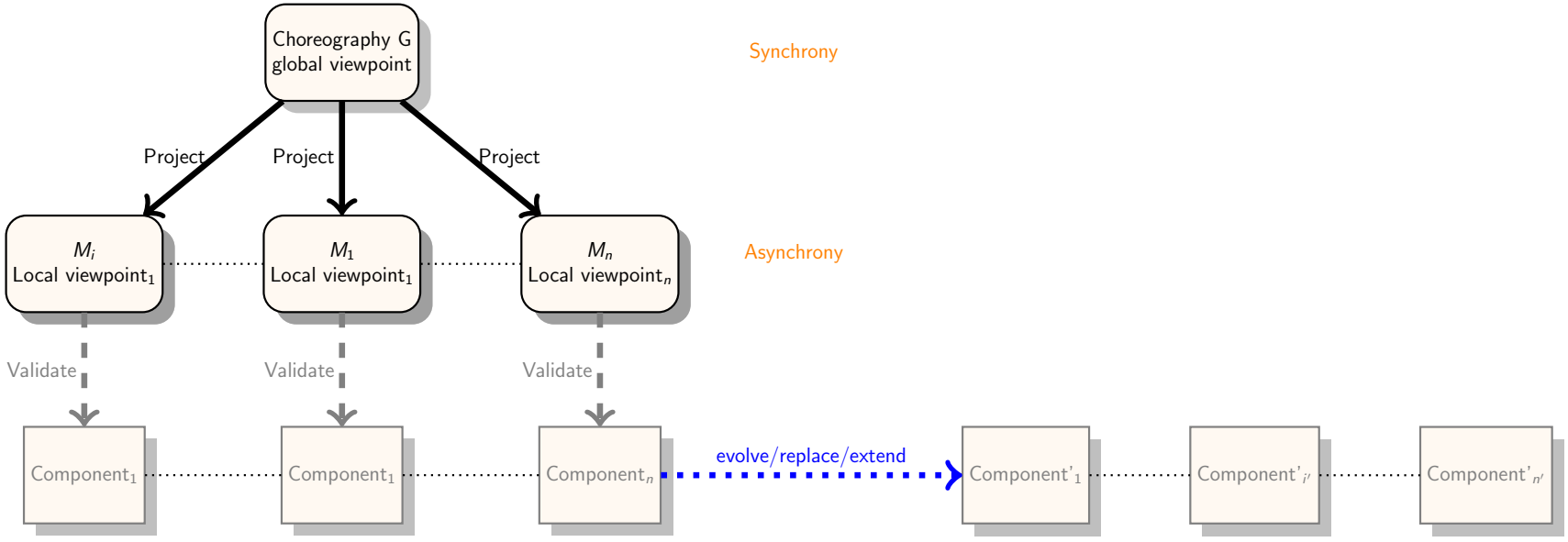
“Top-down”



Software changes

- Correctness-by-Design makes a lot of sense when going top-down

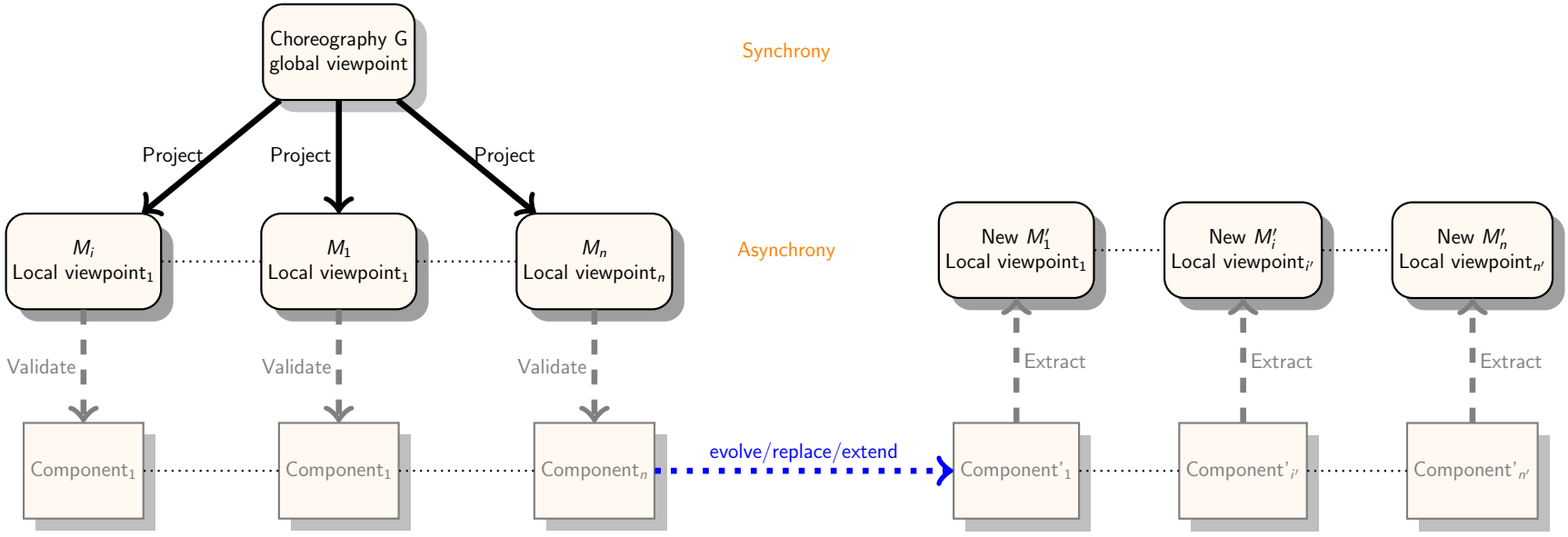
“Top-down” & “Bottom-up” approach



Software changes

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]

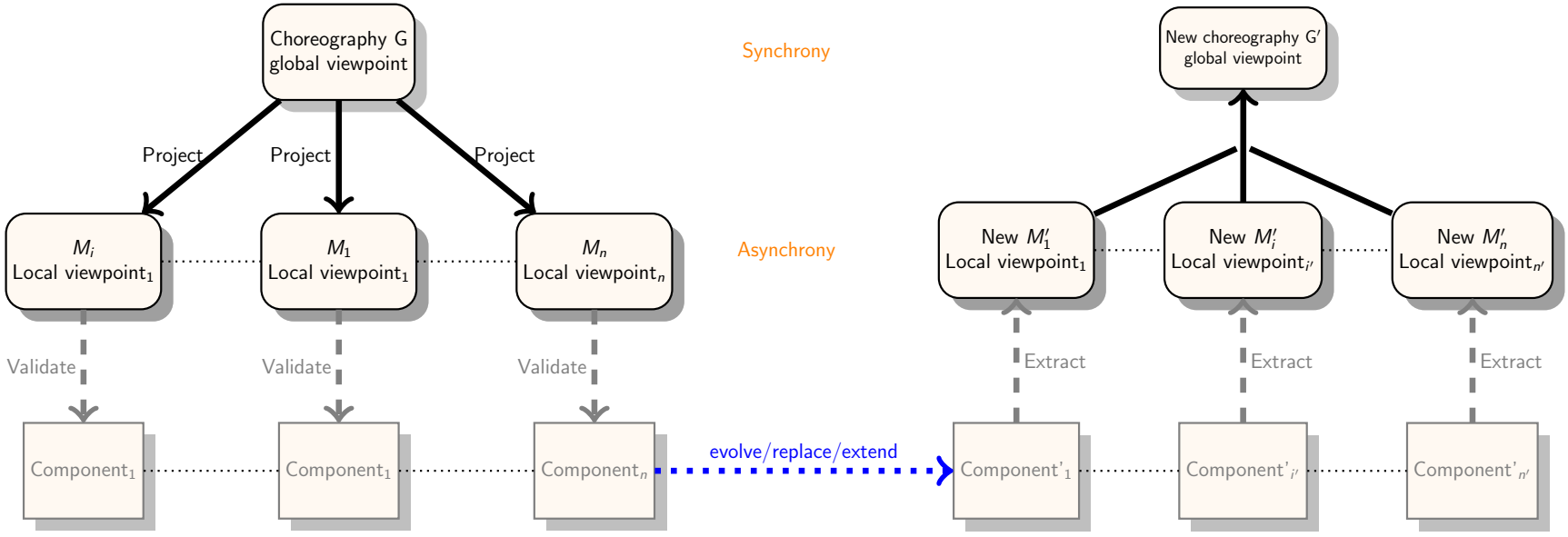
“Top-down” & “Bottom-up” approach



Software changes

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]
- Choreographies may help also going bottom-up

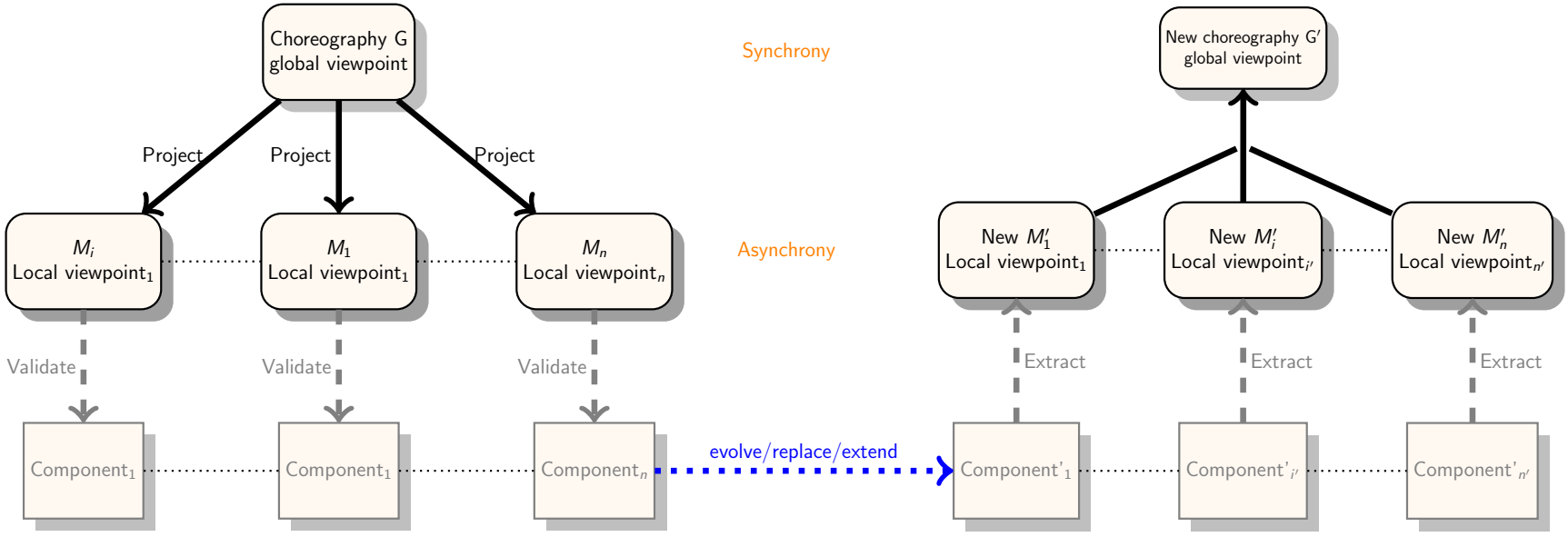
“Top-down” & “Bottom-up” approach



Software changes

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]
- Choreographies may help also going bottom-up

“Top-down” & “Bottom-up” approach



Software changes

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]
- Choreographies may help also going bottom-up
- ...but what if bugs are introduced?

Disruptions (Warning: do not do this at home!)

- Local computations deal with data

$$\mathsf{factorialServer}(\mathsf{req}, \mathsf{res}) = \mathsf{req}?n\,.\,\mathsf{res}!\mathsf{fact}(n) \qquad \text{where}$$

$$\mathsf{fact}(\texttt{int}\ \ n) = \texttt{if}\ \ 0 \leq n \leq 1\ \texttt{then}\ \ 1\ \texttt{else}\ \ n * \mathsf{fact}(n+1)$$

Disruptions (Warning: do not do this at home!)

- Local computations deal with data

$$\mathsf{factorialServer}(\mathsf{req}, \mathsf{res}) = \mathsf{req}?n\,.\,\mathsf{res}!\mathsf{fact}(n) \qquad \text{where}$$

$$\mathsf{fact}(\texttt{int}\ \ n) = \texttt{if}\ \ 0 \leq n \leq 1\ \texttt{then}\ \ 1\ \texttt{else}\ \ n * \mathsf{fact}(n-1)$$

...and this is still not right!

Disruptions (Warning: do not do this at home!)

- Local computations deal with data

$$\mathsf{factorialServer}(\mathsf{req}, \mathsf{res}) = \mathsf{req}?n\,.\,\mathsf{res}!\mathsf{fact}(n) \qquad \text{where}$$

$$\mathsf{fact}(\texttt{int}\ \ n) = \texttt{if}\ \ 0 \leq n \leq 1\ \texttt{then}\ \ 1\ \texttt{else}\ \ n * \mathsf{fact}(n-1)$$

  ...and this is still not right!

- Components' evolution may modify communication patterns

$$\begin{aligned}\mathsf{factorialServer}(\mathsf{req}, \mathsf{res}) = \quad & \mathsf{req}?n\,.\,\mathsf{res}!\mathsf{fact}(n) \\ & \& \\ & \mathsf{req}?0\,.\,\mathsf{res}!\text{`one'}\end{aligned}$$

Disruptions (Warning: do not do this at home!)

- Local computations deal with data

$$\text{factorialServer}(\text{req}, \text{res}) = \text{req}?n\,.\,\text{res}!\text{fact}(n) \qquad \text{where}$$

$$\text{fact}(\texttt{int}\ \ n) = \texttt{if}\ \ 0 \leq n \leq 1\ \texttt{then}\ \ 1\ \texttt{else}\ \ n * \text{fact}(n-1)$$

...and this is still not right!

- Components' evolution may modify communication patterns

$$\begin{aligned}\text{factorialServer}(\text{req}, \text{res}) = \quad & \text{req}?n\,.\,\text{res}!\text{fact}(n) \\ & \& \\ & \text{req}?0\,.\,\text{res}!\text{`one'}\end{aligned}$$

- Openness enables changes to the execution context that may alter "compatibility"

$$\begin{aligned}\text{factorialServer}^{++}(\text{req}, \text{res}) = \quad & \text{req}?n\ . \\ & \texttt{if}\ \ n < 0 \\ & \quad \texttt{then}\ \ \text{res}!\text{`error'} \\ & \quad \texttt{else}\ \ \text{res}!\text{fact}(n)\end{aligned}$$

Specifying tests

- Models abstract away from low level details
- Accurate models as oracles of expected outcome of tests

"Test projection"

Global specifications for model-driven testing can generate tests T_A for a participant A to probe A's ability to handle the choice with the other participants.

Specifying tests

- Models abstract away from low level details
- Accurate models as oracles of expected outcome of tests

"Test projection"

Global specifications for model-driven testing can generate tests T_A for a participant A to probe A's ability to handle the choice with the other participants. Take

$$G = A \to B : m; G_1 + A \to B : n; G_2$$

The test-projection for B are "programs" checking that:

- B can receive message m and then tests B in G_1
- B can receive message n and then tests B in G_2

Specifying tests

- Models abstract away from low level details
- Accurate models as oracles of expected outcome of tests

"Test projection"

Global specifications for model-driven testing can generate tests T_A for a participant A to probe A's ability to handle the choice with the other participants. Take

$$G = A \to B : m; G_1 + A \to B : n; G_2$$

The test-projection for A are "programs"

- that either waits for m and then tests for A in G_1
- or waits for n and then tests A in G_2

...yet tests should be "simple"

Generating tests

Model vs code level

Models generate abstract tests which have to be concretised

Previous example

The concretisations of the tests for B has to be such that the messages sent by A are actually consumed by B

Observability

In an asynchronous setting this may require to use mechanisms of the language at hand or lower level mechanisms.

Previous example

If G above is implemented in Erlang, then the test concretisation has to generate code to check A's mailbox.

Checking coverage

Quality

Testing requires to reach a "good" level of coverage

- there are usually infinitely many tests
  - identify a "good enough" finite subset
  - keep it "small"
- typical criteria for assessing coverage in model-based testing are state or transition coverage
- often choreographies are formalised in terms of transition systems

Black-box testing

Some components may not be "accessible": how to assess coverage in black-box testing?

- choreographies naturally provide (an approximation) of "all" components
- projections yields mock executable components (e.g., C and B in our ATM example)

Why are choreographies good for testing?

Choreographies can be used

- as test case specifications
- to automatically generate executable tests
- to assess coverage of test cases

Many challenges ahead

- What are (good) metrics?
- Is the homing problem easier with global models?
- When is a test passed/failed?
- Can bottom-up approach suggest good tests?
- Global vs local specifications: do they differ testing-wise?
- ...

Thank you!