Abstractions for Collective Adaptive Systems

Omar Inverso @ GSSI Catia Trubiani @ GSSI Emilio Tuosto @ GSSI

ISoLA 2021

Ρόδος
October 25-29, 2021

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

and by MIUR project PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems)

Take-away message

Our CAS equation

emergent behaviour $:=$ partial knowledge $+$ interaction $+$ local decision

Take-away message

Our CAS equation

emergent behaviour $:=$ partial knowledge $+$ interaction $+$ local decision

Emergent behaviour "by-design"

We want abstractions

- to specify CAS (ie to design emergent behaviour as easily as possible)
- to verify CAS

Take-away message

Our CAS equation

emergent behaviour $:=$ partial knowledge $+$ interaction $+$ local decision

Emergent behaviour "by-design"

We want abstractions

- to specify CAS (ie to design emergent behaviour as easily as possible)
- to verify CAS

Behavioural types for CAS

- shortcomings of existing behavioural types
- desiderata for suitable frameworks
  - an immediate by-product: quantitative analysis of CAS

– Prelude –

[Ruminating on CAS]

A simple scenario

Robots “pair up” to recharge batteries

```
def B(prefs, myID):
  # prefs is a finite list
    for charger in prefs:
      send("charging", myID) @ charger
      recv("stop")
```

```
def C(aID, aPID):
  while true:
    recv("charging", idNew)
    if choose(aID, idNew) == idNew:
      send("stop") @ aID
    else: send("stop") to idNew
```

A simple scenario

Robots "pair up" to recharge batteries

```
def B(prefs, myID):
  # prefs is a finite list
    for charger in prefs:
      send("charging", myID) @ charger
      recv("stop")
```

```
def C(aID, aPID):
  while true:
    recv("charging", idNew)
    if choose(aID, idNew) == idNew:
      send("stop") @ aID
    else: send("stop") to idNew
```

- Explicit addressing requires proper configuration (e.g., IDs should be unique, immutable, ...)

A simple scenario

Robots "pair up" to recharge batteries

```
def B(prefs, myID):
  # prefs is a finite list
    for charger in prefs:
      send("charging", myID) @ charger
      recv("stop")
```

```
def C(aID, aPID):
  while true:
    recv("charging", idNew)
    if choose(aID, idNew) == idNew:
      send("stop") @ aID
    else: send("stop") to idNew
```

- Explicit addressing requires proper configuration (e.g., IDs should be unique, immutable, ...)
- Reconfiguration is expensive (e.g., new charge stations $\implies$ update `prefs`...for all bots!)

A simple scenario

Robots "pair up" to recharge batteries

```
def B(prefs, myID):
  # prefs is a finite list
    for charger in prefs:
      send("charging", myID) @ charger
      recv("stop")
```

```
def C(aID, aPID):
  while true:
    recv("charging", idNew)
    if choose(aID, idNew) == idNew:
      send("stop") @ aID
    else: send("stop") to idNew
```

- Explicit addressing requires proper configuration (e.g., IDs should be unique, immutable, ...)
- Reconfiguration is expensive (e.g., new charge stations $\implies$ update `prefs`...for all bots!)
- Tedious with point-to-point communication
  - identify partners
  - information spreads with explicit communications
  - update local knowledge of agents

A simple scenario

Robots "pair up" to recharge batteries

```
def B(prefs, myID):
  # prefs is a finite list
    for charger in prefs:
      send("charging", myID) @ charger
      recv("stop")
```

```
def C(aID, aPID):
  while true:
    recv("charging", idNew)
    if choose(aID, idNew) == idNew:
      send("stop") @ aID
    else: send("stop") to idNew
```

- Explicit addressing requires proper configuration (e.g., IDs should be unique, immutable, ...)
- Reconfiguration is expensive (e.g., new charge stations $\implies$ update `prefs`...for all bots!)
- Tedious with point-to-point communication
  - identify partners
  - information spreads with explicit communications
  - update local knowledge of agents

Q: Is the code above correct? (Assuming we agree about what 'correct' means)

A simple scenario

Robots "pair up" to recharge batteries

```
def B(prefs, myID):
  # prefs is a finite list
    for charger in prefs:
      send("charging", myID) @ charger
      recv("stop")
```

```
def C(aID, aPID):
  while true:
    recv("charging", idNew)
    if choose(aID, idNew) == idNew:
      send("stop") @ aID
    else: send("stop") to idNew
```

- Explicit addressing requires proper configuration (e.g., IDs should be unique, immutable, ...)
- Reconfiguration is expensive (e.g., new charge stations $\implies$ update `prefs`...for all bots!)
- Tedious with point-to-point communication
  - identify partners
  - information spreads with explicit communications
  - update local knowledge of agents

Q: Is the code above correct? (Assuming we agree about what 'correct' means)

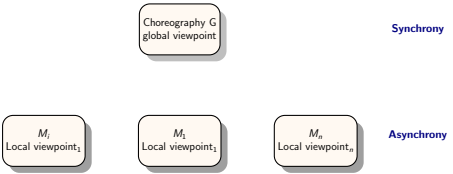
A: Well...it depends on whether (most?) bots pair up eventually

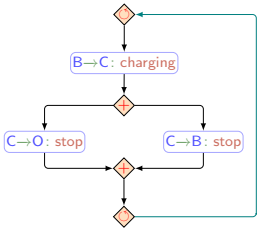
– Act I –

[Why Behavioural Types?]

Behavioural types & distributed applications

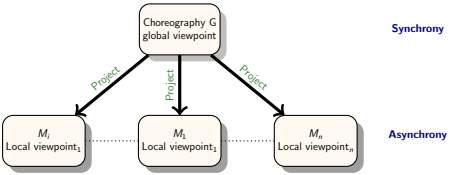
Natural support for choreographic design

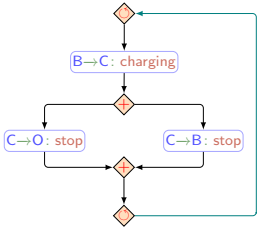




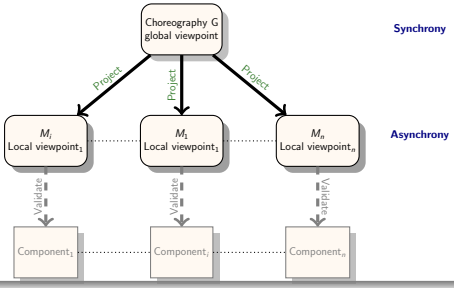
Behavioural types & distributed applications

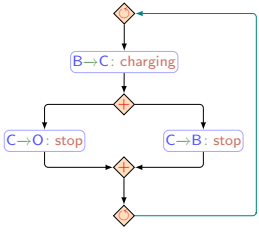
Natural support for choreographic design





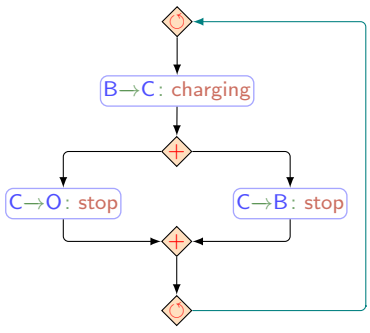
Natural support for choreographic design





Behavioural types & CAS

A shifty protocol

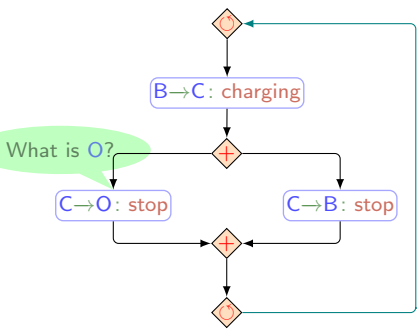


Not fit for purpose

Point-to-point communication is still fine, but...

Behavioural types & CAS

A shifty protocol



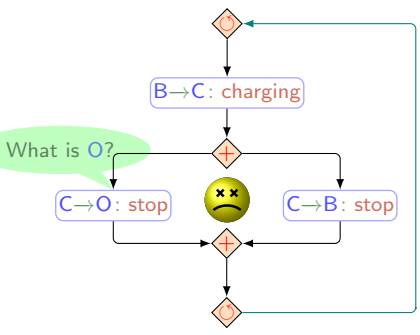
Not fit for purpose

Point-to-point communication is still fine, but...

- arbitrary replication not supported

Behavioural types & CAS

A shifty protocol



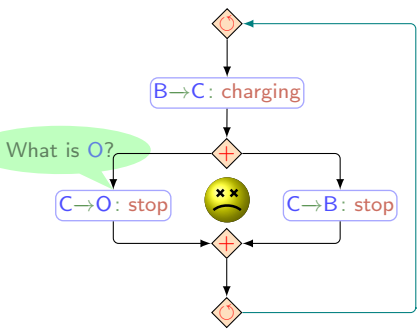
Not fit for purpose

Point-to-point communication is still fine, but...

- arbitrary replication not supported
- well-branchedness is violated

Behavioural types & CAS

A shifty protocol



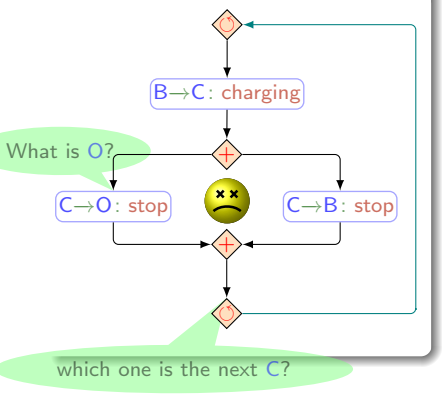
Not fit for purpose

Point-to-point communication is still fine, but...

- arbitrary replication not supported
- well-branchedness is violated
- deadlock IS the goal!

Behavioural types & CAS

A shifty protocol



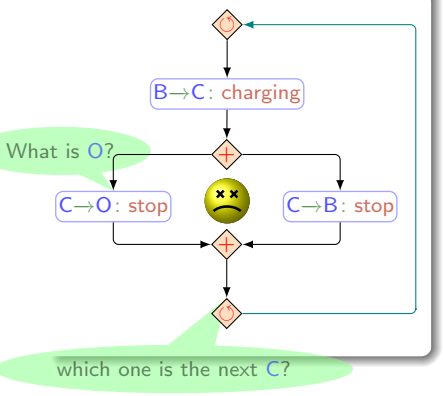
Not fit for purpose

Point-to-point communication is still fine, but...

- arbitrary replication not supported
- well-branchedness is violated
- deadlock IS the goal!
- reasoning about interactions is not enough: "correctness" depends on preference lists

Behavioural types & CAS

A shifty protocol



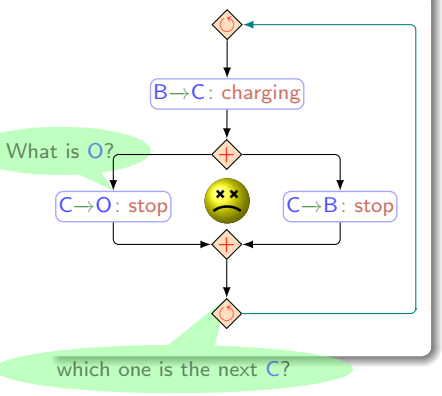
Not fit for purpose

Point-to-point communication is still fine, but...

- arbitrary replication not supported
- well-branchedness is violated
- deadlock IS the goal!
- reasoning about interactions is not enough: "correctness" depends on preference lists
- each instance plays a unique role

Behavioural types & CAS

A shifty protocol



Not fit for purpose

Point-to-point communication is still fine, but...

- arbitrary replication not supported
- well-branchedness is violated
- deadlock IS the goal!
- reasoning about interactions is not enough: "correctness" depends on preference lists
- each instance plays a unique role
- no quantitative analysis

AbC inspired behavioural types

New behavioural types

A new form of interaction

$$A_{|\rho} \xrightarrow{e \quad f} B_{|\sigma}$$

AbC inspired behavioural types

New behavioural types

A new form of interaction

$$\mathsf{A}_{|\rho} \xrightarrow{e \quad f} \mathsf{B}_{|\sigma}$$

interpreted as

- any A satisfying ρ
- generates an expression e
- which any B satisfying σ "can receive"
- provided that f matches e

AbC inspired behavioural types

New behavioural types

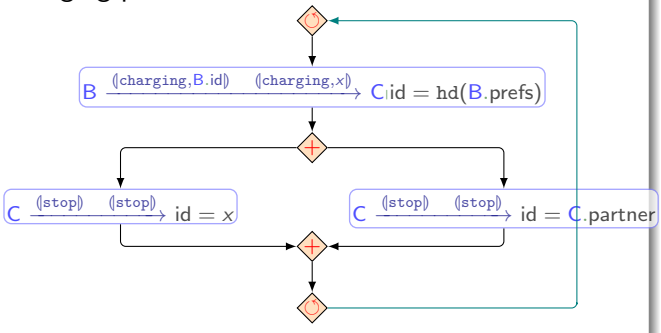
A new form of interaction

$$A \mid \rho \xrightarrow{e \quad f} B \mid \sigma$$

interpreted as

- any A satisfying ρ
- generates an expression e
- which any B satisfying σ "can receive"
- provided that f matches e

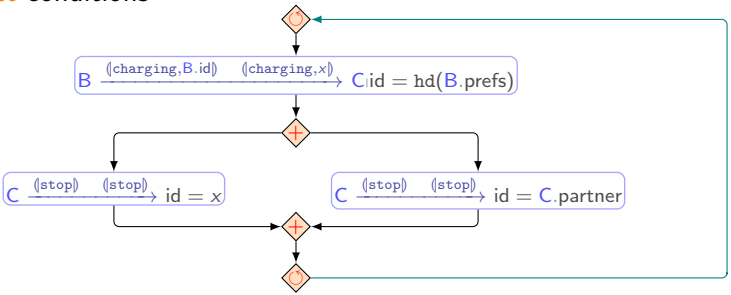
Charging protocol revisited[a]



[a] Tautologies omitted

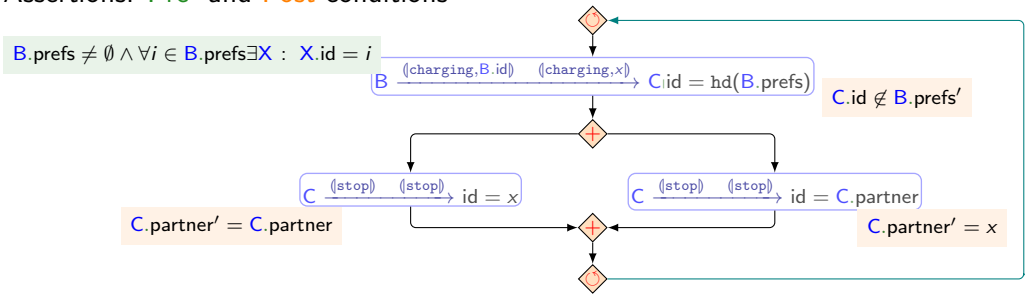
On correctness

Assertions: Pre- and Post-conditions



On correctness

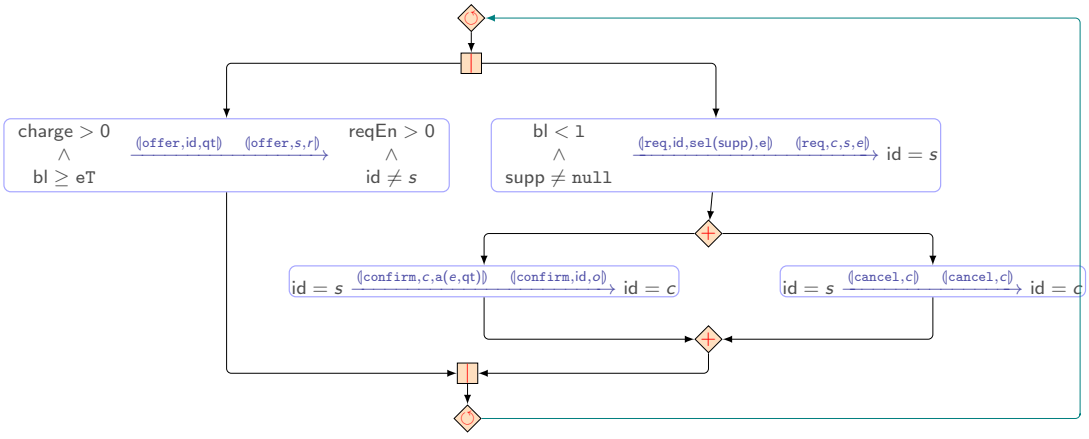
Assertions: Pre- and Post-conditions



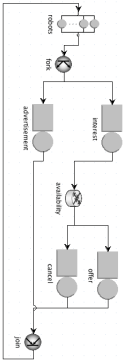
– Act II –

[Some immediate consequences]

Another battery-recharging scenario

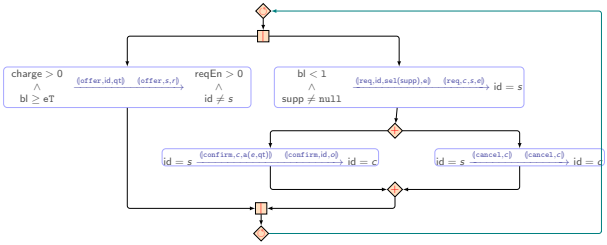


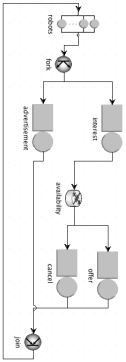
From behavioural types to QN



- A QN model is a rate-regulated service centres (ie set of resources) shared by jobs
- Requests arrive at a think-time dependent rate or at a job arrival rate

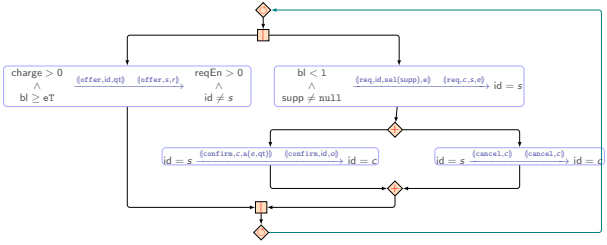
From behavioural types to QN

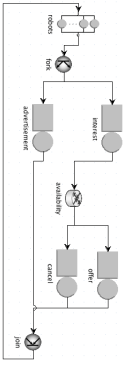




- A QN model is a rate-regulated service centres (ie set of resources) shared by jobs
- Requests arrive at a think-time dependent rate or at a job arrival rate

From behavioural types to QN



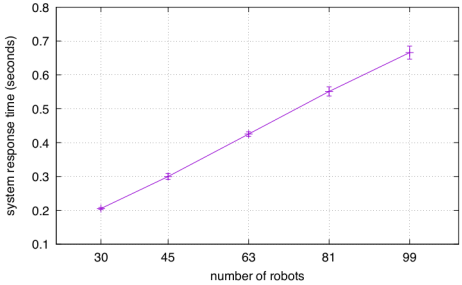


- A QN model is a rate-regulated service centres (ie set of resources) shared by jobs
- Requests arrive at a think-time dependent rate or at a job arrival rate

- $A|\rho \xrightarrow{e \quad e'} B|\rho' \quad \mapsto$ service centre
- $\square \quad \mapsto$ fork/join node
- $\diamond \quad \mapsto$ router node

Quantitative analysis

Parameter	Value
robots	wp=30
advertisement	$\lambda = 10$
interest	$\lambda = 10$
offer	$\lambda = 10$
cancel	$\lambda = 10$
availability	$\pi = 0.5$



– Epilogue –

[What's next?]

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

... whatever one can observe of a system is "emergent"

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

... whatever one can observe of a system is "emergent"

our view: "emergent behaviour" doesn't mean "unexpected/not designed"

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

... whatever one can observe of a system is "emergent"

our view: "emergent behaviour" doesn't mean "unexpected/not designed"

Some questions

- How can we formally characterise general properties of CAS?
  eg, stabilising, oscillating, diverging, ...

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

... whatever one can observe of a system is "emergent"

our view: "emergent behaviour" doesn't mean "unexpected/not designed"

Some questions

- How can we formally characterise general properties of CAS?
  eg, stabilising, oscillating, diverging, ...
- Can models be systematically used for "quantitative" analysis?
  Quantitative analysis seems anyway crucial

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

... whatever one can observe of a system is "emergent"

our view: "emergent behaviour" doesn't mean "unexpected/not designed"

Some questions

- How can we formally characterise general properties of CAS?
  eg, stabilising, oscillating, diverging, ...
- Can models be systematically used for "quantitative" analysis?
  Quantitative analysis seems anyway crucial
- To what extend this can be done statically?

Some thoughts

Back to our equation

emergent behaviour := partial knowledge + interaction + local decision

what do we actually mean by "emergent behaviour"?

... whatever one can observe of a system is "emergent"

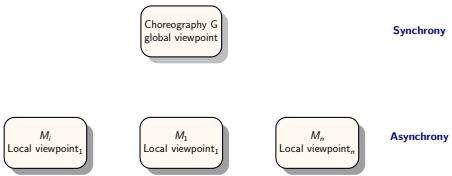
our view: "emergent behaviour" doesn't mean "unexpected/not designed"

Some questions

- How can we formally characterise general properties of CAS?
  eg, stabilising, oscillating, diverging, ...
- Can models be systematically used for "quantitative" analysis?
  Quantitative analysis seems anyway crucial
- To what extend this can be done statically?
- Can emergent behaviour be inferred?

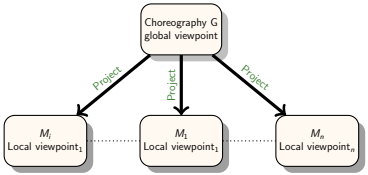
Bhehavioural type inference for CAS?

Round-trip engineering



Bhehavioural type inference for CAS?

Round-trip engineering

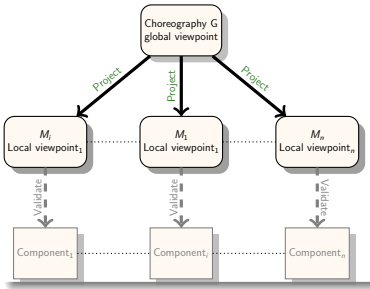


Synchrony

Asynchrony

Bhehavioural type inference for CAS?

Round-trip engineering

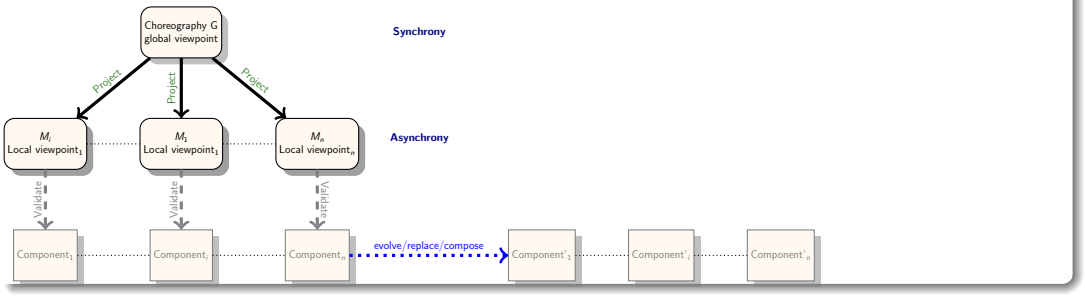


Synchrony

Asynchrony

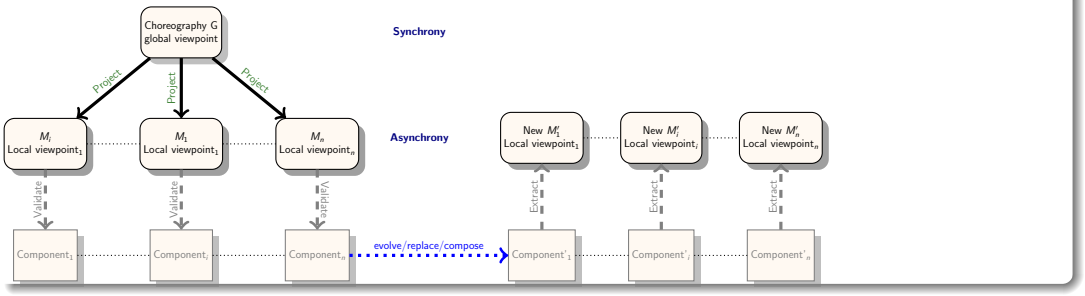
Bhehavioural type inference for CAS?

Round-trip engineering



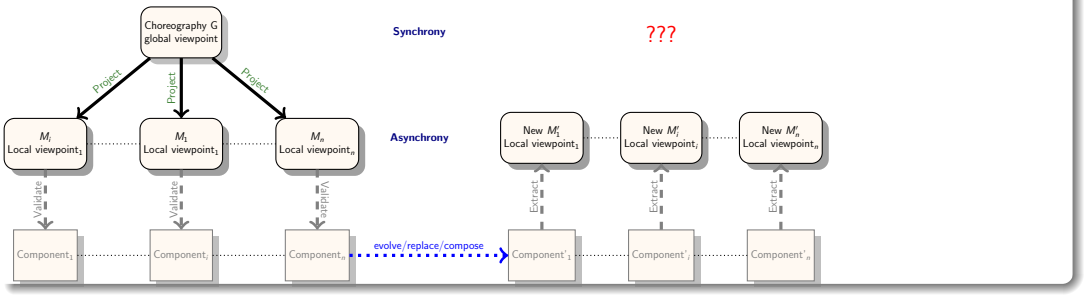
Bhehavioural type inference for CAS?

Round-trip engineering



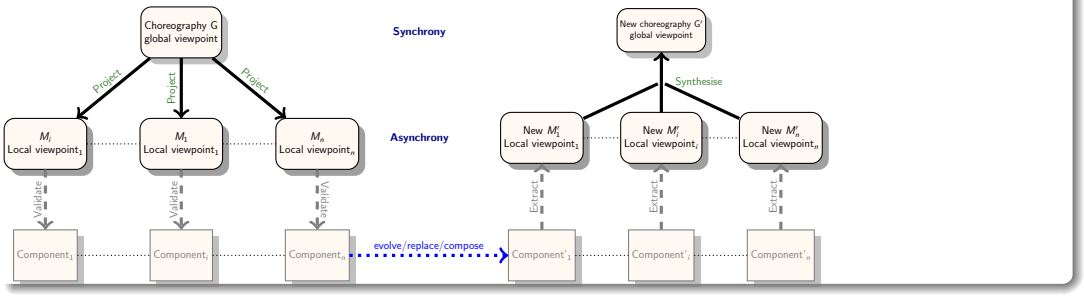
Bhehavioural type inference for CAS?

Round-trip engineering



Bhehavioural type inference for CAS?

Round-trip engineering



Outlook

- Identify typing disciplines
  - global types
  - local types
  - projection

  (it is not clear how much we can reuse from the literature)
- More precise relations with related work

  (expand sec. 7 of the paper)
- Can static specifications help to make attribute-based interaction (more) efficient?

Thank you!