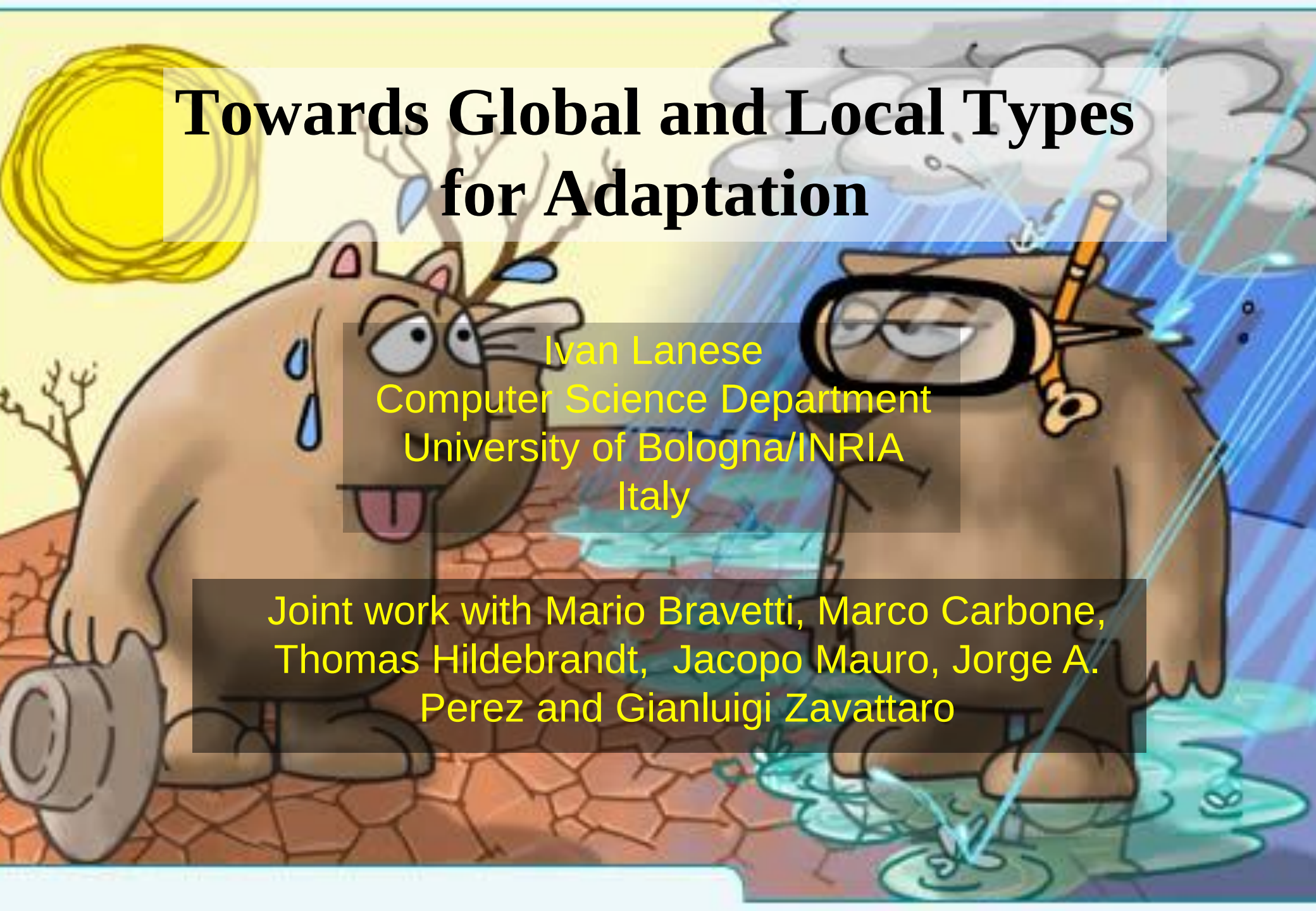


Towards Global and Local Types for Adaptation

Ivan Lanese
Computer Science Department
University of Bologna/INRIA
Italy

Joint work with Mario Bravetti, Marco Carbone, Thomas Hildebrandt, Jacopo Mauro, Jorge A. Perez and Gianluigi Zavattaro

Many distributed participants

Complex multiparty interactions

- Jacopo and Ivan were working on adaptive choreographies (with others)
- Marco was working on multiparty session types (with others)
- Mario, Jorge and Gianluigi were working on adaptable calculi (with others)
- Thomas was working on adaptable case management systems (with others)
- Mario, Marco, Thomas, Ivan, Jacopo, Jorge and Gianluigi wanted to start a collaboration on multiparty session types and adaptation

Message-based communication (e-mail)

- CaseStudy: Thomas → Mario
  Thomas sends a proposal for a case study to Mario
- CommentsReq: Mario → Ivan
  Mario asks Ivan for comments on the syntax
- WriteConcl: Gianluigi → Jorge
  Gianluigi asks Jorge to write conclusions
- Cut: Mario → Jacopo
  Mario asks Jacopo to cut the paper to respect the page limit

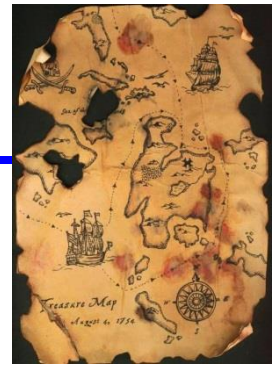
Participants act according to the choreography

- CommentsReq: Mario → Ivan; Comments: Ivan → Mario
- Ivan behavior should follow the type:
  $\text{CommentsReq}_{\text{Mario}};\overline{\text{Comments}}_{\text{Mario}}$
- Ivan code interleaves different sessions:
  k : $\text{CommentsReq}_{\text{Mario}}(x)$;
  k' : $\overline{\text{BuyBread}}_{\text{Wife}}$;
  k' : $\overline{\text{Bread}}_{\text{Wife}}$<1kg>;
  k : $\overline{\text{Comments}}_{\text{Mario}}$<comm.txt>
- Each session respects a given type

Unexpected adaptation needs

- External adaptation need:
  - Simon Gay announces BEAT II: the choreography is adapted to submit a work-in-progress to it
  - Marco notices that most of us will attend DisCoTec 2013: the choreography is adapted to exploit this occasion to work together
    - » Marco is a participant of the choreography, but
    - » DisCoTec attendance is not mentioned in the choreography
    - » It may be part of another choreography Marco is participating to
- Internal adaptation need:
  - Mario finds a bug in the definition of traces: the choreography is adapted to fix it

Our plan



- Write choreographies to describe complex multiparty interactions
- Derive a description of the behavior of each participant
- Type the code of each participant according to its local description(s)
  - The code may involve many interleaved sessions
- Typing ensures good properties
  - The code follows the expected protocol
  - No deadlock

Our plan

- Write choreographies to describe complex multiparty interactions
- Derive a description of the behavior of each participant
- Type the code of each participant according to its local description(s)
  - The code may involve many interleaved sessions
- Typing ensures good properties
  - The code follows the expected protocol
  - No deadlock

Adaptation

Adaptation

- Systems should live for long periods of time
- Systems should adapt to
  - Changes in the environment (new technologies, protocols, unexpected workload)
  - Changes in users minds (new requirements, changing business rules)
- Adaptation happens at runtime
- The system should be adapted with minimal disruption of functionalities
  - No shut down, recompile, and restart

How to face the unexpected?

- Adaptation details (frequently) not known when the system has been designed or even started
- To face those unexpected challenges something should come from outside
  - New code
  - Exploiting the new technology, defining the new business rule, ...
- The system should provide an interface to
  - Interact with an adaptation middleware
  - Get new code
  - Combine it with the existing code

Internal vs external updates

- External updates
  - New code from the environment
    - » A participant of another choreography
    - » The human user via some interface
  - Fundamental to deal with unexpected events
- Internal updates
  - New code from a participant of the choreography
  - Towards another area of the same choreography
  - Useful as a programming construct, e.g., for error handling
  - Enhances compositionality
  - Specifying the choreographies and the updates in the same language useful for refinement

Adaptation and multiparty session types

- Lots of works on adaptation exist
  - Our main contribution is not an innovative way of doing adaptation
- Formal approaches emerging only very recently
  - Our main contribution is in guaranteeing desirable properties
- Trade off between
  - Allowing substantial adaptations
    - » One would be able to change everything
  - Preserving good properties
    - » Easier if one changes very little
    - » Easier if one knows in advance what is changing and how
- Adaptive multiparty session types provide a good trade off

Adaptation constructs

- Impossible to guarentee good properties if adaptations can happen everywhere
  - We need a construct to specify where adaptation can happen
  - We call it a scope
  - A scope contains code, to be executed if no adaptation occurs
  - Running scopes can be adapted too (also from inside)
- A construct is needed for internal update
  - Should provide the new code for a given scope
- Similar constructs at the level of choreographies, endpoints and code

Constructs for external update

- None
- External updates come from outside
  - Not specified in the choreography
  - The system does not know how things will change
- We add external updates to the semantics, extending the notion of traces
- External updates are updates coming from a parallel (unspecified) choreography

Choreography language

- Composed by interactions of the form
  CommentsReq: Mario → Ivan
- Standard composition operators:
  sequence ; parallel | choice + Kleene star *
- Two operators for adaptation
  - X:T[C] scope with name X executing choreography C with set of roles (at most) T
  - $X_r\{C\}$ internal update of a scope done by role r, putting into the scope with name X the new choreography C

Endpoint language

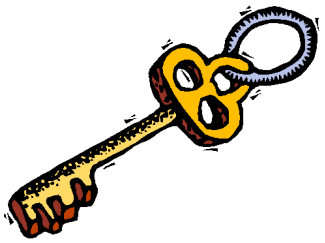
- Processes composed by inputs and outputs of the form $\overline{\text{CommentsReq}}_{\text{Ivan}}$ $\text{CommentsReq}_{\text{Mario}}$
- The same composition operators as before:
  sequence ; parallel | choice + Kleene star *
- Two operators for adaptation
  - X[P] scope with name X executing process P
  - $X_{(r1,...,rn)}\{P_1,...,P_n\}$ update of a scope X sending process P_i to endpoint r_i
    - » A single update involves multiple endpoints
- A system description is a parallel composition of endpoints
  - Each endpoint has a name and executes a process

Projection

- Allow one to automatically derive from a choreography the description of each endpoint
- Moving from more abstract to more concrete
- Op: $r \rightarrow s \mid r = \overline{Op}_s$
  Op: $r \rightarrow s \mid s = Op_r$
  Op: $r \rightarrow s \mid r' = 1$
- $X{:}T[C] \mid r = X[C|r]$ if r in T
  $X{:}T[C] \mid r = 1$ otherwise
- $X_s\{C\} \mid r = X_{(r1,...,rn)}\{C|r_1,...,C|r_n\}$ if r=s, $type(X)=\{r_1,...,r_n\}$
  $X_s\{C\} \mid r = 1$ otherwise
- Other operators are projected homomorphically

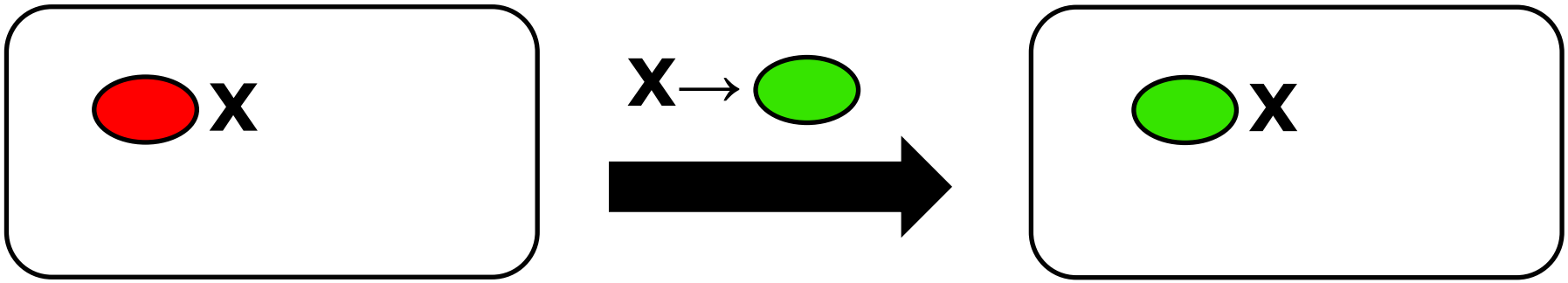
Expected result

- The traces of the projected system are included in the traces of the choreography
  - For all possible adaptations
- Holds only for well formed choreographies and adaptations
- Key challenge: ensuring that all the participants agree on where we are in the choreography
  - Which branch has been taken in a choice
  - Whether a given scope should be adapted or not
  - Which is the new code for a given scope
  - When one should stop executing the old code and start executing the new one

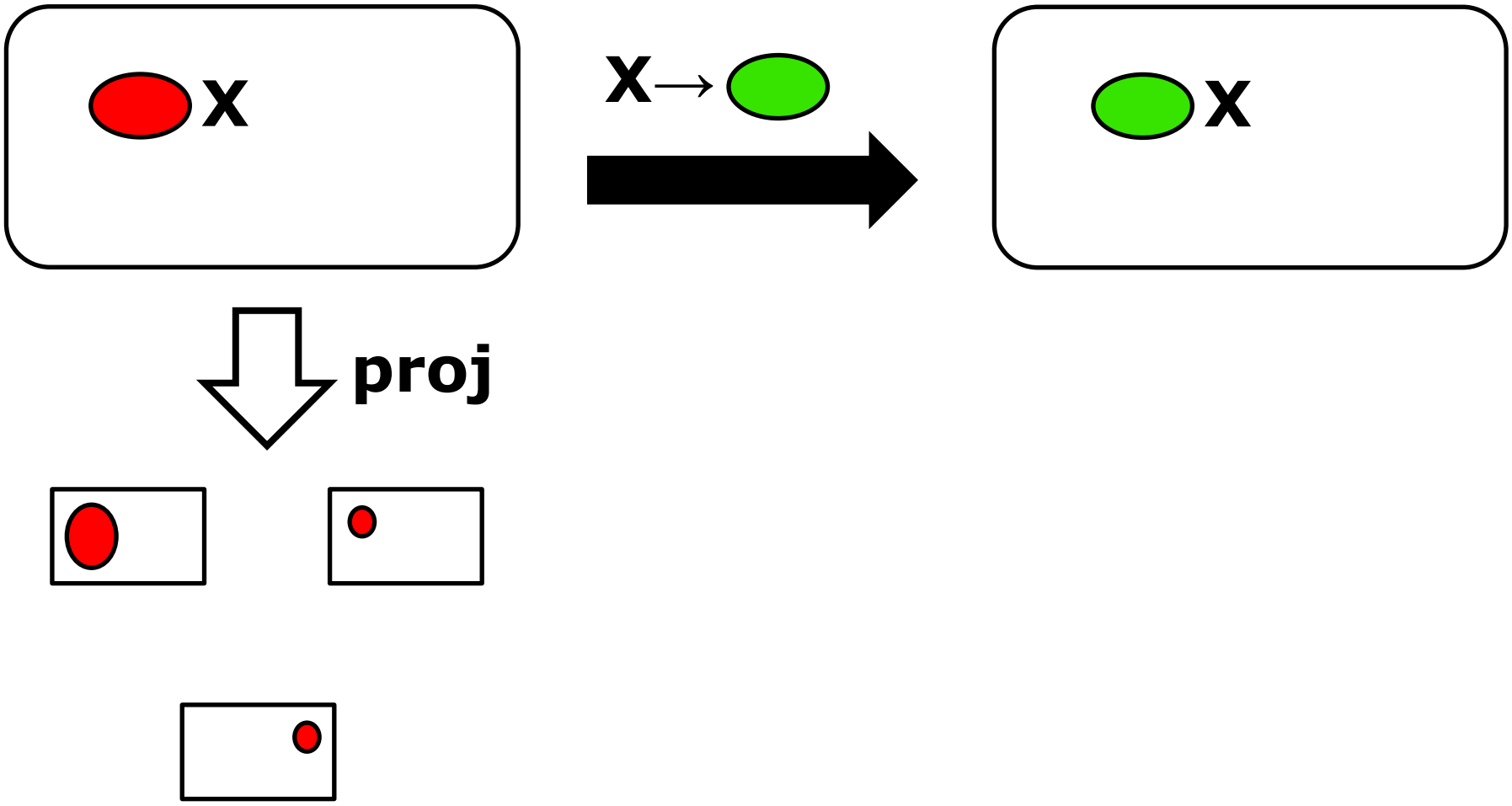


- Semantics carefully crafted to ensure this

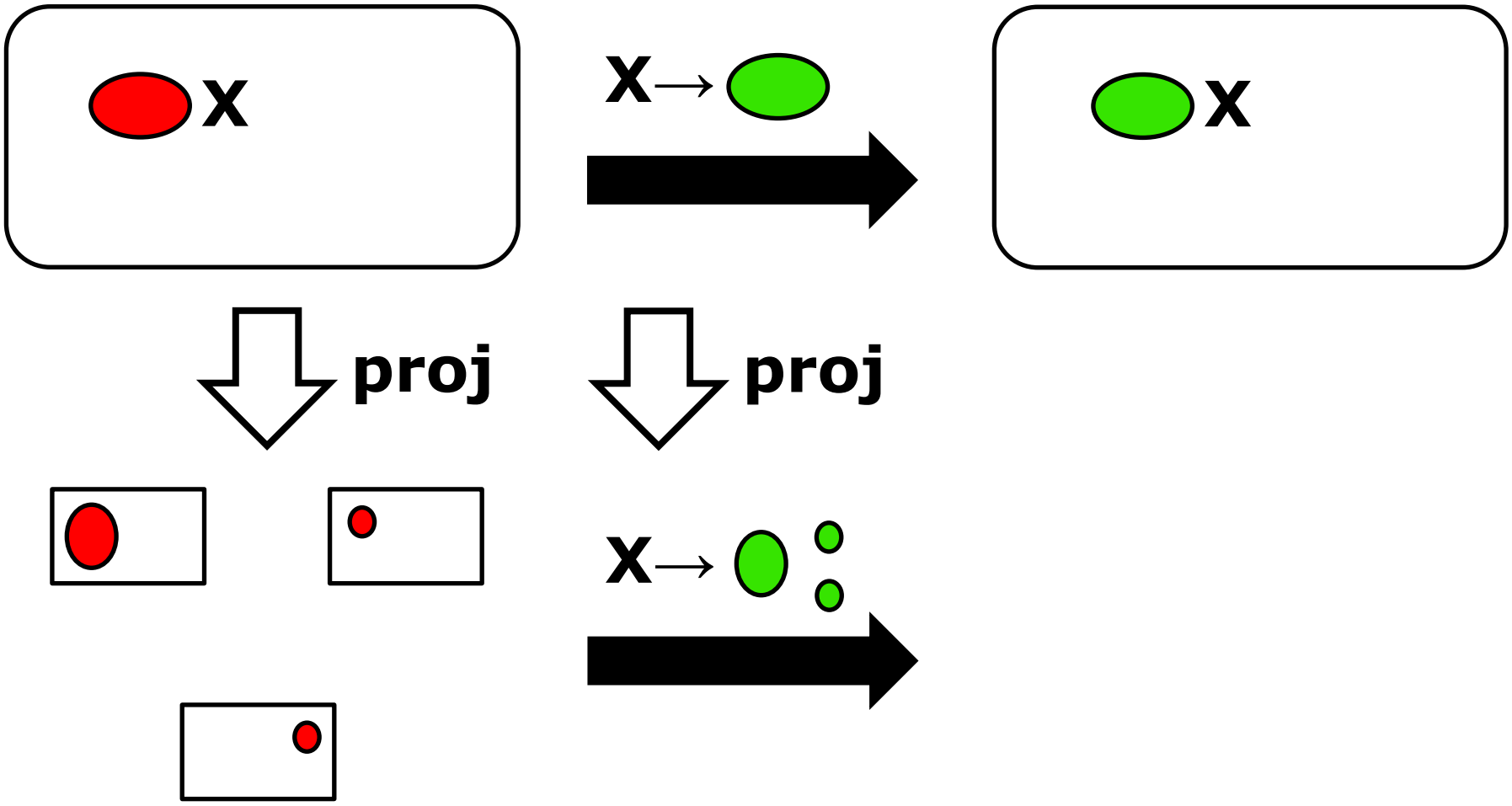
Adapting a scope, graphically



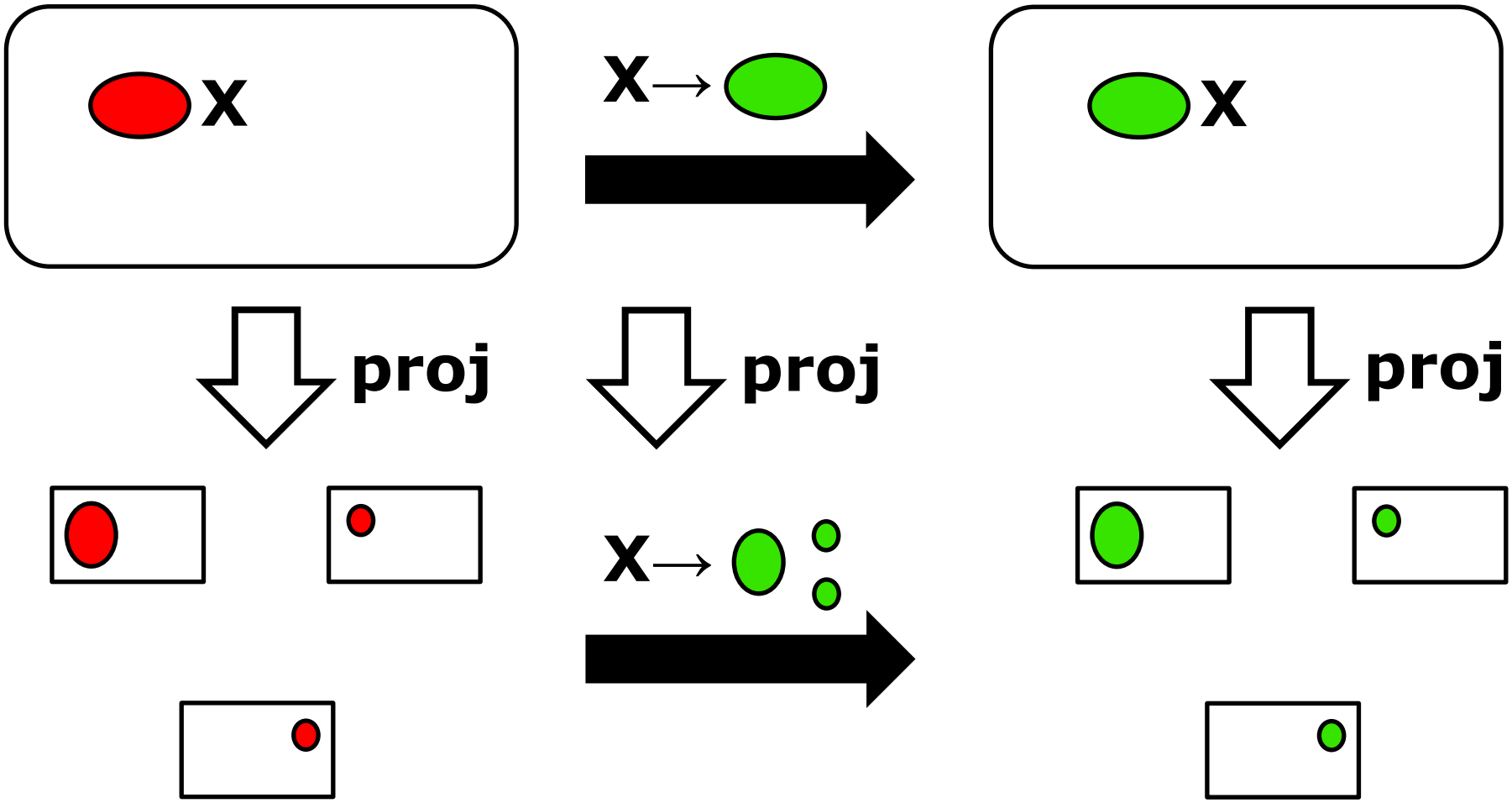
Adapting a scope, graphically



Adapting a scope, graphically



Adapting a scope, graphically



And now the foggy part

Typing a concrete language

- We see endpoint processes as protocol types for programs written in a more concrete language
- A program will execute different sessions in interleaving
  - Ivan program will execute the ‘Working on adaptive session types’ session, the ‘Take care of family’ session, ...
- Each session follows a protocol, defined by the projection of the corresponding choreography on the chosen participant
- This ensures that the good properties of the protocols are reflected in the program
  - More troubles come from the interleaving of sessions

Adapting interleaved sessions

- To adapt the code of a participant
  - All the protocols executed by the code should allow for the adaptation
    - » They should all feature a scope with the given name
  - The adaptation may come from one of them or from outside
- Adapting one participant requires to update the other participants too
  - May be involved in other protocols
  - More participants may need to be considered

Current state

- This is a work in progress
- What we have
  - Syntax and semantics for choreographies and endpoints, projection
  - Similar to adaptive choreographies [LaneseEtAl2013], but differs on some key design choices
    - » Allowing internal updates
    - » Abstract synchronization mechanism
  - A more serious example in the paper
- What we are still working on
  - Correctness proof, concrete language, typing rules, correctness of typing

Future work

- Complete the current work
- Fully understand the interplay between interleaved sessions and adaptation
- Refinement
  - This was our original motivation
  - Having internal updates motivated (also) by this

End of talk