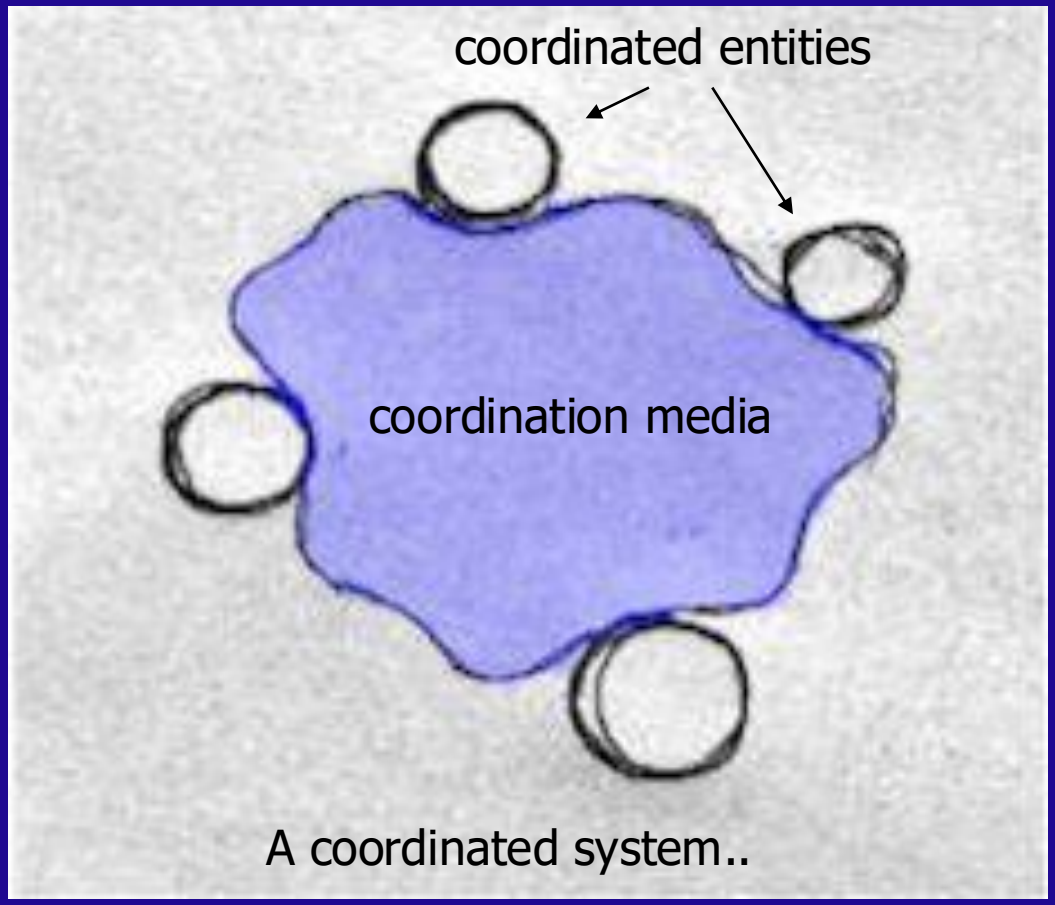
Extending ReSpecT for Multiple Coordination Flows

Alessandro Ricci, Andrea Omicini, Mirko Viroli
DEIS, Università degli Studi di Bologna, Cesena, Italy
{aricci,aomicini,mviroli}@deis.unibo.it

Coordination



ReSpecT coordination model

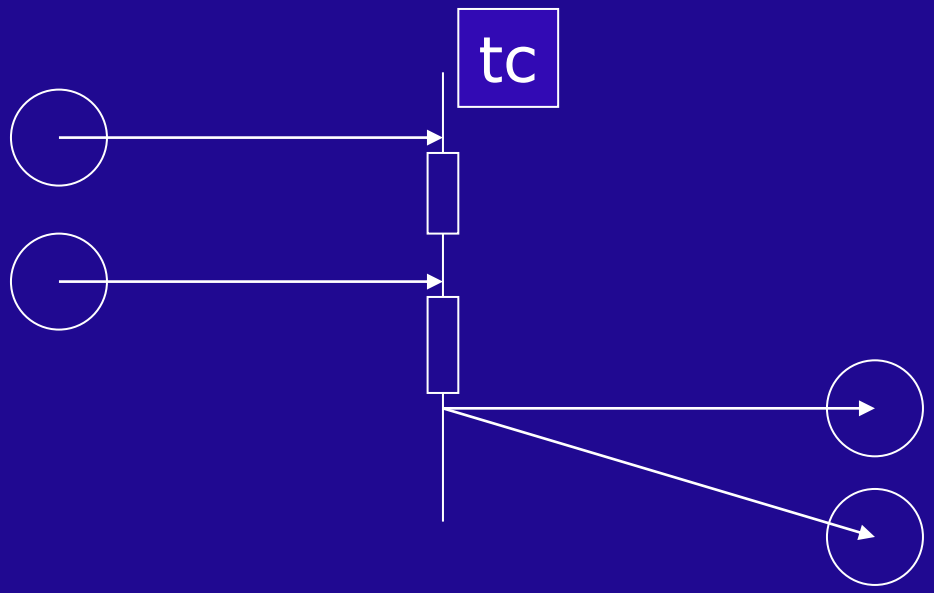
- ReSpecT Tuple Centre (TC)
  - Tuple space programmed with specification tuples (ST)
  - Altering the behaviour of a default Linda tuple space
- Behaviour
  - When an input event arrives or an output event is to be produced
  - reactions are fired (depending on the programmed behaviour)

  $\Rightarrow$ altering the underlying tuple space

  $\Rightarrow$ firing new reactions
- Features
  - Prolog-like programming of coordination media
  - Minimal (and Expressive)

Coordination flows

- Each ReSpecT tuple centre conceptually represents a single coordination flow
  - is a single (conceptually localised) reactive abstraction
  - coordinates a group of coordinated entities (coord. locality)
  - according to the programmed policy



Coordination Infrastructures

- It is possible to have a single TC for the whole system
  - but it would couple conceptually decoupled flows!
  - it is better to provide multiple TCs
- ReSpecT TCs are the basis for coordination infrastructures
  - TuCSoN: each node has one or more TC
  - LuCe: location-unaware group of TCs
- No interaction between different TCs
  - Multiple disjoint coordination flows

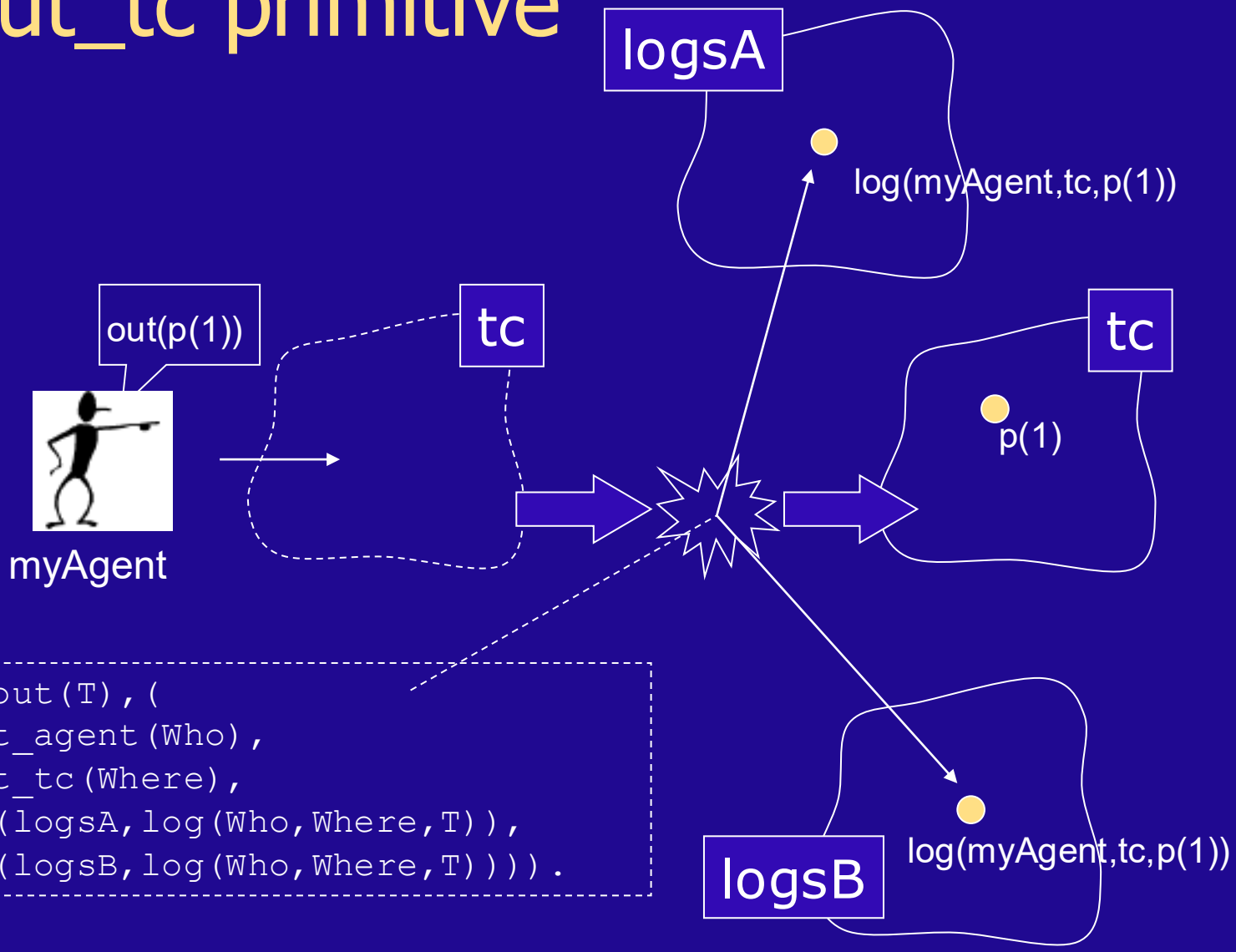
Multiple flows and event scenarios

- Need to deal with
  - open environments
  - unpredictable and possibly highly dynamic behaviours
  - to be addressed through loosely-coupled architectures
- Call for Event-Based architectures
  - different system subparts interact by posting and receiving events
- In the context of coordination infrastructures based on TCs
  - need to deal with loosely-coupled coordination flows
  - making single, conceptually localised coordination flows interacting by exchanging events

Contribution

- Minimal extension to the ReSpecT model
  - ST may include invocations of out_tc(*tc*,*tuple*) primitive
  - causes *tuple* to be sent to *tc* by means of an out primitive
  - a coordination activity may trigger a new one elsewhere
  - coordination flows now can be loosely coupled
- Formal Semantics
- Some application...

The out_tc primitive




```
reaction(out(T),(
    current_agent(Who),
    current_tc(Where),
    out_tc(logsA,log(Who,Where,T)),
    out_tc(logsB,log(Who,Where,T)))).
```

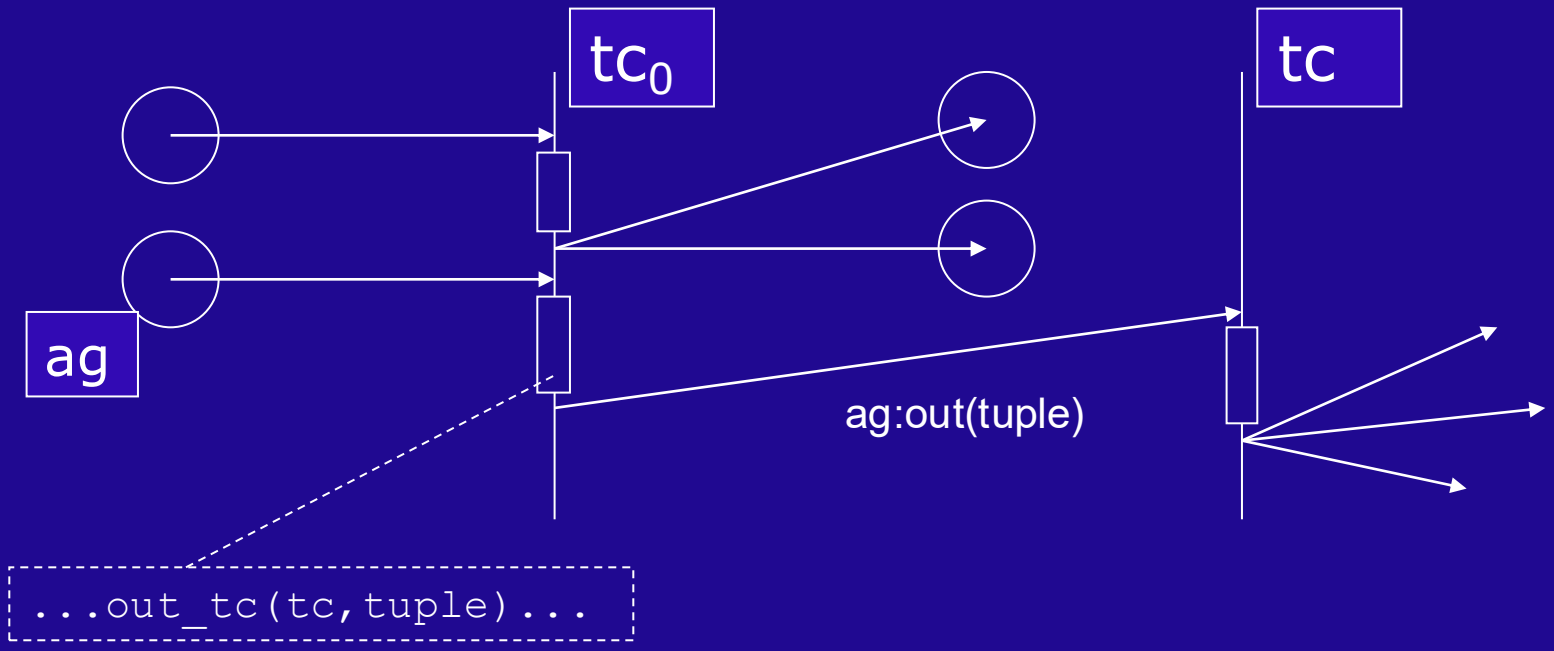
The extended model

- The execution of a reaction can now
  - = change the tuple space state, fire new reactions
  - + produce an ouput event
- The ReSpecT tuple centres state at a given time carries
  - = the tuple space state, the pending reactions, the pending queries
  - + the set of pending output events to be sent
- Semantics of out_tc(*tc*,*tuple*) primitive
  - – leaves the tuple space state unchanged, fires no reactions
  - – schedules production of output event <*tc*,*tuple*>

Multiple flows and out_tc

- In particular, execution of out_tc(*tc*,*tuple*) primitive in tc_o
  - sends *tuple* to *tc*
    - the identity of this message's sender is not tc_o!

  ⇒ is the agent responsible for the triggering activity within tc_o!

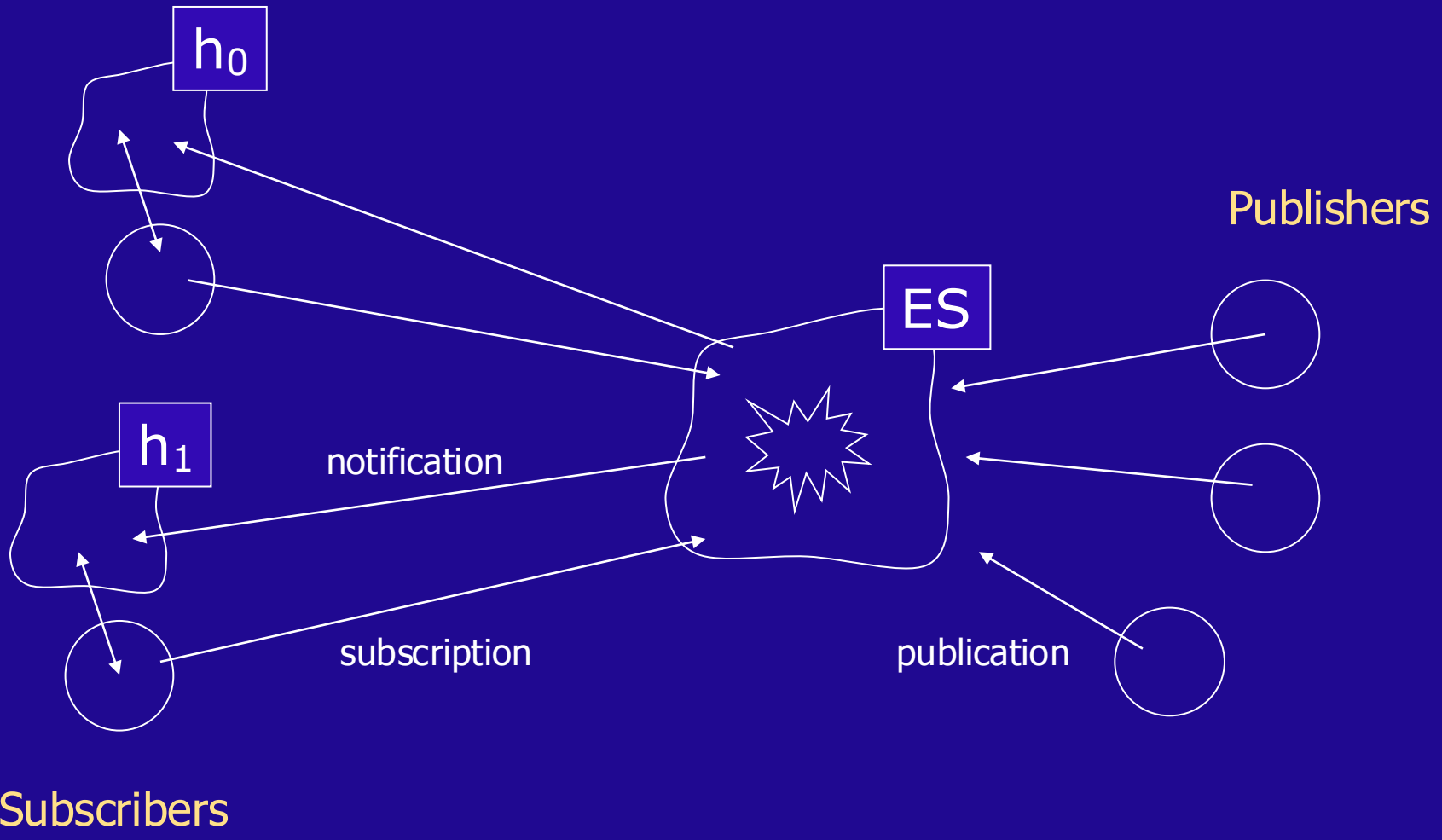


Extended ReSpecT in context

Two basic applications studied in the paper

- Implementing the *copy-collect* primitive [Rowstron&Wood 98]
  - is introduced as a minimal yet powerful support for architectures with multiple Linda tuple spaces
  - can solve the "multiple-rd" problem
  - basically copies a set of tuples from one space to another
- Event-based case studies
  - Publish/Subscribe infrastructure
  - JavaSpaces-like notification
  - Patient Monitoring System

Publish/Subscribe Infrastructure



Features:

- The subscribers do not receive notifications
  - information is stored in their Handler TC
  - can be later retrieved
  - can be archived
  - $\Rightarrow$ separation of policy
- May preserve autonomy of coordinated entities
  - suitable for scenarios such as Multi-Agent Systems
- Possibly implementing the Event Service as multiple TCs
  - e.g. hierarchically managing different scopes [Fiege et.al 2002]

JavaSpaces-like notification

- In JavaSpaces
  - by a *notify(templ,obs)*, entity *obs* is subscribed for being notified when an entry matching *templ* is inserted in the space
- With extended ReSpecT
  - notification implemented as an out_tc to a subscriber TC
- Possibly new features
  - registering not only for tuple insertion (incoming out requests)
  - but also for incoming in, rd, and so on... (T Spaces)
- What about more complex conditions?

Patient Monitoring System

- What about notification of logic events?
  - carrying some information elaborated within the medium
  - supporting content-based and type-based Publish/Subscribe
  - collecting more incoming events

  $\Rightarrow$ they can all be supported by the ReSpecT reaction mechanism
- PMS scenario
  - sensors periodically notify the Event Service

  $\Rightarrow$ e.g. blood_pressure (bp) and blood_sugar (bs)
  - interested *observers* are the nursery and the doctor
    - bp>180 notify the nursery
    - bp>230 & bs>250 notify the doctor

Conclusions

- Minimal yet powerful extension to the ReSpecT model
  - adds one primitive, allowing for cross-media interaction
  - supports loosely-coupled coordination flows
  - supports event-based coordination
- Relationship to σRST framework
  - σRST framework borrows the out_tc idea from ReSpecT
  - ReSpecT enjoys the expressiveness result of σRST
- Main difference
  - The ReSpecT extension applies in a multiple media scenario
  - σRST tackles expressiveness of the single medium
- Future work
  - deepening semantics comparison...
  - implementing event notification systems with ReSpecT TCs