Reaction Specification Tuples (ReSpecT)

Laboratory of Distributed Systems

Elena Nardini

Alma Mater Studiorum – Università di Bologna
elena.nardini@unibo.it

May 30, 2011

An Overview [Omicini, 2006]

- ReSpecT is a Logic-based language for system coordination and
- it is based on the tuple-centres-based coordination model
- The behaviour of ReSpecT tuple centres is programmed through the ReSpecT specification language
  - → The behaviour of a tuple centre can be tailored to the application needs by defining a set of reactions, which determine how a tuple centre should react to interaction events
  - → Agents can operate on a ReSpecT tuple centre in the same way as on a Linda tuple space

The ReSpecT Specification Language

- Has been shown to be *Turing-equivalent* [Denti et al., 1998]

  → Any computable coordination law could be in principle encapsulated into a ReSpecT tuple centre

  → ReSpecT can be assumed as a general-purpose core language for coordination
- Enables the definition of computations within a tuple centre, called reactions
- Makes it possible to associate reactions to events occurring in a tuple centre
- Associations between reactions and events by means of specific logic tuples called specification tuples: `reaction(E,(G,R))`
  - `E` describes the set of interaction events (*in*, *rd*, *inp*, *rdp* and *out*) *Ev* for which the reaction `R` has to be executed
  - `G` represents a set of conditions to be satisfied in order to execute a reaction `R` if the current event `Ev` matches `E`
  - `R` represents the reaction to be executed: a computation mainly composed by interaction primitives upon either the current or a remote tuple centre

Other Reaction Features

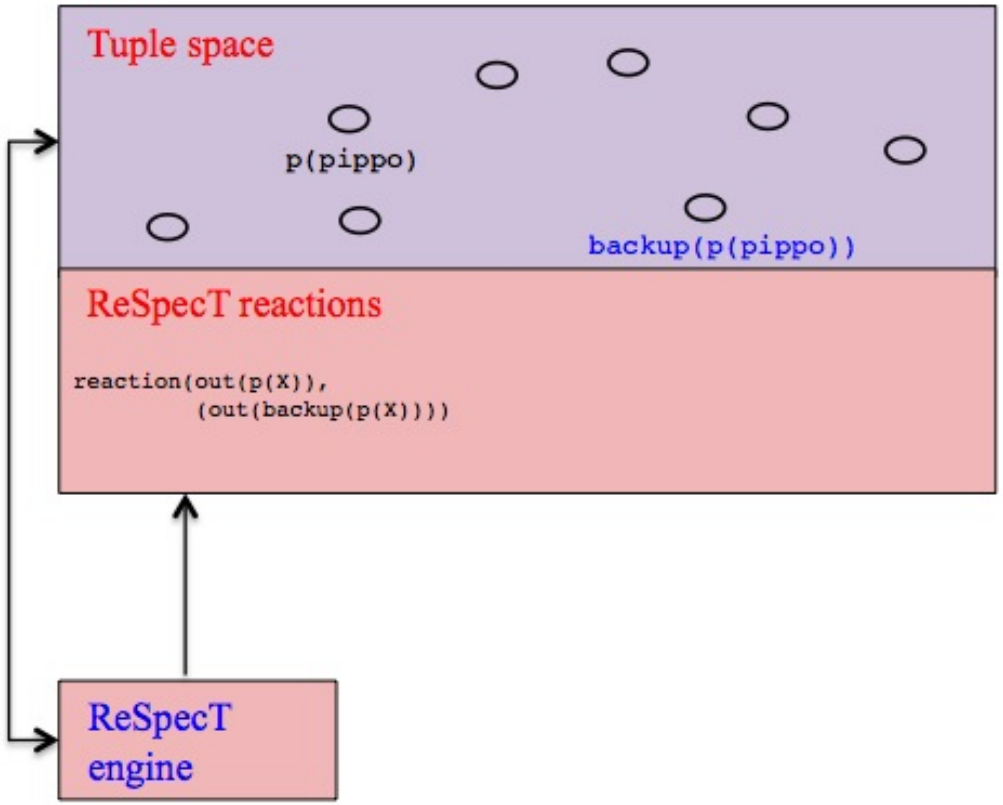
- Each reaction executes sequentially with a *transactional semantic*

  $\longrightarrow$ a failed reaction has no effect on the state of a logic tuple centre

- Reactions triggered by an event are executed before serving any other event

  $\longrightarrow$ agents perceive the result of serving the event and executing all the associated reactions altogether as a single transition state of the tuple centre state

Example


Tuple space
p(pippo)
backup(p(pippo))
ReSpecT reactions
reaction(out(p(X)),
(out(backup(p(X))))
ReSpecT
engine

Specification Language Syntax [Omicini, 2007]

⟨*TCSpecification*⟩ ::= { ⟨*SpecificationTuple*⟩ . }

⟨*SpecificationTuple*⟩ ::= `reaction(` ⟨*SimpleTCEvent*⟩ `,` [⟨*Guard*⟩ `,`] ⟨*Reaction*⟩ `)`

⟨*SimpleTCEvent*⟩ ::= ⟨*SimpleTCPredicate*⟩ `(` ⟨*Tuple*⟩ `)` | `time(` ⟨*Time*⟩ `)`

⟨*Guard*⟩ ::= ⟨*GuardPredicate*⟩ | `(` ⟨*GuardPredicate*⟩ { `,` ⟨*GuardPredicate*⟩ } `)`

⟨*Reaction*⟩ ::= ⟨*ReactionGoal*⟩ | `(` ⟨*ReactionGoal*⟩ { `,` ⟨*ReactionGoal*⟩ } `)`

⟨*ReactionGoal*⟩ ::= ⟨*TCPredicate*⟩ `(` ⟨*Tuple*⟩ `)` | ⟨*ObservationPredicate*⟩ `(` ⟨*Tuple*⟩ `)` |
⟨*Computation*⟩ | `(` ⟨*ReactionGoal*⟩ `;` ⟨*ReactionGoal*⟩ `)`

⟨*TCPredicate*⟩ ::= ⟨*SimpleTCPredicate*⟩ | ⟨*TCLinkPredicate*⟩

⟨*TCLinkPredicate*⟩ ::= ⟨*TCIdentifier*⟩ `?` ⟨*SimpleTCPredicate*⟩

⟨*SimpleTCPredicate*⟩ ::= ⟨*TCStatePredicate*⟩ | ⟨*TCForgePredicate*⟩

⟨*TCStatePredicate*⟩ ::= `in` | `inp` | `rd` | `rdp` | `out` | `no` | `get` | `set`

⟨*TCForgePredicate*⟩ ::= ⟨*TCStatePredicate*⟩`_s`

⟨*ObservationPredicate*⟩ ::= ⟨*EventView*⟩`_`⟨*EventInformation*⟩

⟨*EventView*⟩ ::= `current` | `event` | `start`

⟨*EventInformation*⟩ ::= `predicate` | `tuple` | `source` | `target` | `time`

⟨*GuardPredicate*⟩ ::= `request` | `response` | `success` | `failure` | `endo` | `exo` | `intra` | `inter` |
`from_agent` | `to_agent` | `from_tc` | `to_tc` |
`before(` ⟨*Time*⟩ `)` | `after(` ⟨*Time*⟩ `)`

⟨*Time*⟩ is a non-negative integer

⟨*Tuple*⟩ is Prolog term

⟨*Computation*⟩ is a Prolog-like goal performing arithmetic / logic computations

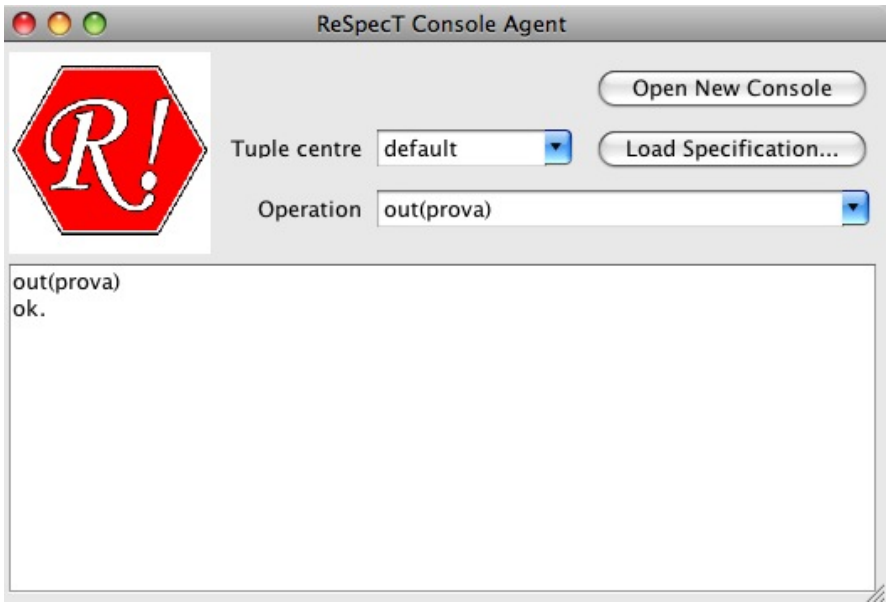
⟨*TCIdentifier*⟩ ::= ⟨*TCName*⟩ `@` ⟨*NetworkLocation*⟩

⟨*TCName*⟩ is a Prolog ground term

⟨*NetworkLocation*⟩ is a Prolog string representing either an IP name or a DNS entry

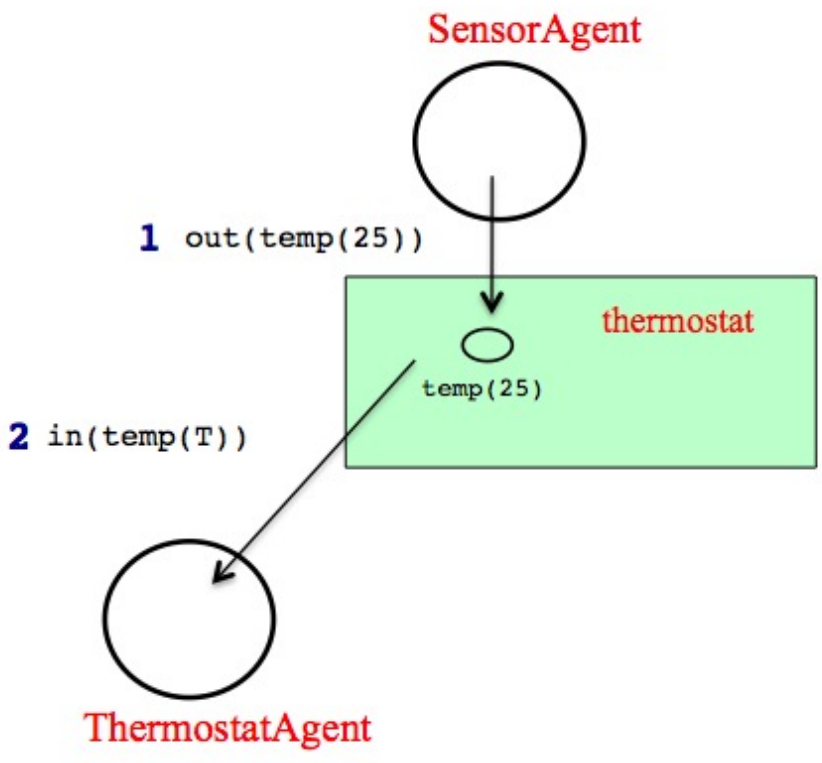
ReSpecT Technology

- ReSpecT technology is freely available in `http://alice.unibo.it/xwiki/bin/view/ReSpecT/`
- ReSpecT Shell
  - Shell interface for human agents
  - With current version (2.2): `alice.respect.tools.ConsoleNew`
- TuCSoN exploit ReSpecT tuple centres: TuCSoN 1.9.1 supports ReSpecT 2.2



Exercise: The Thermostat

- Create the tuple centre thermostat
- Create the agent sensorAgent writing in the tuple centre the sensed external-temperature
- Create the agent thermostatAgent reading the sensed temperature.
  The temperature has to be ≥ 23 and ≤ 26.
  The agent's task is to adapt the external temperature in order to respect the constraint



Bibliography I

Denti, E., Natali, A., and Omicini, A. (1998).
On the expressive power of a language for programming coordination media.
In *1998 ACM Symposium on Applied Computing (SAC'98)*, pages 169–177, Atlanta, GA, USA. ACM.
Special Track on Coordination Models, Languages and Applications.

Omicini, A. (2006).
Formal ReSpecT in the A&A perspective.
In Canal, C. and Viroli, M., editors, *5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06)*, pages 93–115, CONCUR 2006, Bonn, Germany. University of Málaga, Spain.
Proceedings.

Bibliography II

Omicini, A. (2007).
Formal ReSpecT in the A&A perspective.
Electronic Notes in Theoretical Computer Science, 175(2):97–117.
5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, Germany, 31 August 2006. Post-proceedings.

Reaction Specification Tuples (ReSpecT)

Laboratory of Distributed Systems

Elena Nardini

Alma Mater Studiorum – Università di Bologna
elena.nardini@unibo.it

May 30, 2011