

Coordination Models and Technologies towards Self-Organising Systems

Frontiers of Multimedia Systems

Andrea Omicini
andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM—Università di Bologna

ATER 2013/2014

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems



Scenarios for Concurrent / Distributed Systems

Issues

- Concurrency / Parallelism
 - Multiple independent activities / loci of control
 - Active simultaneously
 - Processes, threads, actors, active objects, agents. . .
- Distribution
 - Activities running on different and heterogeneous execution contexts (machines, devices, . . .)
- “Social” Interaction
 - Dependencies among activities
 - Collective goals involving activities coordination / cooperation
- “Environmental” Interaction
 - Interaction with external resources
 - Interaction within the time-space fabric
 - Interaction in context

Basic Engineering Principles

Principles

- Abstraction
 - Problems should be faced / represented at the most suitable level of abstraction
 - Resulting “abstractions” should be expressive enough to capture the most relevant problems
 - Conceptual integrity
- Locality & encapsulation
 - Design abstractions should embody the solutions corresponding to the domain entities they represent
- Run-time vs. design-time abstractions
 - Incremental change / evolution
 - On-line engineering [Fredriksson and Gustavsson, 2004]
 - (Cognitive) Self-organising systems [Omicini, 2012]

Which Components?

Open systems

- No hypothesis on the component's life & behaviour

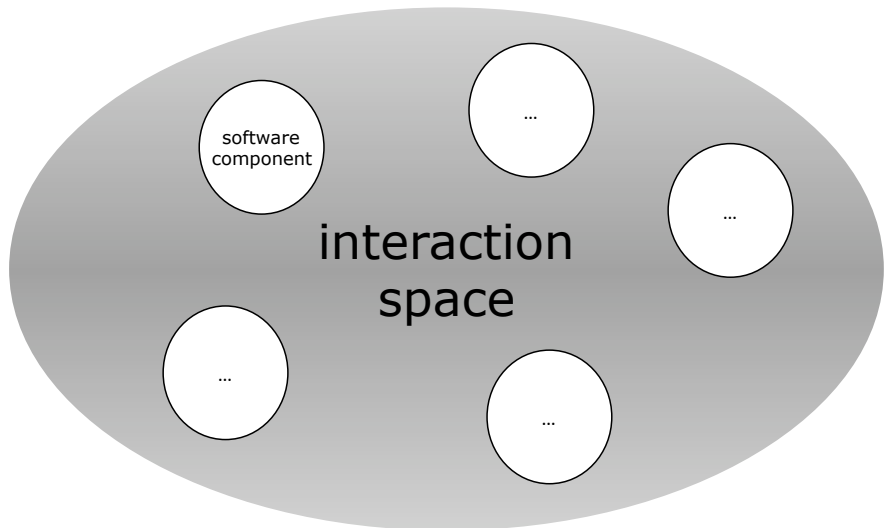
Distributed systems

- No hypothesis on the component's location & motion

Heterogeneous systems

- No hypothesis on the component's nature & structure

“Interactive Systems”: The Space of Interaction



Components of an Interactive System

What is a component of an interactive system?

- A computational abstraction characterised by
 - an independent computational activity
 - I/O capabilities
- Two independent dimensions
 - elaboration / *computation*
 - *interaction*

(Non) Algorithmic Computation I

Elaboration / computation

- Turing Machine (TM)
 - gets an input, elaborates it, throws an output
 - no interaction during computation
- Black-box algorithms
- Church's Thesis and computable functions
 - in short, a function is computable *iff* can be computed by a TM
 - so, all computable functions are computable by a TM

(Non) Algorithmic Computation II

The power of interaction [Wegner and Goldin, 2003]

Real computational systems are not rational agents that take inputs, compute logically, and produce outputs... It is hard to draw the line at what is intelligence and what is environmental interaction. In a sense, it does not really matter which is which, as all intelligent systems must be situated in some world or other if they are to be useful entities. [Brooks, 1991]

... a theory of concurrency and interaction requires a new conceptual framework, not just a refinement of what we find natural for sequential [algorithmic] computing. [Milner, 1993]

(Non) Algorithmic Computation III

Beyond Turing Machines

- Turing's *choice machines* and *unorganised machines* [Wegner and Goldin, 2003]
- Wegner's Interaction Machines [Goldin et al., 2006]
- Examples: AGV, Chess oracle [Wegner, 1997]

Basics of Interaction

Component model

A simple component exhibits

Computation Inner behaviour of a component

Interaction Observable behaviour of a component as *input* and *output*

Coupling across component's boundaries

- Control?
- Information
- Time & Space – internal / computational vs. external / physical

Information-driven interaction

Output shows part of its state outside

Input bounds a portion of its own state to the outside

Compositionality vs. Non-compositionality

An informal view by [Wegner, 1996]

Compositionality

- Sequential composition $P1; P2$
- $behaviour(P1; P2) = behaviour(P1) + behaviour(P2)$

Non-compositionality

- Interactive composition $P1|P2$
- $behaviour(P1|P2) = behaviour(P1) + behaviour(P2) + \mathbf{interaction}(P1, P2)$
- Interactive composition is more than the sum of its parts

Non-compositionality

Issues

- Compositionality vs. formalisability
 - A formal notion of model is required for stating any compositional property
 - However, formalisability does not require compositionality, and does not imply predictability
 - *Partial formalisability* may allow for proof of properties, and for partial predictability
- Emergent behaviours
 - Fully-predictable / formalisable systems do not allow by definition for emergent behaviours
- Formalisability vs. expressiveness
 - Less / more formalisable systems are (respectively) more / less expressive in terms of potential behaviours

Outline

- 1 Premises
- 2 Interaction & Coordination**
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems



Coordination as a General System Issue

Coordination is not just an issue of computational systems

Research in this area uses and extends ideas about coordination from disciplines such as computer science, organization theory, operations research, economics, linguistics, and psychology [Malone and Crowston, 1994]

A general purpose definition

*... **coordination** can be seen as the process of managing **dependencies** among **activities** [Malone and Crowston, 1994]*

- More generally, activities can be extended to include everything that generate **events** in a system, what **makes things happen**
- In systems of any sorts, **complexity** comes from dependencies

Coordination in Distributed Programming

Coordination model as a glue

A coordination model is the glue that binds separate activities into an ensemble [Gelernter and Carriero, 1992]

Coordination model as an agent interaction framework

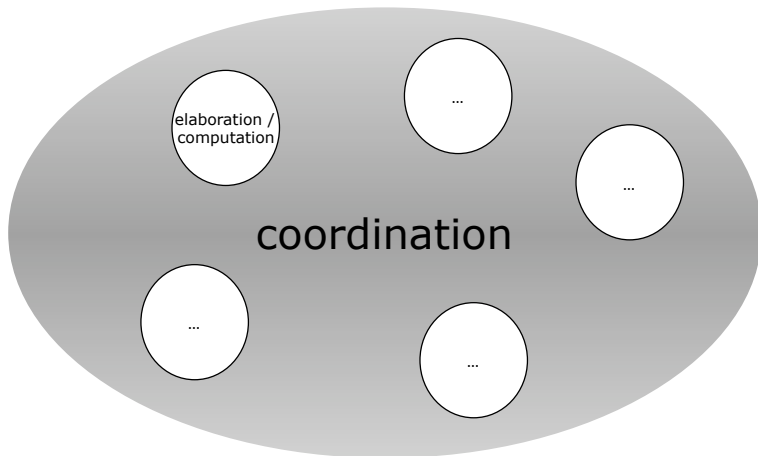
A coordination model provides a framework in which the interaction of active and independent entities called agents can be expressed [Ciancarini, 1996]

Issues for a coordination model

A coordination model should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time [Ciancarini, 1996]

What is Coordination?

Ruling the space of interaction



New Perspective on Computational Systems

Programming languages

- Interaction as an orthogonal dimension
- Languages for interaction / coordination

Software engineering

- Interaction as an independent design dimension
- Coordination patterns

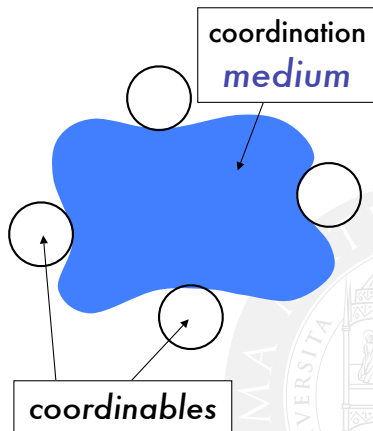
Artificial intelligence

- Interaction as a new source for intelligence
- Social intelligence

Coordination: Sketching a Meta-model

The

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some *coordination laws*
 - enacted by the behaviour of the medium
 - defining the semantics of coordination



Coordination: A Meta-model [Ciancarini, 1996]

A constructive approach

Which are the components of a coordination system?

Coordination entities Entities whose mutual interaction is ruled by the model, also called the *coordinables*

Coordination media Abstractions enabling and ruling interaction among coordinables

Coordination laws Laws ruling the observable behaviour of coordination media and coordinables, and their interaction as well

Coordinables

Original definition [Ciancarini, 1996]

These are the entity types that are coordinated. These could be Unix-like processes, threads, concurrent objects and the like, and even users.

examples Processes, threads, objects, human users, agents, ...

focus Observable behaviour of the coordinables

question Are we anyhow concerned here with the internal machinery / functioning of the coordinable, in principle?

→ This issue will be clear when comparing Linda & TuCSoN agents

Coordination Media

Original definition [Ciancarini, 1996]

These are the media making communication among the agents possible. Moreover, a coordination medium can serve to aggregate agents that should be manipulated as a whole. Examples are classic media such as semaphores, monitors, or channels, or more complex media such as tuple spaces, blackboards, pipelines, and the like.

examples Semaphors, monitors, channels, tuple spaces, blackboards, pipes, ...

focus The core around which the components of the system are organised

question Which are the possible computational models for coordination media?

→ *This issue will be clear when comparing Linda tuple spaces & ReSpecT tuple centres*

Coordination Laws I

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

Coordination Laws II

- Coordination laws rule the observable behaviour of coordination media and coordinables, as well as their interaction
 - a notion of (*admissible interaction*) *event* is required to define coordination laws
 - The interaction events are (also) expressed in terms of
 - the *communication language*, as the syntax used to express and exchange data structures
- examples* tuples, XML elements, FOL terms, (Java) objects, ...
- the *coordination language*, as the set of the admissible interaction primitives, along with their semantics
- examples* in/out/rd (Linda), send/receive (channels), push/pull (pipes), ...

Toward a Notion of Coordination Model

What do we ask to a coordination model?

- to provide high-level *abstractions* and powerful *mechanisms* for distributed system engineering
- to enable and promote the construction of *open, distributed, heterogeneous* systems
- to intrinsically *add properties* to systems independently of components
 - e.g. flexibility, control, intelligence, ...

Enabling vs. Governing Interaction I

Enabling interaction

- Communication languages, middleware, mediators...
- enabling communication
- enabling components interoperation
- no models for coordination of components
 - no rules on what components should (not) say and do at any given moment, depending on what other components say and do, and on what happens inside and outside the system

Enabling vs. Governing Interaction II

Governing interaction

- ruling communication
- providing concepts, abstractions, models, mechanisms for meaningful component integration
- governing mutual component interaction, and environment-component interaction
- in general, a model that does
 - rule what components should (not) say and do at any given moment
 - depending on what other components say and do, and on what happens inside and outside the system

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models**
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems



Origins of Tuple-based Models I

LINDA

- The ancestor of all tuple-based (a.k.a. space-based) models is LINDA
 - In LINDA [Gelernter, 1985], components communicate and synchronise by exchanging tuples through a shared **tuple space**
 - There, communication and coordination occur through a shared data space
 - communication via tuples
 - coordination via space behaviour in response to coordination primitives
- ! LINDA was first conceived to support **parallel** computation in **closed** systems—at least, with no apparent concern for open systems

Origins of Tuple-based Models II

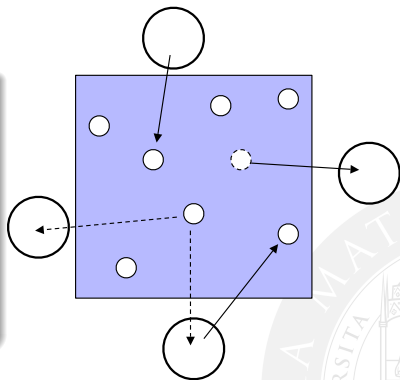
From closed to open systems

- LINDA introduces an environment abstraction devoted to the management of the interaction space
- As a conceptual consequence, computation and **coordination**
 - conceived as the **management of interaction** [Wegner, 1997]
- were to be
 - considered as two **orthogonal dimensions** of computer-based systems [Gelernter and Carriero, 1992]
 - handled – that is, analysed, modelled, designed, programmed – in an **independent** way, by adopting suitable abstractions and mechanisms

The Tuple-space Meta-model

The basics

- *Coordinables* synchronise, cooperate, compete
 - based on *tuples*
 - available in the *tuple space*
 - by *associatively* accessing, consuming and producing tuples



Tuple-based / Space-based Coordination Systems

Adopting the constructive coordination meta-model [Ciancarini, 1996]

coordination media tuple spaces

- as multiset / bag of data objects / structures called *tuples*

communication language tuples

- as ordered collections of (possibly heterogeneous) information items

coordination language tuple space primitives

- as a set of operations to put, browse and retrieve tuples to/from the space

Linda: The Communication Language [Gelernter, 1985]

Communication Language

tuples ordered collections of possibly heterogeneous information chunks

- examples: `p(1)`, `printer('HP',dpi(300))`, `[0,0.5]`, `matrix(m0,3,3,0.5)`, `tree_node(node00,value(13),left(_),right(node01))`, ...

templates / anti-tuples specifications of set / classes of tuples

- examples: `p(X)`, `[?int,?int]`, `tree_node(N)`, ...

tuple matching mechanism the mechanism that matches tuples and templates

- examples: pattern matching, unification, ...

Linda: The Coordination Language [Gelernter, 1985] I

out(T)

- out(T) puts tuple T in to the tuple space

examples out(p(1)), out(0,0.5), out(course('Antonio Natali', 'Poetry', hours(150))) ...

Linda: The Coordination Language [Gelernter, 1985] II

in(TT)

- `in(TT)` retrieves a tuple matching template TT from the tuple space

destructive reading the tuple retrieved is removed from the tuple centre

non-determinism if more than one tuple matches the template, one is chosen non-deterministically

suspensive semantics if no matching tuples are found in the tuple space, operation execution is suspended, and woken when a matching tuple is finally found

examples `in(p(X))`, `in(0,0.5)`, `in(course('Antonio Natali',Title,hours(X)) ...`

Linda: The Coordination Language [Gelernter, 1985] III

rd(TT)

- **rd(TT)** retrieves a tuple matching template TT from the tuple space
- non-destructive reading** the tuple retrieved is left untouched in the tuple centre
- non-determinism** if more than one tuple matches the template, one is chosen non-deterministically
- suspensive semantics** if no matching tuples are found in the tuple space, operation execution is suspended, and awakened when a matching tuple is finally found
- examples** `rd(p(X))`, `rd(0,0.5)`, `rd(course('Alessandro Ricci', 'Operating Systems', hours(X)))` ...

Linda Extensions: Predicative Primitives

$\text{inp}(\text{TT})$, $\text{rdp}(\text{TT})$

- both $\text{inp}(\text{TT})$ and $\text{rdp}(\text{TT})$ retrieve tuple T matching template TT from the tuple space
 - $= \text{in}(\text{TT}), \text{rd}(\text{TT})$ (non-)destructive reading, non-determinism, and syntax structure is maintained
 - $\neq \text{in}(\text{TT}), \text{rd}(\text{TT})$ suspensive semantics is lost: this *predicative* versions primitives just fail when no tuple matching TT is found in the tuple space
 - success / failure* predicative primitives introduce *success / failure semantics*: when a matching tuple is found, it is returned with a success result; when it is not, a failure is reported

Linda Extensions: Bulk Primitives

`in_all(TT)`, `rd_all(TT)`

- Linda primitives (including predicative ones) deal with a tuple at a time
 - some coordination problems require more than one tuple to be handled by a single primitive
- `rd_all(TT)`, `in_all(TT)` get all tuples in the tuple space matching with TT, and returns them all
 - no suspensive semantics: if no matching tuple is found, an empty collection is returned
 - no success / failure semantics: a collection of tuple is always successfully returned—possibly, an empty one
 - in case of logic-based primitives / tuples, the form of the primitive are `rd_all(TT,LT)`, `in_all(TT,LT)` (or equivalent), where the (possibly empty) list of tuples unifying with TT is unified with LT
 - (non-)destructive reading: `in_all(TT)` consumes all matching tuples in the tuple space; `rd_all(TT)` leaves the tuple space untouched
- Many other bulk primitives have been proposed and implemented to address particular classes of problems [Rowstron, 1996]

Linda Extensions: Multiple Tuple Spaces

ts ? out(T)

- Linda tuple space might be a bottleneck for coordination
- Many extensions have focussed on making a multiplicity of tuple spaces available to processes [Gelernter, 1989]
 - each of them encapsulating a portion of the coordination load
 - either hosted by a single machine, or distributed across the network
- Syntax required, and dependent on particular models and implementations
 - a space for tuple space names, possibly including network location
 - operators to associate Linda operators to tuple spaces
- For instance, `ts @ node ? out(p)` may denote the invocation of operation `out(p)` over tuple space `ts` on node `node`

Main Features of Tuple-based Coordination

Main features of the Linda model

tuples A tuple is an ordered collection of knowledge chunks, possibly heterogeneous in sort

generative communication until explicitly withdrawn, the tuples generated by coordinables have an independent existence in the tuple space; a tuple is equally accessible to all the coordinables, but is bound to none

associative access tuples in the tuple space are accessed through their content & structure, rather than by name, address, or location

suspensive semantics operations may be suspended based on unavailability of matching tuples, and be woken up when such tuples become available

Features of Linda: Tuples

- A tuple is an ordered collection of knowledge chunks, possibly heterogeneous in sort
 - a record-like structure
 - with no need of field names
 - easy aggregation of knowledge
 - raw semantic interpretation: a tuple contains all information concerning an given item
- Tuple structure based on
 - arity
 - type
 - position
 - information content
- Anti-tuples / Tuple templates
 - to describe / define sets of tuples
- Matching mechanism
 - to define belongingness to a set

Features of Linda: Generative Communication

Communication orthogonality

- Both senders and the receivers can interact even without having prior knowledge about each others
- space uncoupling** no need to coexist in space for two processes to interact
- time uncoupling** no need for simultaneity for two processes to interact
- name uncoupling** no need for names for processes to interact

Features of Linda: Associative Access

Content-based coordination

- Synchronisation based on tuple content & structure
 - absence / presence of tuples with some content / structure determines the overall behaviour of the coordinables, and of the coordinated system in the overall
 - based on tuple templates & matching mechanism
- *Information-driven coordination*
 - patterns of coordination based on data / information availability
 - based on tuple templates & matching mechanism
- *Reification*
 - making events become tuples
 - grouping classes of events with tuple syntax, and accessing them via tuple templates

Features of Linda: Suspensive Semantics

Blocking primitives

- `in` & `rd` primitives in Linda have a suspensive semantics
 - the coordination medium makes the primitives waiting in case a matching tuple is not found, and wakes it up when such a tuple is found
 - the coordinable invoking the suspensive primitive is expected to wait for its successful completion
- Twofold wait
 - in the **coordination medium** the operation is first (possibly) suspended, then (possibly) served: coordination based on absence / presence of tuples belonging to a given set
 - in the **coordination entity** the invocation may cause a wait-state in the invoker: hypothesis on the internal behaviour of the coordinable

Benefits of LINDA-based Models I

Knowledge-based Coordination

- A **tuple** is an ordered collection of possibly-heterogeneous knowledge chunks
- Synchronisation based on the availability of tuples means essentially synchronisation based on the **availability of structured knowledge** of some sort
- Tuple-based coordination is first of all **knowledge-based coordination**
 - where tuple spaces are possibly interpreted as **knowledge repositories**

Benefits of LINDA-based Models II

Incomplete information

- Tuple spaces are accessed **associatively**
 - queries specify tuple templates that match tuples based on their structure and the data they contain
- Complete uncoupling in communication
 - information neither on the sender nor on the structure of the share space is required for a message to be received
- Synchronisation possible over a partial representation of knowledge—the tuple **template**
 - a fundamental feature in all the contexts where information is often vague, inaccurate, incomplete, or partially specified—as is typical in knowledge-intensive systems

Benefits of LINDA-derived Models I

Two other features characterise tuple-based models as they descend from the original LINDA ancestor

- **distribution** of the coordination abstractions
- **expressiveness** of the coordination abstractions

respectively termed as [Busi et al., 2001]

- “reshaping the coordination media”
- “programming the coordination rules”



Benefits of LINDA-derived Models II

Reshaping the coordination media

- Distribution is essential for any complex system
- In the same way as components of a distributed system are spread all over the system topology, **multiple tuple spaces** fill the system environment, providing for **distributed coordination abstractions**
 - JavaSpaces [Freeman et al., 1999] by Sun
 - TSpaces [Wyckoff et al., 1998] by IBM
- This paves the way toward pervasive coordination systems
- Also, expressing the **environment topology** in a distributed setting is essential for the coordination of **local interaction** as well as of mobile components
 - LIME [Murphy et al., 2006]
 - KLAIM [De Nicola et al., 1998]

Benefits of LINDA-derived Models III

Programming the coordination rules

- **Expressiveness** of coordination media often needs to be tailored to the complexity and peculiarity of the specific coordinated system
- So, a number of LINDA derivatives, e.g.
 - Law-Governed Interaction [Minsky and Ungureanu, 2000]
 - MARS [Cabri et al., 2000]
 - ReSpecT [Omicini and Denti, 2001]
- focus on the **programmability of the tuple space**, so as to
 - make it possible to **explicitly express** the rules of coordination
 - **embed** them within the coordination abstraction
- There, **arbitrarily-complex** coordination policies can be in principle associated to each and every coordination medium, which could be individually programmed so as to embed either global or local coordination policies, as required by the specific coordinated systems

Benefits of LINDA-derived Models IV

Situatedness

- The ability to define arbitrarily-complex coordination policies and to embed them within the coordination media should be in principle coupled with the ability to capture and react to **arbitrary environment events**
- ! ... otherwise, **environment-based coordination** would not be supported directly by the coordination medium
- This provides for the level of **situatedness** typically required by coordination in pervasive computational environments
 - Situated ReSpecT [Casadei and Omicini, 2009]

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models**
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models**
 - **Why Nature-inspired Coordination?**
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems

Why Nature-inspired Models?

Complex natural systems

- such as physical, chemical, biochemical, biological, social systems
- natural system exhibit *features*
 - such as distribution, openness, situation, fault tolerance, robustness, adaptiveness, ...
- which we would like to understand, capture, then bring to *computational* systems

Nature-Inspired Computing (NIC)

- For instance, NIC [Liu and Tsui, 2006] summarises decades of research activities
- putting emphasis on *autonomy* of components, and on *self-organisation* of systems

Why Coordination Models?

Interaction

- most of the complexity of computational systems comes from **interaction** [Omicini et al., 2006]
- along with an essential part of their expressive power [Wegner, 1997]

Coordination

- since **coordination** is essentially the science of **managing the space of interaction** [Wegner, 1997]
- **coordination models and languages** [Ciancarini, 1996] provide abstractions and technologies for the engineering of complex computational systems [Ciancarini et al., 2000]

Why Nature-inspired Coordination?

Coordination issues in natural systems

- coordination issues did not first emerge in computational systems
- [Grassé, 1959] noted that in termite societies *“The coordination of tasks and the regulation of constructions are not directly dependent from the workers, but from constructions themselves.”*

Coordination as the key issue

- many well-known examples of natural systems – and, more generally, of complex systems – seemingly rely on simple yet powerful **coordination mechanisms** for their key features—such as self-organisation
- it makes sense to focus on **nature-inspired coordination models** as the core of complex nature-inspired computational systems

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models**
 - Why Nature-inspired Coordination?
 - Examples**
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems



Stigmergy I

Stigmergy in insect societies

- nature-inspired models of coordination are grounded in studies on the behaviour of social insects, like ants or termites
- [Grassé, 1959] introduced the notion of **stigmergy** as the fundamental coordination mechanism in termite societies
- in ant colonies, pheromones act as environment markers for specific social activities, and drive both the *individual* and the *social* behaviour of ants

Stigmergy II

Stigmergy in computational systems

- nowadays, stigmergy generally refers to a set of nature-inspired coordination mechanisms mediated by the *environment*
- *digital pheromones* [Parunak et al., 2002] and other *signs* made and sensed in a shared environment [Parunak, 2006] can be exploited for the engineering of adaptive and self-organising computational systems

Chemical Coordination

Chemical reactions as (natural) coordination laws

- inspiration comes from the idea that complex physical phenomena are driven by the (relatively) simple chemical reactions
- coordinating the behaviours of a huge amount of components, as well as the global system evolution

Chemical reactions as (computational) coordination laws

- Gamma [Banâtre and Le Métayer, 1990] is a *chemistry-inspired coordination* model—as for the CHAM (chemical abstract machine) model [Berry, 1992]
- coordination in Gamma is conceived as the evolution of a space governed by chemical-like rules, globally working as a rewriting system [Banâtre et al., 2001]

Field-based Coordination

Computational fields as coordination laws

- **field-based** coordination models like TOTA [Mamei and Zambonelli, 2004] are inspired by the way masses and particles move and self-organise according to gravitational/electromagnetic fields [Mamei and Zambonelli, 2006]
- there, computational force fields, generated either by the active components or by the pervasive coordination infrastructure, propagate across the environment, and drive the actions and motion of the component themselves

(Bio)chemical Coordination

Chemical reactions as coordination laws

- **chemical tuple spaces** [Viroli et al., 2010] exploit the chemical metaphor at its full extent—beyond Gamma
- data, devices, and software agents are represented in terms of chemical reactants, and system behaviour is expressed by means of chemical-like laws
- which are actually **time-dependent** and **stochastic**
- embedded within the coordination medium
- **biochemical tuple spaces** [Viroli and Casadei, 2009] add *compartments*, *diffusion*, and *stochastic behaviour* of coordination primitives

Basic Issues of Nature-inspired Coordination I

Environment

- **environment** is essential in nature-inspired coordination
 - it works as a **mediator** for component interaction — through which the components of a distributed system can communicate and coordinate **indirectly**
 - it is **active** — featuring autonomous dynamics, and affecting component coordination
 - it has a **structure** — requiring a notion of **locality**, and allowing components of any sort to **move** through a **topology**

Basic Issues of Nature-inspired Coordination II

Stochastic behaviour

- complex systems typically require **probabilistic** models
 - *don't know* / *don't care* **non-deterministic** mechanisms are not expressive enough to capture all the properties of complex systems such as biochemical and social systems
 - probabilistic mechanisms are required to fully capture the dynamics of coordination in nature-inspired systems
 - coordination models should feature (possibly simple yet) expressive mechanisms to provide coordinated systems with **stochastic behaviours**

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models**
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems

Tuple-based Models for Nature-inspired Coordination I

Expressiveness

- LINDA is a sort of *core* coordination model
- making it easy to face and solve many typical problems of complex distributed systems
- *complex* coordination problems are solved with *few, simple* primitives
- whatever the model used to measure *expressiveness* of coordination, tuple-based languages are highly-expressive [Busi et al., 1998]

Tuple-based Models for Nature-inspired Coordination II

Environment-based coordination

- tuple spaces are provided as **coordination services** by the *coordination infrastructure* [Viroli and Omicini, 2006]
- they can be interpreted as **coordination artefacts** shaping computational *environment* [Omicini et al., 2004]
- as such, they can be exploited to support **environment-based coordination** [Ricci et al., 2005]

Tuple-based Models for Nature-inspired Coordination III

Extensibility

- whatever its expressiveness, LINDA was conceived as a coordination model for closed, parallel systems
- so, in fact, some relevant problems of today open, concurrent systems cannot be easily solved with LINDA either in practice or in theory
- as a result, tuple-based models have been extended with new simple yet powerful mechanisms
- generating a plethora of tuple-based coordination models
[Rossi et al., 2001]

Tuple-based Models for Nature-inspired Coordination IV

Nature-inspired extensions

- LINDA may *not* be nature-inspired, but many of its extensions *are*
- many of the coordination models depicted before
 - stigmergy [Parunak, 2006]
 - field-based [Mamei and Zambonelli, 2004]
 - chemical [Viroli et al., 2010] and biochemical [Viroli and Casadei, 2009]
- along with many others, such as
 - cognitive stigmergy [Ricci et al., 2007]
 - pervasive ecosystems [Viroli et al., 2012]
- are actually **nature-inspired tuple-based coordination models**

Example: Tuples & Stigmergy

Tuple-based models as a computational specimen for stigmergy

- Multiple tuple spaces physically/logically distributed in a computational system could be seen as the building blocks of the system's environment—the “walls” upon which coordinating agents can read and write signs in the form of tuples [Omicini, 2012]
 - Suspensive coordination primitives could be used either by “stupid” processes to synchronise upon pre-determined tuple patterns, or by intelligent agents properly interpreting the symbolic content of tuples—as in the case of logic tuples, for instance [Denti and Omicini, 1999]
- Self-organising patterns based on either stigmergy or cognitive stigmergy can be in principle built upon tuple-based coordination models and technologies

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems**
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems

Toward Self-organising Coordination I

Just *some* is not enough

- capturing just *some* of the principles and mechanisms of natural systems does not ensure to capture their *essence*
- for instance, chemical coordination models such as Gamma and CHAM exploit the raw schema of computation as chemical reaction, but are *not* expressive enough to fully reproduce any non-trivial chemical system
- in fact, *e.g.*, even the simplest model for real chemical reactions requires a notion of *reaction rate*
- neither Gamma nor CHAM provide for such a notion, they are not expressive enough to fully match the behaviour of real chemical systems

Toward Self-organising Coordination II

Towards self-organising coordination

- Generally speaking, **self-organising coordination** can be defined as the management of system interactions featuring self-organising properties
- ... namely, where [Viroli et al., 2009]
 - **interactions** are **local**
 - **global** desired effects of coordination appear by **emergence**

Toward Self-organising Coordination III

The problem of self-organising coordination [Viroli et al., 2009]

- most of the traditional coordination models feature abstractions enacting coordination laws that are typically *reactive*, (mostly) *deterministic*, and *global* as well
- in complex systems featuring self-* properties, instead, coordination patterns typically appear at the global level by *emergence*, from *probabilistic*, *time-dependent* coordination laws based on *local* criteria
- in particular, many coordination models either implicitly or explicitly recognise that full expressiveness requires addressing the issues of *time dependency* and *stochasticity*

Examples I

STOKLAIM

- STOKLAIM [De Nicola et al., 2006] – a *stochastic* extension of the LINDA-derived KLAIM model for mobile coordination [De Nicola et al., 1998] – adds distribution rates to coordination primitives—thus making it possible the modelling of non-deterministic real-life phenomena such as failure rates and inter-arrival times

SwarmLinda

- SwarmLinda [Tolksdorf and Menezes, 2004] enhances LINDA implementation with swarm intelligence to achieve features such as scalability, adaptiveness, and fault-tolerance—by modelling tuple templates as ants, featuring probabilistic behaviour when looking for matching tuples in a distributed setting

Examples II

Time-aware ReSpecT

- ReSpecT [Omicini and Denti, 2001] generally addresses *time dependency* by capturing time events and supporting the definition and enforcement of *timed coordination policies* [Omicini et al., 2005]—so, ReSpecT-programmed tuple centres can work as time-dependent abstractions for the coordination of distributed processes [Omicini et al., 2007]

Enough?

No.

- in the overall, the above-mentioned models *fail* to capture all the essential features of nature-inspired coordination
- this is why many novel research lines stretch existing tuple-based models to achieve the expressive power required to model and build distributed systems with a complexity comparable to natural systems [Omicini and Viroli, 2011]

Chemical & Tuple-based Coordination I

Chemical tuple spaces

- Along this line, a **chemical tuple-space** model and infrastructure have been defined [Viroli et al., 2010] ...
- ... that embodies all the typical features of self-organisation in natural chemical systems
- There, self-organisation could be achieved in two ways
 - either by means of the behaviour of an individual chemical tuple space (**intra-space self-organisation**)
 - or by means of a suitable pattern of interaction among chemical tuple spaces (**inter-space self-organisation**)

Chemical & Tuple-based Coordination II

Chemical tuple-based coordination for pervasive systems

- Tuples as chemical reactants participate to chemical-like reactions in tuple spaces
- Chemical reactions with chemical tuple diffusion can be used in pervasive applications to enact spatial computing patterns of competition and gradient-based interaction
- In [Viroli et al., 2011] such a model is tested against a self-adaptive display infrastructure providing people nearby with several visualization services (advertisements, news, personal and social content)—more generally, for **self-aware pervasive service ecosystems**

Self-aware Pervasive Service Ecosystems

SAPERE

- European Project FP7 – 2010-2013^a
 - <http://www.sapere-project.eu>
 - <http://apice.unibo.it/xwiki/bin/view/SAPERE/>
- Under the hat of the Proactive Initiative AWARENESS
 - <http://www.aware-project.eu/>
- Based on chemical coordination for pervasive computing
- LSA (**Live Semantic Annotation**), as chemical tuples representing individuals, components, services in pervasive scenarios, and triggering eco-laws governing self-organisation of pervasive services

^aThis work has been supported by the EU-FP7-FET Proactive project SAPERE – Self-aware Pervasive Service Ecosystems, under contract no.256873

Tuple-based Models for KIE I

Toward knowledge-intensive environments (KIE)

- In the overall, suitably-extended tuple-based models provide a promising platform for the design and development of self-organising coordinated systems
- Nonetheless, *knowledge-intensive* application scenarios pose a huge challenge for tuple-based models
- There, the aforementioned benefits of tuple-based coordination in terms of knowledge-based coordination fade in front of the problems it induces in terms of syntax & (mostly) semantics
 - e.g., two tuples containing the same data may not match due to differences in the tuple structure
 - e.g., two tuples representing the same information may not match based on a different syntax adopted

Tuple-based Models for KIE II

Two lines of extension in the literature

- Exploiting tuple-based coordination within a middleware for KIE
 - e.g., [Tolksdorf et al., 2008] experiments with a tuple-based coordination within Semantic Web middleware
 - e.g., [Nixon et al., 2008] survey similar approaches
- Enhancing the tuple space abstraction with a semantic interpretation
 - e.g., [Nardini et al., 2010] extend tuple spaces with a description logic framework so as to equip each tuple, template, and operation over tuple spaces with a well-founded semantic interpretation

Self-organising Semantic Coordination (SOSC) I

Adding semantics to tuple spaces

- Generalisation of the basic principles and mechanisms of coordination and self-organisation for application to knowledge-intensive environments
 - Everything still based on tuples and tuple spaces
 - Now equipped with a semantic interpretation
- Exploring a notion of **self-organising semantic coordination**

Self-organising Semantic Coordination (SOSC) II

SOSC

- SOSC as the management of interactions in knowledge-intensive systems
- where
 - interactions are local and involve sharing and processing of knowledge
 - the global desired effects of coordination over distributed knowledge appear by emergence and through self-organisation.
- Coordination middleware – in particular, tuple-based ones – should be adopted to support self-organising semantic coordination
- Examples
 - **eternally adaptive service ecosystems** for pervasive computing [Viroli and Zambonelli, 2010]
 - e-Health systems [Nardini et al., 2011] with Semantic TuCSoN [Nardini et al., 2011]

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems**
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems



Probability

- *Probability* measures how likely some event will occur
- At its core, probability provides a mathematical framework to describe *casual* events
- ... where casual essentially means *non-deterministic*
- By definition, probability deals with *single occurrences* of events
- From a scientific viewpoint, a probabilistic description *per se* has *no predictive value*: it cannot really predict the precise outcome of a phenomenon
- In any case, probability provides an *a priori* model for non-deterministic phenomena

Statistics & Probability

- *Statistics* describes / analyses / interprets phenomena starting from the data available about them
- Whenever a phenomenon has no *a priori* mathematical model (at least, not yet), statistics is concerned with getting one *a posteriori* from the available data
- Accordingly, statistics is concerned with *several occurrences* of (non-deterministic) events
- Probability typically provide the mathematical tools to explain data & build a model

Stochastic Systems

- *Stochastic systems* are non-deterministic systems
 - A stochastic system is one whose states are determined probabilistically
 - More generally, any phenomenon requiring probability for its description is (at least in part) stochastically by definition
 - Roughly speaking, a probabilistic model for a stochastic system provides a *predictive framework* for a non-deterministic system
- We cannot predict the single occurrence of a non-deterministic event, but we can predict the overall outcome of repeated occurrences of a non-deterministic event

Non-determinism, Coordination & Stochastic Behaviour

- Autonomous systems such as *adaptive* and *self-** ones are *stochastic systems* at their very heart
- Accordingly, a foremost feature of computational models for adaptive and self-* systems is *non-determinism*
- In order to obtain stochastic behaviours of computational systems, suitable mechanisms for non-determinism should be provided
- Since most of the complexity featured by adaptive and self-* systems depends on the interaction among components, *coordination models* should feature *non-deterministic coordination mechanisms* for stochastic behaviour

Issues

- Devising out some *basic mechanisms* for *stochastic coordination*
- Finding a *minimal* set of primitives for most (all) of the most relevant stochastic systems
- Showing how such mechanisms could be embedded as *tuple-based* co-ordination primitives, in order to address the general need of complex computational system engineering
- Defining their *formal semantics* and implementing them as TuCSoN primitives

Don't Care Non-determinism in Tuple-based Models

- LINDA features *don't know* non-determinism handled with a *don't care* approach:
 - don't know* which tuple among the matching ones is retrieved by a getter operation (`in`, `rd`) can be neither specified nor predicted
 - don't care* nonetheless, the coordinated system is designed so as to keep on working whichever is the matching tuple returned
 - Instead, adaptive and self-organising systems require *stochastic behaviours* like *"most of the time do this"*, *"sometimes do that"*
 - Possibly with some quantitative specification of *"most of the time"* and *"sometimes"*
- As it is, non-determinism in tuple-based models does not fit the need of stochastic behaviour specification

LINDA “Local” Nature – In Time & Space

- No context** — In a single getter operation, only a *local*, *point-wise* property affects tuple retrieval: that is, the conformance of a tuple to the template, independently of the *spatial* context
- in fact, standard getter primitives return a matching tuple independently of the other tuples currently in the same space—so, they are “*context unaware*”
- No history** — Furthermore, in a sequence of getter operations, don’t know non-determinism makes any prediction of the overall behaviour impossible. Again, then, only a point-wise property can be ensured even in *time*
- sequences of standard getter operations present no meaningful distribution over time

LINDA: How to Roll a Dice?

- We define tuple space dice
- We represent a six-face dice as a collection of six tuples: `face(1)`, ..., `face(6)`
- We roll a dice by `rd`-ing a `face/1` tuple from `dice`:

`dice ? rd(face(X))`

- ! We do *not* obtain the overall (stochastic) behaviour of a dice: for instance, it may reasonably happen that rolling the dice 10^9 times *always* results in `X / 1`—that is, we get “1” 10^9 times in a row.

uLINDA: Probabilistic Non-determinism

- We define **uniform coordination primitives** (uin , urd) – first mentioned in [Gardelli et al., 2007] – as the *specialisation* of LINDA getter primitives featuring **probabilistic non-determinism** instead of don't know non-determinism
- We call the new model **uLINDA** [Mariani and Omicini, 2013]
- Uniform primitives allow programmers to both specify and (**statistically**) **predict** the probability to retrieve one specific tuple among a bag of matching tuples
- Uniform primitives are the *“basic mechanisms enabling self-organising coordination”*—that is, a **minimal set of constructs** able (*alone*) to impact the observable properties of a coordinated system

uLINDA: “Global” Nature

Situation & prediction

Uniform primitives replace don't know non-determinism with *probabilistic non-determinism* to

- **situate** a primitive invocation in space
 - uniform getter primitives return matching tuples based on the other tuples in the space—so, their behaviour is *context-aware*
- **predict** its behaviour in time
 - sequences of uniform getter operations tend to globally exhibit a *uniform distribution* over time

uLINDA: How to Roll a Dice?

- Again, we define tuple space dice
- Again, we represent a six-face dice as a collection of six tuples:
face(1), ..., face(6)
- We roll a dice by urd-ing a face/1 tuple from dice:

dice ? urd(face(X))

! Now, we *do* obtain the overall (stochastic) behaviour of a dice:

context — at every roll, the six faces of the dice $X / 1, \dots, X / 6$ have the same *probability* $P = 1/6$ to be selected

history — in the overall, repeating several times a roll, the six faces will tend to converge towards a uniform distribution

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems**
- 8 Predicting Behaviours of Complex Coordinated Systems



Complexity as a Multi-disciplinary Notion

Complex systems everywhere

- The notion of *complexity* is definitely a *multi-disciplinary* one, ranging from physics to biology, from economics to sociology and organisation sciences
- Systems that are said *complex* are both *natural* and *artificial* ones

Natural vs. artificial complex systems

- We *observe* and *model* complex *physical* systems
- We *design* and *build* complex *computational* systems

Question

- Which features do *all* complex systems *share* independently of their nature?

Complexity & Interaction

... by a complex system I mean one made up of a *large number* of parts that *interact* in a non simple way [Simon, 1962]

Laws of complexity

- Some “laws of complexity” exists that characterise any complex system, *independently* of its specific nature [Kauffman, 2003]
- The precise source of what all complex systems share is still unknown in essence

Interaction

- We argue that *interaction* – its *nature*, *structure*, *dynamics* – is the key to understand some fundamental properties of complex systems of any kind

Interaction in Statistical Mechanics I

Independence from interaction

- Some physical systems are described under the assumption of **mutual independence** among particles—that is, the behaviour of the particles is unaffected by their mutual **interaction**
 - e.g., ideal gas [Boltzmann, 1964]
 - There, the probability distribution of the whole system is the product of those of each of its particles
 - In computer science terms, the *properties of the system* can be **compositionally** derived by the *properties of the individual components* [Wegner, 1997]
- Neither macroscopic sudden shift nor abrupt change for the system as a whole: technically, those systems have **no phase transitions**—of course, while the “independence from interaction” hypothesis holds

Interaction in Statistical Mechanics II

Interacting systems

- Introducing interaction among particles structurally *changes* the *macroscopic properties*, along with the *mathematical* ones
- *Interacting systems* are systems where particles *do not behave independently* of each other
- The probability distribution of an interacting system does not factorise anymore
- In computer science terms, an interacting system is *non-compositional* [Wegner, 1997]

Interaction in Statistical Mechanics III

Interacting vs. non-interacting systems

- Only interacting systems can describe real cases beyond the idealised ones
 - e.g., they can explain phase transitions – like liquid-gas transition – and much more, such as collective emerging effects
- While a system made of independent parts can be represented by isolated single nodes, an *interacting system* is better described by *nodes connected by lines* or higher-dimensional objects
- From the point of view of information and communication theories, an ideal non-interacting gas is a system of *non-communicating nodes*, whereas an interacting system is made of *nodes connected by channels*

Complexity in Statistical Mechanics I

The case of magnetic particles

- The simplest standard prototype of an interacting system is the one made of magnetic particles
- There, individual particles can behave according to a magnetic field which leaves their probabilistic independence undisturbed
- At the same time, two magnetic particles interact with each other, and the strength of their interaction is a crucial tuning parameter to observe a phase transition
 - If interaction is weak, the effect of a magnetic field is smooth on the system
 - Instead, if the interaction is strong – in particular, higher than a threshold – even a negligible magnetic field can cause a powerful *cooperative effect* on the system

Complexity in Statistical Mechanics II

Interaction is not enough

- Interaction is a necessary ingredient for complexity in statistical mechanics but definitely not a sufficient one
 - **Complexity** arises when the possible equilibrium states of a system grow very quickly with the number of particles, regardless of the simplicity of the laws governing each particle and their mutual interaction
 - Roughly speaking, complexity is much more related to **size in number**, rather than to complexity of the laws ruling interaction
- we do *not* need *complex interaction* to make interaction lead to complexity

From Statistical Mechanics to Social Systems I

Large numbers

- The key point in statistical mechanics is to relate the *macroscopic* observables quantities – like pressure, temperature, etc. – to suitable *averages* of *microscopic* observables—like particle speed, kinetic energy, etc.
- Based on the *laws of large numbers*, the method works for those systems made of a *large number* of particles / basic components

From Statistical Mechanics to Social Systems II

Beyond the boundaries

- *Methods for complex systems* from statistical mechanics have expanded from physics to fields as diverse as biology [Kauffman, 1993], economics [Bouchaud and Potters, 2003, Mantegna and Stanley, 1999], and computer science itself [Mézard and Montanari, 2009, Nishimori, 2001]
- Recently, they have been applied to *social sciences* as well: there is evidence that the complex behaviour of many observed socio-economic systems can be approached with the *quantitative tools* from statistical mechanics
 - e.g., *Econophysics* for crisis events [Stanley, 2008]

From Statistical Mechanics to Social Systems III

Social systems as statistical mechanical systems

- A group of isolated individuals neither knowing nor communicating with each other is the typical example of a *compositional* social system
- No sudden shifts are expected in this case at the collective level, unless it is caused by strong external exogenous causes
- To obtain a *collective behaviour* displaying *endogenous* phenomena, the individual *agents* should meaningfully *interact* with each other
- The foremost issue here is that *the nature of the interaction determines the nature of the collective behaviour* at the aggregate level
 - e.g., a simple *imitative* interaction is capable to cause strong polarisation effects even in presence of extremely small external inputs

Modelling vs. Engineering

Physical vs. computational systems

- Physical systems are to be observed, understood, and possibly *modelled*
 - For physical systems, the laws of interaction, and their role for complexity, are to be *taken as given*, to be possibly formalised mathematically by physicists
- Computational systems are to be *designed* and built
 - For computational systems, the laws of interaction have first to be *defined* through amenable abstractions and computational models by computer scientists, then exploited by computer engineers in order to build systems

Coordinated Systems as Interacting Systems I

Coordination media for ruling interaction

- Defining the abstractions for ruling the interaction space in computational systems basically means to define their *coordination model* [Gelernter and Carriero, 1992, Ciancarini, 1996, Ciancarini et al., 1999]
- *Global properties* of complex coordinated systems depending on interaction can be enforced through the *coordination model*, essentially based on its expressiveness
 - For instance, tuple-based coordination models have been shown to be expressive enough to support self-organising coordination patterns for nature-inspired distributed systems [Omicini, 2013]

Coordinated Systems as Interacting Systems II

The role of coordination models

Coordination models could be exploited

- to *rule* the *interaction space*
- so as to *define* new sorts of *global*, macroscopic *properties* for *computational systems*, possibly inspired by physical ones

Coordinated Systems as Interacting Systems III

Research perspectives

One should understand

- how to relate methods from statistical mechanics with coordination models
- whether notions such as *phase*, *phase transition*, or any other macroscopic system property, could be transferred from statistical mechanics to computer science
- what such notions would imply for computational systems
- whether new, original notions could apply to computational systems
- which sort of coordination model could support such notions

Socio-Technical Systems

Humans vs. software

- Nowadays, a particularly-relevant class of *social systems* is represented by *socio-technical systems*
- In socio-technical systems
 - active components are mainly represented by *humans*
 - whereas interaction is almost-totally regulated by the *software infrastructure*
 - where *software agents* often play a key role
- This is the case, for instance, of *social platforms* like FaceBook [FaceBook, 2014] and LiquidFeedback [LiquidFeedback, 2014]

Physical & Computational Social Systems I

A twofold view of socio-technical systems

- The nature of socio-technical systems is twofold: they are **both social systems and computational systems** [Verhagen et al., 2013, Omicini, 2012]
- As **complex social systems**, their complex behaviour is in principle amenable of mathematical modelling and prediction through notions and tools from statistical mechanics
- As **complex computational systems**, they are designed and built around some (either implicit or explicit) notion of coordination, ruling the interaction within components of any sort—be them either software or human ones

Physical & Computational Social Systems II

Computational systems meet physical systems

- In socio-technical systems, macroscopic properties could be
 - described by exploiting the conceptual tools from physics
 - enforced by the coordination abstractions
- Socio-technical systems could exploit both
 - the notion of complexity by statistical mechanics, along with the mathematical tools for behaviour modelling and prediction, and
 - coordination models and languages to suitably shape the interaction space

Physical & Computational Social Systems III

Vision

Complex socio-technical systems could be envisioned

- whose implementation is based on suitable coordination models
- whose macroscopic properties can be modelled and predicted by means of mathematical tools from statistical physics

thus reconciling the scientist and the engineer views over systems

Outline

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems**

Predicting Complex Behaviours

Engineering unpredictable systems around predictable abstractions

- coordination models are meant to harness the complexity of complex MAS [Ciancarini et al., 2000]
- coordination abstractions are often *at the core* of complex MAS
- while this does not make complex MAS generally predictable, it makes it possible in principle to make them *partially predictable*, based on the predictability of the core coordinative behaviour
- suitably-formalised coordination abstractions, along with a suitably-defined engineering methodology, could in principle ensure the predictability of given MAS properties within generally-unpredictable MAS

Conclusion I

In the overall...

- nature-inspired models of coordination already have a long history behind them
- and apparently a huge *potential* for development still to be explored
- to provide core abstractions and technologies for the engineering of complex computational systems, including pervasive, adaptive, and self-organising ones

- 1 Premises
- 2 Interaction & Coordination
- 3 Tuple-based Coordination Models
- 4 Nature-inspired Coordination Models
 - Why Nature-inspired Coordination?
 - Examples
 - Why Tuple-based Models for Nature-inspired Coordination?
- 5 Coordination for Self-organising Systems
- 6 Coordination & Stochastic Systems
- 7 Coordination for Complex Systems
- 8 Predicting Behaviours of Complex Coordinated Systems

Bibliography I



Banâtre, J.-P., Fradet, P., and Le Métayer, D. (2001).

Gamma and the chemical reaction model: Fifteen years after.

In Calude, C. S., Păun, G., Rozenberg, G., and Salomaa, A., editors, *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View*, volume 2235 of *LNCS*, pages 17–44. Springer.



Banâtre, J.-P. and Le Métayer, D. (1990).

The GAMMA model and its discipline of programming.

Science of Computer Programming, 15(1):55–77.



Berry, G. (1992).

The chemical abstract machine.

Theoretical Computer Science, 96(1):217–248.



Boltzmann, L. (1964).

Lectures on Gas Theory.

University of California Press.



Bouchaud, J.-P. and Potters, M. (2003).

Theory of Financial Risk and Derivative Pricing: From Statistical Physics to Risk Management.

Cambridge University Press, Cambridge, UK, 2nd edition.

Bibliography II



Brooks, R. A. (1991).

Intelligence without reason.

In Mylopoulos, J. and Reiter, R., editors, *12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, volume 1, pages 569–595, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.



Busi, N., Ciancarini, P., Gorrieri, R., and Zavattaro, G. (2001).

Coordination models: A guided tour.

In [Omicini et al., 2001], chapter 1, pages 6–24.



Busi, N., Gorrieri, R., and Zavattaro, G. (1998).

A process algebraic view of Linda coordination primitives.

Theoretical Computer Science, 192(2):167–199.



Cabri, G., Leonardi, L., and Zambonelli, F. (2000).

MARS: A programmable coordination architecture for mobile agents.

IEEE Internet Computing, 4(4):26–35.



Casadei, M. and Omicini, A. (2009).

Situated tuple centres in ReSpecT.

In Shin, S. Y., Ossowski, S., Menezes, R., and Viroli, M., editors, *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, Honolulu, Hawai'i, USA. ACM.

Bibliography III



Ciancarini, P. (1996).
Coordination models and languages as software integrators.
ACM Computing Surveys, 28(2):300–302.



Ciancarini, P., Omicini, A., and Zambonelli, F. (1999).
Coordination technologies for Internet agents.
Nordic Journal of Computing, 6(3):215–240.



Ciancarini, P., Omicini, A., and Zambonelli, F. (2000).
Multiagent system engineering: The coordination viewpoint.
In Jennings, N. R. and Lespérance, Y., editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer.
6th International Workshop (ATAL'99), Orlando, FL, USA, 15–17 July 1999. Proceedings.



De Nicola, R., Ferrari, G., and Pugliese, R. (1998).
KLAIM: A kernel language for agent interaction and mobility.
IEEE Transaction on Software Engineering, 24(5):315–330.



De Nicola, R., Latella, D., Katoen, J.-P., and Massink, M. (2006).
StoKlaim: A stochastic extension of Klaim.
Technical Report 2006-TR-01, Istituto di Scienza e Tecnologie dell'Informazione
“Alessandro Faedo” (ISTI).

Bibliography IV



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.



Denti, E. and Omicini, A. (1999).

Designing multi-agent systems around an extensible communication abstraction.

In Meyer, J.-J. C. and Schobbens, P.-Y., editors, *Formal Models of Agents*, volume 1760 of *LNAI*, pages 90–102. Springer-Verlag.

ESPRIT Project ModelAge Final Workshop. Selected Papers.



FaceBook (2014).

Home page.

<http://www.facebook.com>.



Fredriksson, M. and Gustavsson, R. (2004).

Online engineering and open computational systems.

In Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organization*, pages 377–388. Kluwer Academic Publishers.

Bibliography V



Freeman, E., Hupfer, S., and Arnold, K. (1999).
JavaSpaces Principles, Patterns, and Practice: Principles, Patterns and Practices.
 The Jini Technology Series. Addison-Wesley Longman.



Gardelli, L., Viroli, M., Casadei, M., and Omicini, A. (2007).
 Designing self-organising MAS environments: The collective sort case.
 In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*, pages 254–271. Springer.
 3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected
 Revised and Invited Papers.



Gelernter, D. (1985).
 Generative communication in Linda.
ACM Transactions on Programming Languages and Systems, 7(1):80–112.



Gelernter, D. (1989).
 Multiple Tuple Spaces in Linda.
 In Odijk, E., Rem, M., and Syre, J.-C., editors, *PARLE '89 Parallel Architectures and Languages Europe*, volume II: Parall of Springer Berlin Heidelberg, pages 20–27,
 Eindhoven, The Netherlands. Springer.

Bibliography VI



Gelernter, D. and Carriero, N. (1992).
Coordination languages and their significance.
Communications of the ACM, 35(2):97–107.



Goldin, D. Q., Smolka, S. A., and Wegner, P., editors (2006).
Interactive Computation: The New Paradigm.
Springer.



Grassé, P.-P. (1959).
La reconstruction du nid et les coordinations interindividuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs.
Insectes Sociaux, 6(1):41–80.



Kauffman, S. A. (1993).
The Origins of Order: Self-organization and Selection in Evolution.
Oxford University Press.



Kauffman, S. A. (2003).
Investigations.
Oxford University Press.

Bibliography VII



LiquidFeedback (2014).

Home page.

<http://liquidfeedback.org>.



Liu, J. and Tsui, K. C. (2006).

Toward nature-inspired computing.

Communications of the ACM, 49(10):59–64.



Malone, T. and Crowston, K. (1994).

The interdisciplinary study of coordination.

ACM Computing Surveys, 26(1):87–119.



Mamei, M. and Zambonelli, F. (2004).

Programming pervasive and mobile computing applications with the TOTA middleware.

In *Pervasive Computing and Communications*, pages 263–273.

2nd IEEE Annual Conference (PerCom 2004), Orlando, FL, USA, 14–17 March 2004.

Proceedings.



Mamei, M. and Zambonelli, F. (2006).

Field-Based Coordination for Pervasive Multiagent Systems. Models, Technologies, and Applications.

Springer Series in Agent Technology. Springer.

Bibliography VIII



Mantegna, R. N. and Stanley, H. E. (1999).
Introduction to Econophysics: Correlations and Complexity in Finance.
Cambridge University Press, Cambridge, UK.



Mariani, S. and Omicini, A. (2013).
Tuple-based coordination of stochastic systems with uniform primitives.
In Baldoni, M., Baroglio, C., Bergenti, F., and Garro, A., editors, *From Objects to Agents*,
volume 1099 of *CEUR Workshop Proceedings*, pages 8–15, Turin, Italy. Sun SITE Central
Europe, RWTH Aachen University.
XIV Workshop (WOA 2013). Workshop Notes.



Mézard, M. and Montanari, A. (2009).
Information, Physics, and Computation.
Oxford University Press, Oxford, UK.



Milner, R. (1993).
Elements of interaction: Turing award lecture.
Communications of the ACM, 36(1):78–89.

Bibliography IX



Minsky, N. H. and Ungureanu, V. (2000).

Law-Governed interaction: A coordination and control mechanism for heterogeneous distributed systems.

ACM Transactions on Software Engineering and Methodology (TOSEM), 9(3):273–305.



Murphy, A. L., Picco, G. P., and Roman, G.-C. (2006).

LIME: A coordination model and middleware supporting mobility of hosts and agents.

ACM Transactions on Software Engineering and Methodology, 15(3):279–328.



Nardini, E., Omicini, A., Viroli, M., and Schumacher, M. I. (2011).

Coordinating e-health systems with TuCSoN semantic tuple centres.

Applied Computing Review, 11(2):43–52.



Nardini, E., Viroli, M., and Panzavolta, E. (2010).

Coordination in open and dynamic environments with TuCSoN semantic tuple centres.

In Shin, S. Y., Ossowski, S., Schumacher, M., Palakal, M., Hung, C.-C., and Shin, D., editors, *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, volume III, pages 2037–2044, Sierre, Switzerland. ACM.

Bibliography X



Nishimori, H. (2001).

Statistical Physics of Spin Glasses and Information Processing: An Introduction, volume 111 of *International Series of Monographs on Physics*. Clarendon Press, Oxford, UK.



Nixon, L., Simperl, E., Krummenacher, R., and Martin-recuerda, F. (2008).

Tuplespace-based computing for the Semantic Web: A survey of the state-of-the-art. *Knowledge Engineering Review*, 23(2):181–212.



Omicini, A. (2012).

Agents writing on walls: Cognitive stigmergy and beyond.

In Paglieri, F., Tummolini, L., Falcone, R., and Miceli, M., editors, *The Goals of Cognition. Essays in Honor of Cristiano Castelfranchi*, volume 20 of *Tributes*, chapter 30, pages 565–578. College Publications, London.



Omicini, A. (2013).

Nature-inspired coordination for complex distributed systems.

In Fortino, G., Bădică, C., Malgeri, M., and Unland, R., editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 1–6. Springer. 6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, 24–26 September 2012. Proceedings. Invited paper.

Bibliography XI



Omicini, A. and Denti, E. (2001).
From tuple spaces to tuple centres.
Science of Computer Programming, 41(3):277–294.



Omicini, A., Ricci, A., and Viroli, M. (2005).
Time-aware coordination in ReSpecT.
In Jacquet, J.-M. and Picco, G. P., editors, *Coordination Models and Languages*, volume 3454 of *LNCS*, pages 268–282. Springer-Verlag.
7th International Conference (COORDINATION 2005), Namur, Belgium,
20–23 April 2005. Proceedings.



Omicini, A., Ricci, A., and Viroli, M. (2006).
The multidisciplinary patterns of interaction from sciences to Computer Science.
In Goldin, D. Q., Smolka, S. A., and Wegner, P., editors, *Interactive Computation: The New Paradigm*, pages 395–414. Springer.



Omicini, A., Ricci, A., and Viroli, M. (2007).
Timed environment for Web agents.
Web Intelligence and Agent Systems, 5(2):161–175.

Bibliography XII



Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., and Tummolini, L. (2004). Coordination artifacts: Environment-based coordination for intelligent agents. In Jennings, N. R., Sierra, C., Sonenberg, L., and Tambe, M., editors, *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA. ACM.



Omicini, A. and Viroli, M. (2011). Coordination models and languages: From parallel computing to self-organisation. *The Knowledge Engineering Review*, 26(1):53–59.



Omicini, A., Zambonelli, F., Klusch, M., and Tolsdorf, R., editors (2001). *Coordination of Internet Agents: Models, Technologies, and Applications*. Springer-Verlag.



Parunak, H. V. D. (2006). A survey of environments and mechanisms for human-human stigmergy. In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for Multi-Agent Systems II*, volume 3830 of *LNCS*, pages 163–186. Springer.

Bibliography XIII



Parunak, H. V. D., Brueckner, S., and Sauter, J. (2002).

Digital pheromone mechanisms for coordination of unmanned vehicles.

In Castelfranchi, C. and Johnson, W. L., editors, *1st International Joint Conference on Autonomous Agents and Multiagent systems*, volume 1, pages 449–450, New York, NY, USA. ACM.



Ricci, A., Omicini, A., Viroli, M., Gardelli, L., and Oliva, E. (2007).

Cognitive stigmergy: Towards a framework based on agents and artifacts.

In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*, pages 124–140. Springer.

3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers.



Ricci, A., Viroli, M., and Omicini, A. (2005).

Environment-based coordination through coordination artifacts.

In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for Multi-Agent Systems*, volume 3374 of *LNAI*, pages 190–214. Springer.

1st International Workshop (E4MAS 2004), New York, NY, USA, 19 July 2004. Revised Selected Papers.

Bibliography XIV



Rossi, D., Cabri, G., and Denti, E. (2001).
Tuple-based technologies for coordination.
In [Omicini et al., 2001], chapter 4, pages 83–109.



Rowstron, A. I. T. (1996).
Bulk Primitives in Linda Run-Time Systems.
PhD thesis, The University of York.



Simon, H. A. (1962).
The architecture of complexity.
Proceedings of the American Philosophical Society, 106(6):467–482.



Stanley, H. E. (2008).
Econophysics and the current economic turmoil.
American Physical Society News, 17(11):8.
The Back Page.

Bibliography XV



Tolksdorf, R. and Menezes, R. (2004).

Using Swarm Intelligence in Linda Systems.

In Omicini, A., Petta, P., and Pitt, J., editors, *Engineering Societies in the Agents World IV*, volume 3071 of *LNCS*, pages 49–65. Springer.

4th International Workshops (ESAW 2003), London, UK, 29-31 October 2003. Revised Selected and Invited Papers.



Tolksdorf, R., Nixon, L., and Simperl, E. (2008).

Towards a tuplespace-based middleware for the Semantic Web.

Web Intelligence and Agent Systems, 6(3):235–251.



Verhagen, H., Noriega, P., Balke, T., and de Vos, M., editors (2013).

Social Coordination: Principles, Artefacts and Theories (SOCIAL.PATH), AISB Convention 2013, University of Exeter, UK. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.



Viroli, M. and Casadei, M. (2009).

Biochemical tuple spaces for self-organising coordination.

In Field, J. and Vasconcelos, V. T., editors, *Coordination Languages and Models*, volume 5521 of *LNCS*, pages 143–162. Springer, Lisbon, Portugal.

11th International Conference (COORDINATION 2009), Lisbon, Portugal, June 2009. Proceedings.

Bibliography XVI



Viroli, M., Casadei, M., Montagna, S., and Zambonelli, F. (2011).
Spatial coordination of pervasive services through chemical-inspired tuple spaces.
ACM Transactions on Autonomous and Adaptive Systems, 6(2):14:1–14:24.



Viroli, M., Casadei, M., Nardini, E., and Omicini, A. (2010).
Towards a chemical-inspired infrastructure for self-* pervasive applications.
In Weyns, D., Malek, S., de Lemos, R., and Andersson, J., editors, *Self-Organizing Architectures*, volume 6090 of *LNCS*, chapter 8, pages 152–176. Springer.
1st International Workshop on Self-Organizing Architectures (SOAR 2009), Cambridge, UK, 14-17 September 2009, Revised Selected and Invited Papers.



Viroli, M., Casadei, M., and Omicini, A. (2009).
A framework for modelling and implementing self-organising coordination.
In Shin, S. Y., Ossowski, S., Menezes, R., and Viroli, M., editors, *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, volume III, pages 1353–1360, Honolulu, Hawai'i, USA. ACM.



Viroli, M. and Omicini, A. (2006).
Coordination as a service.
Fundamenta Informaticae, 73(4):507–534.
Special Issue: Best papers of FOCLASA 2002.

Bibliography XVII



Viroli, M., Pianini, D., Montagna, S., and Stevenson, G. (2012).
Pervasive ecosystems: a coordination model based on semantic chemistry.
In Ossowski, S., Lecca, P., Hung, C.-C., and Hong, J., editors, *27th Annual ACM Symposium on Applied Computing (SAC 2012)*, Riva del Garda, TN, Italy. ACM.



Viroli, M. and Zambonelli, F. (2010).
A biochemical approach to adaptive service ecosystems.
Information Sciences, 180(10):1876–1892.



Wegner, P. (1996).
Coordination as constrained interaction.
In Ciancarini, P. and Hankin, C., editors, *Coordination Languages and Models. First International Conference, COORDINATION '96 Cesena, Italy, April 15–17, 1996. Proceedings*, Lecture Notes in Computer Science, pages 28–33, Cesena, Italy. Springer Berlin Heidelberg.



Wegner, P. (1997).
Why interaction is more powerful than algorithms.
Communications of the ACM, 40(5):80–91.

Bibliography XVIII



Wegner, P. and Goldin, D. (2003).
Computation beyond Turing machines.
Communications of the ACM, 46(4):100–102.



Wyckoff, P., McLaughry, S. W., Lehman, T. J., and Ford, D. A. (1998).
T Spaces.
IBM Journal of Research and Development, 37(3 – Java Techonology):454–474.



Zavattaro, G. (1998).
On the incomparability of Gamma and Linda.
Technical Report SEN-R9827, CWI, Amsterdam, The Netherlands.

Coordination Models and Technologies towards Self-Organising Systems

Frontiers of Multimedia Systems

Andrea Omicini
andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM—Università di Bologna

ATER 2013/2014