Spatial Multi-Agent Systems

Andrea Omicini, Stefano Mariani, Mirko Viroli
{andrea.omicini, s.mariani, mirko.viroli}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna a Cesena

18th European Agent Systems Summer School (EASSS 2016)
Catania, Italy, 28 July 2016

Outline

Humans & Space

Environment awareness, measure & modelling

For early humans

- *awareness* of the surrounding environment as the premise to self-awareness [Mar98]
- *measure* and *modelling* of the environment as on of the premises to goal-driven actions

Space & activity

For humans, space is a conceptual tool

- to *model* the environment where we live
- to organise *resources* and *activities*
- and their *dynamics* as well

Spatial Reasoning I

Representing space

- the way in which we *represent* space mutually affects the way in which we can *reason about* space
- in some way not well understood, yet
  - for instance, [LG02] and [LKHR02] explicitly express two clearly contrasting views on the matter

Spatial Reasoning II

Reasoning on space

- humans have their own ways to represent space, and to process spatial information [BJL89]
- *spatial reasoning* is a feature of the intelligent process—humans are not alone [Gen07, HCJL06]
- human spatial reasoning is may be either qualitative or quantitative—but is mostly *approximated* [Dut88]

! for a well-organised account of how humans (learn to) represent and reason about space see [CB92]

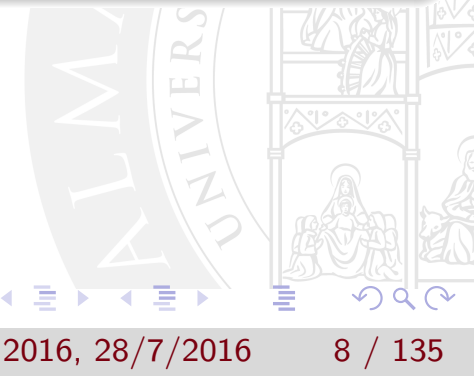
Outline

Geometry

Modelling space

Abstracting away from our perception of reality

- basic geometric concepts (point, line, angle, circle, . . .)
- basic geometric shapes (triangle, rectangle, trapezoid, . . .)
- from Babylonians to Egyptians to Greeks
- Babylonian *astronomy*
- Greek *geometry*



Euclidean Geometry

Axiomatic approach to geometry

- Euclide's *Elements* [Kli72]
- geometry [ABBG12]
  - no longer seen "as a set of empirical observations and practical methods for measuring distances, area of land, etc."
  - instead conceived as "an abstract mathematical theory, which, while rooted in the perceived reality, had nevertheless its own, absolute right of existence and development"
- representing space and reasoning on space through axioms, theorems, proofs, ...

Geometry & Space beyond Human Perception

Modelling "imaginary" space

- beyond Euclidean space
  - physical space may not satisfy Euclid's fifth postulate
  - non-euclidean geometry
  - Riemann (elliptic), Bolyai & Lobachevsky (hyperbolic), ...
- representing *true* physical space beyond our direct sensorial-cognitive perception

[ABBG12]

> *That discovery was an unsurpassed manifestation of the superiority of the abstract, logical approach of mathematics over the empirical approach underpinning the natural sciences, at least when it comes to comprehending such fundamental physical concepts as space and time.*

Outline

From Euclid to Tarski [ABBG12] I

Basic question

Morris Kline, Foreword [Rus56]

> *What geometrical knowledge must be the logical starting point for a science of space and must also be logically necessary to the experience of any form of externality?*

Axiomatic investigations of the foundations of geometry

- axiomatic approach to geometry from Euclid to Peano
- studying relationships between axiomatic systems & basic geometric notions
- (sound) *axiomatic* foundation for geometry by Hilbert [Hil50]

From Euclid to Tarski [ABBG12] II

Analytic method in geometry of space

- Descartes' *coordinatisation* of the Euclidean space
- coordinate systems to solve purely geometric problems by using purely algebraic methods
- geometry as a study not of figures, but of *transformations* (Klein's Erlangen program, [BGKV07]) preserving fundamental geometrical properties
- abstract *algebraic* foundation for geometry

From Euclid to Tarski [ABBG12] III

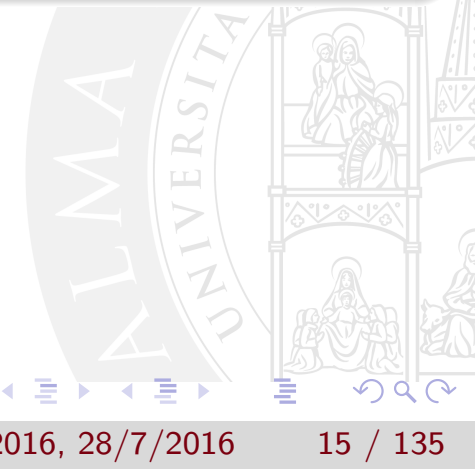
Logical foundation of geometry

- *elementary geometry* as first-order theory of Euclidean geometry [Tar59]
- elementary geometry can be developed axiomatically using just two geometric relations—*betweenness* and *equidistance*
- elementary geometry like field of reals via coordinatisation
- *completeness* and *decidability* of the first-order theory of the field of reals implies an explicit decision procedure for the elementary geometry
- (non efficient) *algorithm* for deciding the truth of any statement in Euclidean geometry

Modal Logics

Non-classical logics

- modelling, analysing, and reasoning about space with non-classical logics
- *modal logics* have proven to be particularly interesting, since
  - they can be more *specific*
  - they have better computational behaviour—very often *decidable*



Modal Logics of Topology I

Distance & metric

- basic problem: *distance*
- approach: notions of *metric*

Topology

- *neighbourhood* as a generalisation of metric
- from neighbourhood to *topology* [ST67]
- "alongside algebra and geometry, topology became one of the fundamental branches of contemporary mathematics" [ABBG12]

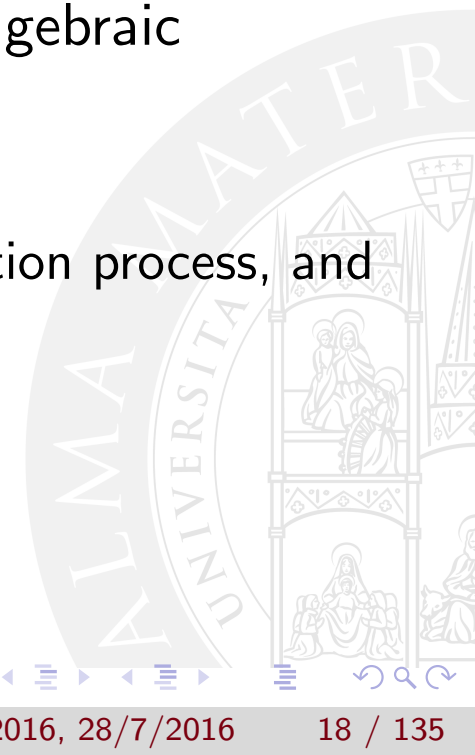
Modal Logics of Topology II

Logic and topology

- modal operators reinterpreted
  - $\Box$ *(necessarily)* as the interior operator
  - $\Diamond$ *(possibly)* as the closure operator of a topological space
- S4 [BS84] is *sound* and *complete* with respect to topological semantics [MT44]
- S4 is the *modal logic* of any Euclidean space

Mathematical Morphology [ABBG12]

- mathematical morphology (MM) analyses *shape*, *spatial information*, *image processing*
- any mathematical theory dealing with shapes can contribute to MM
- *modal morphologic* from the similarity between the algebraic properties of MM operators and of modal operators
- efficient for spatial reasoning
  - to guide the exploration of space, in a focus of attention process, and for recognition and interpretation tasks



Summary

- we have math and logic tools to *represent*, *analyse*, and *reason* about space
- we can *compute* about space and its organisation with diverse levels of efficiency

Outline

Outline

Distributed Systems

A new space for software components

- *physical distribution* of computational systems
  - distribution of *computational units*, *communication channels*, *data*
- local *networks* building the first virtual spaces
- Internet and IP-based locations for first "global" virtual spaces
- WWW as the first global space *shared* by agents and humans
  - a *knowledge-intensive* space

Middleware

Structured environments for software components

- *middleware* as operating systems for distributed systems
  - *software infrastructure* giving structure to *distributed environments*
  - possibly supporting *mobility* [FPV98]
- EAI horizontal integration to build *aggregated environments*

Mapping logical distribution upon physical distribution

- middleware
  - provides topological notions for distributed systems
  - maps logical distribution upon physical distribution

JADE [BCG07] provides agent programmers with the topological notions of *container* and *platform* to represent *locality* for agents

! sometimes, however, physical distribution is what actually matters

Distributed Systems & Situated Computations

Situated distributed systems

- physical distribution of computational systems is essential to cope with the *distributed nature* of many *working environments*
- . . . as well as with the need for *situated computation*
  - that is, computations occurring locally where either perception or action are taking place
  - either elaborating on perception, driving action, or both
- essentially, when system requirements mandate for situated computations within a distributed physical environment, *situated distributed systems* are the only way out
  - e.g., disaster recovery scenarios, environmental monitoring, crowd steering, . . .
- Internet of Things (IoT) just makes the need for situated computation unescapable

Pervasive Systems

Physical space

- *pervasive systems* emphasise the role of *physical environment*
- physical distribution of computational systems to cope with the *distributed nature* of physical application scenarios
- a distributed pervasive system is
  - is part of our surroundings
  - generally lacks of a human administrative control
  - is typically unstable

Example: Home Systems

Systems built around...

... home networks

- no way to ask people to act as a competent network / system administrator

→ home systems should be self-configuring and self-maintaining in essence

... personal information

- huge amount of heterogeneous personal information to be managed
- coming from heterogeneous sources from inside and outside the home system

... our personal, domestic *space*

- resources and activities at home are logically organised around our notion of domestic space
- which home systems need to model, adapt to, and fruitfully exploit

Situatedness I

Situated action [Suc87]

- (intelligent) actions are *purposeful*, and *not abstract*
- "every course of action depends in essential ways upon its material and social circumstances"
- situatedness is in short the property of systems of being *immersed* in their environment
- that is, of being capable to *perceive* and *produce* environment change, by suitably dealing with environment events
- mobile, adaptive, and pervasive computing systems have emphasised the key role of situatedness for nowadays computational systems [ZCF+11]

Situatedness II

Situatedness & context-awareness

- computational systems are more and more affected by their physical nature
- this is not due to a lack of abstraction: indeed, we are suitably layering systems – including hardware and software layers –, where separation between the different layers is clear and well defined
- instead, this is mostly due to the increasingly complex requirements for computational systems, which mandate for an ever-increasing *context-awareness* [BDR07]
- defining the notion of *context* is a complex matter [Omi02], with its boundaries typically blurred with the notion of *environment*, in particular in the field of *multi-agent systems* (MAS) [WOO07]

Situatedness III

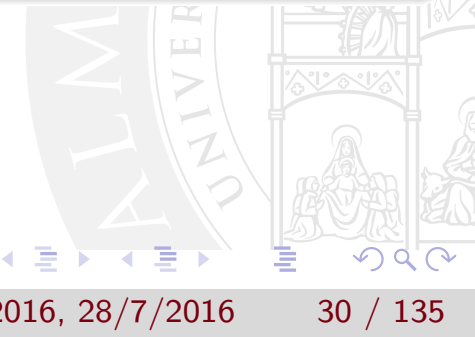
Context-awareness: environment

- situatedness of computational systems – MAS in particular – requires *awareness of the environment* where the systems are deployed and are expected to work
- computational systems and their components need to understand their working environment, its *nature*, its *structure*, the *resources* it makes available for use, the possible *issues* that may harm the systems in any way

Situatedness IV

Context-awareness: space & time

- in particular, mobile computing scenarios have made clear that *situatedness* of computational systems nowadays requires at least *awareness of the spatio-temporal fabric*
- computational systems and their components need to know *where* it is working, and *when*, in order to effectively perform its function
- in its most general acceptation, then, any (working) environment for nowadays non-trivial computational systems is first of all made of space and time



Summary

- distributed systems originate from *physical distribution* of computation, communication, and data
- *middleware* provides logical topology upon physical distribution
- pervasive systems and IoT make *situated computation* a requirement, and *situated distributed systems* the only viable approach
- nowadays non-trivial computational systems needs to be *space-aware*

Outline

Computational Geometry [dBvKOS00]

Computing with geometry to solve problems

- using *space* to *represent problems*—either spatial or non spatial ones
- using *geometry* to make them *computable*
- using *algorithms* to compute solutions

Example

- finding the nearest coffee shop in the campus here in Catania
- finding it for any point in the campus
- dividing the campus in regions around each coffee shop, including all the points for which each shop is the nearest one
- how do we compute this?

Geographical Information Systems (GIS)

Computing with geography

- representing geographic information
- capturing / storing / checking / displaying data representing positions on the Earth—maybe other planets, too, in the (near) future

→ the notion of space is clearly defined in GIS

- more generally,

> *A geographic information system is a computer-based system that supports the study of natural and man-made phenomena with an explicit location in space. To this end, the GIS allows data entry, data manipulation, and production of interpretable output that may provide new insights about the phenomena.[HdB09]*

Virtual Reality (VR)

Activity in a virtual environments

- artificial worlds—with artificial *space*
- digitally-created, represented exclusively as *computational environments*
- where "real people" can actually perform sensorimotor and cognitive activity

Definition [FMG11]

Virtual reality is a scientific and technical domain that uses computer science (1) and behavioural interfaces (2) to simulate in a virtual world (3) the behaviour of 3D entities, which interact in real time (4) with each other and with one or more users in pseudo-natural immersion (5) via sensorimotor channels.

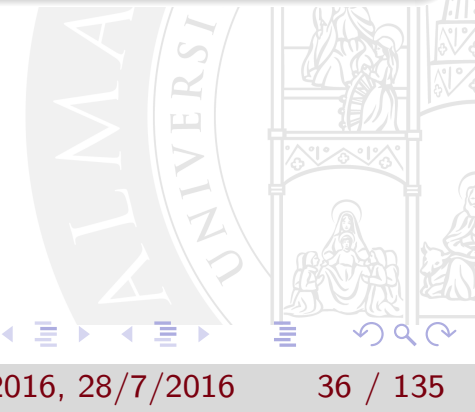
! Here, interaction and immersion are the key words

Gaming

Virtual worlds for gaming

- gaming is the most prominent application of virtual reality
  - e.g., Microsoft Kinect is one of the most well-known applications of VR
- gaming platforms nowadays provide the means for building whole virtual worlds
  - e.g., Unity3D, Unreal Engine

that can be explored from a wide range of diverse devices



Outline

Location-based Services (LBS)

Adding a virtual layer to physical space

- LBS use information about the position of a user / device to provide information, entertainment, security
  - e.g., AroundMe, Pokemon Go
- *mixed reality* is an old term possibly coming back to use
  - yet, with an uncertain meaning

Augmented Reality (AR)

Integrating virtual and physical space

- merging real-world information with *context-sensitive* digital content in a meaningful way [Fur11]
  - the point here is that the virtual and physical layers *mutually affect* each other dynamically
- *gamification* can easily lead to any sort of relevant application—medical, civic, educational, touristic, . . .
  - e.g., GeoZombie [PRS+16]—a step further (and before) Pokémon Go

Outline

Altogether Now: Spatial Computing I

Whenever space cannot be abstracted away from computation [SFA16]

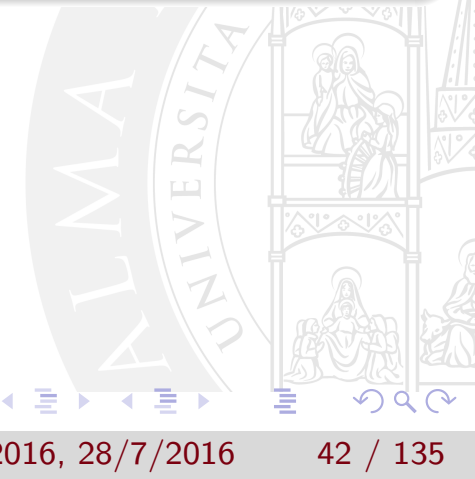
During the "Computing Media and Languages for Space-Oriented Computation" workshop in Dagstuhl (`http://www.dagstuhl.de/06361`), three classes of spatial systems were identified:

- distributed systems, where space is either a means or a resource—a.k.a. intensive computing, or *coping with space*
- situated systems, where location in space (and time) is essential for computation—a.k.a. *embedded in space*
- spatial systems, where space is fundamental to the application problem, is explicitly represented and manipulated, and is essential to express the result of a computation—a.k.a. *representing space*

Altogether Now: Spatial Computing II

Spatial computers, spatial computing [BMS11]

- a *spatial computer* – or, *spatial computing system* – is a computational system where (the logic of) space is essential in representing the problem, define computation, and express the result
- *spatial computing* is any form of computation where (the logic of) space is relevant to express and perform computation

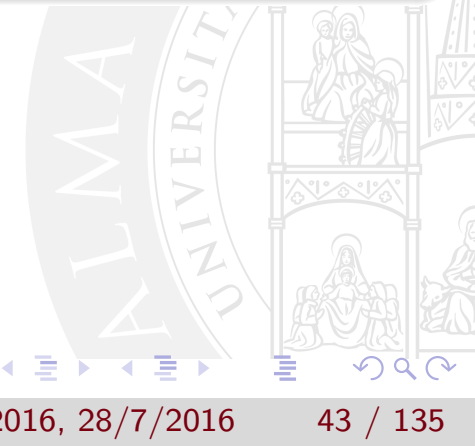


Spatial Computing Languages (SCL)

Languages for spatial computing

Spatial Computing Languages (SCL)

- were born to address the issue of bringing space *into* programming languages
- allowing programmers to explicitly deal with space-related aspects *at the language level*



Proto

Archetypical SCL example

Proto [VBU13]

- is a purely functional language with a LISP-like syntax
- uses a continuous space abstraction (the *amorphous medium*) to model the notion of spatial computer
- has mathematical operations on space-time as primitives
- has programs specified in terms of geometric computations and information flow on a topological space

Proto *is a Domain-Specific Language (DSL) for representing the description of aggregate device behaviour in a spatial computer* [VBU13]

- it embeds the *aggregate computing* model [BPV15] as an original paradigm to organise spatial computations
- it works as a sort of benchmark for SCL of any sort

Expressiveness of SCL [BMS11] I

The Abstract Device Model

In order to allow comparison between different SCL, the Abstract Device Model (ADM) is defined in [BDU+13], working along three basic criteria

communication region — the spatial "coverage" of the communication—e.g., global vs. neighbourhood

communication granularity — the number of receivers—e.g., unicast vs. multicast

code mobility — the relationships between the running code of different devices

Expressiveness of SCL [BMS11] II

Types of space-time operations

Furthermore, three classes of operators are required in SCL in order to achieve a kind of *spatial Turing equivalence* [Bea10]

measure space — to translate spatial properties into computable information—e.g. sensing GPS position

manipulate space — to translate back information into modifications of spatial properties—e.g. starting a motion engine

compute (on) space — any kind of "spatial-pointwise" computation

A fourth class (*physical evolution*) looks more like an assumption about the dynamics that a given program/device must/can consider—e.g. a robot may move while a computation runs

Expressiveness of SCL [BMS11] III

The "T-Program"

As a reference *benchmark* to compare the expressiveness of different SCL, the "T-Program" is proposed in [BDU+13] w.r.t. a set of independent moveable devices

i. cooperatively create a local coordinate system
ii. move or grow devices into a T-shaped structure
iii. determine its center of gravity, then draw a ring around it

SCL Requirements

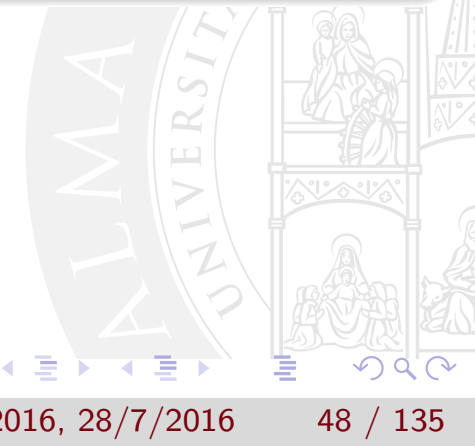
Hence it requires all the three classes of operators afore-mentioned: stage *(i)* requires *measuring space* capabilities; stage *(ii)* requires *manipulating space* capabilities; stage *(iii)* requires both *compute over space* capabilities and, again, measuring capabilities.

Expressiveness of SCL [BMS11] IV

Local Coordinate System

Setting a local coordinate system basically amounts to

i. choosing an origin node
ii. making it *spread* a vector tuple to neighbours
iii. (recursively) making them increment such a vector
iv. forwarding it to neighbours



Expressiveness of SCL [BMS11] V

T-shaped Structure

To arrange nodes (tuple centres) so as to form a T-shaped structure, it is required to

i define spatial constraints representing the T

ii make every node move so as to satisfy them

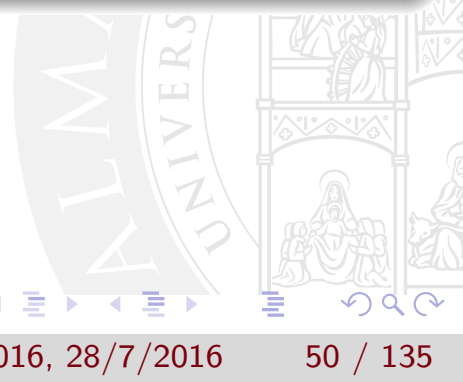
Thus, the basic mechanism needed at the VM level is motion monitoring and control

Expressiveness of SCL [BMS11] VI

Center of Gravity

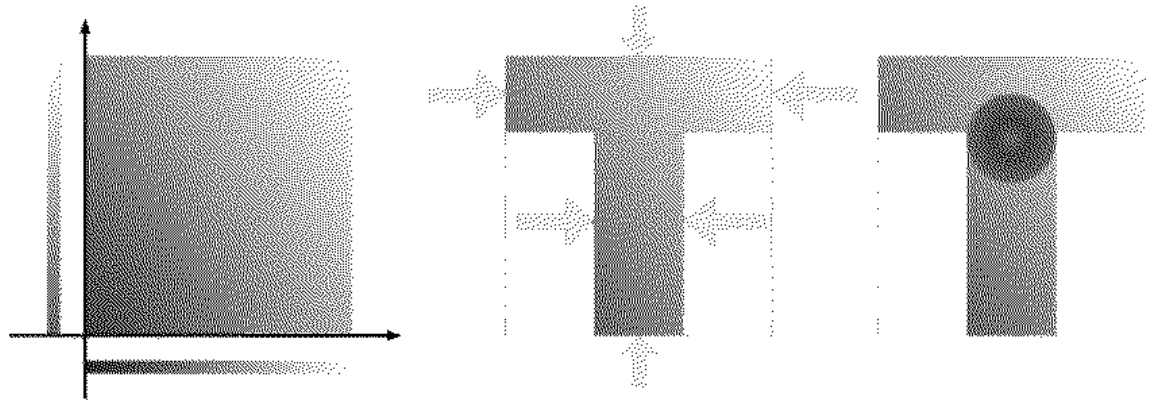
To first compute the focal point (FC) of the T-shape, then draw a sphere around it, two basic mechanisms are needed, both similar to the *neighbourhood spreading* previously shown

- a *bidirectional* neighbourhood spreading to collect replies to sent messages—allowing to aggregate all the node's coordinates and count them
- a *spherical multicast* to draw the ring pattern



Expressiveness of SCL [BMS11] VII

Figure 5. The "T program" reference example exercises the three main classes of space-time operations: measurement of space-time to organize local coordinates (a) and compute the center of gravity (c), manipulation of space-time to move devices into a T-shaped structure (b), and pattern computation to make a ring around the center of gravity (c).



(a) (b) (c)

[BDU+13]

Other SCL and Issues I

- a survey over SCL can be found in [BDU+13]
- diverse classes of SCL are listed
  - amorphous computing
  - biological
  - agent-based
  - wireless sensor networks (WSN)
  - pervasive computing
  - swarm and modular robotics
- and compared against the T-program

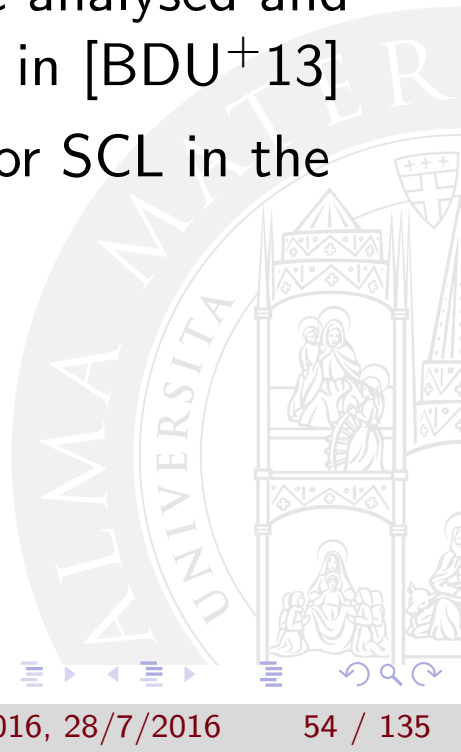
Other SCL and Issues II

Main issue

- the reference framework defined in [BDU+13] perfectly works in terms of language expressiveness when focussing on spatial issues
  - many diverse sort of SCL languages can be analysed and compared
  - so, just to be clear, it is perfect to its purpose
- however, we miss a general conceptual framework where all those language could fit altogether
  - so as to make it possible, for instance, where do they *generally* belong in the engineering of complex software systems
  - e.g., TOTA [MZ09] and Proto [VBU13] cannot really stay in the same place
- a more expressive conceptual framework for SCL – and spatial computing in general – would be dearly needed

Summary

- spatial computing is a sort of "landscape framework" for the many sorts of spatial computations and systems around
- an already-long list of SCL are available, which can be analysed and compared through the conceptual framework provided in [BDU+13]
- a more general framework for finding the right place for SCL in the general-purpose SE process would be needed



Outline

Agents as Autonomous Entities [ORV08] I

Definition (Agent)

Agents are *autonomous computational entities*

genus agents are computational entities

differentia agents are autonomous, in that they encapsulate control along with a criterion to govern it

Agents are *autonomous*

- from autonomy, many other features stem
  - autonomous agents *are* interactive, social, proactive, and situated
  - they *might* have goals or tasks, or be reactive, intelligent, mobile
  - they live within MAS, and *interact* with other agents through *communication actions*, and with the environment with *pragmatical actions*

Agents and Space: Basics I

Autonomy

- autonomy is an essential feature for distributed systems
  - coupling of control is maybe the main potential problem in distributed systems
  - uncoupling is required to prevent delays, deadlocks, faults
  - computational autonomy ensures uncoupling of control, since agents encapsulate control
- this is particularly clear in pervasive systems, where instability makes autonomy an essential feature

$\rightarrow$ spatial distribution mandates for autonomy

Agents and Space: Basics II

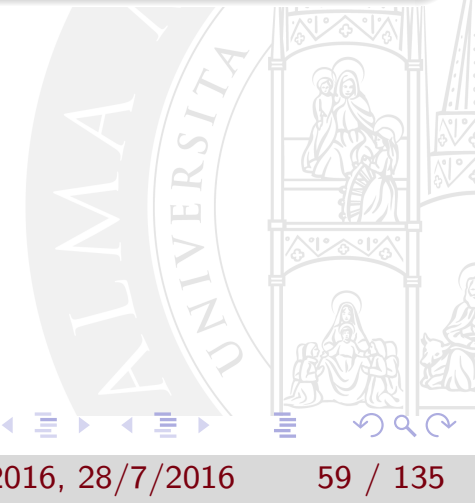
Reactiveness to change

- the ability to be sensitive to environment change is obviously the generalisation of the reactiveness to spatial changes in the surrounding of an agent
- which might be change of
  - either the relative or absolute position / motion of the agent
  - the position / motion of resources in the surrounding
  - the structure of the spatial environment

Agents and Space: Basics III

Situatedness

- the property of an agent of being immersed within its surrounding environment [FM96]
  - is somehow a generalisation of reactiveness
  - is connected to context-awareness
  - is particularly relevant in term of spatio-temporal situatedness



Agents and Space: Basics IV

Mobility

- the ability of an agent of changing its own position within either logical or physical space, by either software or physical agents
- is particularly relevant since it is associated to autonomy—agents can say "Go!" [Ode02]
- may require the ability to move autonomously, through e.g., sensorimotor actions
  - but may also be associated to agents hosted by mobile devices
- may obviously benefit by any form of spatial representation and reasoning

Agents and Space: Basics V

Intelligence

- ! we do not discuss (artificial) intelligence here
- however, intelligent agents are possibly the most suitable vessel for *spatial reasoning* [Kra01]—including
  - languages for spatial representation
  - logics for spatial reasoning

Agents & Spatial Reasoning I

Spatial reasoning by cognitive agents

- cognitive abilities of intelligent agents can be in principle exploited for spatial reasoning [Dut88]
  - e.g., the ability to separately handle and properly use epistemic knowledge is an obvious benefit when dealing with spatial information
- diverse logics can be embedded into an agent architecture, so as to provide for different ways to reason about space
  - fuzzy logic in [Dut88]
  - hybrid logic for common sense reasoning in [BMP07]
  - propositional logic in [Ben92]

Agents & Spatial Reasoning II

Spatial reasoning by physical agents

- actually, the most effective work on spatial reasoning till now comes from robotics [dBvKOS00]
  - robot teams [MW03]
  - robot swarms [Ham10]
  - robots as MAS [WS12]
- a huge flow of literature, including spatial self-organisation [Zam04], robot coordination [MW03], etc.

Summary?

Something missing here

- what?

Outline

Outline

MAS: Basic Abstractions

Agents in MAS

- agents live within an agent *society*
- agents live immersed within an agent *environment*

Basic design abstractions for MAS [WOO07]

agents	are the autonomous components of the systems, embodying the *designed* activities
society	is meant to model and govern the *relationships* among agents
environment	models the either virtual or physical context where the agents are situated, capturing both the effect of agent activities and the unpredictable change brought about by *non-designed* activities

Society and Coordination

Agent society

- autonomous agents encapsulate designed activities in a MAS
- agents live within an agent society
- an agent society models and governs mutual agent relationships

Interaction and coordination

- when agent relationships are expressed in terms of *mutual dependencies*, their govern is a matter of *coordination* [MC94]
- *coordination models* and *languages* [COZ00] rule agent interaction within agent societies and MAS
- *coordination abstractions* – a.k.a. coordination media – model agent societies, and govern agent social relationships by *ruling their mutual interaction*

Coordination Models for MAS I

Coordination model as a glue

A coordination model is the glue that binds separate activities into an ensemble [GC92]

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 [Cia96]

Coordination Models for MAS II

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 [Cia96]*

Examples

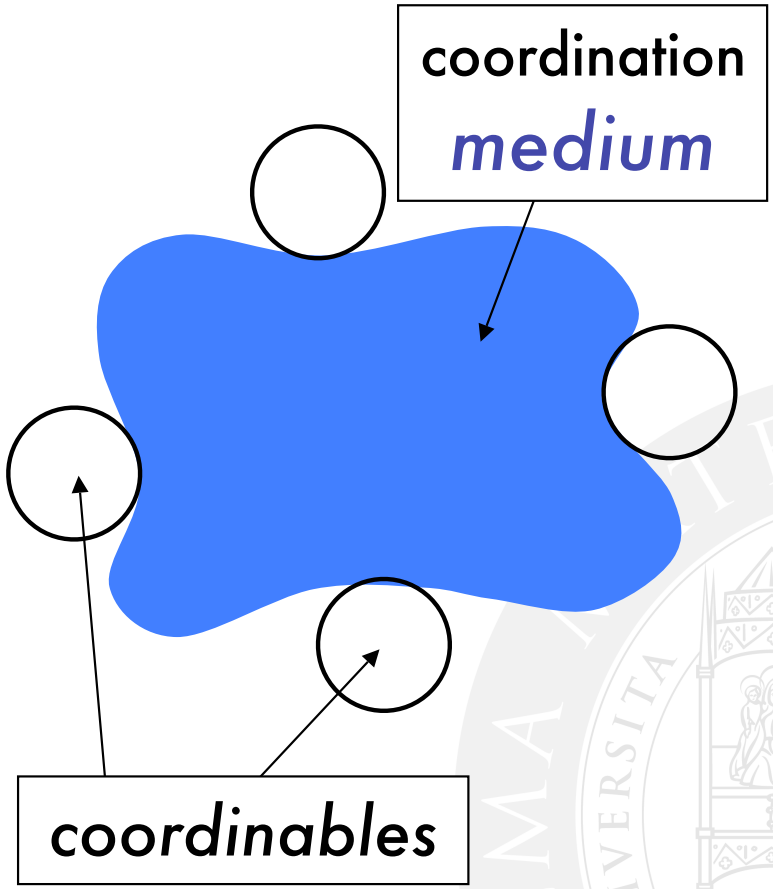
- blackboard-based systems [Cor91]
- LINDA coordination model [GC92, Gel85]

... along with its many derivative, a.k.a. *tuple-based* coordination models [RCD01, Omi99]

Coordination Models for MAS III

The *medium of coordination*

- "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 Models for MAS IV

Coordination: a meta-model [Cia96]

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

Coordination Models for MAS V

Example

For instance, in tuple-based coordination models

coordination entities are the *agents* interacting by exchanging *tuples* via `out`, `rd`, `in` *coordination primitives*

coordination media are the *tuple spaces* where tuples are written (`out`), read (`rd`), and consumed (`in`) by agents

coordination laws are the rules of model, such as

- *pattern matching* between tuples and templates in `rd` and `in`
- *suspensive semantics* for no match
- *non-determinism* for multiple matches

Society and Space

Spatial dependencies

- agent activities can be situated—first of all on their spatio-temporal features
- activities in a situated MAS are logically organised also based on space

→ agent activities may depend on each other on a spatial basis

Spatial coordination

- dealing with spatial dependency between agent activities mandates for *spatial coordination*
- *space-aware* coordination models [MO13] are required
- many remarkable examples have emerged in the last years

Outline

KLAIM I

- KLAIM [DNFP98] consists of a core LINDA with multiple tuple spaces where tuples and operations are located at specific *sites* of a net;
- in particular
  - S is a set of sites, or physical localities, s—e.g., the address of a node where processes and tuple spaces are allocated
  - Loc is a set of (logical) localities l—e.g., the symbolic name for a site; a distinguished locality $\texttt{self} \in Loc$ is assumed, referring to the local execution site of a KLAIM program
- localities permit structuring programs over distributed environments while ignoring their precise allocations
- KLAIM primitives are all located on some logical locality l
- given a finite set of sites, a KLAIMnet is a set of nodes
- a KLAIMnode is a triple (s, P, σ) where s is a site and σ is the allocation environment—e.g., a (partial) function from Loc to S

KLAIM II

- processes at each site can potentially access any other site of the net; however, site visibility is (locally) controllable via the allocation environment
  - a site `s''` is visible at the node `(s, P, σ)` only if `s''` belongs to the image of σ
  - if σ_1, σ_2 are allocation environments, then $\sigma_1 \cdot \sigma_2$ is the environment defined by

$$\sigma_1 \cdot \sigma_2(l) = \begin{cases} \sigma_1(l) & \text{if } \sigma_1(l) \text{ is defined} \\ \sigma_2(l) & \text{otherwise} \end{cases}$$

    where in σ_1, σ_2, σ_1 is the inner environment and σ_2 is the outer environment
- the operational semantics of nets adopts
  - *static scoping* for the evaluation of `out` operations
  - *dynamic scoping* for remote `eval` operations, where the meaning of localities used by a process spawned at a remote site depends on the remote allocation environment

KLAIM III

- indeed, whenever a process P located at the site s_1 wishes to insert a tuple t into the remote tuple space located at s_2, the local environment of P, namely σ_1, is used for evaluating t
- instead, when process P located at s_1 wants to spawn a process Q at the remote site s_2, the local environment σ_2 is used for giving meaning to the localities which may be referred in Q

$\sigma\tau$-LINDA I

- $\sigma\tau$-LINDA [VPB12] is an extension to the original LINDA model toward amorphous and spatial computing [BMS11]
- the claim is that computation in a *dense* and *mobile* system is best understood and designed in terms of *spatial* abstractions (e.g., regions, partitions, gradients, trails, paths) and *temporal* abstractions (e.g., fading, growth, movement, perturbation)
- agents in a MAS interact by injecting into the coordination substrate not tuples, but *space-time activities*—namely, processes that manipulate the space-time configuration of tuples
- activities are expressed in terms of a process-algebra like language, composing LINDA usual primitives (`out`, `in`, `rd`) with additional constructs borrowed from the Proto language [BB06]

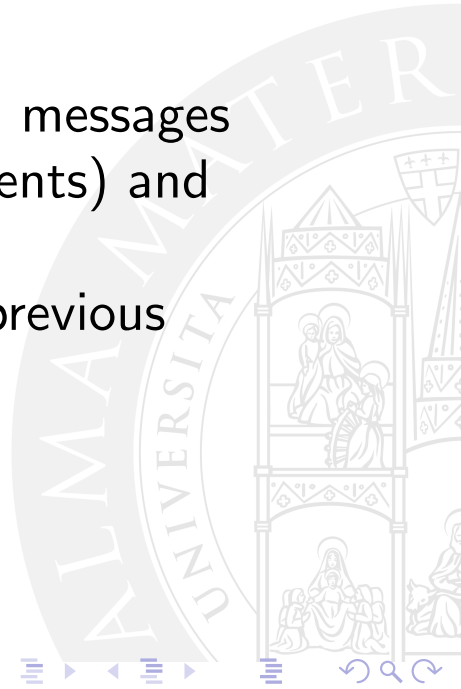
$\sigma\tau$-LINDA II

`next` P to schedule P (the protocol defining an activity) for execution at the next *computation round* of the current node; special variable `$delay` can be used in P and evaluates to the amount of time passed in between the current computation round and the previous one; similarly, variable `$this` can be used to denote the identifier of the node on which it is evaluated

`neigh` P to send a broadcast message containing P to all neighbours of current node, which will then execute P at their next computation round; special variable `$distance` is also introduced, which evaluates to the estimated distance between the node that sent the message and the one that received it; similarly, variable `$orientation` can be used to denote the relative direction from the receiver to the sender

$\sigma\tau$-LINDA III

- $\sigma\tau$-LINDA assumes a (possibly very dense and mobile) set of *situated* computational devices, e.g., located in specific points (i.e. nodes) of the physical environment
- each node hosts software agents (the coordinated components) and a tuple space, and has ability of interaction with nodes in the *neighbourhood*—where proximity can be seen either as a physical or virtual property
  1. it sleeps, remaining frozen
  2. wakes up, gathers all incoming activities (contained in messages received either from neighbour nodes or from local agents) and executes them
  3. executes the continuation of the activity executed in previous computation round
  4. spreads asynchronous messages to neighbourhood
  5. schedules an activity continuation for next round
  6. then it sleeps again, returning to step 1



GeoLinda I

- two approaches to detect movement patterns
  - virtual ⇒ localisation and communication infrastructure creating a virtual model of the physical world ⇒ people and mobile objects regularly update their location
  - physical ⇒ coordination protocols between people and objects, both carrying wireless devices with restricted communication range ⇒ movement detection based on discovery protocols, no global info
- the physical approach can be implemented by distributed tuple spaces ⇒ tuples (dis)appearance based on overlapping of devices' communication range (*physical synchronization*) ⇒ limited number of movement patterns recognisable
  - no relative locations / directions—e.g., arriving from the left
  - detection precision depending on communication range—e.g., too big ⇒ coarse-grained detection

GeoLinda II

- GeoLinda [PCBB07] associates a volume to each tuple (*tuple's shape*) and a volume to each reading operation(*addressing shape*)—e.g., sphere, cylinder, cone, box, sector, and point
- a reading operation is released when the shape of a matching tuple intersects with the addressing shape of the operation
- the programmer defines a tuple's shape *relatively* to the location and the *orientation* of the device which publishes this tuple
- similarly, he defines the addressing shape of a reading operation relatively to the location and orientation of the device which executes this operation

LIME I

- LIME (*Linda in a Mobile Environment*) [MPR06] deals with both physical and logical mobility
  - physical mobility involves the movement of mobile hosts
  - logical mobility is concerned with the movement of mobile agents—processes able to migrate from host to host while preserving code and state
- coordination takes place through *transiently shared* tuple spaces, that ties together physical and logical mobility
- movement, logical or physical, results in *implicit* changes of the tuple space accessible to the individual components
  - the system, not the application, is responsible for managing movement and the tuple space *restructuring* associated with connectivity changes
- tuple spaces are permanently bound to mobile agents and mobile hosts

LIME II

- transient sharing dynamically re-computes the set of *locally accessible* tuples in such a way that, for each mobile agent, the content of its local space gives the appearance of having been *merged* with those of the other mobile agents which are currently *co-located*

TOTA I

- *Tuples On The Air* (TOTA) [MZ09] is a middleware and programming model for supporting *adaptive context-aware activities* in pervasive and mobile computing scenarios
- the key idea in TOTA is to rely on *spatially distributed tuples*, propagated across a network on the basis of application-specific rules
- a tuple can be "injected" into the network from any node and, after cloning itself, can diffuse across the network according to tuple-specific *propagation rules*
- once a tuple is spread over the network, it can be perceived as a single distributed data structure that we call a *tuple field* – made up by all the tuples created during the propagation of the injected tuple – to draw an analogy with physical fields (e.g., gravitational), which have different values (in TOTA, tuples) at different points in space (in TOTA, network nodes)

TOTA II

- the middleware takes care of propagating the tuples and adapting their values in response to the dynamic changes that can (possibly) occur in the network topology
- TOTA assumes the presence of a peer-to-peer network of possibly mobile nodes, each running a local instance of the TOTA middleware
- each TOTA node holds references to a limited set of *neighbour nodes* and can *communicate directly* only with them
- the structure of the network, as determined by the neighbourhood relations, is automatically maintained and updated by the nodes to support dynamic changes, either due to nodes' mobility or to their birth/death
- TOTA tuples can be defined at the application level and are characterized by a *content* `C`, a *propagation rule* `P`, and a *maintenance rule* `M`, hence `T = (C, P, M)`

TOTA III

content `C` an ordered set of typed elements representing the information carried on by the tuple

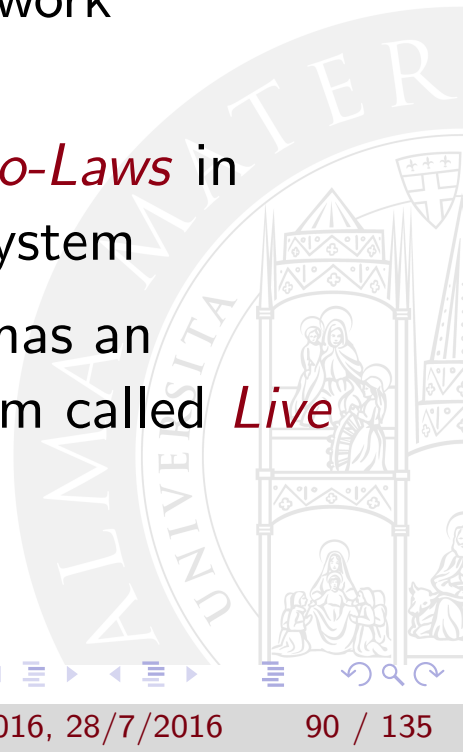
propagation rule `P` determines how the tuple should be distributed across the network; propagation typically consists in a tuple *(i)* cloning itself, *(ii)* being stored in the local tuple space, then *(iii)* moving to neighbour nodes: however, different kinds of propagation rules can determine the "scope" of the tuple – e.g., the distance at which such tuple should be propagated, the spatial direction of propagation, etc. – and how such propagation can be affected by the presence or the absence of other tuples in the system; in addition, the propagation rule can determine how the tuple's content `C` should change during propagation

TOTA IV

maintenance rule `M` determines how a tuple should react to *events* occurring in the environment—including flow of time; on the one hand, maintenance rules can preserve the proper spatial structure of tuple field despite network dynamics—thanks to TOTA middleware constantly monitoring the network local topology and the income of new tuples, eventually re-propagating tuples; on the other hand, tuples can be made *time-aware*, e.g., to support temporary tuples or tuples that slowly "evaporate"—in the spirit of pheromones [Par97]

SAPERE I

- SAPERE (Self-aware Pervasive Service Ecosystems) [ZOA+15] is a EU STREP Project founded within the EU 7FP[1]
- SAPERE considers modelling and architecting a *pervasive* service environment as a non-layered *spatial* substrate – made up of networked SAPERE nodes – laid above the actual network infrastructure
- such substrate embeds the basic "laws of nature" (*Eco-Laws* in SAPERE terminology) that rule the activities of the system
- any individual (e.g., devices, users, software services) has an associated semantic representation inside the ecosystem called *Live Semantic Annotation* (LSA)



SAPERE II

- LSAs are semantic annotations expressing information with same expressiveness as standard frameworks like RDF, whose abstract syntax is $i : [p_1 = v_1, \ldots, p_n = v_n]$ where i is the unique (ecosystem-wide) LSA identifier, p_i is the name of a property, and v_i is the associated value (any atomic or structured data)—p_i need not be different, as some property can be multi-valued
- an LSA *pattern* P should be initially understood as an LSA which may have some variable (written inside a pair of curly brackets) in place of one (or more) values – similar to a LINDA template – and an LSA L is said to match the pattern P if there exists a substitution of variables to values that applied to P gives L—differently from LINDA, the *matching* mechanism here is semantic and *fuzzy*
- eco-laws define the basic policies to rule sorts of *virtual chemical reactions* among LSAs

SAPERE III

- an eco-law is of the kind $P_1 + \ldots + P_n \xrightarrow{r} P'_1 + \ldots + P'_m$ where: *(i)* the left-hand side (reagents) specifies patterns that should match the LSAs $L_1, \ldots, L_n$ to be extracted from the space; *(ii)* the right-hand side (products) specifies patterns of LSAs which are accordingly to be inserted back in the space; and *(iii)* rate expression r is a numerical positive value indicating the *average frequency* at which the eco-law is to be fired
- as far as *topology* is concerned, the framework imposes that an eco-law applies to LSAs belonging to the same space, and constrains products to be inserted in that space or in a *neighbouring* one (to realise space-space interaction)

[1] FP7-ICT-2009.8.5: Self-awareness in Autonomic Systems

Spatial Tuples

Spatial Tuples [RVO+16] is an extension of the basic tuple-based model for distributed multi-agent system coordination, where

- tuples are conceptually *placed* in the physical world and possibly *move*
  - tuples have both a *location* and an *extension*
- the behaviour of *coordination primitives* may depend on the spatial properties of the coordinating agents
- the tuple space can be conceived as a virtual layer *augmenting physical reality*, provided that
  - physical reality is enriched by digital information situated in some physical position
  - visually-augmented reality is perceived by human users by means of specific devices, such as smart-glasses, head-mounted-display, or even smartphones

Spatial ReSpecT I

Goal of Spatial ReSpecT [MO13]

Understanding the basic mechanisms of spatial coordination is a fundamental issue for coordination models and languages in order to govern situated interaction in the *spatio-temporal fabric*

- devising out
  - the *fundamental abstractions*
  - the *basic mechanisms*
  - the *linguistic constructs*

  required to *generally* enable and promote space-aware coordination
- defining their embodiment in terms of
  - the *tuple centre* coordination medium [OD01]
  - the ReSpecT coordination language

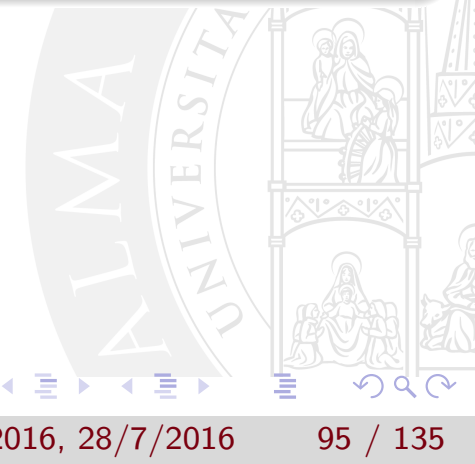
Spatial ReSpecT II

Requirements of spatial coordination

Spatial coordination requires

- spatial situatedness
- spatial awareness

of the coordination media; this translates in a number of *technical requirements*



Spatial ReSpecT III

Situatedness

A *space-aware coordination abstraction* should at any time be associated to an absolute positioning, both physical and virtual

In fact

- *software abstractions* may move along a *virtual space* – typically, the network – which is usually *discrete*
- whereas *physical devices* move through a *physical space*, which is mostly *continuous*

However, software abstractions may also be hosted by mobile physical devices, thus *share* their motion.

Spatial ReSpecT IV

Awareness

The position of the coordination medium should be available to the *coordination laws* it contains in order to make them capable of *reasoning about space*—so, to implement space-aware coordination laws

Also, space has to be embedded into the working cycle of the coordination medium

- a spatial event should be *generated* within a coordination medium, conceptually corresponding to *changes in space*
- then, such events should be *captured* by the coordination medium, and used to *activate* space-aware coordination laws

Spatial ReSpecT V

Tuple centres as general-purpose coordination abstractions

- technically, a tuple centre is a programmable tuple space, i.e., a tuple space whose behaviour in response to (coordination) events can be programmed so as to specify and enact any coordination policy [OD01, Omi07]
- tuple centres can then be thought as *general-purpose coordination abstractions*, which can be suitably forged to provide specific coordination services

Spatial ReSpecT VI

Space-aware tuple centres

- the *location* of a space-aware tuple centre is obtained through the notion of current place
  - *i.e.*, the absolute position of the computational device where the coordination medium is running, or the domain name of node hosting the tuple centre
- *motion* is conceptually represented by two sorts of spatial events:
  - leaving from a starting place
  - stopping at an arrival place

  in any sort of space / place
- analogously to (coordination) operation and time events, it is possible to specify *reactions triggered by spatial events*—spatial reactions

Spatial ReSpecT VII

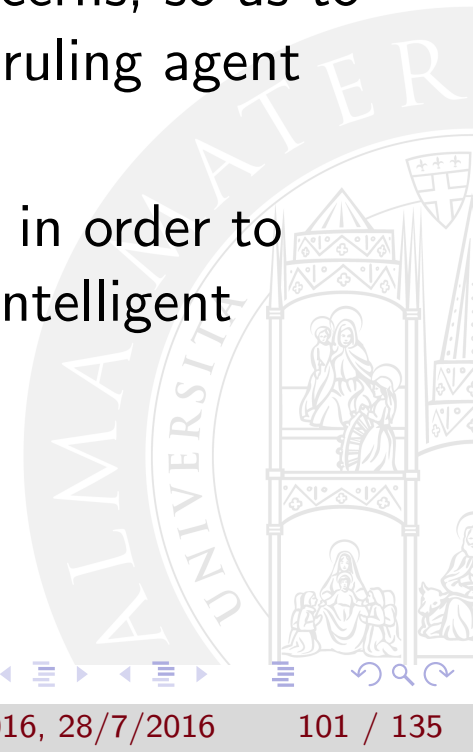
Space-aware coordination policies

As a result, *space-aware coordination policies* can be encapsulated in a spatial tuple centre, which

- can be programmed to *react to motion* in either a physical or a virtual space
- so as to enforce space-aware coordination policies

Summary

- agent societies are a basic brick for MAS modelling and engineering, including spatial aspects
- coordination models can be used to build agent societies
- coordination abstractions needs to include spatial concerns, so as to be capable of expressing *spatial coordination policies* ruling agent societies
- many coordination models nowadays deals with space in order to manage the complexity of systems such as pervasive intelligent systems



Outline

MAS: Basic Abstractions

Agents in MAS

- agents live within an agent *society*
- agents live immersed within an agent *environment*

Basic design abstractions for MAS [WOO07]

agents	are the autonomous components of the systems, embodying the *designed* activities
society	is meant to model and govern the *relationships* among agents
environment	models the either virtual or physical context where the agents are situated, capturing both the effect of agent activities and the unpredictable change brought about by *non-designed* activities

Environment in MAS I

Environment as a first-class abstraction [WOO07]

- environment should be handled as an *explicit* part of MAS
- environment in MAS is a first-class abstraction with two roles
  - providing the surrounding conditions for agents to exist—supporting agent life and activity
  - providing an exploitable design abstraction for building MAS applications

Environment in MAS II

Middleware and infrastructure

- *environment abstractions* [VHR+07]
  - provide agents with services useful for achieving individual and social goals
  - are supported by some underlying software *infrastructure* managing their creation and exploitation

→ modelling and engineering MAS environment based on agent *middleware* and infrastructure

Environment in MAS III

Environment, middleware, and space

- environment for situated MAS means first of all space-time situatedness of agents, societies, and MAS as a whole

→ environment abstractions as provided by MAS middleware should deal with space

! coordination abstractions are middleware abstractions: most of the aforementioned coordination models (should) have an implementation provided via *coordination middleware*

? how do *actually existing* MAS middleware deal with spatial notions?

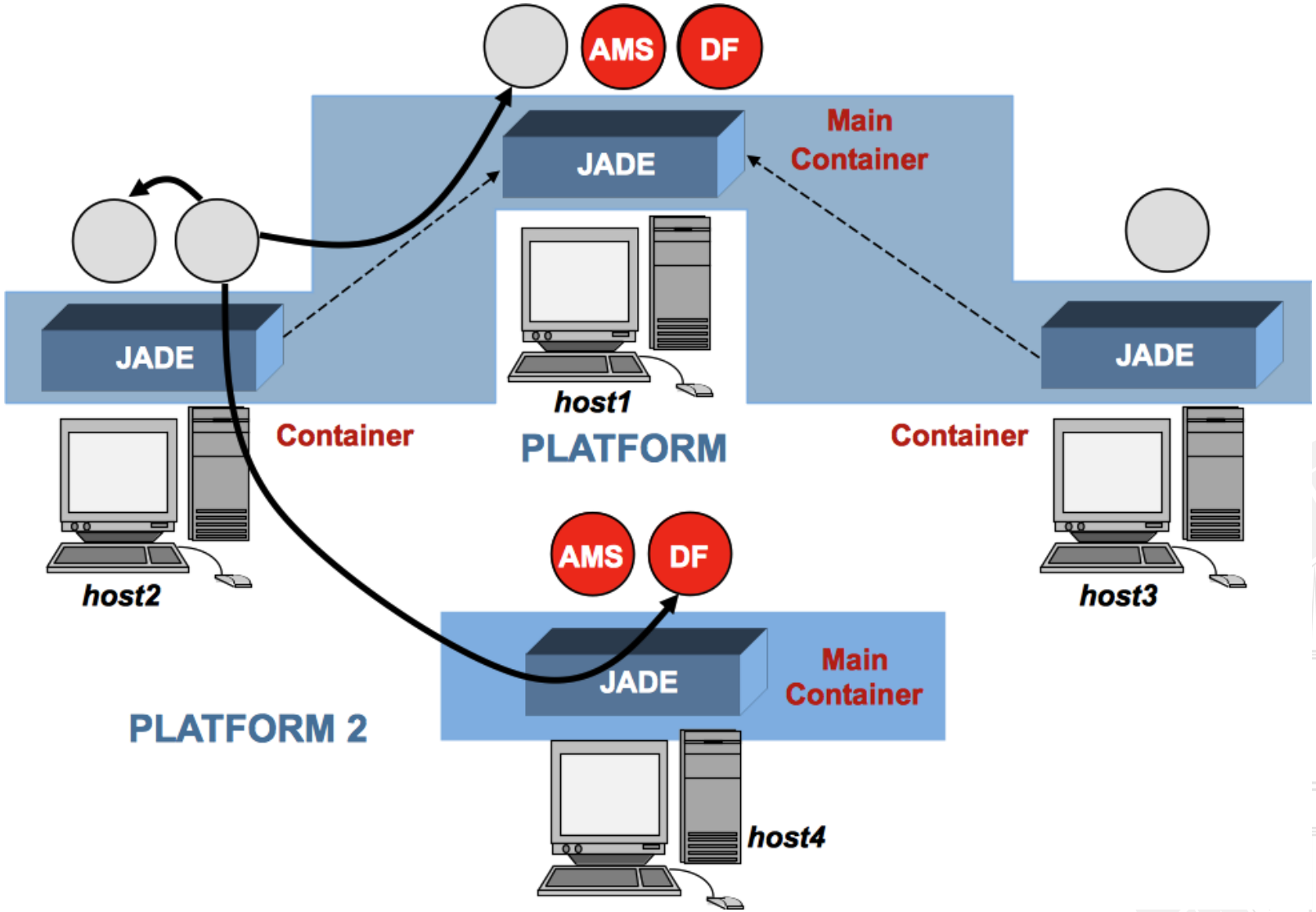
MAS Middleware and Space: Examples I

JADE [BCG07]

- JADE provides a distributed agent *platform*—where "distributed" means that a single (logical) JADE system can be split among different networked hosts
- the platform is FIPA compliant [Fou05]
- it supports intra-platform agent (strong) mobility
- it is composed by one (*main*) or more *containers*—one for each physical hosts of the platform
- each container provides a complete *runtime environment* for JADE agents execution—lifecycle management, message passing facilities, etc.

→ logical notion of locality and topology are provided via containers and platforms, with containers mapping on to physical distribution of hosts

MAS Middleware and Space: Examples II

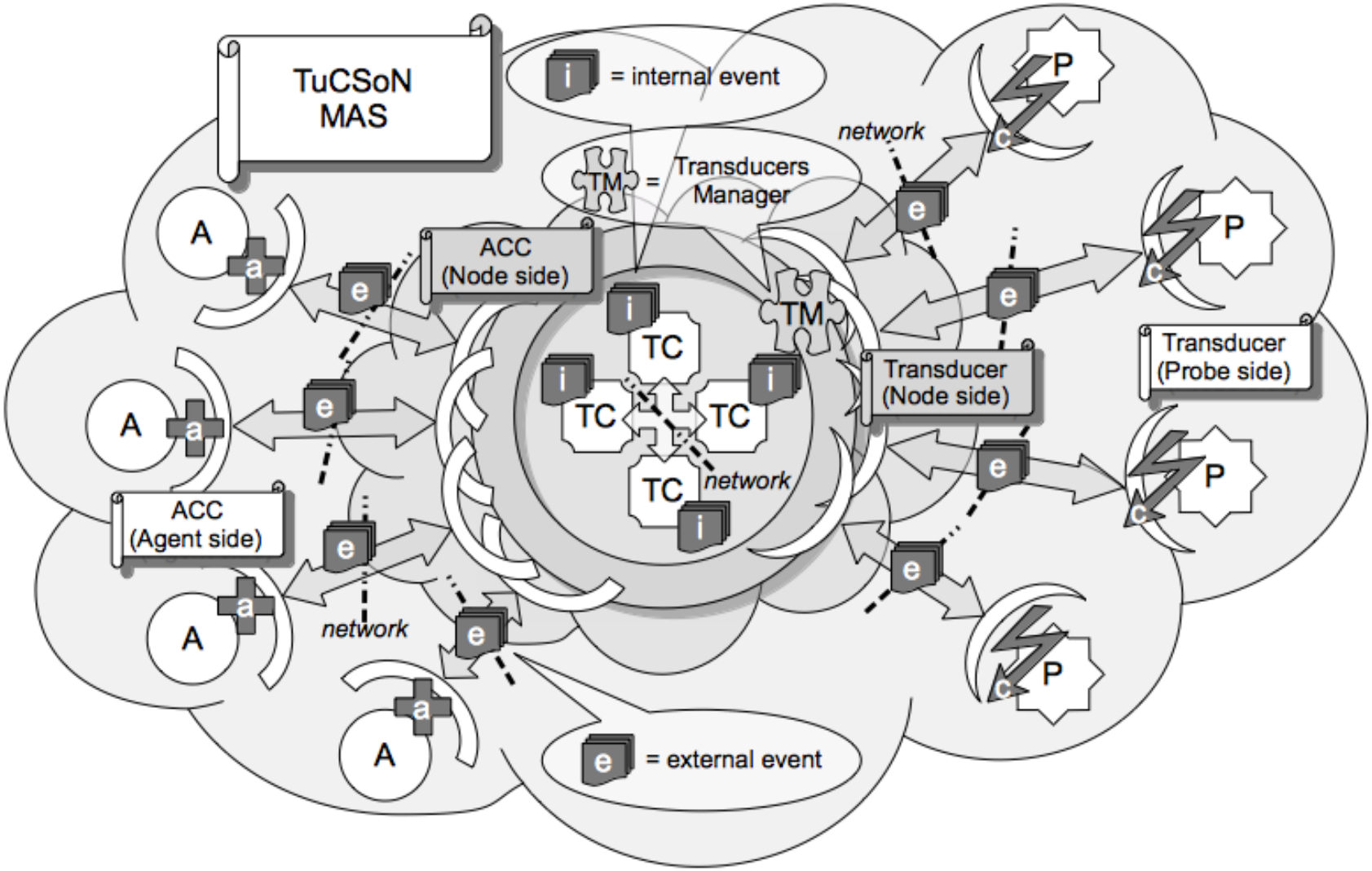


MAS Middleware and Space: Examples III

TuCSoN [OZ99]

- TuCSoN exploits *spatial tuple centres* [MO13] as middleware abstractions for spatial coordination
- tuple centres are associated to a TuCSoN *node*, the basic brick of TuCSoN *topology*
- agent invocations of coordination primitive can be either *local* – towards the node tuple centres – or *global*—towards any tuple centre in the network
- TuCSoN agents and *environment resources* interact with MAS through boundary artefacts [MO15], which are
  - the architectural components representing agents as well as environmental resources within the MAS
  - in charge of associating agents and resources with *spatial information*

MAS Middleware and Space: Examples IV



MAS Middleware and Space: Examples V

CArtAgO [OZ99]

- CArtAgO [RVO07] is a Java-based framework and infrastructure based on the A&A (agents & artefacts) meta-model [ORV08]
- A&A
  - introduces artefacts as the tools [Nar96] that agents use to enhance their own capabilities, for achieving their own goals [ORV06]
  - artefacts can be used to (computationally) represent any kind of environmental resource within a MAS in a uniform way—from sensors to actuators, from databases to legacy OO applications, from real-world objects to virtual blackboards

MAS Middleware and Space: Examples VI

CArtAgO architecture

CArtAgO main architectural components are

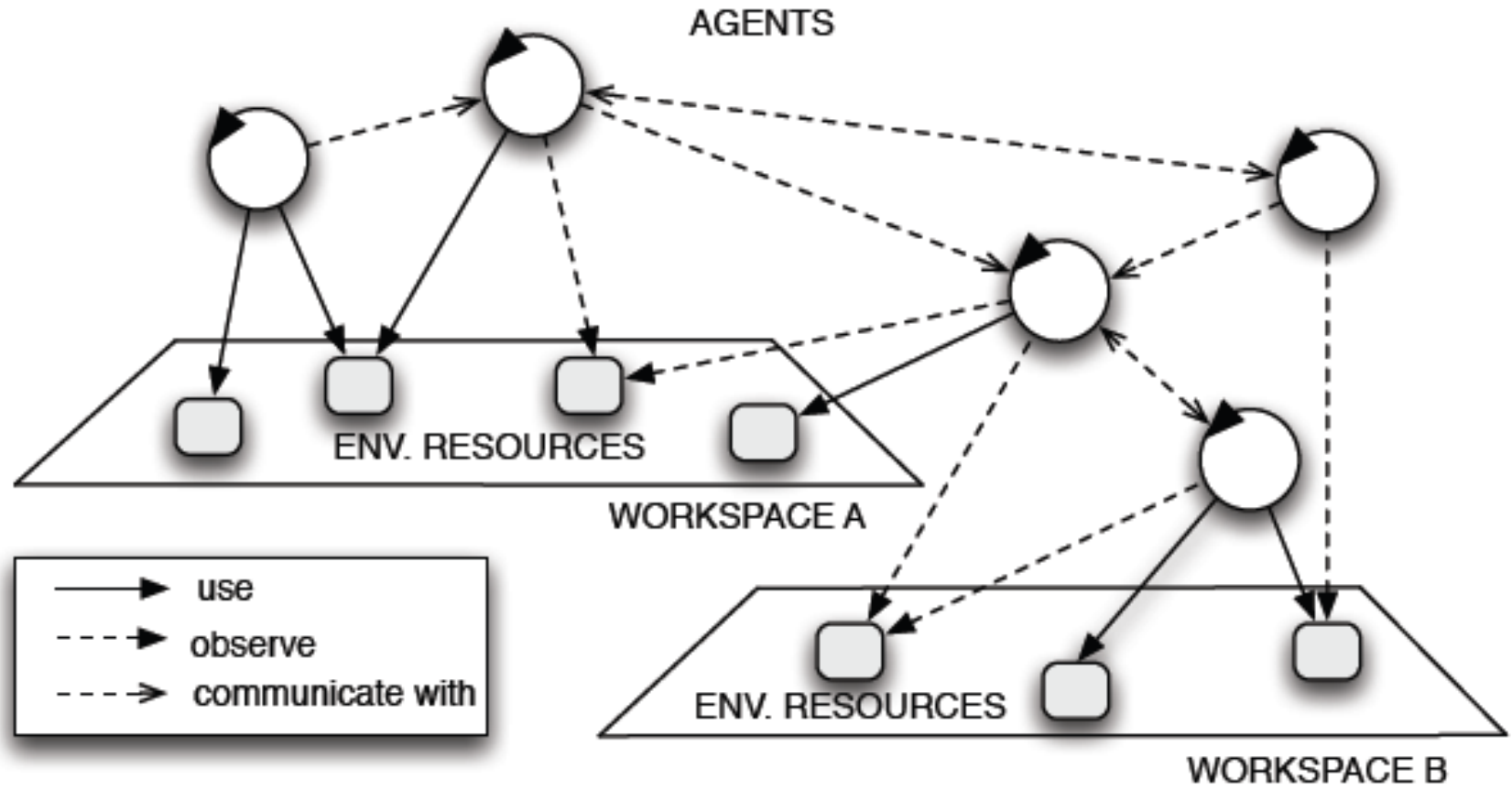
artefacts are the basic bricks in the A&A meta-model, then the basic bricks in the CArtAgO framework, too; they work as the tools for MAS designers to properly model and implement the portion of the *environment* agents can control/should deal with

workspaces play the role of the *topological* containers for agents and artefacts, representing the agent *working environments*: since every agent and every artefact are always associated with a workspace, workspaces can be used to define the scope of event generation/perception for agents and artefacts

agent bodies are the architectural components enabling agent interaction with artefacts—thus, in the very end, *situatedness*, at least from the individual agent viewpoint

observable events are defined by programmers, and are generated in response to specific operation invocations as well as observable states to monitor changes—including environment ones

MAS Middleware and Space: Examples VII



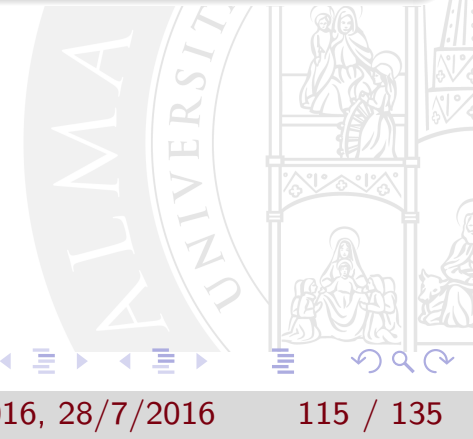
Summary

- agent environment is a basic brick for MAS modelling and engineering, including spatial aspects
- agent middleware is meant to provide agents and MAS programmers with the abstractions and tools to model and engineer complex systems with spatial features

So What?

Landscape, basics & issues for spatial MAS

- so many open issues
- so many open search directions
- for fully exploiting the potential of agents and MAS in spatial computing
- so as to develop the research field of Spatial MAS



Finally...

... we are done!

- time for questions
- and thanks for listening!

References I

Marco Aiello, Guram Bezhanishvili, Isabelle Bloch, and Valentin Goranko.
Logic for physical space: From antiquity to present days.
Synthese, 186(3):619–632, 2012.

Jacob Beal and Jonathan Bachrach.
Infrastructure for engineered emergence on sensor/actuator networks.
IEEE Intelligent Systems, 21(2):10–19, March 2006.

Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood.
Developing Multi-Agent Systems with JADE.
Wiley, February 2007.

Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg.
A survey on context-aware systems.
International Journal of Ad Hoc and Ubiquitous Computing, 2(4):263–277, 2007.

Jacob Beal, Stefan Dulman, Kyle Usbeck, Mirko Viroli, and Nikolaus Correll.
Organizing the aggregate: Languages for spatial computing.
In *Formal and Practical Aspects of Domain-Specific Languages*, pages 436–501. IGI Global, 2013.

References II

Jacob Beal.
A basis set of operators for space-time computations.
In *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW 2010)*, pages 91–97, Washington, DC, USA, 2010. IEEE Computer Society.

Brandon Bennett.
Spatial reasoning with propositional logics.
In Douglas A. Grouws, editor, *Principles of Knowledge Representation and Reasoning. Proceedings of the Fourth International Conference (KR '94*, volume 8, pages 51–62. Macmillan Publishing Co, Inc, 1992.

Philippe Balbiani, Valentin Goranko, Ruaan Kellerman, and Dimiter Vakarelov.
Logical theories for fragments of elementary geometry.
In Marco Aiello, Ian Pratt-Hartmann, and Johan Van Benthem, editors, *Handbook of Spatial Logics*, pages 343–428. Springer Netherlands, Dordrecht, 2007.

Ruth M.J Byrne and P.N Johnson-Laird.
Spatial reasoning.
Journal of Memory and Language, 28(5):564–575, October 1989.

References III

Stefania Bandini, Alessandro Mosca, and Matteo Palmonari.
Common-sense spatial reasoning for information correlation in pervasive computing.
Applied Artificial Intelligence, 21(4-5):405–425, April 2007.

Jacob Beal, Olivier Michel, and Ulrik Pagh Schultz.
Spatial computing: Distributed systems that take advantage of our geometric world.
ACM Transactions on Autonomous and Adaptive Systems, 6(2):11:1–11:3, June 2011.

Jacob Beal, Danilo Pianini, and Mirko Viroli.
Aggregate programming for the Internet of Things.
Computer, 48(9):22–30, September 2015.

Robert Bull and Krister Segerberg.
Basic modal logic.
In *Handbook of Philosophical Logic. Volume II: Extensions of Classical Logic*, volume 165 of *Synthese Library*, pages 1–88. Springer Netherlands, Dordrecht, 1984.

Douglas H. Clements and Michael T. Battista.
Geometry and spatial reasoning.
In *Handbook of research on mathematics teaching and learning: A project of the National Council of Teachers of Mathematics*, pages 420–464. Macmillan Publishing Co, Inc, New York, NY, England, 1992.

References IV


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


Daniel D. Corkill.
Blackboard systems.
AI Expert, 6(9):40–47, 1991.


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


Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf.
Computational Geometry. Algorithms and Applications.
Springer, 2000.


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

References V

Soumitra Dutta.
Approximate spatial reasoning.
In *1st International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems – IEA/AIE '88*, volume 1 of *IEA/AIE '88*, pages 126–140, New York, New York, USA, 1988. ACM Press.

Jacques Ferber and Jean-Pierre Müller.
Influences and reaction: A model of situated multiagent systems.
In Mario Tokoro, editor, *2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pages 72–79, Tokio, Japan, December 1996. AAAI Press.

Philippe Fuchs, Guillaume Moreau, and Pascal Guitton, editors.
Virtual Reality: Concepts and Technologies.
CRC Press, 2011.

Foundation for Intelligent Physical Agents.
FIPA home page, 2005.

Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna.
Understanding code mobility.
IEEE Transactions on Software Engineering, 24(5):342–361, May 1998.

References VI


Borko Furht, editor.
Handbook of Augmented Reality.
Springer New York, New York, NY, 2011.


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


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


Dedre Gentner.
Spatial cognition in apes and humans.
Trends in Cognitive Sciences, 11(5):192–194, 2007.


Heiko Hamann.
Space-Time Continuous Models of Swarm Robotic Systems. Supporting Global-to-Local Programming, volume 9 of *Cognitive Systems Monographs*.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

References VII

Daniel B.M. Haun, Josep Call, Gabriele Janzen, and Stephen C. Levinson.
Evolutionary psychology of spatial representations in the hominidae.
Current Biology, 16(17):1736–1740, September 2006.

Otto Huisman and Rolf A. de By, editors.
Principles of Geographic Information Systems. An Introductory Textbook.
ITC Educational Textbook Series. The International Institute for Geo-Information Science and Earth Observation (ITC), Enschede, The Netherlands, 3rd edition, 2009.

D. Hilbert.
Foundations of geometry.
La Salle, Illinois, USA, 1950.

Morris Kline.
Mathematical Thought from Ancient to Modern Times, volume 2.
Oxford University Press, 1972.

Christian Kray.
The benefits of multi-agent systems in spatial reasoning.
In Ingrid Russell and John Kolen, editors, *14th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2001)*, pages 552–556. AAAI Press, 2001.

References VIII


Peggy Li and Lila Gleitman.
Turning the tables: language and spatial reasoning.
Cognition, 83(3):265–294, April 2002.


Stephen C Levinson, Sotaro Kita, Daniel B.M Haun, and Björn H Rasch.
Returning the tables: language affects spatial reasoning.
Cognition, 84(2):155–188, June 2002.


Gustave Martelet.
Évolution et création, tome 1.
Editions du Cerf, Paris, 1998.


Thomas W. Malone and Kevin Crowston.
The interdisciplinary study of coordination.
ACM Computing Surveys, 26(1):87–119, March 1994.


Stefano Mariani and Andrea Omicini.
Space-aware coordination in ReSpecT.
In Matteo Baldoni, Cristina Baroglio, Federico Bergenti, and Alfredo Garro, editors, *From Objects to Agents*, volume 1099 of *CEUR Workshop Proceedings*, pages 1–7, Turin, Italy, 2–3 December 2013. Sun SITE Central Europe, RWTH Aachen University.
XIV Workshop (WOA 2013). Workshop Notes.

References IX

Stefano Mariani and Andrea Omicini.
Coordinating activities and change: An event-driven architecture for situated MAS.
Engineering Applications of Artificial Intelligence, 41:298–309, May 2015.
Special Section on Agent-oriented Methods for Engineering Complex Distributed Systems.

Amy L. Murphy, Gian Pietro Picco, and Gruia-Catalin Roman.
LIME: A coordination model and middleware supporting mobility of hosts and agents.
ACM Transactions on Software Engineering and Methodology, 15(3):279–328, July 2006.

John Charles Chenoweth McKinsey and Alfred Tarski.
The algebra of topology.
Annals of Mathematics, 45:141–191, 1944.

Reinhard Moratz and Jan Oliver Wallgrün.
Spatial reasoning about relative orientation and distance for robot exploration.
In Walter Kuhn, Michael F. Worboys, and Sabine Timpf, editors, *Spatial Information Theory. Foundations of Geographic Information Science*, volume 2825 of *Lecture Notes in Computer Science*, pages 61–74. Springer Berlin Heidelberg, 2003.

References X


Marco Mamei and Franco Zambonelli.
Programming pervasive and mobile computing applications: The TOTA approach.
ACM Transactions on Software Engineering and Methodology (TOSEM), 18(4):15:1–15:56, July 2009.


Bonnie A. Nardi, editor.
Context and Consciousness: Activity Theory and Human-Computer Interaction.
MIT Press, 1996.


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


James Odell.
Objects and agents compared.
Journal of Object Technologies, 1(1):41–53, May–June 2002.


Andrea Omicini.
On the semantics of tuple-based coordination models.
In *1999 ACM Symposium on Applied Computing (SAC'99)*, pages 175–182, New York, NY, USA, 28 February – 2 March 1999. ACM.

References XI

Andrea Omicini.
Towards a notion of agent coordination context.
In Dan C. Marinescu and Craig Lee, editors, *Process Coordination and Ubiquitous Computing*, chapter 12, pages 187–200. CRC Press, Boca Raton, FL, USA, October 2002.

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

Andrea Omicini, Alessandro Ricci, and Mirko Viroli.
Agens Faber: Toward a theory of artefacts for MAS.
Electronic Notes in Theoretical Computer Sciences, 150(3):21–36, 29 May 2006.
1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, Belgium, 22 April 2005. Proceedings.

References XII

Andrea Omicini, Alessandro Ricci, and Mirko Viroli.
Artifacts in the A&A meta-model for multi-agent systems.
Autonomous Agents and Multi-Agent Systems, 17(3):432–456, December 2008.
Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

Andrea Omicini and Franco Zambonelli.
Coordination for Internet application development.
Autonomous Agents and Multi-Agent Systems, 2(3):251–269, September 1999.
Special Issue: Coordination Mechanisms for Web Agents.

H. Van Dyke Parunak.
"Go to the ant": Engineering principles from natural agent systems.
Annals of Operation Research, 75(0):69–101, January 1997.
Special Issue on Artificial Intelligence and Management Science.

Julien Pauty, Paul Couderc, Michel Banatre, and Yolande Berbers.
Geo-Linda: a geometry aware distributed tuple space.
In *21st International Conference on Advanced Information Networking and Applications (AINA '07)*, pages 370–377. IEEE, May 2007.

References XIII

Catia Prandi, Marco Roccetti, Paola Salomoni, Valentina Nisi, and Nuno Jardim Nunes.
Fighting exclusion: a multimedia mobile app with zombies and maps as a medium for civic engagement and design.
Multimedia Tools and Applications, pages 1–29, July 2016.

Davide Rossi, Giacomo Cabri, and Enrico Denti.
Tuple-based technologies for coordination.
In Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 4, pages 83–109. Springer, January 2001.

Bertrand A. W. Russell.
Essay on the Foundation of Geometry.
Dover Publications, New York, NY, USA, 1956.

Alessandro Ricci, Mirko Viroli, and Andrea Omicini.
CArtAgO: A framework for prototyping artifact-based environments in MAS.
In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*, pages 67–86. Springer, May 2007.
3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers.

References XIV

Alessandro Ricci, Mirko Viroli, Andrea Omicini, Stefano Mariani, Angelo Croatti, and Danilo Pianini.
Spatial tuples: Augmenting physical reality with tuple spaces.
In *10th International Symposium on Intelligent Distributed Computing (IDC 2016)*, 2016.

Shashi Shekhar, Steven K. Feiner, and Walid G. Aref.
Spatial computing.
Communications of the ACM, 59(1):72–81, January 2016.

Isadore Manuel Singer and John A. Thorpe.
Lecture Notes on Elementary Topology and Geometry.
Springer, New York, 1967.

Lucy A. Suchman.
Situated actions.
In *Plans and Situated Actions: The Problem of Human-Machine Communication*, chapter 4, pages 49–67. Cambridge University Press, New York, NYU, USA, 1987.

Alfred Tarski.
What is elementary geometry?
In Leon Henkin, Patrick Suppes, and Alfred Tarski, editors, *The axiomatic method, with special reference to geometry and physics*, pages 16–30. North-Holland, Amsterdam, 1959.

References XV

Mirko Viroli, Jacob Beal, and Kyle Usbeck.
Operational semantics of Proto.
Science of Computer Programming, 78(6):633–656, June 2013.

Mirko Viroli, Tom Holvoet, Alessandro Ricci, Kurt Schelfthout, and Franco Zambonelli.
Infrastructures for the environment of multiagent systems.
Autonomous Agents and Multi-Agent Systems, 14(1):49–60, July 2007.

Mirko Viroli, Danilo Pianini, and Jacob Beal.
Linda in space-time: An adaptive coordination model for mobile ad-hoc environments.
In Marjan Sirjani, editor, *Coordination Languages and Models*, volume 7274 of *Lecture Notes in Computer Science*, pages 212–229. Springer, June 2012.
14th Conference of Coordination Models and Languages (Coordination 2012), Stockholm (Sweden), 14-15 June.

Danny Weyns, Andrea Omicini, and James Odell.
Environment as a first-class abstraction in multi-agent systems.
Autonomous Agents and Multi-Agent Systems, 14(1):5–30, February 2007.
Special Issue on Environments for Multi-agent Systems.

References XVI

Ryan K. Williams and Gaurav S. Sukhatme.
Probabilistic spatial mapping and curve tracking in distributed multi-agent systems.
In *2012 IEEE International Conference on Robotics and Automation (ICRA 2012)*, pages 1125–1130. IEEE, May 2012.

Franco Zambonelli.
Spatial computing and self-organization.
In *13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 4–8. IEEE Computer Society, June 2004.

Franco Zambonelli, Gabriella Castelli, Laura Ferrari, Marco Mamei, Alberto Rosi, Giovanna Di Marzo Serugendo, Matteo Risoldi, Akla-Esso Tchao, Simon Dobson, Graeme Stevenson, Yuan Ye, Elena Nardini, Andrea Omicini, Sara Montagna, Mirko Viroli, Alois Ferscha, Sascha Maschek, and Bernhard Wally.
Self-aware pervasive service ecosystems.
Procedia Computer Science, 7:197–199, December 2011.
Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11).

References XVII

Franco Zambonelli, Andrea Omicini, Bernhard Anzengruber, Gabriella Castelli, Francesco L. DeAngelis, Giovanna Di Marzo Serugendo, Simon Dobson, Jose Luis Fernandez-Marquez, Alois Ferscha, Marco Mamei, Stefano Mariani, Ambra Molesini, Sara Montagna, Jussi Nieminen, Danilo Pianini, Matteo Risoldi, Alberto Rosi, Graeme Stevenson, Mirko Viroli, and Juan Ye.
Developing pervasive multi-agent systems with nature-inspired coordination.
Pervasive and Mobile Computing, 17:236–252, February 2015.
Special Issue "10 years of Pervasive Computing" In Honor of Chatschik Bisdikian.

URLs

Slides

- on APICe

→ `http://apice.unibo.it/xwiki/bin/view/Talks/SpatialmasEasss2016`

- on SlideShare

→ `http://www.slideshare.net/andreaomicini/spatial-multiagent-systems`

Spatial Multi-Agent Systems

Andrea Omicini, Stefano Mariani, Mirko Viroli
{andrea.omicini, s.mariani, mirko.viroli}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna a Cesena

18th European Agent Systems Summer School (EASSS 2016)
Catania, Italy, 28 July 2016