Spatial Coordination of Pervasive Services through Chemical-inspired Tuple Spaces

Mirko Viroli, Matteo Casadei, Sara Montagna
Alma Mater Studiorum – Università di Bologna, Italy
{mirko.viroli,m.casadei,sara.montagna}@unibo.it
and
Franco Zambonelli
Università di Modena e Reggio Emilia, Italy
franco.zambonelli@unimore.it

To support and engineer the spatial coordination of distributed pervasive services, we propose a chemical-inspired model, which extends tuple spaces with the ability of evolving tuples mimicking chemical systems, i.e. in terms of reaction and diffusion rules that apply to tuples modulo semantic match. The suitability of this model is studied considering a self-adaptive display infrastructure providing nearby people with several visualisation services (advertisements, news, personal and social content). The key result of this paper is that general-purpose chemical reactions inspired by population dynamics can be used in pervasive applications to enact spatial computing patterns of competition and gradient-based interaction.

Categories and Subject Descriptors: [**Software Engineering**]:

General Terms: Pervasive Computing, Coordination Models and Languages, Spatial Computing

Additional Key Words and Phrases: Chemical-inspired computation, Tuple Spaces

1. INTRODUCTION

The increasing availability of pervasive sensing and actuating devices (RFID tags, PDAs, localisation devices) will lead to providing a new generation of general-purpose adaptive services. These will include services to coordinate and ease customers' activity (e.g., in an airport, intelligent signs showing the gate for my flight as I get nearby), pervasive location-based information services (finding a shop in the terminal which sells goods I might be interested in), or social services exploiting contextual information (custom advertisements, news or personal content, appearing on the screen installed in my seat while waiting at the gate).

Coordination infrastructures providing a networked set of shared spaces, such as those already proposed in [Mamei and Zambonelli 2009; Picco et al. 1999; Omicini and Zam-

This work has been supported by the EU FP7 project "SAPERE - Self-aware Pervasive Service Ecosystems" under contract No. 256873.

bonelli 1999], can be used to properly reify interactions between people, devices, services and data of the pervasive system. Such interactions can then be managed through proper coordination laws to be designed and deployed over such spaces, supporting properties such as situatedness, adaptivity and diversity (opennes), which are key in order to implement pervasive applications.

According to some recent visions of service systems [Barros and Dumas 2006; Ulieru and Grobbelaar 2007; Agha 2008; Zambonelli and Viroli 2008], we find it useful to see pervasive services (software functionality, pervasive devices, users' preferences, data, knowledge, and signals) as species in a "ecology", defining what we call a *pervasive service ecosystem*. Such pervasive services get injected in some location of the network and possibly evolve/diffuse around, interacting with one another in a context-dependent way, and then being possible active in a region of the whole network space. In other words, their global state can be regarded as a spatial concept – a map from nodes to the service local state, namely, a *computational field* [Beal and Bachrach 2006; Mamei and Zambonelli 2009] – while the dynamics of the whole pervasive ecosystem is naturally seen as a spatial computation.

By focussing on the characterisation of adaptive pervasive systems as "spatial computers", this paper describes a coordination model able to suitably support the requirements that fully adaptive pervasive computing calls for. To this end, we adopt the chemical-inspired tuple space model proposed in [Viroli and Casadei 2009], an extension of standard tuple spaces [Gelernter 1985] in which semantic chemical reactions are used as coordination laws to manage tuples over time (enacting service interaction/evolution) and move them in the neighbourhood (enacting service diffusion). The motivation for adopting this model is twofold: *(i)* when used as a model for pervasive services, chemistry can support the key properties of situatedness (context-awareness), adaptivity and diversity, and *(ii)* it has been proven that ecology dynamics can be suitably modelled in terms of chemical reactions [Berryman 1992; Gillespie 1977].

Via selected use cases related to a pervasive display infrastructure in an airport scenario, we show that our model is able to suitably provide visualisation services to interested users by enacting spatial coordination patterns of competition and gradient-based interaction.

The remainder of the paper is organised as follows. In Section 2, we detail the motivations for a chemical-inspired model of spatial coordination of pervasive services. Section 3 describes the coordination model. Section 4, 5, and 6 incrementally introduce the chemical reactions for defining local/spatial service competition and interaction based on computational gradients. Section 7 discusses related work and Section 8 concludes providing final remarks. On-line appendix A provides a description of the formalisation of the model and discusses some main issues concerning simulation and implementation.

2. BACKGROUND AND MOTIVATION

2.1 Requirements for Adaptive Pervasive Service Systems

In order to better explain the motivation behind the model presented in this paper, we rely on a case study, which we believe well represents a large class of pervasive computing applications in the near future.

We consider a pervasive display infrastructure, used to fill our environments with digital displays, from those in our wearable devices and domestic hardware, to wide wall-mounted screens that already pervade urban and working environments [Ferscha et al. 2006]. How-

ever, instead of considering displays as static information servers as usual today (i.e. showing information in a manually configured manner), we envision a truly general, open, and adaptable information service infrastructure. As a reference domain, we consider an airport terminal filled with wide screens mounted in the terminal area, i.e in the shops, in the corridors and in the gates, down to tiny screens installed in each seat of the gate areas or directly on passengers' PDAs.c

We first notice that information should be generally displayed based on the current state of the surrounding physical and social environment. For instance, by exploiting information coming from surrounding temperature sensors and passenger profiles/preferences, an advertiser could decide to have ice tea commercials – instead of liquor ones – displayed in a warm day and in a location populated by teenagers. Thus, a general requirement for pervasive services is *(i) Situatedness*. Namely, pervasive services deal with spatially and socially situated users' activities, hence, they should be able to interact with the surrounding physical and social world and accordingly adapt their behaviour. As a consequence, the infrastructure itself should act based on spatial concepts and data.

Secondly, and complementary to the above, the display infrastructure, and the services within it, should be able to automatically adapt to changes (or contingencies) in an automatic way. For instance, when a great deal of new information to be possibly displayed emerges, the displayed information should overall spontaneously re-distribute and re-shape across the set of existing local displays, possibly discharging obsolete visualisation services. Accordingly, another requirement is *(ii) Adaptivity*. Namely, pervasive services and infrastructures should inherently exhibit self-adaptation and self-management properties, so as to survive contingencies without any human intervention and at limited management costs.

Finally, the display infrastructure should be not only intrinsically open to any kind of visualisation services that may be added to the system, but also able to allow users – other than display owners – to upload information to displays so as to enrich the information offer or adapt it to their own needs. For instance, a passenger could watch private content uploaded from her/his PDA to a wider screen close to her/his seat, and may be willing also to share it with people nearby. Put it simply, users should act as "prosumers"—i.e. as both consumers and producers of devices, data, and services. Another general requirement is hence *(iii) Diversity*. Namely, the infrastructure should tolerate open models of service production and usage without limiting the number and classes of services potentially provided, but rather taking advantage of the injection of new services by exploiting them to improve and integrate existing services whenever possible—also properly tackling the security concerns that this obviously raises.

2.2 Toward an Ecological Model

Standard solutions to implement services in distributed settings, e.g. service-oriented architectures (SOAs), would be inadequate to support very open scenarios like those that we envision [De Cecco 2009]: we argue that pushing them towards the extreme consequences of supporting situation, adaptivity and diversity, inevitably leads to the service ecosystem vision supported in this paper.

SOAs consider a service as a "locus" of functionality interacting with others through a variety of middleware services, handling discovery, context, orchestration, and shared data-space components to support data-mediated interactions. They neglect *situatedness and spatiality*, or do not regard them as primary abstractions: a sophisticated context ser-

vice has to embed spatial descriptions of each component into its discovery table, causing frequent and costly updates of the table as a result of the inherent mobility of many components. *Adaptivity* can be supported only via expensive pointwise solutions: either all components need to recognise relevant changes in the environment and properly plan corrective actions as a response, or some orchestration server should embed the entire adaptation logic (e.g. as an autonomic manager [Kephart and Chess 2003]), but this would have to be too complex for managing the entire distributed application. Finally, *diversity* cannot be easily accommodated since service deployment, relocation, and disposal – which will become quite as frequent as any other service interaction – are not light and automatic processes, so they cannot be tolerated in the long term without high re-engineering costs.

Complexity and costs can be reduced by a more distributed solution, in which any system locality is managed by its own middleware server, coordinating local devices, users and services, thus simplifying discovery and naturally enforcing spatial interactions. However, by pushing this solution to its extreme consequences – the system is a large, mobile, and context-dependent networked set of very small localities (nodes) – the distinction among the various classes of middleware services tends to disappear: the dynamics of each node would resemble a sort of light shared data-space model (basically just formed by a small local lookup table of service and device descriptions), where components more directly and easily interact with and know of one another. The overall architecture is then perceivable as a (possibly very dense) spatial environment, upon which a number of very diverse, spatially situated components will discover, interact, and get orchestrated with one another via a limited set of rules (subsuming the role of discovery, context, data-space, and orchestration services) embedded in the spatial substrate itself.

We argue that we would end up with something that notably resembles the intrinsic "architecture" of natural ecological systems: a set of spatially situated entities (resembling species) interacting according to a well-defined set of "natural" laws enforced by the spatial environment in which they are situated (e.g. trophic interactions in ecological niches [Agha 2008]). Such a natural metaphor can be pushed further than the simple behavioural similarity, to act as the ground upon which identifying a coordination model supporting the requirements of situatedness, adaptivity, and accommodation of diversity—which are inherent properties of natural ecosystems.

2.3 Chemical-inspired Tuple Spaces for Pervasive Ecosystems

To implement a distributed shared space for supporting pervasive service ecosystems, we start from the tuple space approach—which has already been used in the context of pervasive computing (see an overview in Section 7). Tuple spaces [Gelernter 1985] provide all the "agents" of the system (software agents, users, devices, software developers) with shared spaces where local interactions occur and are reified as tuples (data records), and which provide primitives used to insert a tuple, and read/remove a tuple matching a template—written as a tuple with wildcards in place of some of its arguments. Among the various tuple space models proposed in literature, we adopt the chemical tuple space model [Viroli and Casadei 2009], in which tuples – containing semantic information about the individuals to be coordinated – evolve in a stochastic and spatial way through coordination laws resembling chemical reactions. We observe that this model can properly tackle the requirements sought for adaptive pervasive services:

Situatedness. The current situation in a system locality is represented by the tuples ex-

isting in the tuple space. Some of them can act as catalysts for specific chemical reactions, thus making system evolution intrinsically context-dependent. Moreover, mechanisms resembling chemical diffusion (e.g. chemotaxis [Eyiyurekli et al. 2010]) can be designed to identify localities larger than the single node, facilitating retrieval of services and data in mobile environments as in the computational-gradient approach [Beal and Bachrach 2006; Mamei and Zambonelli 2009].

Adaptivity. It is known from biology that some complex chemical systems are auto-catalytic (i.e. they produce their own catalyst), providing positive-negative feedbacks that induce self-organisation [Camazine et al. 2001] and lead to the spontaneous toleration of environment perturbations. Such systems can be modelled by simple idealised chemical reactions, e.g. prey-predator systems, Brussellator, and Oregonator [Gillespie 1977]. Regarded as a set of coordination laws for pervasive services, this kind of chemical reactions has the potential to intrinsically support adaptivity.

Diversity. Chemical reactions follow a simple pattern: they have some reactants (typically 1 or 2) which combine, resulting in a set of products, through a chemical propensity (or rate) dictating the likelihood for this combination to actually happen. In nature, this generates a plethora of specific chemical reactions that take into account the diversity of chemical species, and the possibility of creating complex molecular structures. In our framework, general reactions can be designed that can be instantiated to the specific and unforeseen services that will be injected in the system over time—using semantic matching to fill the gap.

Designing the proper set of chemical reactions regulating system behaviour is hence crucial. Without excluding the appropriateness of other solutions, in this paper we mostly rely on chemical reactions resembling laws of population dynamics as, e.g., the prey-predator system [Gillespie 1977; Berryman 1992]. Not only has this kind of idealised chemical reactions been successfully used to model auto-catalytic systems manifesting self-organisation properties, but it can also nicely fit the ecological metaphor we want to adopt, namely seeing pervasive services as situated species in a service ecosystem—also making interesting patterns like extinction and competition, or concepts like diseases and food, appearing by emergence.

3. THE SPATIAL COORDINATION MODEL OF CHEMICAL TUPLE SPACES

The coordination model we propose in this paper is based on the idea of reifying the distributed state of a *pervasive service* as a tuple diffused in the network of tuple spaces—a sort of field [Mamei and Zambonelli 2009; Beal and Bachrach 2006] mapping each tuple space (i.e. each network node) to a tuple representing the state of the service in that node. The effect of diffusing a pervasive service is however context-dependent: a service may happen to effectively work only in one or more regions of the whole network space—analogously to the "niches" where individuals of an ecology live and prosper. Namely, the state and behaviour of each service will be understood as a spatial concept, while the dynamics of the whole system of pervasive services is naturally seen in terms of spatial computation. In particular, the notion of spatiality we rely upon is that induced by network (wireless/wired) connections, as reported in other works such as [Mamei and Zambonelli 2009]. Namely, our "space" is the graph connecting each network node (also called location, hosting a tuple space) to those it can directly interact with—a more involved space

notion addressing the position of nodes in a metric space is not considered here, but may be an interesting subject of future work.

The proposed model enhances standard tuple spaces by equipping tuples with a concentration value evolving over time by some chemical-like coordination laws embedded in each tuple space at design-time, which apply to tuples modulo semantic match, and which also account for transferring tuples in neighbouring tuple spaces—as detailed in the following. By making tuples reify the occurrence of people, devices, services, as well as their interactions, this model will be shown to enact self-* properties in pervasive applications.

Tuple Concentration. A key idea underlying the proposed model is to attach an integer value called "concentration" to each tuple, measuring the pertinence/activity value of the tuple in the given tuple space: the higher such a concentration, the more likely and frequently the tuple will be retrieved and selected by the coordination laws to influence system behaviour. Tuple concentration is dynamic, as is typically the pertinence of system activities.

Chemical-like Reactions. Tuple concentration "spontaneously" evolves similarly to what happens in chemical behaviour, namely, a tuple with concentration N is handled pretty much in the same way as if it were a chemical substance made of N molecules of the same species. This is achieved by coordination rules in the form of chemical reactions—the only difference with respect to standard chemical reactions is that they now specify tuple templates instead of molecules. For example, a reaction "$X + Y \xrightarrow{0.1} X + X$" would mean that two tuples x and y matching X and Y are to be selected, get combined, and as a result the concentration of tuple y decreases by one, and that of x increases by one. A reaction is active if reagents occur in the space, in which case it will be selected with a certain "rate" (i.e. frequency): this is computed precisely as the rate of the corresponding reaction in chemistry [Gillespie 1977]. Namely, the transition of the above reaction is seen as a Poisson event with rate $0.1 \times \#x \times \#y$ ($\#x$ is the concentration of x)—see Appendix A for a complete account of this mechanism. In particular, we note that the average interval Δ_r between the selections of any two reactions is computed as $1/R$ where R is the sum of rates of all available reactions [Gillespie 1977]. This model gives rise to a tuple space running as a sort of chemical simulator, picking reactions probabilistically: external agents interact with each other in a mediated way through tuples, hence system coordination will follow the dynamics of the corresponding natural/artificial chemical system described by those reactions.

Semantic Matching. It is easy to observe that standard syntactic matching for tuple spaces can hardly deal with the openness requirement of pervasive services, in the same way as syntactic match making has been criticised for Web services [Paolucci et al. 2002]. This is because we want to express general reactions that apply to specific tuples independently of their syntactic structure, which cannot be clearly foreseen at design time. Accordingly, *semantic matching* can be considered as a proper matching criterion for our coordination infrastructure [Paolucci et al. 2002; Bandara et al. 2008; Bobillo and Straccia 2008; Tolksdorf et al. 2008; Giunchiglia et al. 2007].

It should be noted that matching details are orthogonal to our model, since the application at hand may require a specific implementation of them, ranging from expressive though costly approaches like [Bobillo and Straccia 2008], to more lightweight ones like [Bandara et al. 2008]. We only assume that matching (which can even change over time,

e.g. rely on an external domain ontology description) is fuzzy [Bobillo and Straccia 2008], i.e. matching a tuple with a template returns a *vagueness* value between 0 and 1. Vagueness affects the actual application rate of chemical reactions: given the above chemical reaction, and assuming tuple x matches X with vagueness 0.4, and tuple y matches Y with vagueness 0.5, we would then obtain a transition rate equal to $r \times 0.4 \times 0.5 \times \#x \times \#y$—only 20% of the expected value. Namely, the role of semantic matching in our framework is to allow for coding general chemical laws that can uniformly apply to specific cases, where it is possible for transition rates to be properly decreased—though in some cases we may still want to rely on semantic templates with crisp matching.

Tuple Transfer. We add to the model in [Viroli and Casadei 2009] a mechanism by which a (unit of concentration of a) tuple can be allowed to move towards the tuple space of a neighbouring node, thus generating a *computational field* [Mamei and Zambonelli 2009]—a map from network nodes to tuples. This is again inspired from chemistry: in fact, such a mechanism mimics chemical diffusion through membranes in biochemistry.

—Chemical diffusion involves signal molecules, produced by some intra-cellular reaction and then secreted on the surface of the cell. Accordingly, we introduce the notion of "firing" tuple denoted $t^{\rightsquigarrow}$, which is tuple t scheduled for being sent to a neighbouring tuple space—we shall denote by Δ_m the average interval between production of two firing tuples.

—In the same way chemical transfer can occur only between cells in a proximity, we introduce the concept of (unidirectional) *link* between spaces (reflecting the possibility of communicating): each link has a rate r that measures the maximum transfer of tuples per time unit.

—Chemical transfer can be affected by the concentration of some substance in the source and target cells ([Alberts et al. 2002]). Accordingly, each firing tuple defines a *local gradient tuple* G, and a *local gradient direction* δ in $\{0, +, -\}$: when δ is 0, transfer rate is not influenced by G (it remains fixed to r), when δ is "+", a molecule can only ascend the gradient created by G, and when δ is "−", the molecule can only descend it.

—Due to thermodynamic noise, there is still a probability for a substance to move independently of the gradient. This is modelled through a fixed *noise* value (10^{-2} in this paper[1]), which represents the probability for a firing tuple to move in the direction opposite to that specified by δ—a mechanism introducing a form of simulated annealing.

By the resulting model, the designer can program the self-organising coordination behaviour of the distributed systems in terms of general-purpose chemical-like reactions, to be properly installed in each tuple space of the system—as exemplified in the next sections.

4. LOCAL COMPETITION

We now discuss some examples of chemical reactions enacting general coordination patterns of interest for pervasive service ecosystems. We proceed incrementally: in this section we discuss basic laws for service matching and competition, which will be extended in subsequent sections towards a distributed setting (Section 5), and with chemical-inspired gradients to support other patterns for distributed interaction (Section 6). Each section first introduces chemical reactions, discussing them in general terms, and then a case

[1] A wide range of values lead to analogous results as far as the experiments of this paper are concerned.

study based on the pervasive display infrastructure for the airport scenario is discussed—discussing practical examples of advertisement, news and custom information services.

4.1 Chemical Reactions

We initially consider a simple yet interesting scenario in which a single tuple space mediates the interactions between pervasive services (providing some software functionality) and their users (clients) in an open and dynamic system. In the pervasive display infrastructure, this example is meant to model the basic case where, given the node where a display is installed, visualisation services are to be selected based on the profile of users nearby the display.

We aim at enacting the following behaviour: *(i)* services that do not attract users fade until eventually disappearing from the system, *(ii)* successful services attract new users more and more, and accordingly, *(iii)* overlapping services compete one another for survival, so that some/most of them eventually come to extinction.

An example protocol for service providers can be as follows. A tuple `service` is first inserted in the space to model publication, specifying service identifier and (semantic) description of the service content. Dually, a client inserts a specific request as a tuple `request`—insertion is the (possibly implicit) act of publishing user preferences. The tuple space is charged with the role of matching a request with a reply, creating a tuple `toserve(service,request)`, combining a request and a reply semantically matching. Such tuples are read by the service provider, which collects information about the request, serves it, and eventually produces a result emitted in the space with a tuple `reply`, which will be retrieved by the client.

The rules we use to enact the described behaviour are as follows:

$$\texttt{(USE)} \qquad \texttt{SERV} \mid \texttt{REQ} \xrightarrow{u} \texttt{SERV} \mid \texttt{SERV} \mid \texttt{toserve(SERV,REQ)}$$

$$\texttt{(DECAY)} \qquad \texttt{SERV} \xrightarrow{d} 0$$

Rule (USE) has a twofold role: *(i)* it first semantically matches a service and a request, accordingly creates a `toserve` tuple, and removes the request; and *(ii)* it increases the concentration of the service, so as to provide a positive feedback—resembling the prey-predator system described by Lotka-Volterra equations [Berryman 1992; Gillespie 1977]. We refer to *use rate* of a couple service/request as u multiplied by the match degree as described in the previous section: as a result, it can be noted that the higher the match degree, the more likely a service and a request are combined[2]. On the other hand, rule (DECAY) makes any concentration item of the service tuple disappear at rate d, contrasting the positive feedback of (USE): here, the overall *decay rate* of a service is d multiplied by the match degree—with no match, we would have no decay at all.

Prey-predator dynamics. In Figure 1 (left) we consider a scenario in which requests `r` are injected at average rate 50, and a matching service `s` exists in the system with initial concentration $1{,}000$: we additionally have decay rate 0.01 and use rate 0.05[3]. We can observe that after an initial growth, the number of requests which are not served stabilises

[2]A slightly more involved protocol could make use rate also take into account the user feedback after exploiting the service, reified as a tuple and responsible of increasing service concentration. Hence, in general, we consider the use rate as a measure of the service appropriateness for the specific request.

[3]For this case, and for the others appearing in this paper, we simply simulated the Continuous-Time Markov Chains system described by the chemical reactions at hand, using the approach detailed in Appendix A.

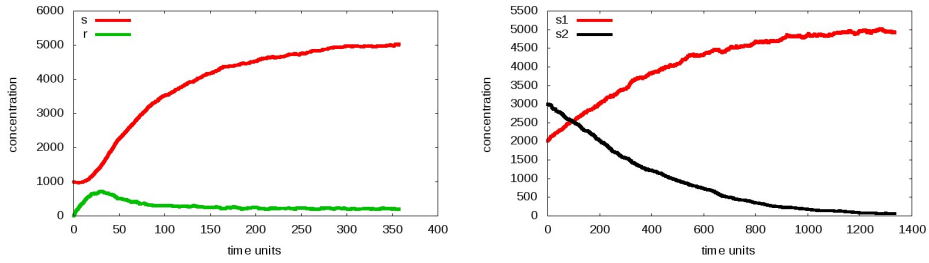

Fig. 1. Service s exploited by matching requests r (left); and competition between services $s1$ and $s2$ (right).

to few hundreds, while the concentration of $\mathtt{s}$ grows to about $5{,}000$. The behaviour of service concentration can be understood in terms of the positive-negative feedback loop of rules (USE) and (DECAY). In general, the positive feedback is caused by the injection of requests, followed by the execution of (USE). For the sake of the discussion, we consider the delay due to use rate negligible (in the scenarios tackled in this paper, the time to find matches is much faster than the average time between the arrival of two requests). Under this hypothesis, service concentration increases at the rate of injection of requests (pumping rate p) and decreases at the service decay rate (decay rate d). We can approximate the resulting stochastic behaviour by the continuous Ordinary Differential Equations (ODEs) interpretation of such chemical reactions—describing service concentration s over time by the *law of mass action* [Cardelli 2008]. Along with its analytical solution, it is as follows:

$$\frac{ds}{dt} = p - ds \qquad \Rightarrow \qquad s(t) = \frac{p}{d}(1 - e^{-dt})$$

Namely, the service concentration increases exponentially and stabilises to about p/d, which is actually $5{,}000$ in our case as shown in Figure 1 (left). Most notably, once the service decay rate is fixed, service concentration is proportional to the rate at which requests are served, without risk of divergence. In a tuple space with these reactions, the overall rate R has two contributions: the pumping rate (independent of service concentration), and the decay rate multiplied by service concentration, namely, $R = p + d \times s$. When stability is reached ($s = p/d$), this becomes $R = 2p$, and hence the minimum average transition time Δ_r is $1/(2p)$—this result will be used in the case study in Section 4.2.

Competition scenario. We now consider a similar scenario, now with two services $s1$ and $s2$ with initial concentration $2{,}000$ and $3{,}000$ respectively, and matching the same requests, though with different use rate: 0.04 for $s1$, and 0.06 for $s2$. This models the situation in which two different services exist to handle requests, one leading to a better match. The result is that $s1$ and $s2$ engage a competition: this is lost by $s1$ which starts fading until completely vanishing (i.e. being disposed) even though it has an initially higher concentration, as shown in Figure 1 (right). In fact, the sum of the concentrations of $s1$ and $s2$ still stabilises to $5{,}000$, but the contribution of $s1$ and $s2$ changes depending on the number of requests they can serve. Hence, matching degree is key when more services are concerned and the shape and dynamics of user requests is unknown, as it is responsible of the rate at which a service is selected each time, and ultimately, of the evolution of service concentration, i.e. of the competition/survival/extinction dynamics. Note that in this case

Parameter	Value	Description
r_req	$141/137\ min^{-1}$	preference injection rate, as passengers per flight over interval between two flights
t_stay	$50\ min$	passenger time nearby the display
r_ads	$100\ year^{-1}$	advertisement service injection rate
t_show	$30\ sec$	showing time for an advertisement service
c_ad	1,000	maximum expected concentration of an advertisement service
r_match	$1{,}000\ sec^{-1}$	match rate

Fig. 2. Airport scenario: competition of services in one display. Simulation parameters.

we still have $\Delta T_r = 1/(2p)$, independently of the number of services.

4.2 Case study of long-term competition

To better ground the discussion, and emphasise the adaptive and ecological character of our model, we consider the airport display infrastructure, and analyse the behaviour of a single display, located near a gate where passengers wait for the departure of their flight. As soon as a passenger gets nearby, her/his preferences are sensed, and become tuples representing requests for a visualisation service—such sensing might be due to either the passenger's PDA or the passenger's data which are stored in an RFID (or alike) placed on the boarding pass. Visualisation services are continuously injected in the system (in the long-term, they could be many): they are meant to tackle passengers' preferences (e.g. sport, food, tourism) and accordingly compete with one another, since the display is meant to probabilistically select the best service/passenger match.

Figure 2 shows all the parameters of the considered simulation scenario, in which 100 visualisation services are injected during a year. Parameters r_req and t_stay are inferred from Heathrow statistics in 2008 [4]. Match rate is the rate at which a single match can be performed: note this is negligible with respect to the time between arrival of two passengers' preferences (namely, about 1 minute due to r_req). For the simulation, we used decay rate $d = 1/30{,}000\ sec^{-1} = 1/(\texttt{c\_ad} \times \texttt{t\_show})$, since the final service concentration c_ad has been shown to be p/d, and the pumping rate for services p is $1/\texttt{t\_show}$ (service concentration increases by 1 each 30 seconds).

This simulation scenario is relative to a class of advertisement services (e.g., concerning food), which can match 5 different "marketing targets" (e.g. beverages, pasta, pizza, meat, and fish). Each passenger is associated to a single marketing target, while each advertisement can cover many marketing targets (e.g. a fast food advertisement somehow matches both beverages and meat). Accordingly, each time an advertisement service is created, we randomly draw its match degree (a number in between 0 and 1) with respect to the 5 different marketing targets, though we keep the sum of such degrees less than the "overlap factor" 1.5 (OF)—to avoid the unrealistic case in which some advertisement perfectly fits all marketing targets (which could happen if $OF = 5$). Note that, since in this application pumping rate p is 30 sec^{-1}, $\Delta T_r = 1/(2p) = 15\ sec$, which poses no significant performance constraints on implementation.

Figure 3 shows a simulation over a whole year. We note that: *(i)* only few services are actually active at a given time (i.e., they have non-negligible concentration), for the others get extinguished throughout system evolution, and *(ii)* some new service can overcome an

[4] http://www.caa.co.uk/.

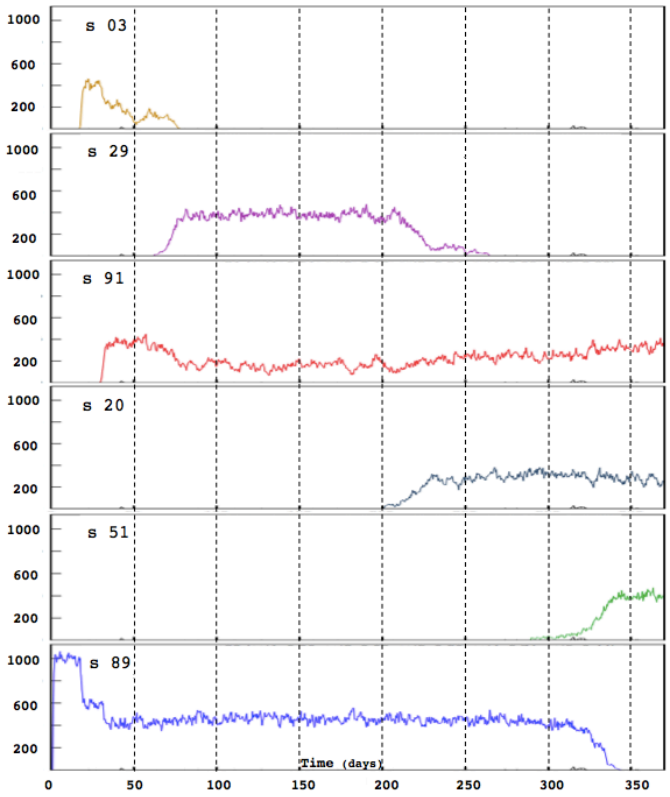

Fig. 3. Airport scenario: competition of services in one display. Charting concentration of the 6 services active throughout the simulation.

existing and established one, causing its extinction (e.g. $s51$ enters the system at day 290, and makes $s89$ extinguishing at day 350)—results of a larger number of simulations show that the average number of active services in the system is about 3.25. At the end of the year, only the following three services are active (reported with their matching degrees):

```
s91[0.52,0.11,0.84,0,0], s20[0.62,0.84,0,0,0], s51[0,0,0,0.75,0.70]
```

Namely, $s91$ is mainly tackling the fourth marketing target (and a good deal of the first), $s20$ is mainly tackling the second marketing target (and a good deal of the first as well), while $s51$ mainly tackles the fourth and fifth targets. By increasing the number of marketing targets and their overlap factor we can deal with more involved situations; for instance, analogous simulations with 20 marketing targets and $OF = 3$ give an average number of 7 visualisation services for the specific class considered. In general, it is predictable that the services that best tackle one ore more marketing targets will survive, while most of the others will end up extinguishing without unnecessarily overloading the system, and with no human intervention. This "ecological" behaviour is typical of today socially situated domains like social networks, and will be likely to play a key role in future pervasive computing systems [Agha 2008; Ulieru and Grobbelaar 2007; Zambonelli and Viroli 2008].

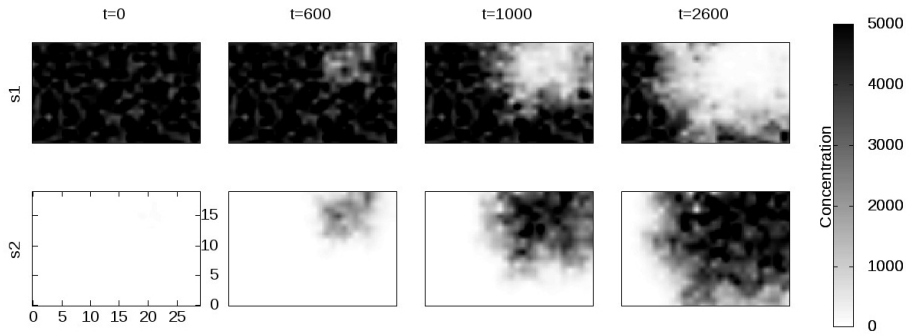

Fig. 4. Spatial competition: after an initial pointwise injection, service `s2` (down) globally overcomes `s1` (top).

5. SPATIAL COMPETITION

We here extend our discussion to a network of tuple spaces, so as to emphasise the spatial and context-dependent character of competing services.

5.1 Chemical reactions

Now suppose that instead of a single tuple space, we have a network of tuple spaces, all programmed with the above set of chemical reactions (USE,DECAY) plus a simple diffusion law for service tuples:

$$\texttt{(DIFFUSE)} \qquad \texttt{SERV} \xrightarrow{m} \texttt{SERV}^{\rightsquigarrow}$$

The resulting system can be used to coordinate a pervasive service scenario in which a service is injected into a node of the network (e.g. the node where the service is more urgently needed, or where the prosumer resides), it starts diffusing around on a step-by-step basis (following no local gradient), until possibly covering the whole network—hence becoming a global service. This situation is typical in the pervasive display infrastructure, since a frequent policy for visualisation services would be to show them on any display of the network—although more specific policies might be enacted to make certain services only locally diffuse.

In this system, we can observe the dynamics by which the injection of a new and improved service may eventually result in a complete replacement of previous versions—spatially speaking, the region where the new service is active is expected to enlarge until covering the whole system, while the old service diminishes. In the context of visualisation services, for instance, this would amount to the situation where an existing advertisement service is established, but a new one targeted to the same users is injected that happens to have greater use rate, namely, it is more appropriate for the average profile of users: we might expect this new service to overcome the old one, which accordingly extinguishes.

Reference topology. For this experiment (and also for the others developed in the remainder of this paper) we need to consider a reference network topology to use. Although this choice may evidently affect the result of simulations, and many topologies could be considered and compared, we here stick to a single case which we consider of general validity for the context of pervasive services. On the one hand, locations are placed as nodes of a square grid, such that each location has in its proximity 8 nodes (4 in the hori-

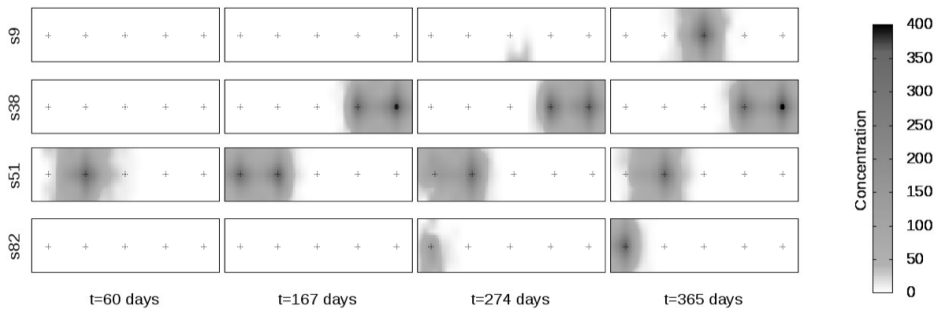

Fig. 5. Airport scenario: competition of services in the terminal. Showing spatial concentration of the 4 surviving services, in 4 snapshots.

zontal/vertical direction, and 4 in the diagonal direction – locations at the boundary of the grid have less neighbours). This choice is motivated by the fact that very often computing devices are placed more or less uniformly over the "space" formed by the buildings, corridors, or rooms of the pervasive computing systems of interest. On the other hand, we find it useful to introduce some randomness in the topology, to tackle heterogeneity of the environment at hand, failures, and so on. Hence, we actually draw the connections between such locations randomly: namely, the probability that a node is connected to one of those 8 in its proximity is 50%: in this way, the average overall number of nodes in the neighbourhood is 4, though the topology is not uniform. We call this a *random grid* topology.

Competition with diffusion. For the sake of explanation, we start from an abstract case, with a reference random grid of 30×20 nodes. In every node, requests for using a service are supposed to arrive at a fixed rate for simplicity, and a service called `s1` is the only available to match the requests (we use the following parameters: use rate $u = 0.01$, decay rate $d = 0.01$, request injection rate $p = 50$, moving rate $m = 0.01$). In particular, in every node, the system stabilises approximately to a concentration of $5{,}000$ $s1$ (p/d as usual), in spite of diffusion.

Another service `s2` is at some point developed that can serve the same requests of $s1$, now with use rate 0.1 instead of 0.05, namely, it is a service developed to more effectively serve requests—it matches requests twice as much as $s1$ does. This service is injected into a randomly chosen node of the network, with an initial very low concentration (10 in our experiment). Figure 4 shows in each column a different snapshot (from left to right), reporting concentration of `s1` on top and `s2` on bottom: we can observe that `s2` starts diffusing where it is injected, until completely overcoming service `s1` after about $3{,}000$ time units. Note that even in this case ΔT_r is $1/(2p)$, since at the equilibrium all locations stabilise to the situation discussed in the previous section. For this case, we can also compute ΔT_m: the rate at which firing tuples are produced is $m \times s$ which is $m \times p/d = 0.5$, hence the infrastructure should guarantee that $\Delta T_m = d/(m \times p)$ to avoid firing tuples to accumulate without being promptly sent—that is, 1 tuple transfer per tuple space each 2 time units.

5.2 Case-Study of Context-Dependent Spatial Competition

We now analyse a more concrete example, extending the airport scenario studied in previous section to show the spatial character of our framework. Instead of a single display we now consider an airport terminal with 5 gates in a row, and 25 displays near each gate disseminated in the corridor and gate areas. This is modelled as a 25×5 random grid (gates are at coordinates $(3,3)$, $(8,3)$, $(13,3)$, $(18,3)$, $(23,3)$).

Advertisement services are now injected from a random node of the network (taken from 8 nodes in the perimeter of the grid considered as entry point nodes), using same dynamics of previous case. Such services diffuse using (DIFFUSE) reaction and diffusion rate $0.0001\, min^{-1}$. Since link rates can be assumed to be much faster (i.e. $l \gg m$), tuples are actually transferred at the (slower) rate at which firing tuples are created, which is $m \times s = m \times p/d = 0.1\, min^{-1}$—each space has an average of one tuple transmission per 10 minutes.

In a scenario in which passenger preferences are uniformly distributed in space and time, we would expect a behaviour similar to that of Figure 4, where winning services diffuse in the whole network reaching an uniform value. But in a real-life situation preferences are not uniform but context-dependent, and this influences the actual region in which certain services can actually win competition. As an example, we consider a class of services containing news about specific locations in the world, and each passenger's profile specifies a preference for just one continent, namely, the one where she/he is flying to, automatically extracted from the RFID in the boarding pass. As in the previous case, the overlap factor is 1.5 (in that some news service might span more continents). Now assume each gate hosts flights towards a given continent: this means that each gate is a context where passengers will more likely be interested in news on the corresponding continent. This is obtained by making the injection rate of preferences dependent on the distance from the gate: the higher the distance, the smaller the rate (and still r_req in the node of the gate, as in previous section).

A simulation result is shown in Figure 5, which emphasises again the ecological character of our framework, now also taking into account spatial aspects. Only 4 services are active at the end of the simulation, which are those actually charted. At day 60, $s51$ is already established at 2^{nd} gate (from left). At day 167, $s51$ is also establishing on 1^{st} gate, while $s38$ established on gate 4^{th} and 5^{th}. At day 274, $s9$ is appearing on 3^{rd} gate and $s82$ is taking over 1^{st} gate winning competition against $s51$. At the end of simulation, both $s9$ and $s82$ completely established.

Note that in this model, `service` tuples act as a reification of the spatial service state as enacted by the coordination infrastructure: the resulting system features situatedness (success of a service in a location depends on requests and existing services there), adaptivity (the best service actually wins, and unused services fade and get garbage-collected), and accommodation of diversity (the arrival of new services is not foreseen at design time, but automatically managed).

6. INTERACTION PATTERNS BASED ON GRADIENTS

Other spatial patterns can arise as soon as we identify mechanisms by which the distributed structure of a tuple extends to a limited locality of the overall space. Based on this idea, computational gradients [Beal et al. 2008] are proposed as a key building brick for pervasive computing systems [Mamei and Zambonelli 2009]. A gradient is a field initiated (i.e.

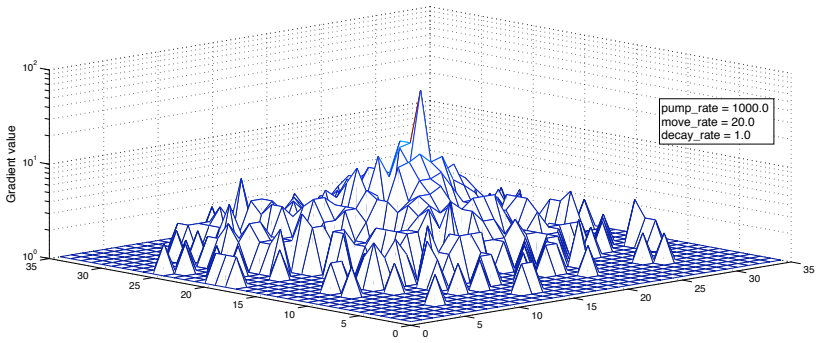

Fig. 6. The stabilised spatial structure induced by a gradient (z-axis reports concentration of field service).

pumped) from a source node, and diffusing in the surrounding until each node of the network features a stable value depending only on the estimated distance of the node from the source. This spatial data structure is primarily used to project some data from the source to a neighbouring region, so that all "agents" in it may not only be aware of such data, but also retrieve the source by simply moving on a step-by-step basis towards the source, namely, choosing at each step the neighbouring node whose gradient value indicates the nearest node to the source.

6.1 Chemical reactions

In our framework there can be many ways of creating a gradient through reactions generating firing tuples; we here focus on the following chemical system:

$$\begin{array}{ll} \texttt{(PUMP)} & \texttt{PUMP} \xrightarrow{p} \texttt{PUMP} \mid \texttt{GRAD} \\ \texttt{(DECAY)} & \texttt{GRAD} \xrightarrow{d} 0 \\ \texttt{(MOVE)} & \texttt{GRAD} \xrightarrow{m} \texttt{GRAD}^{\rightsquigarrow}(\texttt{GRAD}^{-}) \end{array}$$

Initially, assume a PUMP token with concentration 1 is inserted into the source node. Reaction (PUMP) starts spawning units of the GRAD service, such that GRAD concentration starts raising. On the other hand, (DECAY) rule makes any single GRAD unit disappear after an average $1/d$ time units—reaching an equilibrium as discussed in previous section. Additionally, (MOVE) is used to turn tuples into "firing tuples", representing GRAD units that will be moved to a neighbouring node where the concentration of GRAD is lower—note gradient GRAD and direction "−". As a result, the computational gradient starts diffusing in the neighbourhood where it is also subject to decay. The nearer the source is, the higher is the influence of the "pumping force", hence the higher is the gradient value; this value completely fades at a distance called the *gradient horizon*.

The shape of gradients. An estimation of the overall "size" of the gradient – namely the sum of gradient values in all network nodes – can be given by considering that reaction (MOVE) simply replaces units of concentration, hence it does not affect overall size, which is then (as seen in previous sections): $S = p/d$. On the other hand, the gradient value in the location where it is pumped – namely the *gradient peak* P – can be computed by

considering that (MOVE) reaction drops units of concentration with same dynamics of (DECAY), hence[5]: $P = p/(d + m)$. As a result, we can see that while p and d are responsible for the overall size of the gradient, increasing m causes tuples to more quickly escape the peak, resulting in a gradient with a larger horizon and a lower peak.

An example of simulated gradient is reported in Figure 6, showing the spatial structure of a gradient with $p = 1{,}000$, $d = 1$ and $m = 20$: it is easy to verify that peak (about 50) and overall size (about $1{,}000$) are as expected, and that the horizon is about 15 steps. Note that the slope of the field is exponential, but – especially in its boundary – very noisy due to the stochastic dynamics.

A general indication about how move rate m influences the gradient horizon and stabilisation time is given in the simulation results in Figure 7 (top). We observe that: *(i)* final gradient horizon slowly grows with move rate, *(ii)* stabilisation time is mostly independent of the actual final horizon, and *(iii)* in spite of the stochastic nature of these fields, standard deviation is reasonably small—around 5% of the average value. Note that as the final shape of a gradient is only influenced by the relative value of rates, their absolute value influences the time needed to establish the gradient—e.g. by multiplying p, d, m by ten we obtain a gradient with same shape, but stabilising 10 times faster.

Of course, it is the infrastructure that prevents the choice of arbitrary large rates. The location hosting the peak is the one in which we have the tuple space with the highest workload, and there we have the same requirement on performance as seen before: sum of transition rates for reactions (PUMP,DECAY,MOVE) is $R = p + s \times d + s \times m = 2p$, hence again $\Delta T_r = 1/(2p)$. The rate at which firing tuples are produced (due to reaction MOVE alone) is instead $s \times m = P \times m = p \times m/(d + m) \approx p$ since sufficiently large gradients always have $m \gg d$, hence we set $\Delta T_m = 1/p$ to again avoid firing tuples to accumulate without being promptly sent. Given these constraints, we would be forced to keep p rather low: e.g., $100\, sec^{-1}$, assuming we can send 100 tuples per second.

However, relying on an implementation in which firing tuples are packed, and sent together at a fixed rate (as mentioned in Section A.2) can be useful—we call such a rate *packet transfer rate* (PTR). Simulation results shown in Figure 7 (middle) clearly emphasise that by using a PTR equal to $100\, sec^{-1}$, we can support a pump rate $p = 1{,}000$ without significantly affecting accuracy of the result—note that a packet in this case is nothing but a single tuple with concentration greater than 1. Using lower PTRs (e.g., 10) would make the gradient establishing slowly. In general, simulations show that we can easily support up to pump rates equal to $5{,}000$ without losing accuracy.

A final comment concerns the impact of link rates in our model. Note that such rates are typically not a design choice, but just a measure of connection speed useful to run more accurate simulations of diffusion dynamics. In Figure 7 (bottom) we show how a gradient establishes when different link rates are used. What we observe is that when using link rates greater than 10 – a rather low rate, by which transferring a tuple introduces an average $0.1\, sec$ delay – the dynamics of a gradient is not sensibly affected by the actual link rate value. Namely, in our spatial coordination patterns and our reference pervasive scenarios what is more important is the rate of reaction (MOVE), which is the basic parameter we can act upon to design gradient horizon. Of course, link rates are still useful to simulate those situations in which some connection is e.g. particularly slow.

[5] Some units of concentration can actually get back into the peak due to noise, but this contribution is negligible

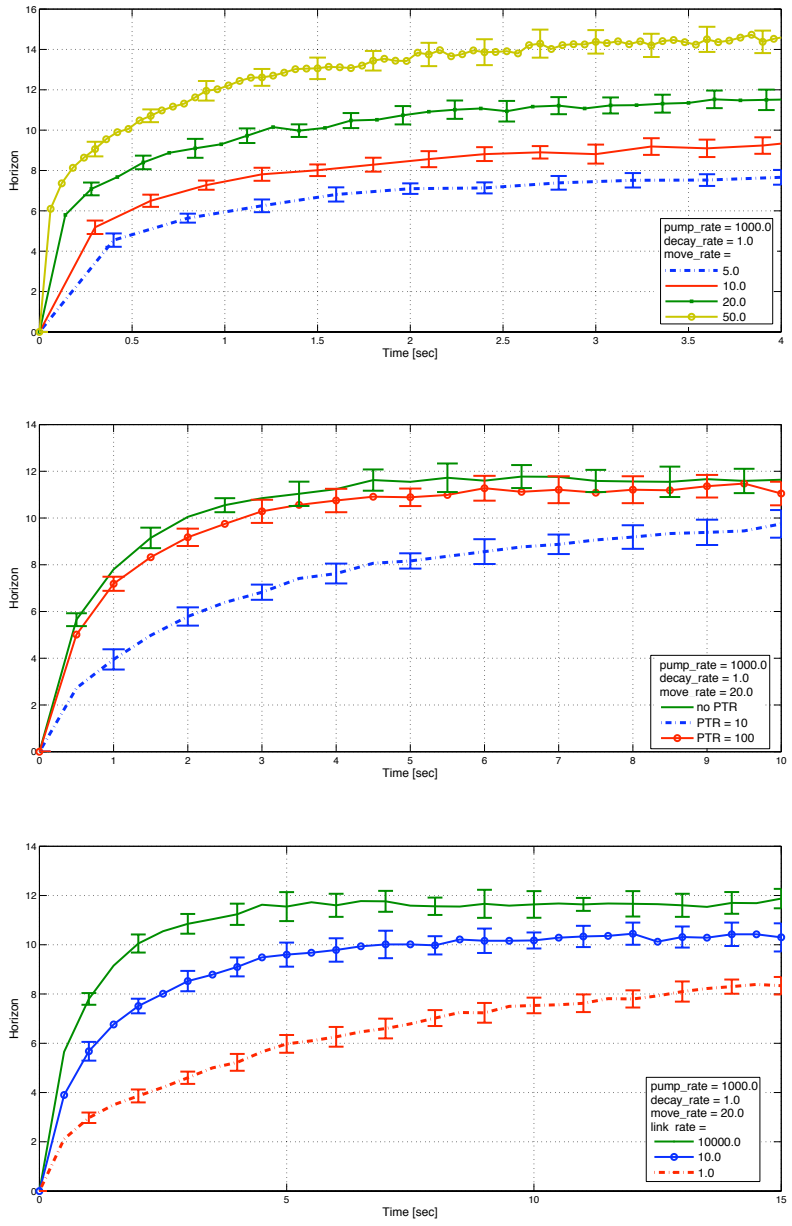

Fig. 7. (top) Gradient horizon over time, with different move rates. (middle) Gradient horizon over time, with different packet transfer rates. (bottom) Gradient horizon over time, with different link rates.

Ascending a field. We now consider a typical retrieval scenario of spatial computing (see e.g. [Mamei and Zambonelli 2009]), which is a main application of computational gradients. Suppose a request `r` located in a node pumps a gradient `g` to look for a given

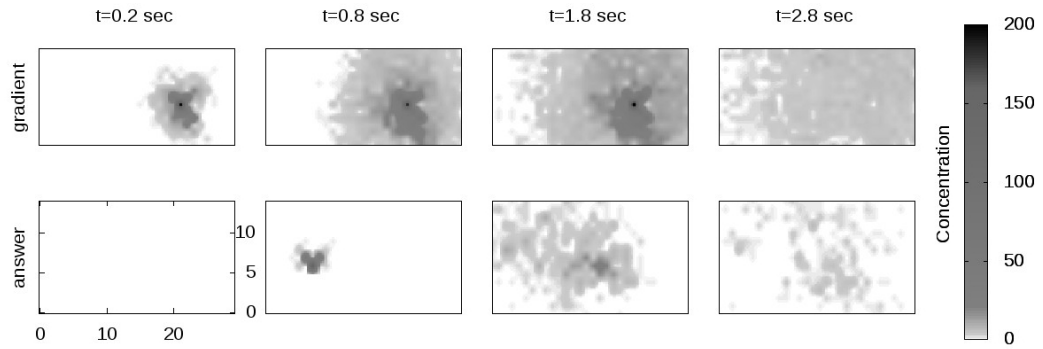

Fig. 8. Field-based attraction: a service enacts a gradient (top) used by an answer service to reach the requester (down).

service `s`: as soon as this gradient reaches a node hosting `s`, an answer `a` is pumped for a limited time which ascends the gradient until reaching `r`. At that point, `a` can simply interact with `r` and interrupt the creation of gradient `g`. This behaviour is realised by the following set of reactions, extending (PUMP,DECAY,MOVE) described in previous section:

$$
\begin{array}{ll}
\texttt{(ANSWER)} & \texttt{SERV} \,|\, \texttt{GRAD} \xrightarrow{p} \texttt{PUMP-A} \\
\texttt{(PUMP-A)} & \texttt{PUMP-A} \xrightarrow{p} \texttt{PUMP-A} \,|\, \texttt{ANSW} \\
\texttt{(DECAY-PA)} & \texttt{PUMP-A} \xrightarrow{d'} 0 \\
\texttt{(ASCENT)} & \texttt{ANSW} \xrightarrow{m} \texttt{ANSW}^{\rightsquigarrow(\texttt{GRAD}^{+})} \\
\texttt{(DECAY-A)} & \texttt{ANSW} \xrightarrow{d} 0 \\
\texttt{(STOP)} & \texttt{ANSW} \,|\, \texttt{PUMP} \xrightarrow{m} 0
\end{array}
$$

Figure 8 provides a visualisation of the corresponding dynamics, using the following rates (expressed as sec^{-1}): $p = 5{,}000$, $m = 20$, $d = 1$, $d' = 1$. At time $t = 0$ the requester located at node $(22, 7)$ starts pumping gradient g, which is already visible at time $t = 0.2\,sec$. At time $t = 0.8\,sec$, the gradient reaches the target node at $(8, 7)$, which creates a pump token pa by reaction (ANSWER). This generates and diffuses an answer service as usual, by reaction (PUMP-A): however, this effect lasts for a limited amount of time ($1\,sec$ in our case due to rate d'), then the answer gradually fades. Meanwhile, the answer diffuses around and probabilistically ascends the gradient by rule (ASCENT). At time $t = 1.8\,sec$, it reaches the requester, so as to complete the interaction. As a result, the gradient pump is dropped by rule (STOP), so that also the gradient fades away, as shown at time $t = 2.8\,sec$.

6.2 Case study of Service Retrieval

We now analyse a more concrete example based on the airport scenario. We again consider the airport terminal with 5 gates in a row, modelled as a 25×5 random grid. We suppose that in every position of the terminal, a passenger passing by a display with her/his boarding pass generates a request service in charge of retrieving updated flight information. According to the interaction pattern described above, the request would pump a gradient from the current display, which generates an answer service upon reaching the gate of interest. In

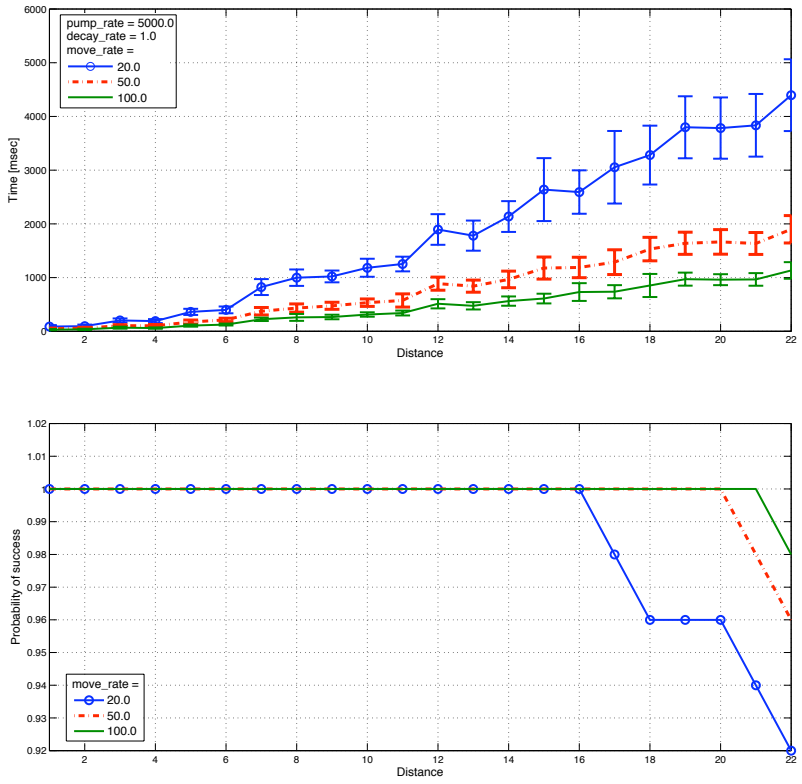

Fig. 9. Airport scenario: retrieving flight information from the gate. Retrieval time over distance for successful interactions, with different move rates (top); and corresponding probability of successful interactions (bottom).

turn, this answer service retrieves the requester by ascending the gradient, such that proper information can be displayed: e.g. if the passenger is late, an alert message with directions to the gate can be shown as the user pass by the display.

To design proper rates for chemical reactions, we can first consider that pump and decay rate can be fixed as usual ($p = 5{,}000$ and $d = 1$) since the size of the network is known; however, move rate m in reaction (ASCENT) can vary. We can expect that with relative high values of m we can more quickly obtain a reply from the gate, though this results in a higher cost for the infrastructure, since the gradient will reach much more tuple spaces of the network.

Figure 9 (top) shows simulation results charting the time to complete the interaction depending on the distance, with different move rates—average values out of 50 simulations have been reported. This figure only reports results for interactions that were successful. In fact, as shown in Figure 9 (bottom), there is a probability that no answer reaches the display—namely, the answer gradient has not yet reached the requester when its pump fades due to reaction (DECAY-PA). If this is the case, simply the passenger would experience no updated information about his/her flight on the display.

After observing that higher move rates mean higher probability of success and quicker

replies, we can design a flight information service by which the actual move rate m is dynamically selected depending on the urgency/priority of the answer. This can be automatically dealt with by semantic matching. Namely, we can impose $m = 100$, and then design semantic template ANSW in reaction (ASCENT) such that it best matches "urgent answers". For instance, we could give higher match degree to requests of passengers over 65 years, or to flights departing soon, or to flights with some important update information; also, it is possible to give no match at all for passengers who do not need any update information, and make sure that no answer at all is created and diffused. This again emphasises the importance of semantic match, and its influence to the selection of chemical reactions to be executed.

7. RELATED WORK

7.1 Coordination Models

The issue we face in this article can be framed as the problem of finding the proper coordination model [Malone and Crowston 1994] for enabling and ruling interactions of pervasive services. Coordination models generated by the archetypal LINDA model [Gelernter 1985], which simply provides for a blackboard with associative matching for mediating component interactions through insertion/retrieval of tuples. A radical change is instead the idea of engineering the coordination space of a distributed system by some policy "inside" the tuple spaces as proposed here, following the pioneer works of TuCSoN [Omicini and Zambonelli 1999] and MARS [Cabri et al. 2000]—in fact, as discussed in Section A.2, TuCSoN can be used as a low-level virtual platform for enacting the chemical tuple-space model. As already mentioned, the work presented in this article is based on [Viroli and Casadei 2009], properly extended to deal with a mechanism for tuple transfer taking into account concentration of tuples in source and target nodes—a key mechanism to exploit gradients, and hence, to support awareness. In [Viroli and Casadei 2010], this model is exploited to deal with self-composition of services, a concept that is essential in pervasive service ecosystems though not deepened here.

Chemistry has been proposed as an inspiration for several works in distributed computing and coordination over many years, like in the Gamma language [Bonâtre and Le Métayer 1996] and the chemical abstract machine [Berry and Boudol 1992]. Although these models already show the potential of structuring coordination policies in terms of chemical-like rewriting rules, we observe that they do not bring the chemical metaphor to its full realisation, as they do not exploit chemical stochastic rates—behaviour is nondeterministic rather than probabilistic as in our model.

7.2 Spatial Computing

The spatial patterns we achieve using the chemical tuple space model very much resemble (and are inspired from) previous works in the context of Spatial Computing. The series of papers [Beal and Bachrach 2006; Beal et al. 2008] discusses various algorithms for defining gradient data structures, featuring different robustness and performance properties. The main idea of this approach is to establish a computational gradient in which the field value in the source is 0, and increases by 1 at each hop—in the case of multiple paths toward a node the smallest value is retained. Variations of this basic algorithm are used to speed up the transition to a new stable state as a result of some topological changes, or to trade speed for precision.

Our chemical-inspired gradient structures are different since: *(i)* our fields decrease with the distance from the source, *(ii)* the slope is exponential and not linear, *(iii)* by acting on rates we can set up size and horizon of a field, *(iv)* the actual field value at any point is established as a result of the probabilistic (and timed) selection of chemical reactions. Similarly to [Beal et al. 2008], our approach is intrinsically robust to changes in the topology and to the position of the source, though the performance of field establishment and evolution are not comparable—however we believe they are reasonable for the context of pervasive service ecosystems. Filling the performance gap between the two approaches is in fact a main future direction of investigation.

It is worth noting that the chemical approach developed in this article is somehow related to the vision of [Beal and Bachrach 2008], which already points out the connection between bio-chemistry and spatial computation. However, while this is focused on adopting spatial computing technologies (e.g. Proto language) to model/simulate biochemical processes, we consider the other way round: we take (existing, idealised, or newly invented) chemical systems and get inspiration from them to build spatial coordination strategies for pervasive computing.

The spatial middleware that most resembles the one presented in this article is TOTA (Tuples On The Air) [Mamei and Zambonelli 2009]. In TOTA each tuple, when inserted into a node of the network, is equipped with a content (the tuple data), a diffusion rule (the policy by which the tuple has to be cloned and diffused around) and a maintenance rule (the policy whereby the tuple should evolve due to events or time elapsing). Hence, while in our approach the coordination laws (chemical reactions) are meant to be fixed for the application domain and apply to all tuples (depending on semantic matching criteria), in TOTA the tuple is responsible for carrying its behavioural rules. So, while we call for specifying the evolution rules of tuples at design-time, when the application goals are identified, TOTA instead promotes a run-time approach: diffusion behaviour is defined by an agent before injecting the tuple in the system. From this viewpoint, a key aspect of our approach is that ontology-based semantic matching can be used to make one general law apply in different ways to different, possibly unforeseen tuples. Accordingly our framework has a better ability of predicting system behaviour at design-time, and tackling diversity.

7.3 Service Environments and Pervasive Middleware

In recent years, we have witnessed a trend combining researches in the areas of service environments and pervasive middleware: there, the need for enforcing adaptivity, other than advanced forms of context-awareness, has been tackled in several ways.

Situatedness and Context-Awareness. Considering the issues of situatedness and context-awareness, extensions or modifications to the traditional SOAs have been recently proposed to address context-awareness and adaptivity in pervasive environments. The PLASTIC approach [Autili et al. 2009] adopts a philosophy somewhat similar to ours, in that service descriptions are coupled with dynamic annotations related to the current context and state of a service, to be used for enforcing contextual and adaptable forms of service discovery. Our approach pushes the idea further, since it gets rid of traditional discovery services and enforces dynamic and adaptive service interaction via simple behaviour rules (chemical reactions) and a minimal middleware, which are key to tolerate diversity.

In many proposals for pervasive computing environments and middleware infrastruc-

tures, the idea of "situatedness" has been promoted by the adoption of shared virtual spaces for services and components interactions. The pioneering system Gaia [Román et al. 2002] introduces the concept of active spaces, a middleware infrastructure enacting distributed active blackboard spaces, acting as the means for service interactions. Later on, a number of proposals have extended upon Gaia, to include additional functionalities and features for active spaces to enforce dynamic semantic pattern-matching for service composition and discovery [Fok et al. 2009] or access to contextual information [Costa et al. 2006]. Other related approaches include: Egospaces [Julien and Roman 2006], exploiting a network of tuple spaces to enable location-dependent interactions across components; LIME [Picco et al. 1999], proposing tuples spaces that temporarily merge based on network proximity, to facilitate dynamic interactions and exchange of information across mobile devices; its extension, TeenyLime [Costa et al. 2007] targeting sensor networks; and the already mentioned TOTA [Mamei and Zambonelli 2009]. Our model shares the idea of conceiving components as "living" and interacting in a shared spatial substrate (of tuple spaces) where they can automatically discover and interact with one another. Yet, our aim is broader, namely, to dynamically and systemically enforce situatedness, service interaction, and data management with a simple language of chemical reactions, and most importantly, enacting an ecological behaviour.

Self-organisation. Several recent works exploit the lessons of adaptive self-organising natural and social systems to enforce self-awareness, self-adaptivity, and self-management features in distributed and pervasive computing systems. At the level of individual component modelling, these proposals take the form of either situated reactive agents [Parunak et al. 2002] or proactive and goal-oriented ones [Ricci et al. 2007]. At the level of interaction models, these proposals typically take the of form of specific nature- and socially inspired interaction mechanisms [Babaoglu et al. 2006] (e.g., pheromones [Parunak et al. 2002], virtual fields [Mamei and Zambonelli 2009], or gossiping [Jelasity et al. 2005]), enforced either at the level of component modelling or via specific middleware-level mechanisms. We believe our framework integrates and improves these works in three main directions: *(i)* it abstracts from the specific internal characteristics of components (no matter whether they are simple reactive components or complex goal-oriented ones) and rather proposes an approach that seamlessly applies to both cases; *(ii)* it tries to identify an interaction model that is able to represent and subsume the diverse nature-inspired mechanisms under a unifying self-adaptive abstraction (i.e. the semantics chemical reactions); *(iii)* the ecological approach we undertake goes beyond most of the current studies that limit to ensembles of homogeneous components, defining a suitable framework for supporting the vision of novel pervasive and Internet scenarios as made up of self-adaptive devices and services, that autonomously cooperate for the creation of global services—a sort of cyber-organisms as envisioned in [Agha 2008].

8. CONCLUSIONS AND FUTURE WORK

The proposed chemical-oriented extension of the tuple space model can be regarded as a ground for building self-organising coordination infrastructures for open and pervasive spatial-oriented service systems. As we showed in this article, the chemical metaphor – along with the semantic character of chemical laws and the possibility of modelling chemical diffusion by tuple transfer – plays a crucial role to enable coordination models to tackle the requirements of engineering adaptive pervasive services. In particular, we

showed that with very simple chemical reactions, it is possible to model reaction/diffusion behaviour that amounts to useful spatial patterns: indeed, our model promotes a view of pervasive systems as spatial computers, and of adaptive services in it as spatial structures that compete and diffuse in a context-dependent way—other patterns of segregation and clustering can be supported which are not described in the article.

As this article is aimed at describing an executable model and spatial pattern emergence via simulation, it paves the way towards several research activities. A main research direction concerns implementation, and in particular semantic matching and performance issues. Currently, we are studying an architecture for tuple spaces which can allow to plug semantic match algorithms in an easy and flexible way. Indeed, different applications may require different semantic approaches, from expressive but costly ones like [Bobillo and Straccia 2008], to more light approaches such as [Bandara et al. 2008].

Concerning performance, we see two main research directions. On the one hand, optimised implementation techniques are to be designed for the tuple space architecture: in particular, τ-leaping [Rathinam et al. 2003; Cao et al. 2006] is worth investigating, which is based on the idea of applying faster reactions many times in one single step, which under certain conditions can be shown to produce a very accurate chemical dynamics. In general, we believe that chemical reaction/diffusion mechanisms can be supported in a rather efficient way by a trade-off with accuracy, still obtaining reasonably good spatial coordination behaviours. On the other hand, different chemical reactions can be designed that may exhibit similar properties to those proposed in this article, though providing better performance.

REFERENCES

AGHA, G. 2008. Computing in pervasive cyberspace. *Commun. ACM 51,* 1, 68–70.

ALBERTS, B., JOHNSON, A., LEWIS, J., RAFF, M., ROBERTS, K., AND WALTER, P. 2002. *Molecular Biology of the Cell*, 4th ed. Garland Science Textbooks. Garland Science.

AUTILI, M., BENEDETTO, P. D., AND INVERARDI, P. 2009. Context-aware adaptive services: The plastic approach. In *Fundamental Approaches to Software Engineering, 12th International Conference, FASE 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*. Lecture Notes in Computer Science, vol. 5503. Springer, 124–139.

BAADER, F., CALVANESE, D., MCGUINNESS, D. L., NARDI, D., AND PATEL-SCHNEIDER, P. F., Eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

BABAOGLU, O., CANRIGHT, G., DEUTSCH, A., CARO, G. A. D., DUCATELLE, F., GAMBARDELLA, L. M., GANGULY, N., JELASITY, M., MONTEMANNI, R., MONTRESOR, A., AND URNES, T. 2006. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst. 1,* 1, 26–66.

BANDARA, A., PAYNE, T. R., ROURE, D. D., GIBBINS, N., AND LEWIS, T. 2008. A pragmatic approach for the semantic description and matching of pervasive resources. In *Advances in Grid and Pervasive Computing*. LNCS, vol. 5036. Springer, 434–446.

BARROS, A. P. AND DUMAS, M. 2006. The rise of web service ecosystems. *IT Professional 8,* 5, 31–37.

BEAL, J. AND BACHRACH, J. 2006. Infrastructure for engineered emergence on sensor/actuator networks. *IEEE Intelligent Systems 21,* 2, 10–19.

BEAL, J. AND BACHRACH, J. 2008. Cells are plausible targets for high-level spatial languages. In *Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008, Workshops Proceedings, October 20-24, 2008, Venice, Italy*. IEEE Computer Society, 284–291.

BEAL, J., BACHRACH, J., VICKERY, D., AND TOBENKIN, M. 2008. Fast self-healing gradients. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. ACM, New York, NY, USA, 1969–1975.

BERRY, G. AND BOUDOL, G. 1992. The chemical abstract machine. *Theoretical Computer Science 96,* 1 (apr), 217–248.

BERRYMAN, A. A. 1992. The origins and evolution of predator-prey theory. *Ecology 73,* 5 (October), 1530–1535.

BOBILLO, F. AND STRACCIA, U. 2008. fuzzydl: An expressive fuzzy description logic reasoner. In *2008 International Conference on Fuzzy Systems (FUZZ-08)*. IEEE Computer Society, 923–930.

BONÂTRE, J.-P. AND LE MÉTAYER, D. 1996. Gamma and the chemical reaction model: Ten years after. In *Coordination Programming*. Imperial College Press London, UK, 3–41.

BUSI, N., GORRIERI, R., AND ZAVATTARO, G. 2000. On the expressiveness of linda coordination primitives. *Inf. Comput. 156,* 1-2, 90–121.

CABRI, G., LEONARDI, L., AND ZAMBONELLI, F. 2000. MARS: A programmable coordination architecture for mobile agents. *IEEE Internet Computing 4,* 4, 26–35.

CAMAZINE, S., DENEUBOURG, J.-L., FRANKS, N. R., SNEYD, J., THERAULAZ, G., AND BONABEAU, E. 2001. *Self-Organization in Biological Systems*. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ, USA.

CAO, Y., PETZOLD, L., AND GILLESPIE, D. 2006. Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics 124,* 4.

CARDELLI, L. 2008. On process rate semantics. *Theoretical Computer Science 391,* 3, 190 – 215. Converging Sciences: Informatics and Biology.

COSTA, P., MOTTOLA, L., MURPHY, A. L., AND PICCO, G. P. 2007. Programming wireless sensor networks with the teenylimemiddleware. Lecture Notes in Computer Science, vol. 4834. 429–449.

COSTA, P. D., GUIZZARDI, G., ALMEIDA, J. P. A., PIRES, L. F., AND VAN SINDEREN, M. 2006. Situations in conceptual modeling of context. In *Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China, Workshops*. IEEE Computer Society, 6.

DE CECCO, M. 2009. Infrastructures for digital business ecosystems: the wrong question? Report of the European Thematic Network on Digital Ecosystems. Available at: `http://www.digital-ecosystems.org/doc/papers/DigitalEcosystemsSyndromes.pdf`.

EYIYUREKLI, M., BAI, L., LELKES, P. I., AND BREEN, D. E. 2010. Chemotaxis-based sorting of self-organizing heterotypic agents. In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, S. Y. Shin, S. Ossowski, M. Schumacher, M. Palakal, C.-C. Hung, and D. Shin, Eds. ACM, Sierre, Switzerland, 1315–1322.

FERSCHA, A., RIENER, A., HECHINGER, M., AND SCHMITZBERGER, H. 2006. Building pervasive display landscapes with stick-on interfaces. In *CHI Workshop on Information Visualization and Interaction Techniques*.

FOK, C.-L., ROMAN, G.-C., AND LU, C. 2009. Enhanced coordination in sensor networks through flexible service provisioning. In *Coordination Languages and Models*, J. Field and V. T. Vasconcelos, Eds. LNCS, vol. 5521. Springer-Verlag, 66–85. 11th International Conference (COORDINATION 2009), Lisbon, Portugal, June 2009. Proceedings.

FUJII, K. AND SUDA, T. 2006. Semantics-based dynamic web service composition. *Int. J. Cooperative Inf. Syst. 15,* 3, 293–324.

GELERNTER, D. 1985. Generative communication in linda. *ACM Trans. Program. Lang. Syst. 7,* 1, 80–112.

GIBSON, M. A. AND BRUCK, J. 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A 104,* 9 (March), 1876–1889.

GILLESPIE, D. T. 1977. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry 81,* 25, 2340–2361.

GIUNCHIGLIA, F., YATSKEVICH, M., AND SHVAIKO, P. 2007. Semantic matching: Algorithms and implementation. *J. Data Semantics 4601*, 1–38.

HORROCKS, I., PATEL-SCHNEIDER, P. F., AND HARMELEN, F. V. 2003. From shiq and rdf to owl: The making of a Web Ontology Language. *Journal of Web Semantics 1*, 2003.

JELASITY, M., MONTRESOR, A., AND BABAOGLU, Ö. 2005. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst. 23,* 3, 219–252.

JULIEN, C. AND ROMAN, G.-C. 2006. Egospaces: Facilitating rapid development of context-aware mobile applications. *IEEE Trans. Software Eng. 32,* 5, 281–298.

KEPHART, J. O. AND CHESS, D. M. 2003. The vision of autonomic computing. *Computer 36,* 1, 41–50.

MALONE, T. W. AND CROWSTON, K. 1994. The interdisciplinary study of coordination. *ACM Comput. Surv. 26,* 1, 87–119.

MAMEI, M. AND ZAMBONELLI, F. 2009. Programming pervasive and mobile computing applications: The tota approach. *ACM Trans. Softw. Eng. Methodol. 18,* 4, 1–56.

MILNER, R., PARROW, J., AND WALKER, D. 1992. A calculus of mobile processes, part I. *Information and Computation 100,* 1 (September), 1–40.

NARDINI, E., VIROLI, M., AND PANZAVOLTA, E. 2010. Coordination in open and dynamic environments with tucson semantic tuple centres. In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, S. Y. Shin, S. Ossowski, M. Schumacher, M. Palakal, C.-C. Hung, and D. Shin, Eds. Vol. III. ACM, Sierre, Switzerland, 2037–2044.

OMICINI, A. AND ZAMBONELLI, F. 1999. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems 2,* 3 (Sept.), 251–269.

PAOLUCCI, M., KAWAMURA, T., PAYNE, T. R., AND SYCARA, K. P. 2002. Semantic matching of web services capabilities. In *International Semantic Web Conference*. LNCS, vol. 2342. Springer, 333–347.

PARUNAK, H. V. D., BRUECKNER, S., AND SAUTER, J. 2002. Digital pheromone mechanisms for coordination of unmanned vehicles. In *Autonomous Agents and Multiagent Systems (AAMAS 2002)*. Vol. 1. ACM, 449–450.

PHAM, H., PALUSKA, J. M., SAIF, U., STAWARZ, C., TERMAN, C., AND WARD, S. 2009. A dynamic platform for run-time adaptation. *Pervasive and Mobile Computing 5,* 6, 676–696.

PICCO, G. P., MURPHY, A. L., AND ROMAN, G.-C. 1999. LIME: Linda meets mobility. In *The 1999 International Conference on Software Engineering (ICSE'99)*. ACM, 368–377. May 16-22, Los Angeles (CA), USA.

PRIAMI, C. 1995. Stochastic pi-calculus. *The Computer Journal 38,* 7, 578–589.

RATHINAM, M., CAO, Y., PETZOLD, L., AND GILLESPIE, D. 2003. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics 119,* 24.

RICCI, A., OMICINI, A., VIROLI, M., GARDELLI, L., AND OLIVA, E. 2007. Cognitive stigmergy: Towards a framework based on agents and artifacts. In *Environments for MultiAgent Systems*, D. Weyns, H. V. D. Parunak, and F. Michel, Eds. LNAI, vol. 4389. Springer, 124–140. 3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers.

ROMÁN, M., HESS, C. K., CERQUEIRA, R., RANGANATHAN, A., CAMPBELL, R. H., AND NAHRSTEDT, K. 2002. Gaia: a middleware platform for active spaces. *Mobile Computing and Communications Review 6,* 4, 65–67.

TOLKSDORF, R., NIXON, L. J. B., AND SIMPERL, E. P. B. 2008. Towards a tuplespace-based middleware for the Semantic Web. *Web Intelligence and Agent Systems 6,* 3, 235–251.

ULIERU, M. AND GROBBELAAR, S. 23-27 June 2007. Engineering industrial ecosystems in a networked world. In *5th IEEE International Conference on Industrial Informatics*. IEEE Press, 1–7.

UNIVERSITY OF BIRMINGHAM. 2007. The PRISM probabilistic model checker. Available at http://www.prismmodelchecker.org.

VIROLI, M. AND CASADEI, M. 2009. Biochemical tuple spaces for self-organising coordination. In *Coordination Languages and Models*, J. Field and V. T. Vasconcelos, Eds. LNCS, vol. 5521. Springer-Verlag, 143–162. 11th International Conference (COORDINATION 2009), Lisbon, Portugal, June 2009. Proceedings.

VIROLI, M. AND CASADEI, M. 2010. Chemical-inspired self-composition of competing services. In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, S. Y. Shin, S. Ossowski, M. Schumacher, M. Palakal, C.-C. Hung, and D. Shin, Eds. Vol. III. ACM, Sierre, Switzerland, 2029–2036.

VIROLI, M., CASADEI, M., AND OMICINI, A. 2009. A framework for modelling and implementing self-organising coordination. In *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, S. Y. Shin, S. Ossowski, R. Menezes, and M. Viroli, Eds. Vol. III. ACM, Honolulu, Hawai'i, USA, 1353–1360.

ZAMBONELLI, F. AND VIROLI, M. 2008. Architecture and metaphors for eternally adaptive service ecosystems. In *IDC'08*. Studies in Computational Intelligence, vol. 162/2008. Springer Berlin / Heidelberg, 23–32.

THIS DOCUMENT IS THE ONLINE-ONLY APPENDIX TO:

Spatial Coordination of Pervasive Services through Chemical-inspired Tuple Spaces

Mirko Viroli, Matteo Casadei, Sara Montagna
Alma Mater Studiorum – Università di Bologna, Italy
{mirko.viroli,m.casadei,sara.montagna}@unibo.it
and
Franco Zambonelli
Università di Modena e Reggio Emilia, Italy
franco.zambonelli@unimore.it

ACM Journal Name, Vol. ?, No. ?, ? 20?, Pages 1–0??.

A. CHEMICAL TUPLE SPACES: FROM FORMAL MODEL TO IMPLEMENTATION

A.1 Formal model

We introduce an executable model of chemical-inspired tuple spaces, which can be used to both compile any system specification into a Continuous-Time Markov Chain (CTMC)–usable to simulate system behaviour – and as an abstract design of the distributed execution platform. To keep the description of this model as simple as possible, we focus on chemical reaction and diffusion of tuples, and hence abstract away from the semantics of the coordination primitives to insert and observe tuples, which is orthogonal—the interested reader can find details on this in [Viroli and Casadei 2009]. Formalisation is provided using notation and style of process algebras [Milner et al. 1992]—which is typical for coordination models [Busi et al. 2000].

Syntax. Let meta-variable σ range over tuple-space identifiers, τ over first-order terms (namely, the actual content of tuples), r over positive real numbers, and n, m over natural numbers—real and natural numbers, as well as literals, can be used as constants for building terms.
The syntax of the model (along with the operational semantics described later) is expressed in Figure 10. Term $\tau\langle n\rangle$ represents a tuple with content τ and concentration value n. Syntax $\langle 1\rangle$ is considered optional, so that tuple τ actually means $\tau\langle 1\rangle$. A firing tuple ft is instead of the kind $\tau^{\rightsquigarrow}_{move}(\tau^{\delta}_{grad})\langle n\rangle$, where τ_{move} is the tuple to be moved, τ_{grad} is the tuple creating the local gradient, and δ is the local gradient direction: when $\delta = 0$, we simply denote the firing tuple as $\tau^{\rightsquigarrow}_{move}$ for there is no gradient influence. Note that in order to represent tuples, we rely on generic terms rather than mere lists of values and wildcards

Syntax:

t	$::=$	$\tau\langle n\rangle$	Tuple
ft	$::=$	$\tau^{\rightsquigarrow}_{move}(\tau^{\delta}_{grad})\langle n\rangle$	Firing Tuple
T	$::=$	$0 \mid t \mid ft \mid (T \mid T)$	Tuple set
L	$::=$	$[T_i \xrightarrow{r} T_o]$	Chemical Law
S	$::=$	$0 \mid T \mid L \mid (S \mid S)$	Tuple Space
C, D	$::=$	$0 \mid [\![S]\!]_\sigma \mid \sigma \overset{r}{\rightsquigarrow} \sigma \mid (C \mid C)$	Configuration

Auxiliary functions:

$$\tau\langle n\rangle \oplus S = \begin{cases} \tau\langle n\rangle | S & \text{if } \tau\langle m\rangle \notin S \\ \bot & \text{otherwise} \end{cases}$$

$$G(T, \tau\langle m\rangle \oplus S) = \begin{cases} \binom{m}{n} G(T', S) & \text{if } T = \tau\langle n\rangle \oplus T' \\ 1 & \text{if } T = 0 \end{cases}$$

$$F(S_s \oplus \tau_g\langle n\rangle, S_t \oplus \tau_g\langle m\rangle, \tau_g, \delta) = \begin{cases} \textit{noise} & \text{if } \delta = + \text{ and } n < m \\ \textit{noise} & \text{if } \delta = - \text{ and } m < n \\ 1 & \text{otherwise} \end{cases}$$

Transition rules:

(R) $$[\![T \mid [T_i \xrightarrow{r} T_o] \mid S]\!]_\sigma \xrightarrow{rG(T, T|S)} [\![T_o\{T/T_i\} \mid [T_i \xrightarrow{r} T_o] \mid S]\!]_\sigma$$

(M) $$[\![\tau^{\rightsquigarrow}_m(\tau^{\delta}_g)\langle n{+}1\rangle \oplus S]\!]_\sigma \mid [\![S']\!]_{\sigma'} \mid \sigma \overset{r}{\rightsquigarrow} \sigma' \xrightarrow{r(n+1)F(S, S', \tau_g, \delta)} [\![\tau^{\rightsquigarrow}_m(\tau^{\delta}_g)\langle n\rangle \oplus S]\!]_\sigma \mid [\![\tau_m | S']\!]_{\sigma'} \mid \sigma \overset{r}{\rightsquigarrow} \sigma'$$

Fig. 10. Computational Model

as in LINDA—similarly to [Omicini and Zambonelli 1999]. This choice is consistent with the need for a general approach to abstractly deal with (semantic) matching.

T denotes a composition of tuples and firing tuples by using composition operator "|" that is assumed to be commutative, associative, and able to absorb term 0—i.e. it is a multiset composition operator. We also assume that term $\tau\langle n\rangle \mid \tau\langle m\rangle$ is syntactically equivalent to $\tau\langle n+m\rangle$—hence a tuple can be seen as either joined into a single term, representing the whole substance in the solution, or split in two (or recursively more) terms down to tuples with concentration 1. L is a chemical-like law (also called reaction), expressing transformation of tuple set T_i (reactants) into T_o (products) with chemical rate r; note that although not explicitly prevented here, laws whose reactants (left-hand side) include firing tuples seem not useful—and hence might be excluded from a surface language for chemical laws. S is a composition of laws L and tuples T, which hence represent a tuple space. Finally, C is a system configuration, which is modelled as a flat composition of tuple spaces $[\![S]\!]_\sigma$ (σ is the space identifier) and links between tuple spaces $\sigma \overset{r}{\rightsquigarrow} \sigma'$ (r represents the link rate).

Note that all the above syntactic elements are considered as terms, e.g. "$.\langle.\rangle$" is considered as a binary functor. The same applies to all the other constructs: hence, firing tuples are a special kind of tuple since $\tau^{\rightsquigarrow}_{move}(\tau^{\delta}_{grad})$ is a special kind of term.

Substitution and Matching. We equip terms with the concepts of substitution and matching, which are application-specific: as a consequence, they are considered abstract in our model. We first assume the existence of a matching function μ for terms, such that $\mu(\tau, \tau') \in [0, 1]$. Secondly, we need a concept for substitution, introduced via a ternary partial function $\tau\{\tau_1/\tau_2\}$. This function first matches two terms τ_1 and τ_2. Such a match

intuitively returns a substitution (e.g. from variables to terms in first-order logic), which is however left implicit: we only consider the effect of applying it to τ, which returns a new term. This ternary operator is partial, since it provides no result in the case τ_1 and τ_2 do not match. Matching function μ is in principle orthogonal to substitution: though, in our calculus the result of $\tau\{\tau_1/\tau_2\}$ is used only if $\mu(\tau_1, \tau_2) > 0$. Considering syntactic matching, we would have as usual that $t(X)\{q(X)/q(1)\} = t(1)$: intuitively, $\{q(X)/q(1)\}$ yields substitution $X/1$ which is then applied to $t(X)$. If we consider semantic matching in our chemical-like scenario, supposing that tuple x matches semantic template X and y matches semantic template Y, we would have $(X|X)\{(X|Y)/(x|y)\} = (x|x)$: intuitively, $X|Y$ matches tuples x and y yielding substitution $\{\text{X/x,Y/y}\}$, which is then applied to $X|X$ giving $x|x$. This mechanism is used when applying chemical laws; it would make chemical law $X|Y \xrightarrow{r} X|X$ applicable to tuple set $x|y$, producing tuple set $x|x$.

Auxiliary functions. The auxiliary infix and partial function $\oplus$ is introduced to extract the overall concentration of a tuple into a tuple space. It takes a tuple $\tau\langle n\rangle$ and a space S: if S does not include τ, it simply composes $\tau\langle n\rangle$ with S, otherwise it provides no result. Note that $\oplus$ is not a syntactic constructor (it is not part of the syntax), but simply a lookup function: for instance, the equation $\tau|\tau|\tau'|\tau\langle 5\rangle|\tau'\langle 4\rangle = \tau\langle n\rangle \oplus S$ has only one solution, which gives $n = 7$ and $S = \tau'|\tau'\langle 4\rangle = \tau'\langle 5\rangle$

Another auxiliary definition concerns function G, which takes the reactants T of a law and the content of a tuple space S, and computes how many different combinations of tuples in T can be found in S—this function is key to properly compute chemical rates according to Gillespie's algorithm [Gillespie 1977]. Suppose T includes n different copies of τ (in natural chemical systems it is often supposed that $n \leq 2$, though our model does not have this limitation), and let m be the overall concentration of τ in the current location, then the multiplicative contribution of τ to the overall number of combinations is given by binomial $\binom{m}{n}$. For instance, we have:

$$G(\tau|\tau|\tau', \tau\langle 10\rangle|\tau'\langle 20\rangle) = G(\tau|\tau, \tau\langle 10\rangle)G(\tau', \tau'\langle 20\rangle) = \frac{10 \times 9}{2} \times 20$$

Finally, function F is used to compute the transfer rate for a firing tuple. It takes the content of the source tuple space, the content of the target tuple space, the local gradient tuple, and the local direction δ, and returns a real number in $[0, 1]$ – to be multiplied by the actual link rate r. After getting the concentration of the local gradient tuple in source and target (n and m), it yields a `noise` value if the tuple has to ascend the local gradient, 1 otherwise.

Operational semantics. The operational semantics of this calculus is given in terms of a Continuous-Time Markov Chains model—following the work by Gillespie [Gillespie 1977] and other existing stochastic languages for biochemistry [Priami 1995]. A transition system $(C, \rightarrow, \mathbb{R}_0^+)$ is defined (bottom part of Figure 10) where transitions are of the kind $C \xrightarrow{r} C'$, meaning that system C moves to C' with dynamics/likelihood expressed by Markovian rate r. Such transition rules are intended as local rewrite rules—they can be applied to any system sub-part in isolation.

Rule (R) handles the execution of a chemical reaction inside a space. Let σ be a space including law $[T_i \xrightarrow{r} T_o]$ as well as reactants T matching T_i; then, a transition can occur which replaces T with $T_o\{T/T_i\}$ with rate $rG(T, T|S)$: this is because each single combination of the molecules in T is equally subject to the chemical law with rate r. As a simple

example, consider a single space with tuple $x\langle 1000\rangle$, and the chemical law of radioactive decay [Gillespie 1977] "$[X \xrightarrow{]} 10.00$" that we denote as L, with x perfectly matching X. The system allows for a chain of transitions leading to the decrease of tuple concentration:

$$[\![x\langle 1000\rangle \mid L]\!]_\sigma \xrightarrow{10000} [\![x\langle 999\rangle \mid L]\!]_\sigma \xrightarrow{9990} \ldots \xrightarrow{10} [\![x\langle 0\rangle \mid L]\!]_\sigma \not\rightarrow$$

Rule (M) handles movement of one firing molecule towards a neighbouring space. Let σ and σ' be two spaces connected by a link with rate r, and suppose that inside σ there is a firing tuple $\tau_m^{\rightsquigarrow}(\tau_g^\delta)$ with overall concentration $n+1$; then, a transition can occur which decreases such a concentration to n, while inside σ' we add a tuple τ_m. The rate of this transition is $r(n+1)F(S,S',\tau_g,\delta)$, in that each single item of $\tau_m^{\rightsquigarrow}$ can move with rate r, multiplied by transfer function as computed by (M). To exemplify the application of this rule, consider now the chemical law $L = [X \xrightarrow{0.01} X^{\rightsquigarrow}]$ used to move any tuple X to a neighbouring space without relying on any local gradient (direction is set to 0), and a space initially holding $x\langle 1000\rangle$. Initially, only the following transition is applicable:

$$C_0 = [\![x\langle 1000\rangle \mid L]\!]_\sigma \mid [\![0]\!]_{\sigma'} \mid \sigma \overset{0.01}{\rightsquigarrow} \sigma' \xrightarrow{10} [\![x\langle 999\rangle \mid x^{\rightsquigarrow} \mid L]\!]_\sigma \mid [\![0]\!]_{\sigma'} \mid \sigma \overset{0.01}{\rightsquigarrow} \sigma' = C_1$$

Then, two transitions are available from C_1, one creating another firing tuple and one moving the firing tuple to σ' due to (F):

$$C_1 \xrightarrow{9.99} [\![x\langle 998\rangle \mid x^{\rightsquigarrow}\langle 2\rangle \mid L]\!]_\sigma \mid [\![0]\!]_{\sigma'} \mid \sigma \overset{0.01}{\rightsquigarrow} \sigma'$$
$$C_1 \xrightarrow{0.01} [\![x\langle 999\rangle \mid L]\!]_\sigma \mid [\![x\langle 1\rangle]\!]_{\sigma'} \mid \sigma \overset{0.01}{\rightsquigarrow} \sigma'$$

Note that the production of firing tuples and their movement to σ' proceed in parallel, until all the tuples are transferred to σ'.

A.2 Putting the model to work

We here outline the main issues arising when using the above model as an executable specification, both for simulating system behaviour and as basis for implementing an execution infrastructure. These details are also useful to understand the performance limits our implementation is subject to.

Simulation Methodology. In order to test the expected behaviour of our chemical-inspired tuple spaces, we rely on formal simulations of the evolution of tuple concentration in given scenarios. Such simulations can be conducted relying on the above operational semantics. As such, once initial tuple space state and laws are fixed, the evolution of a service ecosystem can be simulated using any available framework for CTMCs, like e.g. PRISM [University of Birmingham 2007] (which also allows for stochastic model-checking), typically working via Stochastic Simulation Algorithms based on [Gillespie 1977]. In spite of their optimised variants (e.g. [Gibson and Bruck 2000]), they all are based on the iterative execution of the following procedure: *(i)* the list of all possible transitions is generated, each equipped with its own Markovian rate (computed as shown in our operational semantics), *(ii)* one transition is chosen probabilistically, giving higher probability to those with higher rates (proportionally), *(iii)* the corresponding transition rule is applied to the system state, and *(iv)* the simulation time is increased by ΔT time units computed as $ln(1/\tau)/R$, where τ is a random number between 0 and 1, and R is the sum of the transition rates—note the average ΔT is $1/R$. Specific optimisations as described in [Gibson and Bruck

2000], which turn out to be useful when systems with many compartments are to be simulated, include recomputing only changed rates instead of performing item *(i)* each time, and using a binary (cumulative) search tree to perform item *(iii)* in logarithmic time.

Implementation Infrastructure. As far as implementing an infrastructure for chemical-oriented tuple spaces is concerned, we observe that an implementation from scratch should not be necessary, but existing programmable coordination platforms can be reused. For instance, following the guidelines described in [Viroli et al. 2009], an infrastructure for chemical-oriented tuple spaces can be directly mapped over TuCSoN [Omicini and Zambonelli 1999], which provides tuple centres—namely, tuple spaces programmable via a logic reaction system. In this case, supporting chemical tuple spaces amounts to: *(i)* inject coordination laws in the form of proper tuples into tuple centres, *(ii)* program the suitable matching algorithm for the application at hand (the full power of Prolog language is used to this end), *(iii)* program tuple centres so as to pick up chemical reactions and apply them probabilistically, following the above algorithm. Relying on TuCSoN provides implementation aspects which are orthogonal to our model, such as the management of tuple spaces, their retrieval, and the interaction between agents and tuple spaces.

Concerning semantic matching, we rely on the approach outlined in [Nardini et al. 2010], where W3C standard OWL (Ontology Web Language) [Horrocks et al. 2003] is used to model domain ontology, and Fuzzy Description Logics [Baader et al. 2003; Bobillo and Straccia 2008] for implementing semantic match—tuples are seen as domain objects, templates as domain concepts, and matching as instance checking.

Performance Issues. From the viewpoint of performance, implementation is necessarily constrained by the fact that the average ΔT drawn at each step of Gillespie's algorithm may be too short. On the one hand, actually executing a chemical reaction may require a time that is greater than the average reaction transition time Δ_r of the chemical system (computed through transition rule (R)): if this is the case, obviously the infrastructure would slow system evolution down with respect to the expected Markovian dynamics. Similarly, the average time for transferring a tuple may be greater than the average moving time Δ_m of the chemical system (computed through transition rule (M)). Note that an implementation sending a whole packet of firing tuples altogether, instead of one-by-one as prescribed by the operational semantics, is possible—in Section 6.1 we analyse the corresponding accuracy.

Given the application at hand, the designer of chemical reactions is generally responsible for the task of guessing (or enacting) the rate at which individuals are injected in the system, detecting ΔT_r and ΔT_m in the worst case (e.g. in the tuple space with higher computational load), and accordingly adjusting chemical rates so as to guarantee – whenever possible – an accurate execution of chemical reactions. A deeper description of methodological aspects is out of the scope of this paper, though in this paper we exemplify how performance issues can be analysed in selected case studies.