Chemistry-inspired, Context-Aware, and Autonomic Management System for Networked Objects

Mahmoud ElGammal and Mohamed Eltoweissy
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia, USA
{gammal,toweissy}@vt.edu

***Abstract*—We present a new method for network configuration and management in pervasive computing systems inspired by the Chemical Affinity theory. We coined our approach C_2A_2: Chemistry-inspired, Context-Aware, and Autonomic management system for networked objects. The hypothesis behind C_2A_2 is that by paralleling the model of interaction that takes place between atoms during a chemical reaction, a form of collective intelligence emerges among system components enabling them to achieve a common global objective while relying primarily on preferences expressed at the individual level. In C_2A_2, both physical and logical entities in the network are modeled as atoms with varying levels of affinity toward each other. Network reconfiguration is realized by breaking existing bonds between atoms and establishing new ones based on inter-atom affinities, which change continuously in response to context dynamics. Context is seen as having either a catalytic or inhibiting effect on reactions and is used to guide bond creation in favor of reactions that are most suitable to the situation at hand. Bonds are established by exchanging messages between atoms using a protocol that leverages the Affinity Propagation algorithm, which is used in C_2A_2 as a reaction execution engine. Finally, we use simulation to evaluate the performance of C_2A_2 in clustering and task assignment in wireless sensor networks.**

Keywords-affinity propagation; bio-inspired computing; chemical affinity; context awareness; Internet-of-Things; networked objects; pervasive computing.

I. INTRODUCTION

In this paper we propose a new approach to network configuration and management in pervasive computing systems inspired by the Chemical Affinity theory, which was the key concept in the development of the idea of chemical equilibrium. The hypothesis behind our work is that by paralleling the model of interaction that takes place among atoms during a chemical reaction, we enable a form of collective intelligence to emerge among nodes, enabling them to achieve a common global objective while relying primarily on local decisions.

The objective of our work is to enable effective multi-mission Smart Cyber-Physical Spaces (CPSs) that are context-aware, adaptable, autonomic, and efficient. This would eventually allow for self-managing CPSs capable of hosting multiple applications operating in different contexts with competing demands, whereby for each context the system optimizes its different parameters to the situation at hand. We have coined our system C_2A_2 (short for Chemistry-inspired, Context-Aware, and Autonomic management system for networked objects). In C_2A_2, context-awareness goes beyond affecting the operational state of the system to guiding its self–management, configuration, and optimization. Chemical Affinity and Affinity Propagation [3] –inspired techniques for self-management make the ability to serve multiple concurrent goals efficiently an intrinsic property of the network – one that doesn't require special handling.

The motivation behind our work stems from the observation that a certain degree of congruence exists between the structure and interaction dynamics in a pervasive computing system and their analogues in several natural ecologies. Adding to that the possibility of modeling ecology dynamics using chemical reactions – which, as pointed out in [31], has been proven in [9] and [10] – this motivated us to explore the possibility of using the concept of chemical affinity to model different interaction patterns that take place in a pervasive computing system. The heterogeneous nature and rapidly changing structure of networks that underlie future pervasive computing environments require individual system components to posses a high degree of autonomy and adaptability. We believe that by incorporating the concepts of chemical affinity, affinity profiles, and affinity propagation as outlined in the remainder of this document, we can achieve a system that possesses such qualities. The notion of chemical affinity emphasizes the individuality of each system component, allowing it to freely express the services it can offer to the rest of the system, as well as what it needs to consume from other components in order for it to function properly. This emphasis on individuality is essential in a ubiquitous environment where the relationship between different actors in the system is often based on a supply and demand model. Moreover, the flexibility provided by allowing each entity in the system to dynamically vary its affinities toward other entities allows for fine-grained adaptation and provides a means for subsuming context dynamics into system operation. Finally, the utilization of affinity propagation permits the system to explore different solutions concurrently in the search space of any given problem, allowing for a sort of collective intelligence to emerge as individual entities take local decisions that serve high-level objectives.

The contributions of this work lie in the utilization of the following techniques in order to realize the aforementioned system characteristics:

- **Chemical affinity –inspired collective intelligence:** we introduce the notion of perceiving a CPS as a dynamic chemical solution and utilize the concept of chemical affinity to model the interactions between its components. Each node in C_2A_2 is governed by a mutable *affinity profile*, which determines its affinity to other entities in the system, such as events, tasks, users, or other nodes. Affinity profiles allow nodes to take individual decisions that ultimately serve a global system objective, and they serve to provide a

 ISBN: 978-1-61208-389-6

flexible interconnection framework, which plays a central role in realizing self-organization.

- **Affinity propagation:** we reutilize the Affinity Propagation clustering algorithm [3] as a reaction execution engine, which we then use for executing different network configuration and management tasks. Affinity Propagation is a clustering algorithm that relies on exchanging *messages* between data points to compute pair-wise similarities, which are then used to recursively combine similar points together. The algorithm need not be initialized with exemplar points, which is advantageous in dynamic environments where new resources might join or leave the system at any time.

The remainder of this paper is organized as follows. Section II provides some background on the chemical affinity theory and presents an overview of the related work in literature. Section III discusses the C_2A_2 system. Section IV presents a number of case studies and an experimental evaluation of the system performance. Finally, Section V concludes the paper and discusses future work.

II. BACKGROUND AND RELATED WORK

A common feature among most – if not all – natural computing models that draw their inspiration from the chemical reaction metaphor, is that the system state is represented as a fluid in which reagents of different types move freely and interact with each other according to predefined reaction rules. Developing concrete applications based on this concept requires mature models of computation that can be used to encode real-life problems using the chemical formalism and describe programs to solve them, as well as runtime systems that can actually execute these programs. The former field of study has seen significant research activity, ushered by the Γ language by Banatre et al [18] and continued through various other works such as the Chemical Abstract Machine by Berry et al [1], the Molecular Dynamics model by Bergstra et al [19], Membrane Computing (P Systems) by Păun [2], and more recently in the Biochemical Tuple Spaces model by Viroli et al [20]. Some of these works contributed incremental improvements over previous models while others offered entirely new approaches, but all have served to present the chemical metaphor as a mature and viable option for modeling computational processes, especially for applications where concurrency and self-organization are two necessary characteristics. However, despite the progress on this front, not as much attention has been given to the runtime systems on which chemical computing models can be executed [21].

The earliest runtime system for a chemical machine is perhaps the one described in [18], where an implementation of the Γ language on a massively parallel machine (aka the Γ-machine) is proposed. In order to evaluate a Γ program, the runtime system has to perform two tasks: (a) search for reagents that satisfy reaction conditions (in other words, determine which reactions to fire), and (b) applying the actions associated with fired reactions on the system. It can easily be shown that a roughly similar breakdown of tasks would also apply to any other runtime, not just the Γ-machine. The first task requires solving an NP-hard optimization problem, and with C_2A_2, we attempt to put forward a practical, efficient, and scalable solution to this problem.

Existing runtime systems employ different approaches to address this problem. One approach can be described as the *search-and-match approach*, and it usually relies on some data structure that stores information about reagents and reaction rules where a search algorithm is then used to find reagents that satisfy the left-hand side conditions of any given reaction. The implementation proposed for the Γ-machine in [18] falls under this category. However, it can be considered more of a proof of concept as it assumes a number of processors equal to the number of reagents, which would be faced by strict scalability limits in reality. Additionally, it only considers one form of reactions (more specifically, reactions that take exactly two input values and produce two output values), which would impose further constraints on the practicality of this approach. More efficient methods belonging to this category have also been proposed, such as The Chemical Machine by Rajcsányi et al [22], which relies on the more sophisticated RETE pattern-matching algorithm [23].

Another approach that is mainly used for simulation but is also used in some runtime systems is the *computational chemistry approach*. Methods belonging to this category rely on algorithms that have long been used by theoretical chemists to solve many quantitative chemical problems using simulation with acceptable accuracy. Several algorithms have been developed under this category and which have been improving in efficiency over time, such as Gillespie's First Reaction Method [24], the Next Reaction Method by Gibson et al [25], Slepoy et al's constant-time Monte Carlo algorithm for simulating biochemical reaction networks [26], ALCHEMIST [27] and others. These methods offer a statistically correct depiction of the evolution of species concentration in a chemical solution over time, which is necessary in applications that rely on accurate simulation of the laws of chemical kinetics.

These two approaches have different points of strength and weakness. The *search-and-match* approach can be used to model the behavior of a self-organizing system in terms of microscopic interactions among its lowest-level components, which makes it a more versatile tool for modeling a wide range of applications. On the downside, it offers limited control on the macroscopic behavior of the whole system [28]. The *computational chemistry* approach on the other hand offers greater control over the macroscopic behavior of the system, which allows for better overall stability and predictability, but only if the target application lends itself easily to this approach, such as the case studies given in [27, 29, 30, and 28]. With C_2A_2, we aim to combine some of the advantages of these two approaches.

III. THE C_2A_2 SYSTEM

Before we're able to utilize the chemical metaphor in our target domain, we must first map some key elements that would allow us to mirror the more complex operations that require interaction and cooperation among such elements. At the lowest level, the most basic building block of interest to us is the *atom*, which is paralleled by an individual entity in the modeled system (e.g. network node, event, user, etc.). Atoms from different elements possess distinct qualities, resulting in different affinities toward atoms from other substances. When an atom combines with other atoms, either from the same or

different substance, a *molecule* is formed, which represents a higher-order structure. The counterpart in the network would be a group of cooperating nodes that are structurally *less attached* to the rest of the network – for instance, a cluster within a clustered network, or a group of nodes assigned to a particular task. A molecule can be defined as a specific configuration of *bonds* among atoms of particular elements. In the network, such bonds are paralleled by network connections, where connections between nodes belonging to the same cluster (intra-cluster) mirror the intra-molecular forces holding a molecule together, and connections among different clusters (inter-cluster) would be similar to the weaker inter-molecular forces that exist between different molecules.

The objective of the network is to restructure itself by optimizing *bond* selection between nodes in order to better serve the tasks presented to it. The network is described as *stable* if no such bonds are in the process of being formed. By establishing all necessary bonds, the node groups (*molecules*) required to serve all present tasks are constructed, which parallels the state of *chemical equilibrium*. An important realization regarding the state of chemical stability, or equilibrium, is that despite the apparent steadiness of the system, the potential for dissolution and thereby the formation of new bonds between different reactants is always present. This possibility emanates from the fact that the state of stability is in reality the result of a balance between antagonistic forces that never cease to exist as long as the reactants themselves exist. Whenever a change is introduced in the system – as in the addition or removal of a reactant or a change in system conditions – existing affinity forces become unsettled and they compete again until equilibrium is gradually regained, possibly resulting in new products being formed. This latent potential, even under a stable or steady state, provides the system with a high degree of dynamism in the face of change.

A. Abstract System View

We now present an abstract view of the system we propose. The system is seen as a pool of networked resources with different roles, capabilities, and other attributes of different degrees of relevance according to hosted applications – such as function, geographic location, resource levels, and so forth. Throughout the lifetime of the system, it is presented with events that can be categorized into three classes: *application events*, *system events*, and *circumstantial events*. An example for application events is user queries, while an example for system events is change in resource levels, and an example for circumstantial events is any change in the conditions within which the system operates. The system is programmed to respond to each event by executing a set of tasks. In order to do that, a group of resources with appropriate capabilities must be selected and assigned to each task. This process involves optimizing resource selection, configuring selected resources, and reorganizing the pool of resources by establishing the required connections among its members.

B. C_2A_2 Abstraction Domains

One of the fundamental challenges that manifest themselves when embarking on the task of designing a pervasive computing system is the heterogeneity of its elements. Ubiquity is a highly desirable feature in all but the most narrowly scoped of such systems, and this immediately entails the involvement of a host of highly diverse real-life entities. Such entities play different roles in the system in terms of whether they produce or consume information, the type of information they produce or consume, the tasks they partake during information processing, and the depth of their impact on how the system is configured and the selection of its short-term objectives. Accordingly, this implies that such entities would possess extremely heterogeneous and non-normalized characteristics, and consequently interaction models. It is then only imperative that an appropriate abstraction tool be utilized so as to normalize and homogenize such diverse elements of the system for it to be able to deal with them uniformly and efficiently. As illustrated in Fig 1, C_2A_2 can be perceived at three levels of abstraction, or as having parallel manifestations in three different domains, which are discussed below.

1) The Physical Domain

In the first – and lowest – level, lies the *physical domain*, which encompasses all the elements of the system in their real-life representation. These are heterogeneous physical and logical entities that are either among the system stakeholders or are necessary for its operation. The physical entities can be categorized into *active* and *passive* ones. Active entities are those that participate in imposing the (short-term) functional objectives of the system, in addition to elements that partake in the execution of such objectives. An example of the former is system users, whether they are human users or software agents, while an example of the latter is the various hardware components that carry on the execution of requests generated by users. These components can either be infrastructural (e.g. communication devices, generic processing devices, network health monitoring hardware, etc.) or application-specific, which are customized and properly equipped to generate, collect, and process the exact information needed by the possibly different applications running on the system. Passive entities in the physical domain are necessary for proper operation of the system, but they play a role that is rather assistive, as they mainly comprise resources that handle storing and managing network state and application data, deploying and distributing middleware updates, and overseeing the knowledgebase responsible for defining and formulating the system response to various events. The third and final element in the physical domain is the different types of events that constitute contextual information.

2) The Logical Domain

In the next level lies the *logical domain*, which constitutes the first level of abstraction. In this domain traditional object-oriented software modeling techniques are applied in order to represent all entities in the physical domain as well as their respective interrelationships. Each entity is represented by a proxy object that serves three purposes: (1) it provides a faithful depiction of the characteristics of its corresponding real-life object in the physical domain, represented using a normalized form that permits and simplifies manipulation via software, (2) it acts as an observer of its physical counterpart, and as an upstream channel that propagates any changes to its state to the higher layers of the system, and (3) it acts as a surrogate that provides access to the physical entity, allowing

 ISBN: 978-1-61208-389-6

control and configuration commands to flow downstream through the system to be finally effected in the physical domain.

3) The Chemical Domain

The third and final level is the *chemical domain*, which adds homogeneity to the normalized representation attained in the previous level. Semantic mapping is used to reduce the high dimensionality of the OOP representation, and produce a homogenous model where all entities are represented uniformly and their different characteristics are abstracted via different quantification techniques. Each entity is represented as an atom with an associated *affinity profile*. Events constituting the context are represented as either catalysts or inhibitors, which according to each atom's affinity profile, either attract or repel an atom with a certain strength. This is used as a platform for executing *reaction rules* that emulate chemical reactions with the objective of reorganizing and reconfiguring the system, taking into consideration user-generated requests, the internal state of the system, as well as contextual information. In the rest of this document, we will be focusing more onto the components of this layer, which will be referred to as the Chemical Abstraction Layer (CAL), and discuss the various operations that take place therein.

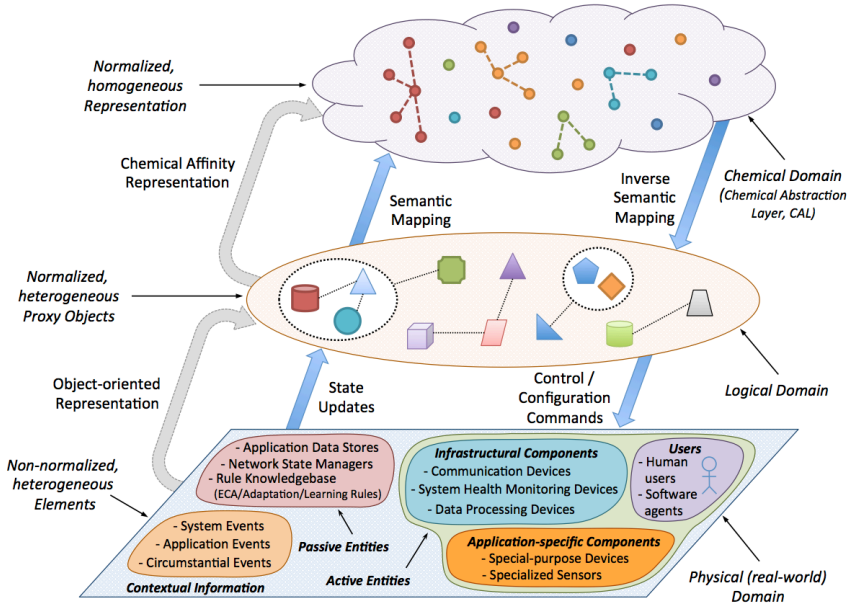

Figure 1. C_2A_2 at different levels of abstraction.

C. The Chemical Abstraction Layer (CAL)

In chemical-inspired systems (e.g. CHAM [1]), the state of a system is like a chemical solution in which floating reactants can interact with each other according to specific reaction rules. A certain stirring mechanism is responsible for causing motion within the solution, allowing for possible contacts between reactants. A solution can be heated to break complex molecules into smaller ones up to ions. Conversely, a solution can be cooled to rebuild heavy molecules from simpler reactants. Furthermore, to make it possible to model and solve problems of hierarchical structure, a molecule is allowed to contain a sub-solution enclosed in a membrane, which can be somewhat porous to allow communication between the encapsulated solution and its environment [1]. In this section we show how all of these concepts are realized in C_2A_2.

1) Affinity Profiles

One of the goals that we attempt to achieve in C_2A_2 is adaptability. In order to optimize the system under changing circumstances and varying tasks, the system configuration should be easily and dynamically adaptable to both external and internal stimuli. Several factors must be considered when assigning tasks, allocating resources, or carrying out any network configuration operation. For instance, in a network of heterogeneous nodes, the decision to allocate a node to a certain task must take into consideration the compatibility between such task and the capabilities of the selected node. Other factors may also affect the decision, such as node location, its residual power, whether the node is allocated to another task with a higher priority, and so forth. We use the concept of affinity to express the strength of such compatibility.

In our proposed scheme, we would like each node to behave as independently as possible in order to minimize the communication overhead, but without compromising the functionality of the network as a whole. In other words, we would like to guide nodes to take local decisions that collectively serve global goals. Each real-life entity, be it physical or logical, is represented in CAL as a reactant (atom or molecule) with the capacity to interact with other reactants to establish new bonds and possibly disband existing ones. In C_2A_2 this capacity is expressed using an *affinity profile*. An affinity profile is akin to atomic valence, which determines the ability of an atom to form bonds with other atoms. Unlike valence, however, affinity profiles are mutable. In CAL, affinity between two reactants might change over time due to changes in context or in the internal state of the physical

 ISBN: 978-1-61208-389-6

domain counterpart of either reactant. This allows us to incorporate context dynamics in the system configuration loop and adapt each individual component accordingly.

An affinity profile can be modeled mathematically as a vector $P_i = \langle f_{ij} \rangle$, where f_{ij} is the affinity of reactant i toward reactant j. Affinities are expressed as real numbers where maximum affinity, minimum affinity, and neutrality are denoted by ∞, $-\infty$, and 0 respectively. One advantage of this representation is its neutrality to the application-specific interpretation of affinity values, which provides a flexible and generic tool to carry out various system configuration tasks. CAL is absolutely agnostic to application semantics, and thus, the process of assigning initial affinities and updating them as different events occur during system operation is handled independently via a semantic mapping mechanism that resides between the logical layer and CAL. Such mechanism is cognizant of application semantics, contextual information, as well as the precise state of the entities being mapped, which is made possible by its access to their representation in the logical domain. Affinities are not necessarily symmetrical; i.e. f_{ij} and f_{ji} don't have to be equal, which allows for more independence for reactants to bond with the most suitable peer. As will be shown in the examples in Section IV, each entity independently controls its initial affinity toward other entities, and it is up to the reaction execution mechanism to resolve conflicts that arise between them. Initial affinities can be computed using a variety of ways that can be as plain or as elaborate as the system permits. For instance, neural networks can be used to produce the affinity values based on arbitrary inputs, including, but not restricted to, context parameters. This would provide a means for incorporating machine learning techniques, which can be used to retrain the neural networks in order to produce affinities that optimize system performance over time.

2) Context Representation

In CAL, context is seen as having either a catalytic or inhibiting effect on ongoing reactions. A catalyst is a substance that causes or accelerates a reaction without itself being consumed, while an inhibitor has the opposite effect. In addition to its congruity with the chemical frame of reference employed herein, we also find that this perception of context is in accordance with how context information is used in a context-aware system, where the presence of a certain context element (also referred to as *parameter*, *variable* or *dimension*) typically triggers an adaptational response where the system incorporates the newly sensed context information into the process of deciding how the system configuration should be changed to best serve the task at hand. The catalytic or inhibiting effect owing to the presence of a specific context parameter is exhibited at two levels. At a high level, it controls the overall attraction force by which a reaction rule pulls reactants. This has the effect of regulating the rate at which each reaction takes place by controlling the amount of reactants consumed by it. At a lower level, context is used to mutate the affinity profiles of individual reactants, leading to different bonding choices than what would have taken place in absence of contextual influence. Both effects are modeled as a multiplicative factor that either strengthens or weakens the affinity between a rule and a reactant or between two reactants that are potentially to be consumed by the same reaction.

Now that we have discussed the role of affinity profiles, which is one of the key factors in determining the behavior of reactants in CAL and the influence of context on them, we now turn our attention to the final factor that constitutes the driving force behind reaction execution in CAL.

3) Affinity Propagation

Affinity propagation (AP) is a clustering algorithm that operates by exchanging messages between data points [3]. In essence, AP is a heuristic approach for finding an approximate solution to the maximization problem in graphs, which is known to be NP-hard [14]. AP relies on the max-product belief propagation algorithm [12, 13] to optimize an objective function that aims to maximize the sum of similarities between nodes and their exemplars. Despite being conceived as a clustering algorithm, AP is utilized in this work as the reaction execution engine in CAL, where it is responsible for exchanging affinities between reactants and updating them iteratively during the course of a reaction until final bonds are established. In order to explain how this is achieved, we first need to discuss how AP works in slightly more detail.

AP takes as input the initial measures of similarity between each pair of data points and simultaneously considers all data points as potential exemplars to each other. These values also include self-similarities, or *preferences* [3], which express the a priori willingness of individual points to become exemplars. Message exchange then takes place between data points until a high quality set of exemplars and corresponding clusters gradually emerge. In this paper, we use the binary variable model formulation of AP presented in [11] because of its ease of extension. In this model, cluster affiliations are obtained by computing an $N \times N$ array of binary variables h_{ij} using the factor graph shown in Fig 2, where $i, j \in \{1..N\}$ and N is the number of data points. $h_{ij} = 1$ if j is the exemplar of i, otherwise $h_{ij} = 0$. As pointed out in [11], in order for exemplar assignments to be valid, two constraints must hold in the solution obtained by the factor graph:

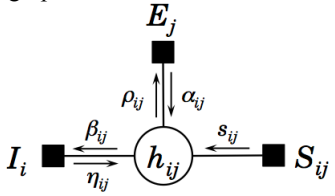

Figure 2. Factor graph for computing cluster affiliation between nodes i and j [11].

• ***1-of-N Constraint*:** Each node i must select exactly one exemplar (i.e. $\forall i \in [1,N]: \sum_{j=1}^{N} h_{ij} = 1$), which can be i itself if it is an exemplar node (in which case $h_{ii} = 1$).

• ***Exemplar consistency constraint*:** A node i may only choose j as its exemplar if j has chosen itself as an exemplar (i.e. $\forall i \neq j, h_{ij} = 1 \rightarrow h_{jj} = 1$).

It can easily be seen that any solution satisfying the two constraints above would be correct, although not necessarily optimal. There are three function nodes that contribute to the value of h_{ij}, which are shown in the factor graph in Fig 2: S_{ij} represents the similarity between nodes i and j, while I_i and E_j correspond to the two constraints above, respectively. Cluster affiliations are found by executing the max-sum (or log-

domain max-product) algorithm on the graph, which works by exchanging the five messages shown in Fig 2:

- s_{ij} is a scalar that expresses how similar node *i* is to node *j*. In this formulation of AP, this value also subsumes *i*'s preference.
- *Responsibility* messages (ρ_{ij}) are sent from a node *i* to a candidate exemplar *j*, and they represent the accumulated evidence for how well suited point *j* is to serve as an exemplar for point *i.*
- *Availability* messages (α_{ij}) are sent from *j* to *i*, reflecting the accumulated evidence of how appropriate it would be for point *i* to choose *j* as its exemplar, given the support *j* has from other candidate cluster members. Together with ρ_{ij}, these two messages are used to verify the fulfillment of constraint E_j.
- Similarly, β_{ij} and η_{ij} are used to test whether constraint I_i is violated for node *i*.

The derivation of these messages is given in detail in [11]. If either constraint is violated, its corresponding factor assumes a value of $-\infty$, otherwise it is set to *0*. The objective function being maximized by the algorithm is given in (1). The function maximizes the sum of intra-cluster similarities but also ensures that invalid cluster assignments are discarded by incorporating the constraints, which would cause the function to yield $-\infty$ if either constraint is violated for any node.

$$\mathcal{F}(\{h_{ij}\}) = \sum_{i,j} s_{ij} h_{ij} + \sum_i I_i(h_{i:}) + \sum_j E_j(h_{:j}) \quad (1)$$

Evidently, (1) is only suitable for clustering. However, [11] shows how this formulation of AP can be extended to solve other problems, such as the Facility Location [15] and Maximum Bipartite Matching [16] problems. Shamaiah et al. [17] also apply the same technique to develop distributed routing mechanisms for networks. In C_2A_2 we utilize the extensibility of the binary AP model to produce cluster affiliations that obey the reaction rules defined in CAL. More precisely, we modify the factor graph and objective function above by manipulating node similarities and defining different constraints such that the resulting clusters would resemble the structure of molecules produced by the reaction rules. We now describe how this was achieved in detail.

4) Utilizing AP as a reaction execution engine in CAL

CAL is an implementation of a restricted P System [2] where reactions can only take one form in either direction. Reactions in the forward direction take the form $ca \rightarrow cu$, while reverse directions take the form $cu \rightarrow ca$. In both forms, *c* is an optional catalyst, and *a* and *u* are multisets such $|a|>1$ and $|u|=1$. The forward form is used to compose complex molecules from multisets of atoms and/or simpler molecules, while the reverse form is used to dissolve molecules produced previously by forward reactions to their simpler components, thereby returning them back to the environment. Both forms may optionally require the presence of a catalyst as a condition to be triggered. Reactions of different forms that produce an arbitrary number of outputs are not allowed.

Under this restriction, the problem of assigning reactants to forward reactions can be seen as a clustering problem in which reaction rules play the role of exemplars while reactants (atoms and molecules) play that of cluster members. However, one property of P Systems necessitates a slight departure from traditional clustering, which is that reactions must be performed in a *maximally parallel* way. This means that reactants are assigned to rules until no further assignments are possible, which imposes the consequence that a single rule may assume exemplarity of multiple cluster instances, all of identical structure, where the members of each instance have a one-to-one correspondence to the elements in the left-hand side multiset in the forward reaction form, *a*.

This reformulated clustering problem can be solved by constructing a factor graph as the one shown in Fig 3. In this graph, we assume that the system contains *N* reactants and *R* reaction rules, and the goal is to match subsets of these *N* reactants with reaction rules such that each subset (or reactant group) is congruent with the left-hand side multiset of the reaction it is matched with (the multiset *a* in the forward reaction form), and no single reactant is involved in more than one reaction. One important practical consideration to note is that even though multiple instances of the same reactant type might exist simultaneously in the system, their inclination to partake in the same reaction may differ due to application semantics. Similarly, if the left-hand side multiset of a reaction consists of two reactant types, r_1 and r_2, where multiple instances exist of the former but only one exists of the latter, the different r_1 instances are not to be considered interchangeable, but rather, the one with the highest affinity to the r_2 instance should be picked if possible. This is important because it might not be preferable for the physical entities that correspond to the reactant instances to communicate with each other for any reason, such as being physically separated by a large distance. Because of this, the input provided to the algorithm consists of similarities between reactants and reaction rules as well as reactants and each other, where the latter is based entirely on application semantics, while the former is based additionally on whether the reactant appears in the left-hand side multiset of the rule. If it does not, then the two are incompatible and their similarity is set to $-\infty$. In this scheme, self-similarities (or preferences) have no significance for reactants, whereas for reaction rules, they represent the context influence.

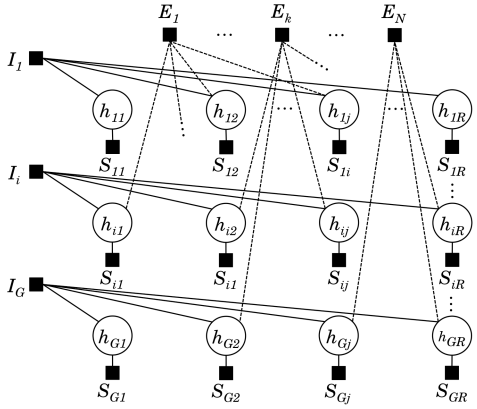

Figure 3. Factor graph of reformulated clustering problem for executing reactions in CAL.

As illustrated in Fig 3, the factor graph is 2-dimensional and is constructed such that reaction rules are organized horizontally and are indexed by the variable $j \in \{1..R\}$, while

symbols representing reactant groups are organized vertically and are indexed by the variable $i \in \{1..G\}$. Reactant groups represent the elements of the power set over all reactants, which has a size of 2^N. Obviously, this can be prohibitively expensive. In practice, however, reactant groups that don't appear in any reaction rules as well as those that don't satisfy a minimum similarity threshold among its members can be eliminated, reducing the size of the set by several orders of magnitude. Similar to the original AP formulation, assigning a value of *1* to h_{ij} indicates that the reactant group *i* is to be consumed by reaction rule *j*. Due to the fact that a single reactant may belong to multiple groups, if group *i* is assigned to a rule (i.e. $\exists\ j$ s.t. $h_{ij} = 1$), we must guarantee that all reactants in group *i* are not assigned to any other rule. This is achieved by adding the *single-assignment constrain*t factors (labeled $E_{\{1..N\}}$ in the graph), one for each reactant, which collectively guarantee that if $h_{ij}=1$ then all other variables that appear in rows corresponding to groups that share any reactants with *i* are set to *0*. Finally, the I_i factors represent the *1-of-N constraint*, which plays the same role as in the standard AP problem.

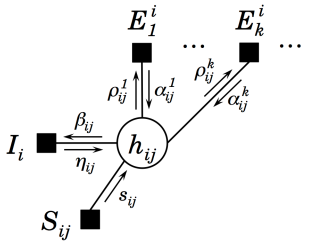

Figure 4. Factor graph fragment for computing the affiliation of reactant group *i* with reaction rule *j*.

A fragment of the factor graph for computing a single variable is shown in Fig 4, which is similar to that of AP (Fig 2) except that the exemplar consistency constraint E_j is now replaced by a varying number of single-assignment constraints E_k^i (the notation stands for the constraint in $E_{\{1..N\}}$ that corresponds to the k^{th} reactant in group *i*). The number of single-assignment constraints imposed on each h_{ij} variable is equal to the number of reactants in group *i*, and each constraint is connected to all other $h_{gr|\,(g,r)\neq(i,j)}$ where reactant group *g* includes the reactant associated with the constraint. As shown in (2) and (3), the *1-of-N* constraint is satisfied only if each reactant group is assigned to at most one reaction rule, while the single assignment constraint is satisfied if no more than one of the h_{ij} variables linked to the constraint is set to *1*. The objective function in (4) is similar to that of AP, where the sought solution maximizes the sum of similarities between reactant groups and reaction rules while satisfying all constraints.

$$I_i(h_{i:}) = \begin{cases} 0 & : \ if\ \sum_{j=1}^{R} h_{ij} \le 1 \\ -\infty & : \ otherwise \end{cases} \quad (2)$$

$$E_k(h_{:::}^k) = \begin{cases} 0 & : \ if\ \sum_{(i,j)^k} h_{ij}^k \le 1 \\ -\infty & : \ otherwise \end{cases} \quad (3)$$

$$\mathcal{F}\left(\{h_{ij}\}\right) = \sum_{i=1}^{G}\sum_{j=1}^{R} s_{ij}h_{ij} + \sum_{i=1}^{G} I_i(h_{i:}) + \sum_{k=1}^{N} E_k(h_{:::}^k) \quad (4)$$

Messages passed between nodes in the modified graph have the same semantics as in the original AP formulation. However, due to the varying number of constraints imposed on each variable and their slightly different roles in our problem, the equations for computing the values of these messages must be rederived. The binary formulation of AP simplifies this process, as it lets us derive the formula for each message by calculating its value for each setting of the hidden binary variable h_{ij} then taking the difference [11]. We now discuss the derivation of the five message types shown in Fig 4:

- s_{ij} : this message is independent of h_{ij} and it represents the initial affinity between reactant group *i* and reaction rule *j*. If reactants of group *i* match those of rule *j*, then the value of the message is set to the average similarity between each reactant in the group and the reaction rule, multiplied by the preference value of the rule. Otherwise, the pair is not a candidate for matching and the message is assigned a value of $-\infty$. In the former case, the average is taken as a normalization mechanism for countering the fact that the number of reactants (and subsequently constraints) may vary from one reactant group to another. Multiplying by the preference value is used as a means for adjusting the rate at which the reaction takes place as a result of contextual influence.
- β_{ij} and η_{ij} have the same role as in AP, which is to enforce the *1-of-N* constraint, and thus their formulas are unchanged, except that the average over the α messages is taken for the same reason explained earlier.
- α_{ij}^k : this message represents the accumulated evidence of how appropriate it would be for the k^{th} reactant in group *i* to partake in reaction rule *j*. To calculate its value, we fix the value of h_{ij} to *1* or *0*, then we find an optimal assignment for the other variables associated with the same reaction. When $h_{ij} = 1$, this means that the k^{th} reactant (and all of group *i*) is assigned to rule *j*, and thus all other variables that correspond to reactant groups that contain the same reactant must be set to *0*, yielding $\alpha_{ij}^k(1) = \text{avg}_{(g,r)^k}\ \{\rho_{gr}^k(0)\}_{(g,r)\neq(i,j)}$. For $h_{ij} = 0$, the other variables are unconstrained and the maximum over both configurations should be taken, resulting in $\alpha_{ij}^k(0) = \text{avg}_{(g,r)^k}\ \{\max_{h_{gr}} \rho_{gr}^k\left(h_{gr}\right)\}_{(g,r)\neq(i,j)}$. We obtain α_{ij}^k by taking the difference between $\alpha_{ij}^k(1)$ and $\alpha_{ij}^k(0)$ where we end up with the formula in (8), in which we have relied on the fact that $x - \max(x,y) = \min(0, x-y)$.
- ρ_{ij}^k : this message represents the accumulated evidence for how well suited reaction *j* is to consume the k^{th} reactant in group *i*, and it has a similar formula as that of AP but takes into consideration the availabilities of the other reactants involved in the group. A summary of the formulas for all messages is given in (5) – (9):

$$s_{ij} = \begin{cases} p_j \cdot \underset{k}{\text{avg}}\left\{sim\left(k^i, j\right)\right\} & : \ if\ i \equiv rule\ j \\ -\infty & : \ otherwise \end{cases} \quad (5)$$

$$\beta_{ij} = s_{ij} + \text{avg}_{k^i}\{\alpha_{ij}^k\} \quad (6)$$

$$\eta_{ij} = -\max_{r\neq j} \beta_{ir} \quad (7)$$

$$\alpha_{ij}^k = \text{avg}_{(g,r)^k}\{\min\left(0, \rho_{gr}^k\right)\}_{(g,r)\neq(i,j)} \quad (8)$$

$$\rho_{ij}^k = s_{ij} + \eta_{ij} + \text{avg}_{r^i}\{\alpha_{ij}^r\}_{r\neq k} \quad (9)$$

When the algorithm terminates, decoding the final solution is straightforward. Reaction rules with any variables set to *1* in their respective columns are considered to have been triggered. Those reactions are denoted by *j* s.t. $\sum_{i=1}^{G} h_{ij} \ge 1$. Similarly, a

reactant group *i* is considered to have been consumed by a rule if one of the variables in its row is set to *1*, i.e. $\sum_{j=1}^{N} h_{ij} > 0$. Individual reactants within each such group are substituted by a molecule that represents the whole group in the system, while all other reactants are remain unaffected.

IV. EVALUATION

We now provide a number of examples where we put the concepts behind C_2A_2 to use. This serves to elucidate how such concepts can be applied in a concrete application and also to gauge their feasibility and efficiency in a practical setting.

A. Clustering in Wireless Sensor Networks

The first application we use to present C_2A_2 is clustering in wireless sensor networks (WSNs). In WSNs, clustering is often used to increase the longevity of the network and ensure balanced resource utilization. In its most basic form, WSN clustering involves just two parameters: node location and residual energy. In this model, each node is represented as an atom with an affinity profile that specifies its affinity toward other nodes in the network. The affinity profile can be sparse, where missing values are taken to mean $-\infty$, indicating that the two nodes can never be in the same cluster (due to their being too far from each other, for instance). Messages exchanged between nodes are calculated using the equations (10) – (13), which are a slight variation from the original AP formulas:

(a) Similarities between each pair of nodes are calculated according to (10), which in this model is designed to minimize the sum of squared error (i.e. the distance between each node and its exemplar).

(b) Availability of a node *i* to serve as an exemplar for another node *j* is initialized as shown in (11), which takes into consideration the exponential increase in communication cost as the distance between the two nodes increases. In subsequent iterations, availabilities are updated using (12) which takes into account the accumulated evidence that node *j* should be chosen as an exemplar in addition to the positive responsibilities received by *j* from other nodes.

(c) Responsibility of a node *j* to serve as an exemplar for a node *i* (from the latter's point of view) is calculated using (13), which relies on the similarity between the two nodes but also taking into consideration the suitability of *i* to be chosen as an exemplar by other nodes.

$$s(i,j) = c_1 \cdot distance(i,j)^2 \quad (10)$$

$$a_o(i,j) = c_2 \cdot \frac{residual\_power(i)}{distance(i,j)^2} \quad (11)$$

$$a(i,j) = min\{0, r(j,j) + \sum_{k:k\notin\{i,j\}} \max\{0, r(k,j)\}\} \quad (12)$$

$$r(i,j) = s(i,j) - max_{k:k\neq j}\{a(i,k) + s(i,k)\} \quad (13)$$

In this experiment we used ns-2 [8] to compare our results to those obtained using two widely used WSN clustering protocols; generalized LEACH [4] and Average Minimum Reachability Power (AMRP) HEED [5]. The simulation parameters are identical to those used in [5], and the experiments were carried out using ns-2 [8]. The sensors are deployed randomly in a *100×100* m^2 area with a base station located at *(50,175)*. During each TDM frame, each node collects data and sends a data packet to its cluster head, and each cluster head fuses these packets into one message and sends it to the base station. Clustering is triggered at the end of each round, where a round is *5* TDM frames.

The simulation results are shown in Fig 5, which shows the ratio of the total number of messages to the total number of nodes. The ratio seems to follow a logarithmic curve, where the overhead is inversely proportional to the number of nodes, approaching an asymptote of *1*. The effect of this on network longevity is illustrated in Fig 6, where WSN clustering using C_2A_2 can be seen to outperform both LEACH and HEED clustering algorithms in terms of the number of clustering rounds that can be performed under identical conditions.

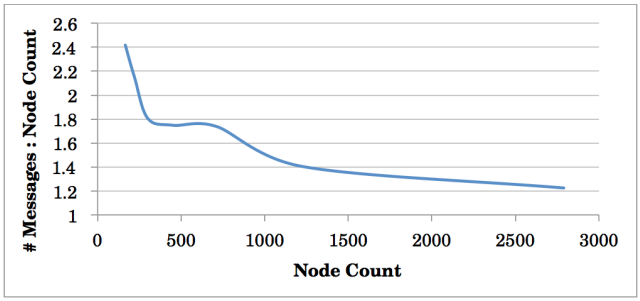

Figure 5. The ratio of the total number of messages exchanged during each clustering round is inversely proportional to the number of nodes with an asymptote of *1*.

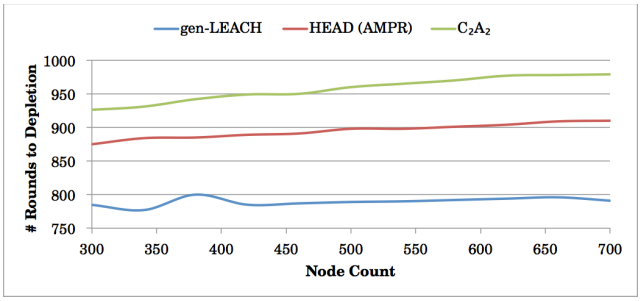

Figure 6. Number of clustering rounds until power depletion for different deployment sizes.

B. Constrained Task Assignment

The next case study we utilize to demonstrate C_2A_2 is the problem of constrained task assignment in a network of heterogeneous nodes, which has been formulated in several works in the literature, such as [6] and [7]. The problem can be stated as follows: given a set of tasks *T* and a set of nodes *N*, where each task $t \in T$ is associated with a demand vector *d(t)*, and each node $n \in N$ is associated with a supply vector *p(n)*, it is required to find an assignment $M(N \rightarrow T)$ such that:

(a) For each task $t \in T$, *M* must contain an assignment associating *t* with a set of nodes whose aggregate supply satisfies *d(t)*.

(b) The assignment *M* should be selected such that the objective function $f = \sum_{t\in T} w(t) - \sum_{m\in M} c(m_{n\rightarrow t})$ is maximized, where *w(t)* is the reward gained by the system for executing task *t*, and $c(m_{n\rightarrow t})$ is the cost incurred by the system for assigning node *n* to task *t*.

This problem is known to be reducible to the sub-graph isomorphism problem, which is NP-Complete, and therefore finding an optimal solution (denoted henceforth as f^*) could be prohibitively expensive for large inputs. However, the problem

 ISBN: 978-1-61208-389-6

is particularly suited for C_2A_2 since it is aligned with the system's ability to reconfigure network resources into separate taskforces. The goal is to approximate f^* better than typical greedy solutions while keeping the cost of obtaining the solution within the same order of magnitude. We would like each taskforce to be represented in CAL as a molecule that has an atom (corresponding to a task) in its center with bonds linking it to a number of atoms representing the nodes assigned to the task. The formation of such bonds depends on the mutual affinity between a task and a node, which is based on matching the demand and supply vectors, and gauging the reward gained and the cost incurred by the system. It is important to note that a single node atom might be involved in two molecules, which could happen if the supply vector of the node can satisfy the demand of more than one task. The AP parameters we use to achieve this can be calculated as follows:

(a) To ensure that bonds are only created between task atoms and node atoms (and not between two tasks or two nodes), the similarity between two tasks or two nodes is set to $-\infty$, indicating absolute dissimilarity.

(b) To guide bond formation such that task atoms are always in the center of molecules and node atoms are always peripheral, the preference (or self-similarity) of a task atom is set to $+\infty$ while that of a node atom is set to $-\infty$ (as mentioned previously, the self-similarity parameter determines the willingness of a data point to become an exemplar).

(c) Since we want all task atoms to be exemplars and all node atoms to be followers, the responsibility of a task atom and the availability of a node atom are both set to $-\infty$. The justification of this is that no atom should be *responsible* – in AP terms – toward a task atom, since the latter is always an exemplar. Similarly, a node atom is never *available* to any other atom, since it should never be an exemplar.

(d) Similarity between a node and a task is based on the compatibility between their respective supply and demand vectors, as reflected in (14), which also considers the reward associated with the task.

(e) Availability messages sent from a task to a node are initialized according to (15), which takes into account the cost associated with the node in addition to its ability to satisfy the demand of the task.

(f) Availabilities and responsibilities are updated using the same equations shown in (12) and (13), respectively.

$$s(n,t) = c_1 \cdot w(t) \cdot |\{i : p(n)[i] \geq d(n)[i]\}| \qquad (14)$$

$$a_o(n,t) = c_2 \cdot \frac{|\{i : p(n)[i] \geq d(n)[i]\}|}{c(m_{n \to t})} \qquad (15)$$

The efficiency of our task assignment solution in approximating f^* is shown in Fig 7, where the obtained results are compared to the optimal solution as well as a greedy solution for different combinations of node and task counts. In an average of *100* runs, our approach consistently outperforms the greedy solution, and is a very close approximation of f^*. Moreover, the average number of AP iterations required was below *10* in our experiments, which means that the improvement over the greedy solution does not come at the expense of a much higher time complexity, as seen in Fig 8 which shows a logarithmic scale of the time complexity of the three methods. Our solution is slightly more costly that the greedy solution, but lies within the same order of magnitude. Both algorithms, however, are significantly faster than the optimal one.

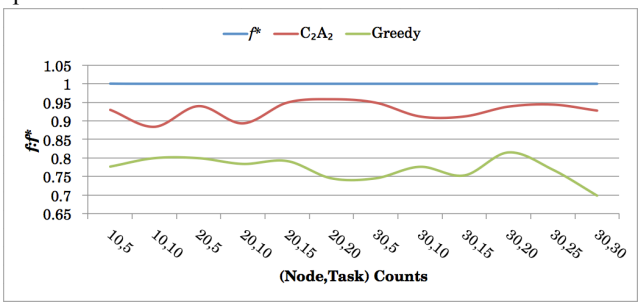

Figure 7. Normalized f^* vs. C_2A_2 and greedy approximations.

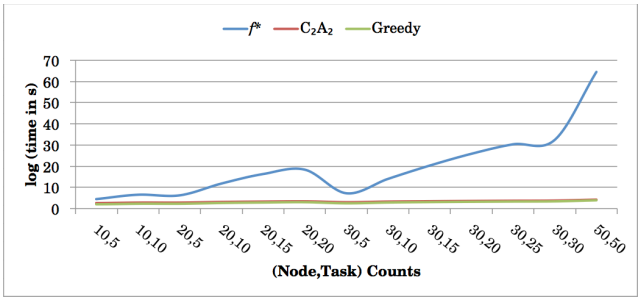

Figure 8. Time complexity (logarithmic scale).

V. CONCLUSION

In this paper, we presented C_2A_2, a chemistry-inspired, context–aware, and autonomic management system networked objects. We proposed a framework for implementing a pervasive computing environment built around the chemical affinity theory. We presented an abstraction through which network components can be mapped to the chemical domain, allowing us to carry out several network operations by simulating the interaction model that takes place between atoms during a chemical reaction. We introduced the concept of dynamic context-aware affinity profiles, which govern the behavior of individual system components, ensuring adaptability in response to context changes and other interesting events. We also extended and repurposed the Affinity Propagation clustering algorithm as a reaction execution engine in our Chemical Abstraction Layer (CAL), allowing distributed exchange of affinities among individual nodes while steering them toward convergence on a common goal. We used simulation to verify the efficacy of our approach using the problems of clustering and constrained task assignment in WSNs. As a natural extension to this work, we are exploring the utilization of reinforcement learning techniques and exploratory self-adaptation, where the system associates past decisions with the monitored effect on performance, thereby allowing the system to self-optimize in anticipation of potential events expected to take place in the future.

REFERENCES

[1] Berry, G. and G. Boudol (1992). "The Chemical Abstract Machine." Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. POPL '90.

[2] Păun, G. (2003). "Membrane Computing." Fundamentals of Computation Theory.
[3] Brendan Frey and Delbert Dueck (2007). "Clustering by passing messages between data points." Science.
[4] Heinzelman, W.R. and H. Balakrishnan (2000). "Energy-Efficient Communication Protocol for Wireless Microsensor Networks." Proceedings of the 33rd Annual Hawaii International Conference on System Sciences.
[5] O. Younis, S. Fahmy, and P. Santi (2004). Robust communications for sensor networks in hostile environments. Proceedings of the 12th IEEE International Workshop on Quality of Service (IWQoS).
[6] Zhao, B., M. Wang, et al. (2008). "Topology-Aware Energy Efficient Task Assignment for Collaborative In-Network Processing in Distributed Sensor Systems." Distributed Embedded Systems: Design, Middleware and Resources.
[7] Xu, S. (2010). "Energy-efficient task assignment of wireless sensor network with the application to agriculture." Iowa State University Digital Repository.
[8] Network Simulator (ns-2): http://nsnam.isi.edu/nsnam/index.php/User_Information
[9] Berryman, A. A. (1992). "The Origins and Evolution of Predator-Prey Theory." Ecology 73, 5 (October): 1530-1535.
[10] Gillespie, D. T. (1977). "Exact Stochastic Simulation Of Coupled Chemical Reactions." The Journal Of Physical Chemistry 81, 25: 2340–2361.
[11] Givoni, I. (2012). "Beyond Affinity Propagation: Message Passing Algorithms for Clustering." PhD Thesis, University of Toronto.
[12] Pearl, J. (1988). "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference."
[13] Kschischang, F.R. et al. (2001). "Factor Graphs and the Sum-Product Algorithm." IEEE Transactions on Information Theory 47, 2 (February): 498-519.
[14] Dueck, D. (2009). "Affinity Propagation: Clustering Data by Passing Messages." PhD Thesis, University of Toronto.
[15] Drezner, Z. (1995). "Facility location: a survey of applications and methods." Springer Verlag.
[16] Cormen, T. H. el al. (2001). "Introduction to Algorithms, Second Edition." The MIT Press.
[17] Shamaiah, M. et al. (2011). "Distributed routing in networks using affinity propagation." IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011: 3036-3039.
[18] J.-P. Banâtre et al. (1988). "A Parallel Machine for Multiset Transformation and its Programming Style." Future Generation Computer Systems 4(2): 133-144.
[19] Bergstra, J. and I. Bethke (2002). "Molecular Dynamics." The Journal of Logic and Algebraic Programming. 51: 193–214.
[20] Viroli, M. and M. Casadei (2009). "Biochemical Tuple Spaces for Self-organising Coordination." Proceedings of the 11th International Conference on Coordination Languages and Models (COORDINATION 2009). 5521: 143-162.
[21] Kreyssig, P. and P. Dittrich (2011). "On the Future of Chemistry-Inspired Computing." *Organic Computing — A Paradigm Shift for Complex Systems. C. Müller-Schloer, H. Schmeck and T. Ungerer, Springer Basel. 1: 583-585.*
[22] Rajcsányi, V. and Z. Németh (2012). "The Chemical Machine: An Interpreter for the Higher Order Chemical Language." *Euro-Par 2011: Parallel Processing Workshops.*
[23] Forgy, C. (1982). "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem." *Artificial intelligence. 19.*
[24] Gillespie, D. T. (1977). "Exact Stochastic Simulation Of Coupled Chemical Reactions." *The Journal Of Physical Chemistry 81, 25: 2340–2361.*
[25] Gibson, M. and J. Bruck (2000). "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels." *The Journal of Physical Chemistry. 104: 1876-1889.*
[26] Slepoy, A., et al. (2008). "A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks." *The Journal of Chemical Physics 128(20).*
[27] Pianini, D., et al. (2013). "Chemical-oriented Simulation of Computational Systems with Alchemist." *Journal of Simulation. 7: 202-215.*
[28] Monti, M., et al. (2013). "Chemistry-Inspired Algorithm for Emergent Distributed Consensus in WSNs." *Distributed Computing in Sensor Systems, 2013 IEEE International Conference on.*
[29] Meyer, T. (2010). "On Chemical and Self-healing Networking Protocols." *PhD Thesis. University of Basel.*
[30] Monti, M., et al. (2013). "Stability and Sensitivity Analysis of Traffic-Shaping Algorithms Inspired by Chemical Engineering." *Selected Areas in Communications, IEEE Journal on. 31: 1105-1114.*
[31] Viroli, M., M. Casadei, et al. (2011). "Spatial Coordination of Pervasive Services through Chemical-Inspired Tuple Spaces." *ACM Transactions on Autonomous and Adaptive Systems 6(2).*

 ISBN: 978-1-61208-389-6