Self-Adaptation in Collective Self-Aware Computing Systems

Jeffrey O. Kephart, Ada Diaconescu, Holger Giese, Anders Robertsson, Tarek Abdelzaher, Peter Lewis, Antonio Filieri, Lukas Esterle, and Sylvain Frey

Jeffrey O. Kephart
IBM Thomas J Watson Research Center, 1101 Kitchawan Rd., Yorktown Heights, NY 10598, USA, e-mail: kephart@us.ibm.com

Ada Diaconescu
Telecom ParisTech, CNRS LTCI, Paris Saclay University, 46 Rue Barrault, Paris, 75013, France
e-mail: ada.diaconescu@telecom-paristech.fr

Lukas Esterle
Vienna University of Technology, Treitlstrasse 3, 1030 Vienna, Austria, e-mail: lukas.esterle@tuwien.ac.at

Holger Giese
Hasso-Plattner-Institut für Softwaresystemtechnik GmbH, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam
e-mail: holger.giese@hpi.de

Anders Robertsson
Lund University, Department of Automatic Control LTH, SE-221 00, Lund, Sweden
e-mail: Anders.Robertsson@control.lth.se

Tarek Abdelzaher
University of Illinois at Urbana-Champaign, Computer Science Department, 201 N. Goodwin Ave, Urbana, Illinois 61801, USA
e-mail: zaher@illinois.edu

Peter Lewis
School of Engineering and Applied Science, Aston University, B4 7ET, Birmingham, UK
e-mail: p.lewis@aston.ac.uk

Antonio Filieri
Imperial College London, Department of Computing, 180 Queen's Gate,SW7 2AZ, London, UK
e-mail: a.filieri@imperial.ac.uk

Lukas Esterle
Vienna University of Technology, Department of Computer Engineering, Treitlstrasse 3, 1040 Wien, Austria
e-mail: lukas.esterle@tuwien.ac.at

Sylvain Frey
Lancaster University, Computing and Communications Department, LA1 4WA, Lancaster, UK
e-mail: s.frey@lancaster.ac.uk

Abstract The goals of this chapter are to identify the challenges involved in self-adaptation (including learning and knowledge sharing) of multiple self-aware systems (or system collectives). We shall discuss the techniques available for dealing with the challenges identified (e.g. algorithms for conflict resolution, collective learning, negotiation protocols, etc.), and which are appropriate given assumptions regarding the collective system architecture. We refer to notions of knowledge, learning and adaptation; various self-awareness levels; and reference scenarios introduced in chapter 1.5.

1 Introduction

Whereas Chapter 5.1 dealt with issues of learning and adaptation by individual self-aware entities, the purpose of this chapter is to explore challenges, opportunities, and methods that arise in the context of learning and adaptation by collectives consisting of multiple self-aware entities. In other words, we treat issues of learning, adaptation and self-awareness at the system level. Nevertheless, we are interested in learning mechanisms at at least two levels of abstraction: in the individual systems which compose the collective, and at the level of the collective itself. An important question concerns how the structure of distributed knowledge, and local learning and adaptation, can affect and give rise to global learning and adaptation behavior. We elaborate upon reference scenarios of Chapter 1.5 to explore issues of learning, adaptation and self-awareness at the collective system level in both cooperative and competitive settings. For a focus on architecture in collectives of self-aware systems, including assumptions relating to modes of interaction and relation within the collective, please see Chapter 2.2.

According to the terminology introduced in Chapters 1.2 and 2.2, we focus on collectives that are self-aware at the level of individual entities, and which may or may not be self-aware at the level of the collective itself. For the most part, we will assume that there is no entity responsible for coordinating or otherwise managing the collective as a whole in accordance with goals that are described at the level of the collective. In other words, imagine that the entities adapt using the techniques described in chapter 5.1, with no conception of or regard for the existence of other adaptive entities in the system. However, in individual agents, the scope of adaptive or other behavior is clearly enabled or limited by its self-awareness. While this is true of the component members of the collective, it is also the case for the collective itself. When considering the case of collective adaptation therefore, two important questions arise: how does distributed self-awareness enable or limit adaptation at the level of the collective? and, more precisely perhaps: how do different types (cf. Chapter 1.2) and organizations (cf. Chapter 2.2) of collective self-awareness impact on collective adaptation?

Based on the more detailed concepts related to collectives and self-aware collectives introduced in Chapter 2.2, we employ UML hierarchies and UML collaborations to denote collectives of self-aware systems. We also broaden the scope of

these concepts by *not* requiring that direct communication must be present between the involved systems, and we only assume that we want to study the correlation (or anti-correlation) between the systems in the collective. These definitions follow the design taxonomy introduced in [36]. In addition, we distinguish the special case where the correlation observed between systems results from information-sharing during runtime and thus is a form of *coupling*. Therefore, various additional cases can be covered, including local random behavior of systems in a collective with no coordination, direct forms of coordination of systems in a collective using messages, and indirect forms of coordination of systems in a collective based on stigmergy.

As detailed in Chapter 2.2, sometimes there is only a negligible correlation between systems at the considered level of abstraction, such that it is possible to consider the systems involved in the collective as independent. In such cases, the behavior of the collective at the considered level of abstraction is more-or-less the superposition of the behaviors of the separate systems, as long as certain constraints are fulfilled that guarantee that the coupling can be neglected. However, in cases where the design of the collective has to consider that some correlation (usually some coupling) occurs that is not negligible at the considered level of abstraction, this correlation may have either adverse effects, due to the correlation, that have to be mitigated; or expedient effects from which the collective behavior can benefit.

In line with Chapter 2.2 we will discuss the role that self-awareness and the related coordination has with respect to mitigating or exploiting the correlation between the systems of a collective. For this purpose, we will look into the elements of learning, reasoning, and acting processes making-up the LRA-loop, as per the definition of self-aware computing systems in Chapter 1.1. In particular, this will include discussions on how to mitigate or exploit the correlation by choosing the right design for the collectives concerning the self-awareness scope (see Chapter 1.2 and 2.2) and coordination approach (see Chapter 2.2).

This chapter is structured as follows. Each of the next several sections deals with a general class of collective challenge or opportunity. The first subsection of each section describes one or more scenarios, several of which are extensions of scenarios introduced in Chapter 1.5. The focus is upon behavioral phenomena that result from interactions among self-aware entities — many of which are undesirable, but some of which are desirable. In the second subsection of each section, we discuss approaches that may be taken to cope with or eliminate undesirable phenomena. The third and final subsection discusses ways in which desirable collective phenomena might be encouraged and capitalized upon.

In the first few sections, we consider non self-aware (or pre-reflective) collectives in which the entities are individually self-aware, but for which there is no awareness at the level of the collective, and for which no entity is aware of the self-awareness of other entities in the collective. First, in section 2, we discuss interactions among entities whose goals are not in direct conflict, and which act in ignorance of one another's goals, demonstrating situations in which the actions undertaken to reach their individual goals are in conflict. In section 3, we discuss situations in which the goals among entities are in direct conflict, and in which the entities act in ignorance of or despite the goals of other entities. Next, in Section 4, we consider collectives in

which the individual entities learn, treating both the case in which they are unaware of one another's existence and the opposite case, in which they know about (and take into account) one another's existence. Here we explore effects that can occur when multiple self-aware entities are all adapting to their environment and to one another simultaneously. Section 5 then treats situations in which entities coordinate deliberately with one another, either via centralized or decentralized techniques and either based on cooperation or competition relations (Cf. Chapter 2.2). Finally, in Section 6 we summarize the chapter and make some general observations.

2 Actions

In this section, we consider collectives in which the entities are only *locally* self-aware. That is, they possess an awareness of themselves, but they either do not recognize or otherwise do not account for the behavior, state, or self-awareness of other entities in the system. Additionally, we suppose that the individual goals that govern each individual entity's behavior are not inherently in conflict. Through a series of scenarios, we illustrate negative and positive global behaviors that may occur under such conditions. At the end of the section, we summarize our observations regarding these phenomena and the means that may be taken to ameliorate or eliminate undesirable global behaviors, or to capitalize on the desirable ones.

2.1 Scenarios

We consider two domains in which self-aware entities strive to realize goals that do not explicitly conflict, but for which actions taken in service of those goals may cause unintended interactions that result in desirable or undesirable collective effects. First, we describe two scenarios from the cyber-physical system domain (Cf. section 5 of chapter 1.5) involving smart appliances in a smart home. Second, we describe an IT scenario involving the self-aware sorting algorithm (Cf. section 3 of chapter 1.5) plus a power manager.

2.1.1 Smart Appliances

Consider interactions that could conceivably take place between a thermostat and a smart window. The thermostat aims to maintain room temperature within a targeted range, for instance between 21°C to 23°C, while consuming as little power as possible. In the same room, a smart window controller opens the window periodically to maintain air freshness, and it also opens and closes the blinds to regulate room luminosity. The aforementioned behaviors are governed by the home owner's pref-

erences, along with environmental conditions (such as the weather and the presence of people in the home).

The goals of the thermostat and the smart window are not explicitly related, but nonetheless their actions can affect one another. For example, if the smart window is open on a cold day, the thermostat may struggle to maintain the targeted temperature — and even if it succeeds in doing so, the heater it controls may consume much more power than it would have had the window stayed shut. Similar issues may occur if the thermostat controls an air-conditioning device during summer months. For example, if the smart window controller opens the blinds on a sunny day to increase luminosity, this could force the thermostat to choose between consuming extra power to attain the temperature goal, or deliberately falling short on the temperature goal in order to avoid excessive power consumption. On the other hand, there may be other conditions under which the smart window and the thermostat unwittingly help one another accomplish their respective goals. For example, if the smart window opens the blinds to increase luminosity on a cold sunny day, the open blinds may help warm the room, allowing the thermostat to attain its temperature goal with less effort than would have been required had the blinds been closed.

2.1.2 Adaptive Sorting and Power Management

Consider the adaptive sorting service introduced in Chapter 1.5, which strives to perform the sort as quickly as possible without exceeding a preset limit on the amount of CPU that should be used. It uses linear regression to adaptively estimate the number of CPU cycles required per basic sort operation, using observations over the last 5 minutes. Then, it sets the concurrency to a level that is calculated to keep the CPU usage in the neighborhood of 90%. Consider as well the adaptive power management algorithm introduced in Chapter 5.1, embodied as a service. It allows a server to use power up to a pre-defined limit, above which it uses a feedback control mechanism to reduce the chip frequency to a value that is just below that limit.

When taken individually, the sorting service and the power manager are using reasonable approaches to controlling the system, ensuring that it operates efficiently and stably. Each goal is quite reasonably trying to achieve a balance between accomplishing work and reducing resource consumption. However, consider what may happen when the power management algorithm controls the server on which the sorting service is running. Suppose that the power manager detects a slight exceedance in the power, and adjusts the chip speed downward. After the chip has been operating at a slower speed for a while, the sorting service will notice a decrease in the rate at which sorting operations are performed, and conclude that the number of operations required per sort has increased. To obtain more resource for sorting, the sorting service adjusts the concurrency level downward. With the decreased concurrency, there is less demand placed on the server, whereupon the power manager decides that it can raise the chip speed. The increased chip speed increases the rate at which sorting operations are performed, causing a reversal of the logic that was used to decrease concurrency, and the sorting service now increases concurrency. This in

turn may cause an exceedance of the power limit, beginning the cycle anew. A very related phenomenon was reported by Kephart et al. [22], who observed spontaneous oscillations generated by an unanticipated feedback loop between a power manager and a performance manager (illustrated in Figure 1).

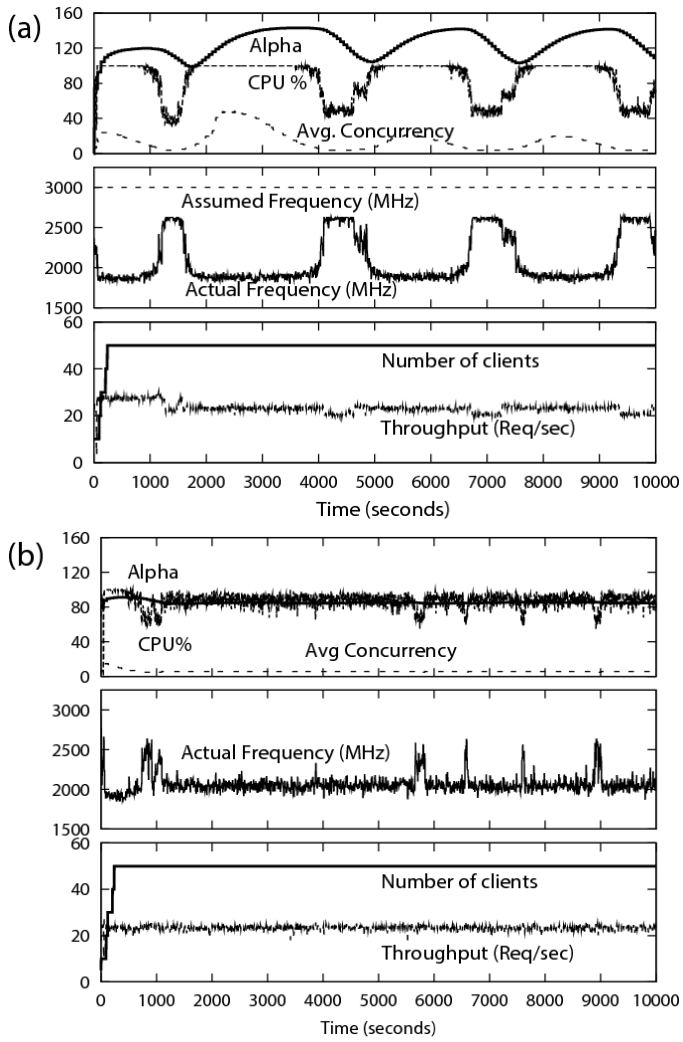

Fig. 1 Effect of CPU frequency feedback on system behavior. (a) WXD receives no feedback (b) WXD receives feedback.

2.2 Mitigating undesirable collective behaviors

The scenarios described in this section exhibited two basic classes of undesirable collective behavior that may occur when two or more self-aware entities attempt to satisfy goals that are not inherently in conflict, but for which actions taken in an effort to satisfy those goals may create inadvertent conflicts or misalignments that result in suboptimal behavior:

1. Conflicting actions that waste resources and/or thwart attainment of one or more individual goals; and
2. Spontaneous instabilities that waste resources and/or thwart attainment of one or more individual goals

We now examine each of these classes in turn, first diagnosing their cause and then (based on that diagnosis) proposing and critiquing various possible approaches to reducing or eliminating these undesirable collective behaviors.

The first class of behavior was exhibited in the scenario involving the thermostat and the smart window. Abstracting from the specifics of that scenario, it is apparent that such phenomena may occur when actions taken by one entity induce an environmental state change that affects the goal of another entity, i.e. the actions taken by the entities are coupled through the impact those actions have on the environment in which both are situated. In more mathematical terms, an action a_e taken by entity e causes the system state S to evolve to state S', and the difference between S and S' matters from the perspective of a second entity ε. In that particular example, when the smart window controller opens the blinds to increase luminosity, the act of opening the blinds causes not just the luminosity to increase, but it also has the side effect of causing the temperature to rise. To the thermostat, the increased luminosity is of no consequence — that aspect of the difference between S and S' does not matter to it — but the increased temperature *does* matter, as the thermostat's goal concerns temperature.

The underlying cause of the conflict is that the smart window controller is unaware of two important facts: a) opening the blinds affects temperature; and b) temperature affects the behavior of another adaptive entity. Note that the situation is asymmetric, i.e. the action taken by the thermostat to increase or decrease heating or cooling does *not* affect luminosity, the variable of interest to the smart window controller. Thus the thermostat's lack of awareness of the existence of the smart window controller and its interest in the luminosity of the room does not contribute to the problem.

What can be done to mitigate this class of collective behavior? In general, an entity A whose actions create a state change that matters to some other entity B must first of all become aware of its impact upon B, and second modify its behavior in some way that uses that awareness to reduce or eliminate the conflict. For the thermostat scenario, the smart window controller must somehow become aware that its actions affect temperature, and that temperature matters to some other adaptive entity in the system, and moreover it must somehow change its behavior to take this new awareness into account.

Many general approaches can be contemplated, some of which we enumerate below. For clarity, all are expressed in terms of the thermostat/smart window scenario, but the generalization should be readily apparent:

1. **Joint control**. Anticipating that they will be used in conjunction with one another, a joint controller is designed to manage the behavior of the smart window and the thermostat (or perhaps the thermostat can be eliminated entirely). The user specifies joint luminosity and temperature goals, along with any tradeoffs that may be needed. The joint controller can be situated in either appliance — in fact it could be designed into both, and if both the smart window and the thermostat happen to operate in the same room then one of the joint controllers can voluntarily turn itself off and let the other take control. The joint controller can also be placed in a dedicated device, which monitors and inhibits other devices when their actions risk causing conflicts. From an architectural perspective, this approach corresponds to a *Hierarchy pattern* where devices implement a *cooperation relation* with the joint controller (as discussed in chapter 2.2). A valid criticism of this approach is that such conflicts may be difficult to envisage at design time because the heater and window operate a priori in different domains – temperature versus air freshness and luminosity, respectively. Similar issues may occur if an air-conditioning device is added in the summer, as the window may open the blinds on a sunny day to increase luminosity, which in turn will also cause the temperature to rise.
2. **Distributed control with derived individual goals**. Anticipating that the smart window may be used in an environment where a thermostat is present, the smart window controller is designed to take into account data provided by temperature sensors located in the room in which it operates. The user specifies joint luminosity and temperature goals, along with any tradeoffs that may be needed. The joint goals are transformed into a set of derived goals or policies to be followed by the smart window controller, which if followed are expected to produce nearly the same behavior as would be exhibited by a joint controller. The smart window then operates according to these derived goals, which now take temperature into account. Since devices do not communicate with each other directly but only via their impacts on their joint environment (i.e. temperature), this case corresponds to a solution implementing a Stigmergy pattern (as discussed in chapter 2.2). Also, with respect to the relation types identified in chapter 2.2, systems implement a *synergy relation* with respect to their goals, since they have positive effects on each-other's goals yet without being explicitly aware of this. At the same time, systems implement ignorance relations with respect to their knowledge and actions, since they are unaware of each other and do not exchange any direct information to coordinate their knowledge and actions; coordination is provided by-design instead.
3. **Distributed control based upon individual goals derived from global feedback**. The smart window controller is given access to temperature readings as a potentially interesting environmental variable, and provided with moment-by-moment readings of a utility variable that indicates the degree to which the overall joint goal of luminosity plus temperature is satisfied. The smart window con-

troller can then learn an association between its actions, the temperature, and the overall utility, and using that model can modify its behavior to try to maximize the joint utility. As in the case above, this corresponds to a Stigmergy pattern, since devices only react to each-other's actions indirectly, via feedback from the environment. The types of relations they implement are also similar to the previous case. The main difference consists in the increased adaptability (e.g. via learning) of this feedback-based case, with respect to the previous one where goals were hard-coded at design time.

4. **Changing one or both strategies**. In the smart home example, the thermostat may choose to achieve a temperature goal by opening the window shutters during a sunny winter day (hence heating by solar energy and saving energy for optimising a power goal). This, in turn, may conflict with the window shutters luminosity goal, which would require closing the shutters partially to avoid direct sunlight. If the thermostat chose to switch on its power instead, the conflict would be solved, or more precisely avoided. Such strategy adaptation can be performed either at design-time, to avoid conflicts, or at runtime, when the conflict is detected automatically.

In general, the solutions discussed above can be implemented either at design-time, when conflicts are being predicted, or during runtime, in cases where systems are able to detect conflicts dynamically (e.g. [20]) and adapt accordingly.

While joint control can certainly solve the problem of conflicting actions, it is impractical under many conditions. It can be extremely difficult for designers to anticipate all of the possible combinations of controllers that could be co-present in a given environment, and to anticipate the couplings that might occur. A more flexible, decentralised solution can be more suitable for unpredictable open environments, where both the initial execution context and the participating devices may change during runtime. Here as well, the levels and kinds of self-awareness that devices require of each other (chapters 1.2 and 2.2) will depend on the extent to which device discovery and coordination can be predicted at design-time. At the same time, while more decentralisation and higher self-awareness capabilities increase the adaptability of devices and of the entire collective, it also increases the overall system complexity and raises several risks (e.g. more unpredictability or decreased performance). Ideally, design-time solutions should be provided to address aspects that are known and unlikely to change, in order to ensure desirable outcomes and stable behaviours – e.g. important in a smart home scenario; and, self-adaptive solutions with various degrees of self-awareness should be provided to deal with unpredictable aspects – e.g. where a safe, yet perhaps non-optimal solution is better than system failure.

The second class of behavior, spontaneous oscillation, was exhibited in the sorting algorithm/power manager scenario. At a high level of abstraction, one can see that spontaneous oscillation has the same basic cause as the first class, except that the situation is now symmetric rather than asymmetric: each of the entities induce an environmental state change that affects the goal of the other, thereby creating the potential for an infinite cycle. The environmental coupling in the sorting algorithm and power manager scenario occurs, not through a single shared resource,

but instead through two different environmental variables: the sorting algorithm inadvertently affects power consumption by adjusting concurrency, while the power manager inadvertently affects computational speed by adjusting chip frequency. A system with such couplings could be modeled to a first degree of approximation as:

$$\begin{aligned} x'(t) &= \alpha y(t) \\ y'(t) &= \beta x(t), \end{aligned} \tag{1}$$

from which one can derive $y''(t) = \alpha\beta y(t)$. When the product $\alpha\beta$ is negative (as it is in the case of the power manager and the sorting algorithm), $y(t)$ is sinusoidal; when it is positive then the solution is a growing exponential (a positive feedback loop that runs amok until something in the system saturates).

For spontaneous oscillations that occur due to couplings to two different environmental variables, we offer the following set of mitigations:

1. **Breaking the feedback loop** by using any of the methods listed above for mitigating asymmetric resource conflicts, such as joint control or distributed control with derived individual goals. Note that applying such a mitigation to just one of the two couplings may suffice to break the feedback loop, but it may leave the system with the asymmetric goal conflict problem, resulting in suboptimal behavior. In such a case, the mitigation may be applied to the second coupling as well.
2. **Giving one or both of the entities knowledge of the variables through which their actions are coupled**, such that it can update its model appropriately. Such a method was employed by Kephart et al [22] to eliminate the spontaneous oscillation shown in Figure 1. Specifically, the power manager conveyed to the performance manager the chip frequency setting at which it was operating. With this information, the performance manager was able to change its model for the speed at which computations were being performed, enabling it to stop reacting too strongly when the chip frequency was changed. This simple change eliminated the oscillations entirely.

Another case of undesirable synchronisation can occur when multiple entities within a collective react in the same way to the same stimuli detected in their shared environment. For instance, if all devices detect a power consumption peak (e.g. by monitoring the frequency of a shared micro-grid) then they may all react simultaneously to reduce their consumptions and to lower the overall load on the grid. This may in turn cause an abrupt fall in overall consumption and risk a blackout. If devices then detect this lack of consumption and start consuming, oscillations may occur and threaten the grid. Similarly, when all thermostats in a room react independently to temperature fluctuations, undesirable oscillations may also occur. Direct coordination may be employed for addressing this issue when dealing with relatively small numbers of devices. Alternatively, randomising reactions to common events can also be employed when dealing with large-scale collectives. This latter solution has been used [1] for desynchronising electric devices connected to a shared power grid, as discussed above.

2.3 Capitalizing on desirable collective behaviors

In other scenarios, global synchronisation can be a desirable property of a collective system, which can help achieve a global goal collectively. In such cases, system synchronisation helps them achieve goals more efficiently. This is the case for instance in robot swarms where all robots self-synchronise their speeds, and/or their movement directions in order to better achieve some collective goal [33].

In addition, most cases of spontaneous synchronizations can be capitalised upon if some extra design and tuning are introduced to regulate their behaviours. Most often, some form of randomisation that is proportional to the number of entities and dependent on the desired aggregated effect can be introduced to obtain lightweight, highly flexible and scalable self-adaptation solutions.

3 Reasoning & Goals

In this section, we consider the individual goals that drive the reasoning and subsequent actions of self-aware entities, and discuss collective behaviors that arise. As in Section 2, we consider here collectives in which the entities are only *locally* self-aware; that is, they possess an awareness of themselves, but they either do not recognize or otherwise do not account for the behavior, state, or self-awareness of other entities in the system. In contrast to Section 2, however, here we suppose that the individual goals that govern each individual entity's behavior are inherently in conflict, either overtly or indirectly through their mutual need for the same limited resource. collective. This is i) because individual components may themselves differ, for example in terms of capabilities or resources, ii) because they may be in different locations, and hence be subject to different experiences, and iii) because the individual entities may have different domains, processes, accuracy or levels of self-awareness. Therefore, learning, adaptation and knowledge present will all vary between individuals in typical collective systems (e.g. [16, 31]). Through a series of scenarios, we illustrate negative and positive global behaviors that may occur under such conditions. At the end of the section, we summarize our observations regarding these phenomena and the means that may be taken to ameliorate or eliminate undesirable global behaviors, or to capitalize on the desirable ones.

3.1 Scenarios

3.1.1 Heater vs. air conditioner

As a first, very simple example of direct goal conficts, consider the case of two (misconfigured) appliances: a heater that aims to maintain temperature above 24°C,

and an air conditioner that aims to keep the temperature below below 22°C[1]. Several different behaviors might arise in such a situation:

1. If the heater and air conditioner are approximately equal in heating or cooling capacity, and if both operate continually, the system might stabilize at a temperature in between the two set points. Each would labor continually to achieve their goal, wasting enormous amounts of energy in the process.
2. If the heater or the air conditioner are substantially more powerful than the other appliance, and both operate continually, the temperature may stabilize at the set point of the more powerful appliance. As above, tremendous amounts of energy could be wasted in the process.
3. If the heater and air conditioner operate sporadically, each turning off when their set point is met, the system could oscillate: when the heater reaches its goal, it turns off for a while, allowing the air conditioner to start bringing the temperature down. If the air conditioner accomplishes its goal before the heater can turn on, it too will turn off. Then, when the heater turns on, it will once again warm the room — and so the cycle may continue indefinitely, wasting lots of energy. Even if the heater and air conditioner don't act quickly enough to completely reach their set points before the other appliance turns on, significant oscillation (and energy wastage) may occur.

Of course, if one of the appliances were to become aware of the other's goal, it could at least detect the conflict and warn the user about the conflict, in hopes that the user would then rectify the conflict by changing one of the goals.

3.1.2 Dishwasher vs. oven

As a second example of direct conflicts among goals, consider interactions that might take place between a smart dishwasher and a smart self-cleaning oven. The dishes must be washed and the oven must be clean by 9am, and furthermore the cost of electricity consumed by these appliances must be minimized. Due to a constraint on the total amount of power that may be consumed by the smart home, the oven (which consumes up to 3kW) and the dishwasher (which consumes up to 2kW) may not consume more than 4kW in total for a period of longer than 5 minutes (i.e. there is a soft circuit breaker). Each appliance checks every 5 minutes to ensure that its power consumption is not causing the total power consumption for the house to exceed the limit. Suppose further that the cost of electricity is $0.20 for most of the day, but reduced to $0.10 between midnight and 6am.

Now imagine that each appliance pursues its own objectives independently, ignorant of the goals (or even the existence) of the other appliance. In an effort to minimize cost and ensure that their cleaning jobs are done before 9am, both appliances might turn on automatically as soon as the rates go down, at midnight. After 5 minutes, each would sense that it is causing the total power to exceed the limit,

[1] As ridiculous as it may sound, anecdotally such situations have been observed in industrial buildings, and one basic rule in doing energy audits is to check for this type of conflict.

and voluntarily turn itself off. After waiting another 5 minutes, each would sense that the current power consumption of the house plus what they anticipate adding to that consumption would fall under the limit, and so each would turn itself on again. Five minutes later, each would discover that they are causing the limit to be exceeded again, whereupon each would turn itself off. This oscillatory cycle would continue for a while, until one or the other (probably the dishwasher) finally finishes its job, leaving the other to continue uninterrupted. Alternatively, if the warm-up time for each appliance is longer than 5 minutes, neither would ever finish their job. In any event, regardless of whether either or both finish cleaning, the continual cycling would potentially shorten the lifetime of both appliances, cause frequent exceedance of the power limit, and create instabilities in total power consumption that might be problematic for neighboring homes. The instability might even affect the electric grid as a whole. If similar smart appliances are installed in many homes throughout the grid, the common electric utility pricing policy (exacerbated further by a propensity for consumers to leave appliance goals at common factory settings) might trigger synchronized instabilities in smart homes throughout the grid.

On the other hand, one can also envision circumstances under which independently acting appliances might function efficiently with no explicit coordination. Suppose that the goals of the two appliances are exactly as described above, but a small bit of randomness is introduced into the algorithms used to realize those goals, such that the oven waits until 12:02am to turn itself on, while the dishwasher waits until 12:04am. Consider the dishwasher's perspective first. When the dishwasher wakes up and decides whether to turn itself on, it will decide not to do so, because the extra 2kW that it will add will cause the total consumption to be too high. Thenceforth, at 5 minute intervals, the dishwasher will check again and come to the same conclusion — until the oven finishes its job, at which point the dishwasher will find that it can turn itself on. From the oven's perspective, it will reconsider its state at 12:07am and find that it is fine to stay on, as the total power consumption for the house remains below the limit (because the dishwasher has decided not to turn on). A similar thought experiment shows that even if the starting times for the two appliances are not randomized, randomizing the time intervals between decisions will also prevent disastrous synchronous power cycles. If some randomness is incorporated into the algorithms, one cannot predict which appliance will turn on first, but whichever one does so will stay on until finished, whereupon the second will turn itself on and then finish. This is an example of a beneficial spontaneous (or emergent) coordination that could occur among two (or even more) self-aware appliances that have no direct awareness of one another.

3.1.3 Community of smart houses interacting with electric utility

As a third example, we shift our perspective a level up in scale from the smart home. Consider a multitude of smart homes, connected by a power grid owned by a utility that prices energy dynamically according to supply and demand. Each smart home may have a power manager responsible for ensuring that the overall

power consumption of each house does not exceed some limit. As illustrated in previous scenarios, this may be accomplished by curtailing energy use by various appliances. Note that, in its effort to intelligently manage the consumption of power by household appliances, a smart home's power manager has a slight effect upon the overall demand within the grid, and that its level of demand for power in turn affects prices in the grid. Since all homes see and can respond to the same price at the same time, prices and power consumption across all of the homes in the grid are coupled to one another, and one can therefore envision a variety of dynamics that include oscillations in price (price cycles) and power consumption.

As a crude approximation to this coupling, imagine that the aggregate demand for power p is approximately inversely related to the price of power π, while the price π is linearly related to the aggregate power consumption p, i.e.

$$\begin{aligned} p'(t) &= -\alpha\pi(t) \\ \pi'(t) &= \beta p(t), \end{aligned} \tag{2}$$

One can readily see that Eq. 2 is essentially identical to Eq.1, and therefore capable of exhibiting the same cyclical dynamics. Such cycles in price and power consumption may hurt both consumers and the electricity provider, as they introduce extra uncertainty into prediction (and therefore planning).

3.2 Mitigating undesirable collective behaviors

The heater vs. air conditioner scenario exemplifies direct goal conflicts that can result in considerable waste of resources, or even spontaneous instabilities in system behavior. Recovering from such a situation requires that the goal conflicts be detected and then resolved in some way. One method by which self-aware entities could recognize that they are involved in a goal conflict with one or more other entities is to recognize that they are consistently failing to meet objectives, and advertise this fact to other entities in the system, along with some information about the variables or metrics that are not behaving according to expectations. Upon receiving such information, other self-aware entities could check the variables and/or metrics that are most relevant to their function. If there is overlap, the overlapping goals or metrics could be exposed to a human user of the system, or else to some automated authority operating within the system. Once alerted to the conflict, a user may then specify additional goals or preferences that resolve the conflict. One approach to resolving such conflicts is to prioritize some goals higher than others. Another approach is to define a utility function that maps the state (as defined by all of the metrics that matter to the user) to a scalar, in which the system goal is to reach a feasible state that maximizes the utility subject to any constraints that might also be specified. Determining how a self-aware system might exploit models of itself

and its environment to elicit additional goals and preferences that suffice to resolve detected conflicts is a worthy research challenge [30, 28].

In the smart homes and power grid scenario, the periodic oscillations in the price and the usage of electric power result from a cause similar to that which drove the spontaneous oscillations in the sorting algorithm and power manager scenario of section 2.1. In that case, the sorting algorithm and the power manager each induced an environmental state change that affected the goal of the other. Here, the smart homes each adjust their power consumption in reaction to the electricity price set by the utility, while the utility adjusts its price in response to the aggregate power consumption of the smart homes. All of the mitigation mechanisms discussed in section 2.2 apply here. Since the smart homes are a collective rather than a single individual entity, some additional mitigations are possible. For example, the smart homes might collaborate with one another via negotiation or some other mechanism to coordinate with other homes to make the overall consumption more inherently stable, thereby making prices more stable, which in turn results in less susceptibility to consumption oscillations. Flexible houses with consumption reduction and / or storage capabilities might anticipate peak prices and deliberately consume less energy at times when other households are demanding more, resulting in more stability and lower overall payments to the utility. Houses that are less flexible in their energy consumption might still contribute to scheduling by advertising their consumption predictions, thereby enabling the more flexible houses to schedule their power consumption to avoid oscillations.

In the dishwasher and oven scenario, we also observed spontaneous oscillations. However, the underlying cause is of a different nature, and therefore the mitigation strategies are necessarily different. The environmental coupling in the sorting algorithm and power manager scenario occurs through two different environmental variables: the sorting algorithm inadvertently affects power consumption by adjusting concurrency, while the power manager inadvertently affects computational speed by adjusting chip frequency. However, in the case of the dishwasher and oven, there is a single environmental variable through which their actions are coupled — a shared resource of which there is a limited supply, electric power.

This situation is closely related to *computational ecosystems*, large-scale, distributed, decentralized computing systems composed of agents that each require a specific type of resource for their operation. Each agent uses exactly one resource at any given moment in time, and asynchronously and independently reconsiders which to use, based upon an expected payoff that depends on their belief about the current usage of that resource by other agents. An agent's belief about the current usage of each resource may not be entirely correct, due to inherent uncertainty and/or time delays. Kephart et al. [26, 27] showed that, in the limit of large numbers of agents, the dynamics of resource consumption in computational ecosystems could be modeled as a differential-delay equation. If the uncertainty is sufficiently large and/or the delay is sufficiently low, the solution to the differential-delay equation is damped oscillations that settle to a fixed equilibrium, but for small uncertainty and/or large delays the solution may be persistent oscillations, possibly quite complex — even chaotic — in nature. The dishwasher-oven scenario involves only two

agents, and for that case the electricity consumption is best modeled as a difference equation, but the principles and the dynamics are very similar.

For spontaneous oscillations that arise from constraints on shared resources, some possible approaches to stabilizing the system include:

1. **Introducing randomness into the actions of the entities**. For computational ecosystems, Kephart et al. [26] showed that introducing randomness into the decision about which resource to use (modeled as softening the decision function for choosing a resource as a function of the usage of all resources) could eliminate oscillations at the expense of shifting the stable operating point to one that is less optimal globally.
2. **Introducing heterogeneity into the goals of the entities**. For computational ecosystems, Kephart et al. [26] also showed that if the agents' resource needs are heterogeneous then the ecosystem as a whole is much less vulnerable to spontaneous instability; in effect, the agents settle into different niches. Such a strategy is only possible in situations in which the goals of the individual agents may be controllable by a system designer. As above, if this causes the goals to differ from those really intended by the designer, then this greater stability is achieved at the cost of suboptimal behavior.
3. **Reducing information delays**. Since this type of spontaneous instability arises from a differential-delay equation, and the susceptibility to such instability tends to increase with the delay, stability can be restored by reducing delays in information regarding resource usage (if it is possible to do so).
4. **Endowing entities with an awareness of other entities, and some ability to approximately predict their behavior.** For computational ecosystems in which some of the entities are able to predict the behavior of other entities (or at least the collective behavior of the system as a whole), Kephart et al. [27, 23] showed that the overall behavior may improve provided that the proportion of predictive agents is small. However, the collective behavior can become even more unstable (and strongly suboptimal) if too large a proportion try to predict the collective behavior of the system and act on that basis.
5. **Introducing a resource broker to resolve resource conflicts**. If entities can not resolve resource conflicts cooperatively, one or more resource brokers can be charged with governing resource usage in the system. Rather than each entity placing a direct demand on resource usage, they submit requests to a resource broker that describe the extent to which they need resource, and the broker decides how much resource to allocate to each such entity. Walsh et al. [38] described a system that allocated compute resources, in which each of several application managers could send to a resource broker utility functions describing the value they would realize if they were to as a function of the amount of resource that they might be granted by that broker, and the broker determined the resource allocation by maximizing a (perhaps weighted) sum of utility functions.

3.3 Capitalizing upon desirable collective behaviors

The scenarios of this section focussed exclusively on undesirable collective effects that may occur when multiple self-aware adaptive entities pursue their own goals without considering adaptive goal-driven behavior by other entities operating within the same environment. However, there are conditions under which no serious conflicts will emerge among the goals pursued independently by the self-aware entities. For example, resource conflicts will not emerge if resources are relatively plentiful. If goals do not explicitly conflict, and the actions undertaken to realize those goals do not result in unanticipated couplings, then they may be pursued independently without negative consequences. In such a situation, the system as a whole benefits from the individual adaptive goal-driven behaviors of the individual self-aware entities from which it is composed, as no explicit coordination is needed.

4 Learning

A defining characteristic of self-aware entities is that they learn models of themselves and/or the environment in which they are situated, and use these models to reason about what actions to take so as to best realize their goals. Consider for example the various self-aware entities that have appeared in scenarios described in this chapter: smart appliances of various kinds, or a house power manager, or a grid power manager. In all of these cases, and quite generally, default design-time settings cannot encompass the full variability of situations in which an agent will find itself during its lifespan. Moreover, it is often the case that system goals will only be provided by a user at run-time, i.e. they are inherently not predictable by the system designer – and therefore some sort of learning will be required in order to determine a sequence of actions or behaviors that best realizes the system's goals. Yet another motivation for endowing self-aware entities with an ability to learn is so that they can avoid suboptimal or unstable behaviors resulting from conflicts among actions or goals (as described in Sections 2.2 and 3.2). In short, for many different reasons, learning is a must.

A variety of learning mechanisms are described in detail in chapter 5.1. The purpose of this section is to explore the impact that learning by multiple individual self-aware entities may have upon the behavior of a collection of self-aware entities that are learning models of themselves and their environments. When multiple self-aware entities are situated within an environment, they typically interact with one another — either directly, or indirectly through the impact their actions have upon the environment. In effect, they form part of one another's environment. Therefore — for better or for worse — whether or not the entities are explicitly aware of one another's existence, their learning algorithms may respond to one another's behavior.

In sections 2 and 3, we considered collectives in which the entities are only *locally* self-aware; that is, they possess an awareness of themselves, but they either

did not recognize or they did not otherwise account for the behavior, state, or self-awareness of other entities in the system. In this section, we first treat learning self-aware entities with such limited awareness of other entities, and then we extend our treatment to agents that are aware of the existence of other self-aware entities, and (in some cases) aware of the fact that those other self-aware entities are learning. While it is possible in some situations to contemplate approaches that model the entire collective as a single learning problem, we give centralized approaches very little consideration here, as they require a global system view [4], which is typically not available to local components. One can approximate a centralized approach in a decentralized setting through the use of joint-action learning [5], but such techniques require strong assumptions about the cooperative nature of the learning problem, or observability of others' actions.

Through a series of scenarios, we shall illustrate negative and positive global behaviors that may occur when agents learn, both for the case where they are not directly aware of other self-aware entities, and for the case where they are directly aware of other self-aware entities. In either case, very interesting dynamics can be created in systems where agents try to learn simultaneously, as they create moving targets for one another. Several interesting questions arise in this context:

1. How do individual learning strategies of agents influence the environment in which they are situated?
2. Conversely, how should the fact that an individual self-aware entity is a member of a collective affect the choice of learning strategies for that entity?
3. Overall, how does the scope of awareness affect this individual-collective interplay?

At the end of the section, we summarize our observations regarding these phenomena and the means that may be taken to ameliorate or eliminate undesirable global behaviors, or to capitalize on the desirable ones. [2]

[2] It is worth noting in passing that data privacy is a very real issue in systems of self-aware learning agents. With sufficient monitoring accuracy and long enough observation periods, accurate models of user activities can be learnt and exploited for commercial and non-commercial purposes. In the same way search engines and social media can intrude in the private life of their users, power utilities would be able to reconstruct their user's life patterns and habits, their usual presence at home and absence periods, the nature and type of their electrical appliances, their usage of these appliances, down to very specific details such as the multimedia contents they are watching — reconstructed via specific consumption patterns, for instance, of a TV set playing a particular movie. Therefore, the question of the scope of learning — which agents learn about which others and under which conditions – and of the dissemination of the produced knowledge is becoming a critical privacy and security matter. This privacy risk related to the development of self-aware systems must be understood from the earliest design phases. However, as this chapter is concerned with dynamical behaviors, we shall not pursue these important issues further here.

4.1 Scenarios

For our learning scenarios, we extend the smart home scenarios of sections 2 and 3 at three levels: smart appliances, smart homes, and multiple smart homes connected to a smart grid.

4.1.1 Smart appliances

Here we extend the smart appliance scenarios of sections 2 and 3 to include learning, and explore some issues that may result.

Consider a set of smart appliances operating within a smart home, which may include thermostats and windows in each room, an oven, a dishwasher, and one or more batteries. Each entity is equipped with algorithms and compute resources required for building long-term models. Depending upon the nature of the learning algorithms and the availability of data, such models may range from very elementary to very sophisticated in terms of their complexity and predictive power. For example, a smart thermostat might attempt to learn the temperature preferences and habits of the human occupants of the rooms (perhaps even keyed to each individual). It might further try to learn mappings between the external temperature and the time and effort needed to cool or heat the room. It might even try to augment its models of external temperature by contacting a service to obtain weather forecasts.

The other appliances could exhibit the same broad range in modeling sophistication and awareness of one another's existence and capacity for adapting and learning. A smart dishwasher might learn the consumption patterns of other appliances and schedule its own washing cycles at times when the house's total consumption is expected to be low; a smart battery might learn and anticipate dynamic price variations on energy markets to schedule its load phases to times when prices are low and unload phases when prices are high.

There is also a wide variation in the degree to which the appliances could be aware of one another's existence, and might attempt to develop models of one another's behavior or intent. For example, a smart thermostat that is aware of the existence of other thermostats in other rooms might incorporate into its model the fact that the temperature of the room it is controlling will be affected by not just the external temperature, but also the temperature of neighboring rooms in the house. Given this realization, it would be sensible for the smart thermostat to request from the other thermostats current temperature readings from nearby rooms. Such information would be useful for both adjusting the demands the smart thermostat places upon the heating or cooling system, but also as data that could be used to adapt its model of how the room temperature depends on the external temperature, the temperatures of nearby rooms, and the heating or cooling effort that it demands. Suppose further that the thermostat is aware of the other thermostats, not just as entities capable of reporting temperature, but as controllers of temperature. Then it might request not just the current temperature, but also the anticipated demand that the other thermostats intend to place on the house's cooling or heating system.

An even more sophisticated thermostat might try to learn the models employed by the other thermostats by associating reported temperatures with reported heating or cooling demand, and use this information to anticipate how the other thermostats might respond to its own actions. A yet more sophisticated thermostat might understand that the other thermostats are themselves adaptive and therefore take into account that the models they employ are potentially dynamic. The thermostat might even take into account that its own adaptivity might be anticipated by other thermostats that possess a similarly high scope of awareness, and indeed this can in principle be taken to an infinite level of regress: "I know that *A* and *B* know that I know that they know that I know that .... they adapt and learn."

Self-aware entities that learn are potentially vulnerable to all of the potential pitfalls that have been described in sections 2 and 3. For example, the aggregate demand that thermostats, window controllers, smart ovens, smart dishwashers, and smart batteries place upon electricity consumption may exceed a limit imposed by the user or the utility company, resulting in the resource conflict described in section 3. All of the same mitigations described in section 3.2 may apply. Of particular relevance is the mitigation strategy according to which entities are endowed with an awareness of other entities and an ability to predict their behavior, as behavior prediction could be based upon a learned model. Another particularly relevant mitigation strategy is the one that introduces a resource broker. In the case of a smart home, the resource broker could be a power usage scheduler that takes into account power requests from the various self-aware entities (which are derived from user preferences) and performs some sort of optimization to determine how much and when each appliance may consume power.

Consider what might happen under such circumstances when one or more of the smart appliances has the capacity to learn about the typical daily power consumption profile and reschedule their consumption during low-consumption periods. A typical domestic consumption profile might exhibit peaks in the morning when residents wake up and start their activities, at lunch time, and primarily at night when domestic appliances are used intensively for utilitarian or recreational purposes. In principle, this could enable the smart appliance to optimize its use of power, and if many or all of the smart appliances learn then the home as a whole could operate very efficiently.

However, one can also contemplate scenarios in which learning by one or more smart appliances has undesirable consequences. To take a specific example, the dishwasher agent might well discover through (via reinforcement learning, for example) that its performance is optimized when it over-reports its consumption needs to a house manager that allocates electric power to the various appliances. Other appliances that are more honest about their electricity needs may suffer, and so may the inhabitants too if they find that the dishes are always washed whenever they want, but the room temperature is often too hot or too cold, and the oven is hardly ever cleaned. In other words, an imbalance in which a single smart appliance learns to optimize its own performance may end up violating global objectives that express tradeoffs among tasks performed by the various appliances. Suppose further that one or more smart thermostats and a smart self-cleaning oven also learn that over-reporting their demands secures enough power for them to optimize their perfor-

mance. Then these and the other appliances will benefit at the expense of others that do not learn — just so long as their total demand for power does not exceed the limit. However, if all of the appliances learn to over-report their demands — or enough do so that the aggregated demand for power exceeds the limit — then it is easy to imagine that a disastrous form of co-evolutionary learning [32] could occur. As each appliance ratchets up its resource estimate in an effort to grab more resource for itself, the other appliances must do so as well, resulting in a never-ending arms race; a destructive feedback loop that makes it impossible for the house's power manager to really know the actual resource requirements. Under such circumstances, the house's power manager might itself learn that the appliances are all lying to it, and try to develop its own models of their actual needs based upon their record of actual consumption — and in effect become a central controller that largely or completely ignores the requests made to it by the smart appliances.

Note that the arms race described above could happen without the appliances being aware of one anothers' existence; they would merely be adjusting their actions to maximize their own reward. Now consider what might happen if self-aware entities were to become aware of one another's existence, without trying to deliberately model one anothers' behavior. A scenario such as this was explored by Kephart and Tesauro [29] in the context of two selling agents (pricebots) that use Q-learning to learn pricing policies that govern the price they should charge for a commodity as a function of the price charged by the other selling agent. The policies did not embody a prediction of the other agent's price; it merely expressed the price that a seller should charge given the other's observed price. The dynamics in this case were surprising. Under some conditions, the pricing policies converged to a well-defined symmetric pair. Under other conditions, they appeared to converge to an asymmetric pair of pricing policies, but when viewed at long time scales these policies proved to be unstable — shifting abruptly to a new pair of policies. One can imagine similar behaviors being exhibited by smart appliances that formulate policies governing how much resource to demand as a function of how much has been requested by other appliances.

Now consider the case where agents *do* attempt to model one anothers' behaviors [4]. They could do this by observing one anothers' actions, often in response to their own. In some cases, tagging agents according to their type or their membership in a social group may help, as it enables agents to cluster observations about other agents and thereby potentially reduce the time required to learn models of other agents' behavior. Being able to simulate the likely actions of other agents in the system has the potential to let agents anticipate and potentially avoid oscillations or other unfortunate dynamical collective behaviors. For example, the symmetric price policies learned by the two competing Q-learning pricebots studied by Kephart and Tesauro [29] resulted in pricing dynamics with much shorter (and higher-priced) price war cycles than the naïve policy achieved without learning, resulting in a higher profit for both sellers. As another example, Kephart et al. [27, 23] studied computational ecosystems into which some agents were "smart", i.e. they were endowed with the ability to predict the resource consumption decisions of other agents. Such predictive capabilities could be based upon reinforcement or other forms of

learning. The smart agents experienced gains in utility for themselves, and even sometimes for the agents whose behavior they were predicting. However, when too many of the agents in the system became smart, the system dynamics changed in such a way that the smart agents could no longer predict them, and the resource usage of the system became highly unstable and inefficient.

The field of game theory provides another family of techniques that self-aware entities may use to learn the behavior of other self-aware entities. In fictitious play, agents use the historical frequency of actions taken by other agents as a model for their behavior, and play their best (possibly randomized) strategy against that distribution. Detailed descriptions of NIR and NER and algorithms that exhibit these properties can be found in many references [10, 21]; the basic idea is to use the observed frequency of other agents' actions to converge to a policy (a mapping from an opponent's action to an agent's response to that action) that minimizes the regret that one would feel in hindsight. Depending on the details of the game (the payoffs received by the agents under all possible joint actions), and the details of the learning algorithms, the policies of the players may evolve to various forms of game-theoretic equilibria, such as a Nash equilibrium or correlated equilibria. However, convergence to an equilibrium is not guaranteed. Jafari and Greenwald [21] observed that in the game of "rock, paper, scissors", two agents that both use a no-regret learning algorithm introduced by Hart and Mas-Colell [19] cycle indefinitely among the various strategies of "rock", "paper" and "scissors", and yet interestingly the frequency of their play averaged over time does achieve the Nash equilibrium, in which each strategy is played 1/3 of the time. A similar phenomenon was observed by Greenwald and Kephart [17] who studied a probabilistic pricebot scenario in which between two and five seller agents used NER and NIR algorithms to adapt their pricing strategies. The seller agents did not always settle into a stable mixed (probabilistic) strategy, but their long-term empirical frequency of play did coincide with a Nash equilibrium. In some instances, a very long-term period of stability would end spontaneously, and after a very brief transition period the system would settle into a new Nash equilibrium.

An alternative, less explored approach to self-aware collective learning and adaptation is through the sharing and aggregating of knowledge about global state and progress towards goals. Furthermore, collective adaptation could include adaptation of the architecture of the collective itself, such as to support and optimize such knowledge sharing. A promising direction could be to integrate self-organization mechanisms (e.g. [9]) in order to support such knowledge sharing. Indeed, as agents in a collective learn new architectures based on their self-awareness, so may more suitable architectures facilitate more effective collective learning and self-adaptation.

As a final note, it is worth pointing out that the time scale on which learning occurs is typically a good deal slower than that on which operational decisions are made, and actions taken. For example, the pricing policies that are learned by Q-learning [29] or by no-regret algorithms [17] evolve on a time scale that is 3 to 6 orders of magnitude slower than the scale on which prices are reconsidered. Therefore, while oscillations can occur both for actions and for the policies that govern

those actions, and they may even share similar mathematical bases at some level of abstraction, they are so different in time scale that the shifts in policy can often be too slow to be coupled with the actions themselves.

4.1.2 Smart home

It is also worth considering one or more self-aware entities that could operate at the level of the smart home as a whole. One such entity was already introduced in section 4.1.1: a power manager responsible for managing the total power consumption by all of the appliances and other electrical equipment in the house. One can draw an analogy between the power manager and mitigation strategy #5 introduced in section 3.2), which entails introducing a resource broker to stabilize systems that are prone to spontaneous oscillation. In this analogy, the various smart appliances could play the role of application managers that each appeal to the power manager for resource. In scenarios where the smart appliances discover that they can increase their resource allocation by exaggerating their resource needs, the power manager might benefit from learning the mapping from an appliance's requests to their actual consumption. It is not clear a priori whether this practice would improve stability and efficiency, or worsen it. Another form of learning that a power manager might exhibit would be to learn the tradeoffs among the various functions provided by the appliances, which could be interpreted as weights on their individual utility functions. While one might think that such weights could be provided explicitly, in practice it is difficult for people to provide them, and therefore the power manager would more likely have to infer the weights from observations of human behavior.

Another key role that a power manager for the smart home would play is that of an economic software agent that makes purchasing decisions that determine how much power there is to divide among the smart appliances. For this purpose, the power manager could employ various learning mechanisms to build models of the environment in which it is situated, which includes other smart houses in the district, the global behaviour of the city grid, and the local and regional weather. Such models would enable the house power manager to predict consumption patterns in the house and in the local grid with better accuracy. Of special importance in this context would be learning mechanisms that enable the power manager to function competently as an economic player. Each smart home's power manager could then be seen as one economic player among an entire economy consisting of all of the smart homes plus the utility (smart grid). From various works on multi-agent learning in economic systems that have already been referred to in this section [29, 17], it is clear that there is a rich set of dynamical phenomena that can be exhibited in this context.

A few additional observations are worth making here. First, in a related scenario, the power manager might be expanded in scope to be a smart home manager that makes intelligent tradeoffs among power consumption, various notions of comfort, and other attributes that matter to the home's occupants. Second, learned models are useful not just for making minute-by-minute decisions about resource allocation;

they can also be valuable for making informed decisions about long-term investments, such as more energy storage capacities (batteries), local energy generators (such as solar panels), replacing existing appliances with more efficient versions (e.g. more energy-efficient water heaters or air conditioners or washing machines), or even providing advice to users on their usage patterns (thereby helping the human end user more self-aware with regard to energy consumption).

4.1.3 Smart grid

At the level of the smart grid as a whole, the self-aware entities could include myriad smart home managers and a resource manager that represents the utility. Having already discussed in the previous scenario the issue of economic learning by the smart home managers, it suffices here to discuss what forms of learning might be valuable to incorporate into the utility resource manager. The utility resource manager might benefit from learning collective generation (e.g. via solar panels) and consumption patterns through their interactions with the smart home managers. Knowledge of the weather, vacation times, and special events that may have a noticeable impact on aggregate demand can help energy providers to provision their capabilities accordingly, on a day to day basis. Learning such patterns is also key to anticipating the long-term evolution of the grid, in terms of required production and storage capabilities, and to fine-tune fair tariffs according to offer and demand. Such global models become key economical and political decision support tools regarding the development of grid infrastructures.

Many of the previously-cited advantages and pitfalls of learning in an environment in which other self-aware entities are simultaneously learning apply in the smart grid context as well. One possible form of oscillation that might occur was described in section 3.1 (see Eq. 2), in which electricity prices and consumption fluctuate at the level of the grid as a whole. Note that this oscillation occurs in the actions; given the observations reported above for Q-learning [29] and no-regret algorithms [17] applied to economically-motivated software agents it may be possible to observe oscillations in the pricing policies themselves, at a much slower time scale. Just as a smart appliance might learn to do a better job of satisfying its individual goal by over-reporting its resource need to the smart home manager, so might the smart home managers try to game the utility resource manager or the other smart home managers by behaving in ways that misrepresent their true needs and interests. In scenarios such as this, one reasonable approach is to create coordination mechanisms that make it difficult or even impossible for smart home managers to game the system. The design of incentive-compatible auctions (such as Vickrey or second-price auctions [37]), which encourage bidders to honestly report their valuations for a good, is motivated by such a goal.

4.2 Mitigating undesirable collective behaviors

As has been illustrated several times in this section, learning can be both a cure and a cause of undesirable collective behaviors in self-aware systems. Sometimes a learning mechanism can be both at the same time — curing a problem at one level but introducing a new problem at another, somewhat akin to the infamous efforts of the Cat in the Hat and a succession of Little Cats to get rid of a pink bathtub ring in the The Cat in the Hat Comes Back [13] by employing progressively more aggressive methods that only serve to exacerbate the original problem.

In the smart appliances scenario, learning can help appliances to improve their ability to behave in accordance with human preferences, but it can also lead to an arms race in which each appliance tries to grab more resource for itself by overestimating its resource needs. The arms race can occur whether or not the appliances are unaware of one another's existence. This arms race bears some resemblance the resource conflict described in section 3, resulting from constraints on shared resources. An important difference is that, whereas the resource conflicts of section 3 resulted in direction actions by the contending entities to grab resource, in the case described in this section, the actions taken to acquire more resource were more subtle and indirect, since resource allocations were mediated through the resource broker instead of being obtained directly from the resource. Even more importantly, the dynamics of the system change on a slower time scale, because alterations in the amount of resource requested from the broker are due to learning, and such learning would typically require several resource allocation cycles.

Differences in directness and timescale notwithstanding, some of the mitigation techniques introduced in section 3 may still be applied to learning agents. The first two approaches (randomness and heterogeneity) are still applicable in many situations, but of course they still suffer from the drawback that, while they may stabilize the system, the equilibrium to which they stabilize is typically suboptimal. The third option, reducing information delays, is potentially of interest, but it is not immediately clear how to apply it in a situation where the unstable dynamics are generated at least as much by the learning process itself as they are by the resource usage. It is conceivable that, given that the timescale on which information propagates through they system can profoundly affect the dynamics of that system, such slowing down of the system dynamics from the timescale on which actions take place to the timescale on which policies governing those actions evolve may help reduce or prevent over-reactions and oscillations, but more research would be required to determine whether or not this is the case. The fifth option, introducing a resource broker, appears not to be available as a mitigation in the smart appliances scenario because a resource broker has already been introduced into that scenario.

The fourth option (endowing agents with an awareness of one another, and an ability to predict one anothers' actions) bears more discussion, as it was also discussed as one variant of the smart appliances scenario. In the example of the Q-learning pricebots [29], simultaneous multi-agent learning can result in action policies that converge to an equilibrium, but it can also lead to pseudo-convergence that is punctuated by brief episodes of transition among different near-equilibria that,

while less optimal than would be obtained by a benevolent dictator, are still preferable to the case where the agents act in ignorance of one another. However, as noted previously [27, 23], in some cases efforts to endow agents with awareness of other agents' behavior can succeed only if a sufficiently small proportion of agents possess this awareness. As was discussed in section 4.1.1, game-theoretic approaches involving fictitious play or learning algorithms based upon no external regret or no internal regret show some promise, as they can result in convergence to Nash or other game-theoretic equilibria, at least in a time-averaged sense.

In addition to the mitigation strategies discussed in Section 3.2, another option for avoiding or mitigating undesirable collective behaviors in systems of self-aware learning agents is to introduce mechanisms used in economies to coordinate the actions of self-interested entities, of which auctions are a prime example. Incentive compatible auction designs (such as Vickrey, or second-price mechanisms) [37]) have the potential to eliminate learning and resource allocation dynamics because they encourage agents to be honest about their resource needs. Such mechanisms are conceptually similar to the idea of using a resource broker to coordinate actions of multiple self-aware entities, except that they are explicitly designed to handle systems of self-interested agents, whereas the resource broker concept is inherently designed for cooperative systems in which agents honestly report their resource needs. In a properly designed auction, agents are not just *assumed* to be honest; they are *compelled* to be honest out of self-interest. Auctions are attractive in that they strike a good balance between centralized coordination (which is essentially the joint control mitigation introduced in section 2) and independent action by self-aware agents. The number of different auctions types available to system designers is enormous, ranging at least into the thousands [39], and the mechanisms can become quite complex when agents require multiple types of resource to accomplish their tasks (see various works on combinatorial auctions [35]. Fortunately, there has been some research on how a system designer can translate overall system goals into auction mechanisms that best achieve them [6].

4.3 Capitalizing on desirable collective behaviors

A system consisting of self-aware entities that can adapt, learn and interact with one another without suffering deleterious emergent effects such as those described in section 4.1 can exhibit a number of very desirable system-level properties. One important advantage that manifests itself at both design-time and run-time is modularity, and the flexibility and evolvability that stem from it. Rather than having to build systems containing pre-determined, fixed sets of agents, and controllers designed specifically for that fixed set, one can design each entity individually, and be confident that it will settle into the system alongside the other entities that cohabit the same environment, and discover and use the agents, services, or other resources that it needs to satisfy its goals. These qualities enable one to build self-aware systems and applications from self-aware components that were designed before the systems

or appliations were conceived. At run-time, these capabilities translate into autonomic [25], or self-managing capabilities such as self-configurability, self-healing, self-optimization and self-protection.

To the extent that the various mitigations discussed in section 4.2 prevent or at least reduce suboptimal and/or unstable collective behavior, they enable these natural desirable collective behaviors of self-aware systems to shine through. Several examples of desirable collective phenomena were illustrated in Section 4.1 alongside the undesirable behaviors, including evolution to Nash or other game-theoretic equilibria (at least in a time-averaged sense) by observing other agents' behaviors. As stated in Section 4.2, auctions appear to show considerable process as a mechanism that can support favorable collective behaviors, and indeed the use of economic mechanisms in general seems particularly appropriate and suitable for very large-scale systems, although on the other hand simulations of price war and related behaviors in information economies suggest that economic approaches are not a panacea [24].

5 Advanced Coordination with Mutual Awareness

Thus far in this chapter, we have considered scenarios in which adaptive entities within the system learn, reason and act independently of one another, with various levels of mutual awareness. In this section, we shift our focus to systems of adaptive entities that learn, reason and/or act in an explicitly coordinated fashion. While it is not the only rationale, one important reason for coordination is that it can avoid or mitigate undesirable collective behaviors such as those described earlier in this chapter. For example, recall that the following forms of coordination have been introduced earlier in this chapter as methods for mitigating various types of undesirable collective behavior in adaptive systems:

- Technique 1 (Joint Control), which was suggested in Section 2.2 as a means for mitigating the problem of independent controllers taking actions that waste resources and thwart attainment of individual goals;
- Technique 5 (Resource Broker), which was suggested in Section 3.2 as a means for mitigating spontaneous oscillations that migh arise from constraints on shared resources; and
- Auctions and other economic mechanisms, which were suggested in Section 4.2 as a means for migitating undesirable collective behaviors in systems of self-aware learning agents.

This section is organized as follows. First, we illustrate various forms of coordination and resultant collective adaptive behaviors through two scenarios: extensions of the familiar smart home and power grid scenario, and the autonomous shuttle scenario of Chapter 1.5, which exemplifies large-scale but inherently cooperative adaptive systems. Then, in the second and final subsection, we discuss the pros and

cons of these various techniques for achieving system-level adaptation through coordination.

5.1 Scenarios

5.1.1 Smart home and smart grid

Consider variants of the smart home and power grid scenario in which the entities comprising the system explicitly coordinate with one another [12, 11].

At the level of smart appliances within a single home, appliances (such as a smart window and thermostat) might coordinate their actions with one another through a variety of means. They could submit information about their goals (and perhaps their state) to a central authority such as a joint controller to act on their behalf (as in Technique 1 of Section 2.2). Another approach is for them to communicate their resource needs to a resource broker (Technique 5 of Section 3.2) and the resource broker could then provide either an instantaneous power allocation or a power allocation schedule to each appliance. A third approach that avoids a central authority is for the entities to exchange information regarding their proposed actions and perhaps their goals to one another, and to engage in some sort of bilateral or multilateral negotiation [18] to determine which actions (e.g. window opening times and thermostat heating periods that maximise their respective efficiencies while minimising negative impacts (for instance, avoiding the window to open during cold weather while the heater is just beginning a heating cycle). In the same vein, the smart oven and dishwasher – and other electrical appliances – can use coordination in one form or another to implement a sophisticated scheduling algorithm taking into account complex constraints on their respective schedules (the dishes have to be cleaned before deadline D, and the oven must run its cleaning cycle for M minutes during the evening, considering that the 7pm-9pm slot is reserved for potential cooking on the stove, etc.). When trade-offs are necessary – for instance, heating must be reduced so as not to exceed a power consumption limit – minimal sacrifices (such as letting the temperature drop more in the living room and the study) can be identified and chosen.

Laws and social norms that encourage or restrict certain types of individual behavior constitute another type of coordination mechanism that can enable a collective to adapt in ways that promote the goals of the collective as a whole. The deviation of Nash equilibria from solutions that maximize societal welfare is a well-known phenomenon in game theory, exhibited in the well-known Tragedy of the Commons /citeHardin1968 (in which villagers are compelled through self-interest to overgraze the commons even though they know it will hurt the whole village) and the Prisoner's Dilemma /citeAxelrod1984. If the villagers in the Tragedy of the Commons scenario were governed by a central authority, that authority could compute a societally better plan, such as rationing the commons equitably among villagers. Similarly, if the prisoners in the Prisoner's Dilemma were permitted to

communicate with one another, they could each compute the joint action that maximizes their joint welfare and negotiate an agreement whereby each will execute their contribution to that optimal joint action (which in this case is for both to cooperate). Of course, for this to work they would have to be subject to some authority that either forces or strongly encourages them to hold to their agreement. In the smart home scenario, norms governing power consumption scheduling could resolve resource contention by giving essential appliances (e.g. medical ones) priority over less important ones used for cooking or entertainment. These norms could either be built into the appliances, or they could be enforced by a smart home controller or resource broker.

At the level of the power grid as a whole, smart home managers might use other coordination mechanisms to determine power allocations for individual homes. Since the situation is inherently competitive across different homes, the joint control method of section 2.2 is not appropriate, nor is the Resource Broker as described in section 3.2, as it requires that entities communicate their true goals (in the form of utility functions) to the resource broker. On the other hand, auctions and related market mechanisms (which were introduced in Section 4.2), are inherently suited to competitive situations. For example, suppose house A has planned visitors for the night and expects higher than usual consumption levels. A bid to reserve extra power that night could be placed in a local energy market, and matched by a bid to provide power by another house B that has stored extra energy in its smart battery by collecting solar power during the day. Note that communicating bids to a central auctioneer is analogous to sharing utility functions with a Resource Broker, except that the former approach reveals far less information to competitors. Various forms of negotiation are also appropriate in competitive scenarios such as this.

Might social norms also work at the scale of a power grid? Possibly. For example, in a mutual-assistance smart city, some neighbourhoods may adopt norm-oriented solutions to deal with the unpredictability of power production and consumption at a local level and over the long term. Smart houses could agree to offer overproduction to houses that lack power resources at a certain instant, in return for having the favour returned to them in the future, when the situation may be inverted. Such behaviour could be regulated based on mutually-agreed norms, negotiated and updated as needed by the smart houses and/or the users [8]. For instance, participants might agree to only share power when do not intend to use it, or to share overproduction whenever they do not need it for critical tasks. The agents can achieve advanced coordination by specifying and adapting the norms (as in the cited example), in an alternative to market-oriented approaches.

At the largest scales, however, it seems doubtful that social norms could support collective adaptation — but laws certainly could. For example, in the event of an energy shortage, it may be deemed more important for society as a whole to keep hospitals running at the expense of office buildings or sports stadiums. Laws enforcing such prioritization could be implemented by local governments, and enforced either in smart devices that control power allocation, or through the more standard practice of threatening legal action againstd violators.

5.1.2 Autonomous shuttle fleet

The autonomous shuttles scenario introduced in Chapter 1.5 exemplifies a large-scale, inherently cooperative system. In such a situation, coordination mechanisms may be introduced to improve the system's ability to learn, reason, or act.

For example, imagine that the individual shuttles within a fleet each sense aspects of their environment (such as track conditions [3] or traffic conditions) or themselves (such as the number of passengers who embarked or disembarked at each stop, or the remaining battery power), and share their measurements with other shuttles (which could be accomplished via peer-to-peer messages or through a central hub or authority). Sharing such information enables each member of the fleet (or a central authority acting on their behalf) to operate upon more accurate and up-to-date knowledge of system state, and to learn more accurate models of system behavior (cf. [14]).

Members of the fleet might also share with one another their goals (e.g. deadlines or schedules). Based upon this shared information, a central authority could compute an itinerary that optimizes some combination of energy and scheduling goals and constraints (such as avoiding collisions) [2, 15]. This could be thought of as an instance of the Joint Control method of section 2.2 or the Resource Broker method of section 3.2. In addition to avoiding the problems of resource contention and spontaneous oscillation to which systems of uncoordinated and mutually unaware entities are vulnerable, these and other forms of coordination can avoid problems that arise in systems of entities that possess a high degree of mutual awareness, but whose individual incentives compel them to behave in ways that hurt the collective.

As an inherently cooperative systems, social norms are more likely to be an effective means for coordination than in large-scale competitive scenarios such as the power grid scenario above. For example, a protocol that clearly defines which shuttle has the right of way if two meet at an intersection promotes the societal goal of safe, collision-free operation by the shuttle fleet as a whole.

5.2 Coordinated vs. distributed adaptation

Much of this chapter has been devoted to cataloging collective behaviors that may ensue when individual entities adapt without awareness of or consideration for actions or goals of other adaptive entities within the system. We have outlined various approaches that can under some circumstances mitigate the more harmful of these collective behaviors. The coordination mechanisms discussed in this section are all designed to avoid these problems by enabling some forms of global computation to be performed over the entire collective – either actively by a central agency such as a joint controller or resource broker, or at design time by a government or other entity that has the power to enforce norms or laws that are calculated to encourage or enforce individual behaviors that will lead to a desired collective result.

Nonetheless, one should not conclude that coordination mechanisms that enable global computation are preferred universally over more distributed approaches. While centralized approaches may be attractive in principle because they offer the possibility of computing a social optimum, in practice they may suffer from the drawback that they require global knowledge, and (perhaps even worse) computational resources that may scale superlinearly with the number of entities or other variables over which they must compute a solution. Computing a globally optimal solution may be impractical at sufficiently large scale. If the computation involves a game-theoretic calculation, for example, the compute resources required can grow astronomically with the number of players and strategies.[3]

Fortunately, fully-distributed and fully-centralized approaches to adaptation and control are just two ends of a spectrum. Hierarchical control, in which local controllers operate within a scope that is defined by a higher-level controller (typically at a slower time scale) is often a successful approach. For example, Raghavendra et al. showed that a multi-level hierarchy of controllers using a variety of control techniques at different levels can be effective for data center power management [34].

Moreover, it is worth noting that one could adapt the overall method of adaptation itself in response to observed behavior. In other words, one could contemplate an optimistic approach to designing a adaptive system in which decentralization is attempted, but in which the individual entities are endowed with the ability to monitor their own behavior and perhaps that of their neighbors for signatures of undesirable emergent behaviors. A technique of this nature was employed by Heo and Abdelzaher in their work on coupled feedback control systems [20]. One could even introduce into the system special-purpose watchdog entities that contain signatures of undesirable behaviors. When such signatures are detected, the entities could send their observations to a central authority, or exchange those observations among themselves to determine that they are suffering from a known type of collective behavior. Upon such a determination, the system could introduce into itself an appropriate mitigation, such as one of those described in this chapter.

6 Discussion and Summary

Characterizing the vast universe of possible systems of interacting self-aware entities is a daunting task. Rather than attempting a full taxonomy in this chapter, we have mentioned a few non-orthogonal dimensions of that space (borrowing from nomenclature introduced in Chapter 2.2. where possible) and sampled it rather sparsely with a set of scenarios, many of which are extensions of the reference scenarios in Chapter 1.5. In order to achieve some measure of coherency, this chapter has been organized broadly according to collections of scenarios that share a common characteristic. However, it should be acknowledged that other groupings that emphasized different dimensions of the space might have been equally coher-

[3] More precisely, computing Nash equilibria has been shown to be PPAD-complete, where PPAD is a subclass of NP that contains problems that are suspected of being hard. [7].

ent, such as arranging them according to the degree of awareness that the self-aware entities have of one another's existence, actions or goals.

Looking across all of the scenarios covered in this chapter, the many forms of collective behavior exhibited by the system described in those scenarios, and the many approaches proffered for suppressing or fostering these behaviors, we draw some general observations:

- For many reasons, including scalability and evolvability, it is generally desirable to design self-aware systems as collections of adaptive, learning self-aware entities that govern their own actions. While designing joint controllers to manage two or more functions (e.g. smart windows and smart thermostats) can help attain optimality and stability, it is very constraining to try to anticipate all possible pairs or n-way combinations of functions that might be found together in a system.
- Conflicts among self-aware entities can arise under a wide variety of circumstances:
  - The most obvious case is where the goals explicitly conflict; in such a case the user must somehow clarify how that conflict is to be resolved via a policy or precedence rule, utility function, or other mechanism.
  - Even when goals do not conflict directly, actions guided by goals of different self-aware entities may conflict. One common cause of conflict is contention for a commonly-needed resource. Another is via unintended side-effects that end up coupling the actions of self-aware entities in unanticipated ways
  - The self-aware entities may conflict with one another directly, or indirectly through their impact on the environment in which they are situated.
- In the common situation where more than one of the self-aware entities learns, each learning agent potentially creates a dynamic environment for other agents, and that dynamism creates a need for all self-aware entities to continue learning, further perpetuating the dynamicity of the system. Thus convergence to stable behavior may be inherently difficult in such systems.
- While many different methods can be used to thwart undesirable collective phenomena, at a sufficient level of abstraction some common themes emerge:
  - Introducing randomness or heterogeneity where possible.
  - Reducing information delays where possible.
  - Engineering individual goals to achieve desired collective behavior (this may be difficult and somewhat against the objective of developing self-aware entities as individual agents that are not necessarily explicitly coordinated)
  - Detecting and breaking feedback loops
  - Endowing agents with an awareness of one another's existence, behavior, impact and/or models
  - Introducing an intermediary such as a resource broker or an auction or other economic/market mechanism (thereby providing a small amount of centralization, or at least localization)

- Conflicts and instabilities that result from learning can bear a mathematical resemblance to conflicts and instabilities that occur among agents that are merely adaptive or reactive (but which don't learn). However, the instabilities tend to originate in the realm of policies that govern actions, as opposed to the actions themselves, and thereafter the instabilities can occur on a much longer time scale than for non-learning agents.
- Some mitigation strategies entail learning, and yet learning can itself induce instabilities or suboptimal behavior.
- While economic mechanisms such as auctions show promise as general mechanisms for decentralized management of self-aware systems, they too can suffer from instabilities

Elaborating on the last point somewhat, it is apparent that there is no one panacea. A thread through this chapter has been to better illustrate the advantages of self-awareness and demonstrate mitigation strategies that often relied on information sharing, communication, or learning. It remains to observe that the very mechanisms that improve awareness and help with mitigation often also reduce *robustness* in the presence of failures. The intuitive high-level reason for this conflict lies in the extent of dependencies. Attaining awareness in a multi-agent system often requires one component or agent to receive information (e.g., measurements or state) from another. The receiving component then uses this information in its own algorithms and/or reasoning. This exchange creates a dependency of algorithms in one component on information arriving from another. In turn, this depedendency acts as a conduit for failure propagation. When a component fails, those that depend on its outputs may fail as well. The more extensive the web of dependencies, the larger the global fallout from local failures. In the absence of dependencies, components can fail independently without impacting the rest of the system. Hence, in designing adaptive and collectively self-aware systems, it becomes imperative to understand the implications of information exchange (needed to support adapive behavior and collective awareness) on robustness. Care should be taken not to create information dependencies or functional dependencies that give rise to failure cascades, whereby local component malfunctions propagate along dependency chains to disrupt operation of large chunks of the system.

References

1. Jacob Beal, Jeffrey Berliner, and Kevin Hunter. Fast precise distributed control for energy demand management. In *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*, pages 187–192. IEEE, 2012.
2. Basil Becker, Dirk Beyer, Holger Giese, Florian Klein, and Daniela Schilling. Symbolic Invariant Verification for Systems with Dynamic Structural Adaptation. In *Proc. of the 28th International Conference on Software Engineering (ICSE), Shanghai, China*. ACM Press, 2006.
3. Sven Burmester, Holger Giese, Eckehard Mnch, Oliver Oberschelp, Florian Klein, and Peter Scheideler. Tool Support for the Design of Self-Optimizing Mechatronic Multi-Agent Sys-

tems. *International Journal on Software Tools for Technology Transfer (STTT)*, 10(3):207–222, June 2008.
4. Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, 2008.
5. Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 746–752, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.
6. Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 132–141. ACM, 2004.
7. Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.
8. Ada Diaconescu and Jeremy Pitt. *Coordination, Organizations, Institutions, and Norms in Agent Systems X*, volume 9372 of *Lecture Notes in Artificial Intelligence*, chapter Holonic Institutions for Multi-scale Polycentric Self-governance, pages 19–35. Springer International Publishing, 1 edition, 2015.
9. JoseLuis Fernandez-Marquez, Giovanna Di Marzo Serugendo, Sara Montagna, Mirko Viroli, and JosepLluis Arcos. Description and Composition of Bio-inspired Design Patterns: a Complete Overview. *Natural Computing*, 12(1):43–67, 2013.
10. Dean P Foster and Rakesh V Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21(1):40–55, 1997.
11. Sylvain Frey, Ada Diaconescu, David Menga, and Isabelle Demeure. A holonic control architecture for a heterogeneous multi-objective smart micro-grid. In *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*, pages 21–30. IEEE, 2013.
12. Sylvain Frey, François Huguet, Cédric Mivielle, David Menga, Ada Diaconescu, and Isabelle M Demeure. Scenarios for an autonomic micro smart grid. In *SMARTGREENS*, pages 137–140, 2012.
13. Theodore Geisel. *The Cat in the Hat Comes Back*. Random House, 1958.
14. Holger Giese, Sven Burmester, Florian Klein, Daniela Schilling, and Matthias Tichy. Multi-Agent System Design for Safety-Critical Self-Optimizing Mechatronic Systems with UML. In Brian Henderson-Sellers and J Debenham, editors, *OOPSLA 2003 - Second International Workshop on Agent-Oriented Methodologies*, pages 21–32, Anaheim, CA, USA, Center for Object Technology Applications and Research (COTAR), University of Technology, Sydney, Australia, October 2003.
15. Holger Giese and Wilhelm Schäfer. Model-Driven Development of Safe Self-Optimizing Mechatronic Systems with MechatronicUML. In Javier Camara, Rogrio de Lemos, Carlo Ghezzi, and AntÂnia Lopes, editors, *Assurances for Self-Adaptive Systems*, volume 7740 of *Lecture Notes in Computer Science (LNCS)*, pages 152–186. Springer, January 2013.
16. Harry Goldingay and Peter R. Lewis. A Taxonomy of Heterogeneity and Dynamics in Particle Swarm Optimisation. In Thomas Bartz-Beielstein, Jrgen Branke, Bogdan Filipi, and Jim Smith, editors, *Parallel Problem Solving from Nature PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 171–180. Springer International Publishing, 2014.
17. Amy R Greenwald and Jeffrey O Kephart. Probabilistic pricebots. In *Proceedings of the fifth international conference on Autonomous agents*, pages 560–567. ACM, 2001.
18. James E Hanson, Gerald J. Tesauro, Jeffrey O Kephart, and Edward C Snible. Multi-agent implementation of asymmetric protocol for bilateral negotiations. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 224–225. ACM, 2003.
19. Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.
20. Jin Heo and Tarek F. Abdelzaher. Adaptguard: guarding adaptive systems from instability. In Simon A. Dobson, John Strassner, Manish Parashar, and Onn Shehory, editors, *Proceedings*

of the 6th International Conference on Autonomic Computing, ICAC 2009, June 15-19, 2009, Barcelona, Spain, pages 77–86. ACM, 2009.
21. Amir Jafari, Amy Greenwald, David Gondek, and Gunes Ercal. On no-regret learning, fictitious play, and nash equilibrium. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
22. J O Kephart, H Chan, R Das, D W Levine, G Tesauro, and C Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In *Proceedings of the Fourth International Conference on Autonomic Computing*. IEEE, 2007.
23. Jeffrey O Kephart. Can predictive agents prevent chaos. *Economics and cognitive science*.
24. Jeffrey O Kephart. Software agents and the route to the information economy. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7207–7213, 2002.
25. Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
26. Jeffrey. O. Kephart, Tad Hogg, and Bernardo A. Huberman. Dynamics of computational ecosystems. *Phys. Rev. A*, 40:404–421, Jul 1989.
27. Jeffrey O Kephart, Tad Hogg, and Bernardo A Huberman. Collective behavior of predictive agents. *Physica D: Nonlinear Phenomena*, 42(1):48–65, 1990.
28. Jeffrey O Kephart and Jonathan Lenchner. A symbiotic cognitive computing perspective on autonomic computing. In *Proceedings of the 2015 IEEE International Conference on Autonomic Computing*. IEEE, 2015.
29. Jeffrey O Kephart and Gerald J Tesauro. Pseudo-convergent q-learning by competitive pricebots. In *Proc. 17th Intl Conf. Machine Learning*, pages 463–470, 2000.
30. Jeffrey O Kephart and William E Walsh. An artificial intelligence perspective on autonomic computing policies. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 3–12. IEEE, 2004.
31. Peter R. Lewis, Arjun Chandra, Funmilade Faniyi, Kyrre Glette, Tao Chen, Rami Bahsoon, Jim Torresen, and Xin Yao. Architectural Aspects of Self-Aware and Self-Expressive Systems: From Psychology to Engineering. *Computer*, 48(8), August 2015.
32. Peter R. Lewis, Paul Marrow, and Xin Yao. Resource Allocation in Decentralised Computational Systems: An Evolutionary Market Based Approach. *Autonomous Agents and Multi-Agent Systems*, 21(2):143–171, 2010.
33. Fernando Perez-Diaz, Ruediger Zillmer, and Roderich Groß. Firefly-inspired synchronization in swarms of mobile agents. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 279–286, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.
34. Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No power struggles: Coordinated multi-level power management for the data center. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 48–59. ACM, 2008.
35. Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial intelligence*, 135(1):1–54, 2002.
36. H. Van Dyke Parunak, Sven Brueckner, Mitch Fleischer, and James Odell. A Design Taxonomy of Multi-agent Interactions. In Paolo Giorgini, JrgP. Mller, and James Odell, editors, *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 123–137. Springer Berlin Heidelberg, 2004.
37. William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.
38. William E Walsh, Gerald Tesauro, Jeffrey O Kephart, and Rajarshi Das. Utility functions in autonomic systems. In *Autonomic Computing, 2004. Proceedings. International Conference on*, pages 70–77. IEEE, 2004.
39. M P Wellman, W E Walsh, P R Wurman, and J K MacKie-Mason. Auction protocols for decentralized scheduling. *Games and economic behavior*, 35(1):271–303, 2001.