Fundamenta Informaticae 63 (2004) 1–34
IOS Press

A Formal Model for Situated Multi-Agent Systems

Danny Weyns* **and Tom Holvoet**
AgentWise, DistriNet
Department of Computer Science
K.U.Leuven, Belgium
danny.weyns@cs.kuleuven.ac.be

Abstract. Contrary to cognitive approaches of agency where a lot of effort is devoted to the formalization of agent concepts, little work has been done on the formalization of *situated multi-agent systems* (situated MASs). In this paper we present a generic model for situated MASs. This model formally describes an abstract architecture for situated MASs. In this architecture each agent is situated in his local context that he is able to perceive and in which he can act. Since intelligence in situated MASs results from the interactions of agents with the environment rather than from their individual capabilities, the model takes an action-centric approach. The model deals with (1) the actions of agents in the environment, (2) ongoing activities in the environment, such as moving objects, and (3) the interactions between agents and ongoing activities through the environment.

One model for situated MASs was described by J. Ferber and J.P. Müller. In this model all agents of the MAS act at one global pace, i.e. the agents are globally synchronized. Drawbacks of global synchronization are centralized control and poor scalability. We present a model that allows agents to synchronize locally. In this model there is no centralized entity that imposes all agents to act at one global pace, but instead agents themselves decide when they perform their next actions. The model supports simultaneous actions through regional synchronization. With regional synchronization agents form synchronized groups on the basis of their actual locality. Different groups can act asynchronously, while agents within one group act synchronously. The result is a model that does not suffer from the drawbacks of global synchronization while it preserves the properties for handling simultaneous actions.

In the paper we apply the model to a simple MAS application. We show how the abstract model can be instantiated for a practical application. Then we follow a trace in the evolution of the application and demonstrate how the model deals with each particular step.

* Address for correspondence: AgentWise, DistriNet, Department of Computer Science, K.U.Leuven, Belgium

1. Introduction

In this paper we present a generic model for *situated multi-agent systems* (situated MASs). This model formally describes an abstract architecture for situated MASs. The focus of the model is on interactions between situated agents and the effects these interactions produce in the world. This contrasts to popular knowledge level approaches such as BDI [6][25] that focus on manipulation of a symbolic representation of the environment and the desired behavior, and that use such representations as specifications for agents and their decision making [35].

1.1. Situated multi-agent systems

Situatedness is a property of agents adopted by most researchers in the domain of MAS. A well known example is Wooldridge and Jennings' definition of an agent in [36]: 'an agent is a computer system that is *situated* in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives'. In this definition, situatedness expresses the fact that an agent is not an isolated entity but *exists in* an environment, however the concept of environment is kept abstract. The definition does not make explicit *what* it means for an agent to be situated in an environment, e.g. nothing in the definition explicitly refers to the fact that the existence of an agent in an environment entails a social component. In *situated MASs*, agents are particularly social entities. The emphasis in situated MASs is on 'M' in MAS, rather than on 'A'. Agents and environment constitute complementary parts of a multi-agent world. Situatedness expresses the *local relationships* between agents and objects in the environment. Exactly these relationships give the system meaning and drive the evolution of the MAS. Through its situatedness an agent is placed in a context that he is able to perceive and in which he can interact with other agents. Intelligence in a situated MAS originates from these *interactions*, rather then from capabilities of individual agents.

The approach of situated MASs has a long history. R. Brooks [7][8] identified the key ideas of *situatedness*, *embodiment* and *emergence of intelligence*. L. Steels [30] and J. L. Deneubourg [9] introduced the basic mechanisms for agents to coordinate through the environment: *gradient fields* and *marks*. P. Maes [21] adopted the early robot-oriented principles of reactivity in a broader context of software MASs. A. Drogoul [11], M. Dorigo [10], V. Parunak [23] and many other researchers drew inspiration from social insects and adopted the principles in situated MASs. Where the school of reactive MASs originates from the rejection of classical agency based on symbolic AI, nowadays the original opposition tends to evolve towards convergence with different schools emphasizing different aspects. The researchers, although having different points of view, are very complementary, and each have their own applications [26].

Situated MASs have been applied with success in numerous practical applications over a broad range of domains. Some examples are: manufacturing scheduling [24] and supply chains systems [27], artificial worlds [20] and social simulation [18], network support [3] and peer-to-peer systems [2]. The benefits of situated MAS are well known, the most striking being efficiency, robustness and flexibility. However, in [35] M. Wooldridge has pointed to a number of limitations of situated MASs. Some of the quoted limitations are inherent to the approach, e.g. decision making of situated agents is based on local, current information so situated agents have by definition a short-time view on the world. Other topics are rather unsolved problems yet, e.g. the engineering of situated agents with respect to desired overall behavior.

1.2. Situated agents

Situated agents live and act in the present. Their decision making is not based on extensive reasoning upon mental issues. Situated agents do not use long-term planning to decide what action sequence should be executed, but instead perform *situated actions*. Situated actions are actions selected on the basis of an agent's position, the state of the world he perceives and limited internal state [1]. Contrary to knowledge based agents, situated agents do not emphasize internal modeling of the environment. Instead, they favor to employ the environment itself as a source of information. Internal representations are often sub-symbolic, e.g. evolving threshold values that express the dynamics of an agent's preferences over time for certain behaviors [4][29]. Other forms of internal state can be representations of goals with respect to the immediate environment (e.g. I see objects, so I can get one) or a commitment to a particular role in a local ongoing collaboration (e.g. as long as there are objects available, I pass them to the agent in front of me). Describing representation only in terms of the immediate environment is known as indexical-functional representation [22][13]. Situated agents use representations as an instrument to direct their decision making process, however this is done 'here and now'. Such representations do not oblige the agent to keep track of a hypothetical future state or investigate the implications of it on a plan.

1.3. Simultaneous actions

Different agents in a situated MAS may execute actions simultaneously and these actions possibly interfere with one another. We use the term of *simultaneous actions* as a general designation for actions that happen together, see also [34]. We make a distinction between two kinds of simultaneous actions: *independent actions* and *interfering actions*. Independent actions are actions that do not interfere with one another. When for example two agents decide to make a step to a different location, these two actions can happen independently of one another. Interfering actions on the other hand, bring two or more agents in contact with each other. Interfering actions correspond to an interaction between agents [2]. Depending on the nature of these interactions, we distinguish between *concurrent actions*, *influencing actions* and *joint actions*. Concurrent actions are of a conflicting nature. An example is two agents that simultaneously try to pick up the same object. The result of such interaction may be that one agent gets the object, while the other fails to get it. Influencing actions are actions that positively or negatively affect each other. When for example two agents push the same object in different directions, the object moves according to the resultant of the two actions. Whether or not this resulting movement is profitable for the individual agents depends on their potential intentions. Joint actions are actions that must be executed together in order to produce a successful result. An example of joint actions is a group of agents that carry a large object and make a move together.

Other researchers make a similar distinction between different kinds of simultaneous actions. Some examples: Allen and Ferguson [1] differentiate between 'actions that interfere with each other' and 'actions with additional effects'; Griffiths, Luck and d'Iverno [17] talk about 'joint actions that a group of agents perform together' and 'concurrent actions that are performed at the same time'; Boutilier and Brafman [5] distinguish 'actions with a positive or negative interacting effect'.

[1] Wavish and Connah [31] adopted the notion of situated action in MASs. They associate situated action with stimulus/response-like behavior that is only related to external perception. In this paper we relax this definition and allow agents also to integrate limited internal state to select their situated actions.

[2] Throughout the paper, we use interaction and interfering actions interchangeably. However, in general, the notion of interaction is not only used for interfering actions, e.g. the allocation of a task by means of a contract net is also called an interaction.

1.4. Towards a locally synchronous model for situated MASs

Contrary to cognitive approaches of agency where a lot of effort is devoted to the formalization of agent concepts, little work has been done on the formalization of situated multi-agent systems. Genesereth and Nilsson [16] have described a set of simple architectural models, including models for purely reactive agents and simple agents with memory. Based on this work a number of other researchers have developed formal theories for agent architectures and interaction. An interesting reference is [19] that analysis the differences between rational and reactive agent architectures and proposes a unified architecture that aims to capture both architectures as special cases.

Ferber and Müller [14] pursued a similar course and developed a formal model for MAS that is applicable to both purely reactive agents and agents with memory. That model is based on Ferber's theory for action, described in [12]. In essence, this theory separates what an agent wants to perform from what actually happens. Agents produce influences into the environment and subsequently the environment reacts by combining the influences to deduce a new state of the world from them. The reification of actions as influences enables the environment to combine simultaneously performed activity in the MAS. Built upon this theory, the Ferber-Müller model is able to deal with complex interactions both in the environment and between agents through the environment. However, in that model all agents of the MAS act at one global pace, i.e. the agents are globally synchronized. Global synchronization implies centralized control, in general an undesirable property of MASs. Besides, in order to calculate the reaction, the environment has to combine the influences of all agents in each cycle and that does not fit with scalability of the MAS.

The contribution of this paper is a formal model for situated MASs that allows agents to synchronize locally. In this model there is no centralized entity that imposes all agents to act at one global pace, but instead agents themselves decide when they perform their next actions. The model supports simultaneous actions through *regional synchronization*. With regional synchronization agents form synchronized groups on the basis of their actual locality. Different groups can act asynchronously, while agents act synchronously within their group. This results in a model that does not suffer from the drawbacks of global synchronization while it preserves the properties for handling simultaneous actions.

1.5. Design choices of the model

By taking a formalized approach, the model we present gives a rigorous *specification* of a situated MAS. This specification describes the structural decomposition of a MAS. Splitting up the MAS in constituent components reduces complexity and offers a means for reasoning upon situated MASs. But more than only a means for communication, the specification can also serve as a basis for *constructing* situated MASs towards *implementation*.

No model is without bias. Decomposing a system implies *design choices*, and as such it reflects the vision of the designers on the modeled system. The basic choices that underlie the model presented in this paper are:

1. to model actions of agents and their consequences in the environment
2. to deal with complex interactions in the environment and between agents
3. to balance autonomy of agents with simultaneity of actions
4. to decompose the behavior of an agent into a set of functional modules that can further be refined

5. to describe the evolution of the MAS
6. to use a formalized approach

We motivate each of these choices. Choices 1 and 2 clearly express the underlying action-centric approach of the proposed model. The motivation for these choices is that interactions between agents are central to the problem solving abilities of situated MASs. Choice 3 however, expresses that agents may not be slaves of the collective. We advocate that the autonomy of a situated agent, with respect to the decision *when* to act, can only be limited by other agents as far as this agent is involved in (possible) interactions with other agents. Independently acting agents should be able to act autonomously, i.e. at their own pace, enhancing the flexibility of the MAS. This motivation may sound evident, however finding a good balance between the agents' autonomy and the ability to perform simultaneous actions is a non trivial matter. Therefore, in practice the autonomy of agents to decide when to act is often sacrificed to keep the implementation simple, e.g. with global synchronization. The focus of choice 4 is on the architecture of individual agents. The motivation for a structural decomposition is to build a model that can serve as a basis for constructing situated agents. A number of design choices we have made reflect common principles of agent architectures, such as limited perception, memorization or decision making. The motivations for other specific choices are explained in the discussion of the agent model in section 5. The goal of choice 5 is to integrate activity of agents with other ongoing activities in the environment. The motivation is to build a model that supports dynamical evolution in the environment that not directly depends on agent activity, but that potentially interferes with the activity of agents. An example is a moving object that is influenced by an agent, e.g. a soccer player that kicks the rolling ball or a robot arm that removes a malformed product from a conveyor belt. Finally, the motivation for choice 6 is to be precise. It is not a first goal of the model to perform mathematical verification of situated MASs. Nevertheless verification is an important topic, but in this paper we limit the discussion to a formal description of a MAS. A final remark concerns non-determinism in the MAS. To avoid overloaded expressions we have not included non-deterministic evolution of the MAS in the model. In the elaborated example application discussed in the paper, we touch upon non-determinism briefly.

1.6. Overview

The paper is structured as follows. In section 2 we briefly discuss Ferber and Müller's model for action based on influences and reactions to influences. Section 3 explains regional synchronization. Next, in section 4, we present a general theory for a dynamical system. Subsequently we apply this theory. First we develop a model for agents in section 5. Next, in section 6 we describe a model for other ongoing activities in the environment. Then we integrate both these models in an integral model for situated MASs in section 7. In section 8 we apply the model for situated MASs to a simple example application. We show how the abstract model can be instantiated and follow a trace in the evolution of the application. Finally, we evaluate the model and conclude, respectively in sections 9 and 10.

2. Influences and reactions

In [14], J. Ferber and J.P. Müller describe a model for actions in situated MASs. In this section, we give a brief summary of the model as presented in that paper. At the end we evaluate the model and point to a number of limitations.

The model of Ferber and Müller is based on three main principles. First, it distinguishes between influences and reactions to influences. Influences come from inside the agents and are attempts to modify the course of events in the world. Reactions, which result in state changes, are produced by the environment by combining influences of all agents, given the local state of the environment and the laws of the world. This clear distinction between the products of the agents behavior and the reaction of the environment enables to handle simultaneous actions. Second, the model decomposes the system dynamics into two parts, the dynamics of the environment and the dynamics of the agents situated in the environment. And third, the model describes the different dynamics of the MAS by means of abstract state machines.

Contrary to classical theories that only use the state of the world to describe evolution in a MAS, in Ferber and Müller's model evolution is described as the transformation of what they call *dynamical state*. Such dynamical state is defined as a 3-tuple:

$$\delta \in \Delta \ = < s, \sigma, \gamma >$$

$s \in S$ is the set of internal states of the agents, representing the 'mental state' of the MAS. s is defined as a vector $< s_1 \times \ldots \times s_n > \in S_1 \times \ldots \times S_n$ for all agents $\{a_1, \ldots a_i, \ldots a_n\}$ of the MAS. $s_i \in S_i$ denotes the internal state of agent a_i with S_i the set of all possible states of agent a_i. $\sigma \in \Sigma$ represents the state of the environment and $\gamma \in \Gamma$ the set of influences simultaneously produced in the environment. The evolution of the MAS is then defined as:

$$Evol : S \times \Sigma \times \Gamma \rightarrow \perp$$
$$Cycle : S \times \Sigma \times \Gamma \rightarrow S \times \Sigma \times \Gamma$$
$$Evol(< s_1 \times \ldots \times s_n >, \sigma, \gamma) = Evol(Cycle(< s_1 \times \ldots \times s_n >, \sigma, \gamma))$$

Ferber and Müller define the evolution of a MAS as "an infinite recursive function", called *Evol* that takes as argument a dynamical state of the world. In each step the $Cycle$ function transfers the dynamical state to the next dynamical state. The $Evol$ function runs in an infinite loop and as such returns no results except errors or impossible values, denoted by $\perp$[3].

The *Cycle* function is further split into three sub-functions, $Behavior_i$, $React$ and $Exec$. We start with $Behavior_i$ that is typed as follows:

$$Behavior_i : S_i \times \Sigma \rightarrow S_i \times \Gamma$$
$$Behavior_i(s_i, \sigma) = < s_i', Decision_i(s_i') >$$ [4]

$Behavior_i$ defines how an agent a_i transforms his internal state and produces the next influence, given its own internal state and the state of the environment. s_i' is the internal state of the agent updated with the last percept, while $Decision_i$ is the function that decides which operation must be executed based on the most recent information.

$React$ and $Exec$ are typed as follows:

$$React : (\Lambda, ||) \times \Sigma \times \Gamma \rightarrow \Sigma$$
$$Exec : (O, ||) \times \Sigma \times \Gamma \rightarrow \Gamma$$

[3] The authors do not explain how $Evol$ can produce a result in $\perp$ if none of its substituents ever does.

[4] We make the assumption that the behavior of an agent is based on the perceived state of the environment. In [14] there is some confusion about this issue, but in [13] Ferber affirms our assumption.

Λ is the set of laws of the world that describe how the next state is computed given the previous state and the set of influences and O is the set of operators defined for the MAS that produce influences. Operators as well as laws are composed in parallel, denoted by the $||$ operator [5]. The $React$ function takes the influences and according to the current state of the world and its laws, produces the next state of the world. The $Exec$ function produces the influences in the next dynamical state. According to the authors, parallel composition of operators simply produces the union of influences produced by the different operators. For laws, the parallel composition must be commutative to allow a flexible description of state changes by an unordered set of laws. Applied for two parallel composed laws, respectively operators, we get:

$$React(\lambda_1 \mid\mid \lambda_2, \sigma, \gamma) = React(\lambda_2, React(\lambda_1, \sigma, \gamma), \gamma)$$
$$Exec(o_1 \mid\mid o_2, \sigma, \gamma) = Exec(o_1, \sigma, \gamma) \cup Exec(o_2, \sigma, \gamma)$$

To describe the dynamics of the system, Ferber and Müller integrate the $React$ and $Exec$ function with the behavior of the agents in the $Cycle$ function:

$$Cycle(< s_1 \times \ldots \times s_n >, \sigma, \gamma) = < s'_1 \times \cdots \times s'_n, \sigma', Exec((o_1 \mid\mid \ldots \mid\mid o_p), \sigma', \gamma) \cup \bigcup^n \gamma_i >$$
where $\sigma' = React((\lambda_1 \mid\mid \ldots \mid\mid \lambda_m), \sigma, \gamma)$ and $< s'_i, \gamma_i > = Behavior_i(s_i, \sigma)$

$(o_1 \mid\mid \ldots \mid\mid o_p)$ denotes the set of parallel composed operators, while $(\lambda_1 \mid\mid \ldots \mid\mid \lambda_m)$ denotes the applied set of parallel composed laws. The $Cycle$ function expresses the evolution of a MAS with *n* agents. This evolution is described as "an infinite recursive function" that produces in each cycle: (1) a new state of the environment as reaction of the environment to the set of produced influences; (2) a new set of internal states as a result of the behavior of the individual agents and (3) a new set of influences produced by the agents and the dynamics of the environment.

The evaluation of the model brings up the following considerations. The model deals with complex interactions in the environment and between agents, solving the fundamental problem of simultaneous actions in an elegant way. Besides, the model is applicable to purely reactive agents as well as to agents with memory [6]. However, in the model *all* agents are forced to perform simultaneously the perception-to-action cycle in one step, i.e. the model is restricted to synchronous description of MAS evolution. This is a hard requirement and it implies a number of drawbacks of the model. First, the scalability of the model with respect to the number of agents in the MAS is limited. Since the influences of *all* agents are treated as if they happened together, each influence can possibly interfere with any other influence. This makes the costs for calculating reactions in the order of square to the complete number of agents in the MAS, i.e. $O(n^2)$ for a MAS populated with n agents. And second, all agents are globally synchronized and that implies centralized control of the evolution of the MAS. Centralized control conflicts with the distributed nature of MAS.

3. Regional synchronization

To resolve the drawbacks of global synchronization without losing the properties for handling complex interactions, we introduce regional synchronization. Regional synchronization shifts the responsibility

[5] Obviously, the type of the first argument in the $React$ function in [14] should be $(2^{\Lambda}, ||)$ and similar $(2^{O}, ||)$ for the first argument of the $Exec$ function.

[6] In this brief overview, we only examined the more general model for agents with memory. For more information on purely reactive agents, see [14].

of synchronization from the environment towards the agents. With regional synchronization, there is no longer one global synchronizer, but instead each agent is equipped with his own local synchronizer. Each synchronizer is responsible to setup synchronization for its associated agent with the synchronizers of other agents. Since the goal of synchronization is to handle simultaneous actions, in particular interfering actions [7], the range for an agent to synchronize with other agents should be in accordance with the range for such interfering actions. In the context of situated agents it is quite natural to limit this range to the perceptual range of the agents themselves. Synchronization between two agents then boils down to reaching a mutual agreement about synchronization. To reach such an agreement, synchronizers negotiate with one another by exchanging messages.

In general, setting up such regional synchronization is not a trivial matter. First, an agent may establish synchronization with more than one agent. This implies that a group as a whole must reach an agreement about synchronization. Second, agents are autonomous entities, running in asynchronous processes. This fundamental property makes that any agent that has entered the phase of synchronization setup can be disturbed, at any time, by a new agent that enters the perceptual range of the synchronizing agent. And finally, a third tough nut to crack concerns the situatedness of agents. Since situated agents have an explicit position in the environment, each agent has a personal view on the environment. So when two agents are positioned inside each others perceptual range, they may see different candidates to synchronize with. This property makes that at the end of synchronization setup an agent may know only a limited number of agents of the synchronized group to which he belongs. We call the whole group a *region* of synchronized agents. Each member of a region of synchronized agents is synchronized with every other member of that group, however each member is *directly synchronized* only with a subset of the whole group, i.e. with the agents he perceives himself.

We developed an algorithm that enables individual agents to establish regional synchronization. For a detailed discussion of this algorithm we refer to [33]. Here we limit the discussion to an intuitive description of the algorithm.

As stated above, each agent is equipped with a synchronizer who is responsible for handling synchronization. Before each action an agent enters synchronization setup. Synchronization setup starts when the synchronizer receives its *view–set*, together with the *synchronization–time*. The view–set is the initial set of candidates for synchronization, containing all synchronizers within the perceptual range of the associated agent. Agents that do not perceive any other agent within their perceptual range skip synchronization setup and continue immediately, acting asynchronously. The synchronization–time is the value of the logical clock at the moment when the synchronizer's view-set was composed. This logical clock is a counter maintained by the environment [8]. Each time a group of synchronized agents has concluded the acting phase, the value of the logical clock is incremented and new view–sets for the agents are composed. With the view-set and synchronization-time the synchronizer composes a *synchronization–set*. Besides its own name and synchronization–time, such synchronization–set contains a *member–set*. In the member–set each synchronizer in the view–set of the synchronizer is represented by a *member*. A member is a 3-tuple, containing the *name* of the candidate for synchronization, a *state* and a *time stamp*. Initially each member of the member–set is in the initial state denoted by *ini*, while the time stamps have the value t^0 that stands for the initial value, zero. During the execution of the algorithm, synchronizers

[7] Interfering actions are the kinds of simultaneous actions we described in the introduction section: concurrent actions, influencing actions and joint actions.

[8] Notice that the value of the logical clock is not a global variable. In a distributed setting, the local environment of the MAS on each host maintains its own local clock.

progressively try to synchronize with the members of their synchronization–set by means of sending messages back and forth. During this interaction, negotiating synchronizers pass two phases. During the first phase they decide whether they agree about synchronization and subsequently during the second phase they mutually commit to the agreement. During this process, synchronizers exchange the value of their synchronization–time and mutually register the received values for the member of that particular synchronizer. Throughout the algorithm, the state of each member evolves from *ini* to *ack* (synchronization accepted), *com* (committed) and finally *sync* (mutually synchronized). The decision whether a synchronizer continues synchronization with a particular synchronizer depends on (1) the membership of a requesting synchronizer; (2) the comparison between the value of the synchronization–time of the member and the value of the synchronizer's own synchronization–time; and (3) the combination of states of all members of the member–set. In case synchronization can not be achieved, the rejecting synchronizer informs its colleague. As far as they belong to each others member–set, both synchronizers then remove the corresponding member from their member–set. As soon as all members of the member–set of a synchronizer have reached the state $sync$, the synchronizer concludes synchronization setup and activates its associated agent to continue acting. Based on the individual sets of synchronized agents, the environment composes regions of agents whose influences are handled synchronously.

To conclude this section we briefly describe how the algorithm integrates two building blocks of distributed algorithms: a two-phase commit protocol and a logical clock. In a classical two-phase commit protocol, one coordinator manages the votes of all participants and decides about the outcome of the interaction. In our algorithm, synchronizers are peers and can play both the role of participant as well as coordinator during one ongoing synchronization setup. Which role one synchronizer plays with respect to the other depends on the comparison of the values of both their synchronization–time, i.e. the value of the logical clock they received when they entered synchronization setup. As for the two-phase commit protocol, the result of our algorithm is a set of synchronizers that have reached an agreement, i.e. their associate agents execute the next action synchronized. However, contrary to the two-phase commit protocol where reaching an agreement is a matter of all or nothing between a fixed set of participants, the algorithm for regional synchronization can result in an agreement between only a subset of synchronizers involved in the ongoing negotiation.

4. Dynamical system

In this section, we present a theory that forms the basis of the model for situated MAS we discuss in the following sections. This theory describes the dynamics of a system composed of a set of actors $At = \{a_1, \ldots, a_n\}$ that exist in an environment. Actors are processes that can perform actions in the environment. We denote a group of actors that simultaneously perform an action as $\alpha \subseteq At$. Furthermore, we introduce $A = 2^{At}$, i.e. the set of all possible subsets of At, thus $\alpha \in A$. Besides the activity invoked by the actors, other activities may be going on in the environment too. Examples of such activities are a moving object or, in the context of ant-like systems, an evaporating pheromone. We describe such activities as *ongoing activities*, denoted as $d_j \in Ac$, with $Ac = \{d_1, \ldots, d_m\}$ the set of all possible ongoing activities in the MAS and $D \subseteq 2^{Ac}$ the set of all subsets of ongoing activities that simultaneously can be active in the environment. The theory we discuss is based on the distinction between influences and the reactions of the environment upon them. We use a different notion of *dynamical state* as Ferber and Müller:

$$\delta \in \Delta \ = < \sigma, \psi >$$

$\sigma \in \Sigma$ is the state of the environment, while $\psi \in \Psi$ describes a set of *consumptions* [9]. A consumption is an effect from the environment reserved for a particular actor. Such consumption is a result from the reaction of the environment to the most recently produced influences for that actor. When an actor 'consumes' a consumption, the consumed effect can be absorbed by the actor (e.g. food that is turned into energy), the actor may simply hold an element (e.g. an object he has picked up) or the consumption may affect the actor's state (e.g. the arm of a robot is wrenched through an external force). We motivate the different choice of dynamical state in comparison to the Ferber-Müller model at the end of this section.

The dynamics of the system is defined as the $Cycle$ function that maps a dynamical state on the next dynamical state:

$$Cycle : \Sigma \times \Psi \rightarrow \Sigma \times \Psi$$
$$Cycle(\sigma, \psi) = < \sigma', \psi' >$$

To clarify the activities invoked by the actors and the ongoing activities in the environment on the one hand and the reaction of the environment upon both activities on the other hand, we split the $Cycle$ function in two parts. The first part is composed of two sub-functions: $Exec^A$ and $Apply^D$. $Exec^A$ represents the activities invoked by the actors, while the ongoing activities in the environment are represented by the $Apply^D$ function. The second part is a single function $React$ that represents the reaction of the environment to the simultaneously performed activity of actors and ongoing activities. We start with $Exec^A$ that is typed as follows:

$$Exec^A : (2^{O^A}, ||) \times \Sigma \times \Psi \ \rightarrow \ \Gamma^A \times \Psi$$
$$Exec^\alpha((o_p \ || \ \ldots \ || \ o_t), \sigma, \psi) =< \gamma^\alpha, \psi^I >$$

Here, $\alpha = \{a_p, \ldots, a_t\}$ is the set of simultaneously acting actors and $\gamma^{\alpha} \in \Gamma^A$ the set of influences produced by α, with $\Gamma^A \subseteq \Gamma$ all influence sets that can be produced by actors and Γ all influence sets that can be produced in the system. Influences are produced through the application of *operators*. Operators of simultaneously acting actors are composed in parallel, denoted by the $||$ operator. The operator applied by $a_i \in \alpha$ is denoted as $o_i \in O^A$, with $O^A \subseteq O$ the set of operators available for actors and O the operators available in the system. We define an operator as a 3–tuple:

$$o_i \in O : < name, conditions, influence >$$

$name$ is an expression with variables that can appear both in $conditions$ and in $influence$. $conditions$ is a set of expressions that determine when the operator $name$ is applicable. $influence$ finally denotes the activity in the world that the application of the operator attempts to realize. Operators are composed in parallel to define the resulting set of influences of the simultaneous acting actors, i.e. simply the union of influences produced by the different operators. $Exec^{\alpha}((o_p \ || \ .. \ || \ o_t), \sigma, \psi)$ then expresses the execution of the operators invoked by the actors, in words: in the state of the environment σ, a set of actors α consume a subset of consumptions of ψ, say ψ^E, resulting in a reduced set of consumptions [10] $\psi^I = \psi - \psi^E$, and produce, with a set of composed operators $(o_p \ || \ .. \ || \ o_t)$, a set of influences γ^{α}.

The effects of the ongoing activities in the environment are induced by the $Apply^D$ function that is typed as follows:

[9] We borrow the notion of consumption from Ferber who introduced the concept in [13].

[10] The binary operator '$-$' simply denotes the subtraction of two sets (operand 1 minus operand 2), while the '$+$' operator analogously denotes the addition of two sets.

$$Apply^D : (2^{O^D}, ||) \times \Sigma \rightarrow \Gamma^D$$
$$Apply^d((o_r \mid\mid \ldots \mid\mid o_v), \sigma) = \gamma^d$$

$O^D \subseteq O$ is the set of operators that can be applied by ongoing activities, while $\Gamma^D \subseteq \Gamma$ represents the set of influence sets that can be produced in the environment through the application of the operators by ongoing activities. $Apply^d((o_r \mid\mid \ldots \mid\mid o_v), \sigma)$ expresses the application of the operators resulting from the ongoing activities in the environment, with $d = \{d_r, \ldots, d_v\}$ the current set of ongoing activities, i.e. $d \in D$. Putting it to words: in the state of the environment σ, d produces the set of influences γ^d through the application of a set of parallel composed operators $(o_r \mid\mid \ldots \mid\mid o_v)$, one operator o_j for each ongoing activity $d_j \in d$.

Since the activities invoked by the actors and the ongoing activities in the environment happen simultaneously, we have to combine the influences resulting from $Apply^d$ and $Exec^\alpha$. For convenience, we use a relaxed definition of function composition, i.e. $h(a, b) = g \circ (f_1(a), f_2(b))$ denotes that function $h(a, b)$ is defined as: first calculate $f_1(a)$ and $f_2(b)$ are then use the results to calculate g. Based on this definition, we represent the combination of activity invoked by the actors and the ongoing activities by the binary operator $\uplus$ that is typed as follows:

$$(\uplus) : ((2^{O^D}, ||) \times \Sigma) \times ((2^{O^A}, ||) \times \Sigma \times \Psi) \rightarrow \Gamma^D \times \Gamma^A \times \Psi$$
$$\uplus(((o_r \mid\mid \ldots \mid\mid o_v), \sigma), ((o_p \mid\mid \ldots \mid\mid o_t), \sigma, \psi)) = < \gamma^d, \gamma^\alpha, \psi^I >$$

$$\text{or } \uplus(((o_r \mid\mid \ldots \mid\mid o_v), \sigma), ((o_p \mid\mid \ldots \mid\mid o_t), \sigma, \psi)) =$$
$$\cup_T \circ (Apply^d((o_r \mid\mid \ldots \mid\mid o_v), \sigma), Exec^\alpha(((o_p \mid\mid \ldots \mid\mid o_t)), \sigma, \psi))$$
$$\text{with } \cup_T : (\Gamma^D) \times (\Gamma^A \times \Psi) \rightarrow \Gamma^D \times \Gamma^A \times \Psi$$

The reaction of the environment to the simultaneously performed activity of the actors and the ongoing activities in the environment is described by the $React$ function that is typed as follows:

$$React : (2^\Lambda, ||) \times \Sigma \times \Gamma^D \times \Gamma^A \times \Psi \rightarrow \Sigma \times \Psi$$
$$React((\lambda_q \mid\mid \ldots \mid\mid \lambda_u), \sigma, \gamma^d, \gamma^\alpha, \psi^I) = < \sigma', \psi' >$$

Λ denotes the *laws* of the world. These laws describe how the next state of the environment is computed, given the previous state and the produced influences. Laws are defined as 3-tuples:

$$\lambda \in \Lambda : < influence\text{-}set, conditions, effects >$$

$influence\text{-}set$ is the set of influences involved in the law, i.e. a collection of possibly interfering influences originated from the execution of a set of parallel composed operators. $conditions$ can be state representations of the environment or other parameterized boolean expressions. Every term in $conditions$ must hold to apply the $effects$, otherwise no effect at all is induced by the law. $effects$ expresses the results of the successful application of the law in the MAS, i.e. state changes or other effects the actors experience. Note that the outcome of $effects$ can be non-deterministic, e.g. for a set of concurrent actions, a law may specify that a random selection of different possible outcomes is made. As for operators, laws are composed in parallel [11]. $React((\lambda_q \mid\mid \ldots \mid\mid \lambda_u), \sigma, \gamma^d, \gamma^\alpha, \psi^I)$ then expresses the reaction of the environment to the simultaneously performed activity in the environment. Putting it to words: in the state of the environment σ, for the united sets of influences γ^d and γ^α, produce according

[11] In section 8.2.2 we elaborate on the semantics of parallel composition of laws, illustrated by a concrete set of laws.

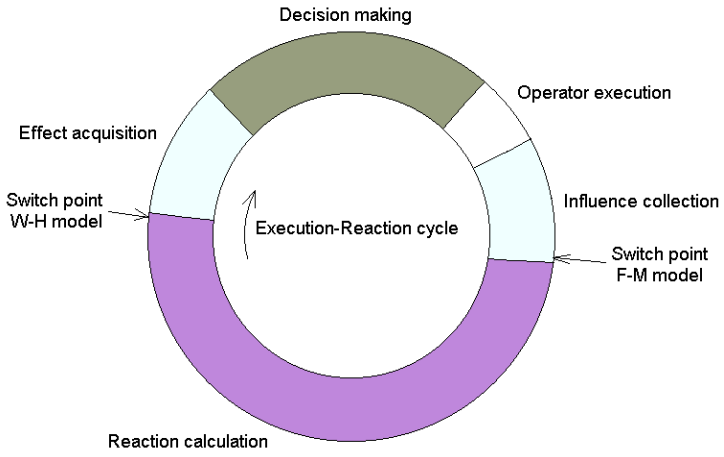

Figure 1. Structural overview of the execution-reaction cycle.

to the set of parallel composed laws $(\lambda_q \mid\mid \ldots \mid\mid \lambda_u)$, the next state of the environment σ' and add a set of consumptions, say ψ^R, to ψ^I resulting in an updated set of consumptions $\psi' = \psi^I + \psi^R$.

We can now describe the evolution of the dynamical system:

$$Cycle(\sigma, \psi) = React((\lambda_q \mid\mid \ldots \mid\mid \lambda_u), \sigma, \gamma^d, \gamma^\alpha, \psi^I)$$
$$\text{with } < \gamma^d, \gamma^\alpha, \psi^I > = \ (Apply^d((o_r \mid\mid \ldots \mid\mid o_v), \sigma) \uplus Exec^\alpha((o_p \mid\mid \ldots \mid\mid o_t), \sigma, \psi)))$$

We define the evolution of the dynamical system as a sequence of cycles. In each cycle the $Cycle$ function transfers the dynamical state into the next dynamical state, i.e. it produces (1) a new state of the environment and (2) a new set of consumptions. This twofold transfer is the result of the *reaction* of the environment to the *execution* of a set of parallel composed operators $(o_p \mid\mid \ldots \mid\mid o_t)$ invoked by a subset of actors α that exist in the system together with the *application* of a set of parallel composed operators $(o_r \mid\mid \ldots \mid\mid o_v)$ resulting from the ongoing activities d in the environment, given the state of the environment σ, a set of consumptions ψ, and a set of parallel composed laws of the world $(\lambda_q \mid\mid .. \mid\mid \lambda_u)$.

To conclude this section, we reflect on the model and clarify in what respect and why the proposed model differs from the model of Ferber and Müller. Two obvious differences between the models are the definition of dynamical state and the granularity of the groups of synchronized agents. We discuss the differences of the models from the perspective of the execution-reaction cycle for situated MASs, graphically depicted in Fig. 1. In the Ferber-Müller model dynamical state is composed with *influences*. As such, the dynamics of the MAS can be expressed as the reaction of the environment to the set of influences and subsequently the production of a new set of influences, given the state of the environment and the laws of the world. So, the execution-reaction cycle runs from the point where the influences are collected to the next point where influences are collected, indicated by the ”Switch point F-M model” in Fig. 1. In that model, the start of the cycle is initiated by the environment and as such the model takes an *environment-centered* view on MAS evolution. This fits well with synchronous evolution of the MAS whereby each agent in the system has to produce a new influence in each new cycle. As such, the granularity of synchronous acting agents in the Ferber-Müller is the whole group of agents in the MAS. All agents act at one global pace, i.e. the influences of all agents in each cycle are considered as happen simultaneously.

In the model proposed in this paper, dynamical state is composed with *consumptions*. The dynamics of the MAS can be expressed as the consummation of a subset of consumptions and the production of a set of influences to which the environment subsequently reacts (according to the applicable laws) by updating its state and producing a new set of consumptions. So in this model the execution-reaction cycle runs from the point where the reactions are calculated to the next point where the reactions are calculated, indicated by the "Switch point W-H model" in Fig. 1. In this model actors take the initiative to start their cycles, and as such the model takes an *agent-centered* view on MAS evolution. A subtle difference with the Ferber-Müller model is that in this model subsets of actors can consume their consumptions independently and run asynchronously through the execution-reaction cycle. In this model, the granularity of synchronous acting actors are the subsets. Influences of actors within a subset are considered as happen simultaneously, however different subsets can act asynchronously.

5. Model for situated agents

In this section we describe an architecture for a situated agent that fits in the theory of a dynamical system as we discussed in the previous section. In this theory we used the notion of actor as an abstract concept for a process that is able to act in an environment and consume the results of his actions. In this section, we refine the notion of actor towards a generic model for a situated agent. More than only observable behavior, a situated agent is an encapsulated entity with several capabilities. We consider agents to have memory, however this should be seen as a generalization and does not exclude the model to be applicable to purely reactive agents. In section 7 we integrate the agent model, together with the model for ongoing activities in the environment we discuss in the next section, in a generic architecture for situated MASs.

We start by introducing a number of definitions:

$Ag = \{a_1, \ldots a_i, \ldots a_n\}$: the agents of the MAS
$y_i \in Y$: the identity of a_i with $Y = \{y_1, \ldots, y_n\}$ the set of unique identities, one for each agent in the MAS
$Id : Ag \rightarrow Y$ is a function that returns an agent's identity, i.e. $Id(a_i) = y_i$
$\Upsilon = 2^Y$ the set of all possible subsets of identities of agents
$p_i \in P_i$: a percept [12] of a_i, with P_i the set of all possible percepts of a_i
C: the set of possible consumptions in the MAS
$c_i \in C_i$: a consumption consumed by agent a_i, with $C_i \subseteq C$ the set of all consumptions that can be consumed by agent a_i
$s_i \in S_i$: the internal state of a_i, with S_i the set of all possible states of a_i
$\upsilon_i \in \Upsilon$: a set of identities of agents with whom agent a_i is *directly synchronized* : $(\forall a_i \in Ag : Id(a_i) \in \upsilon_i) \wedge (\forall a_j, a_k \in Ag$: if $Id(a_j) \in \upsilon_k$ then $Id(a_k) \in \upsilon_j)$
$\mu_i \subseteq M$: a set of synchronization messages exchanged by a_i, with M the set of all possible synchronization messages in the MAS
O: the operators available in the MAS
$O^{Ag} \subseteq O$: the operators available for agents in the MAS
$o_i \in O_i^{Ag}$: an operator for agent a_i, with $O_i^{Ag} \subseteq O^{Ag}$ the set of all operators available for a_i

Based on these definitions, we define a situated agent as a 12-tuple:

[12] A percept denotes the perceptual input an agent is able to perceive in his environment.

$$a_i = < y_i, P_i, C_i, S_i, \Upsilon, M, O_i^{Ag}, Perception_i, Consumption_i, Memorization_i, Synchronization_i, Decision_i >$$

An agent is capable to perceive its neighboring environment. This is expressed by the $Perception_i$ function that is typed as follows:

$$\begin{aligned} &Perception_i : \Sigma \rightarrow P_i \\ &Sense_i : \Sigma \rightarrow \Sigma|_\Omega \\ &Interpret_i : \Sigma|_\Omega \rightarrow P_i \\ &Perception_i(\sigma) = Interpret_i(\sigma_i) \circ Sense_i(\sigma) \\ &\qquad\qquad = Interpret_i(Sense_i(\sigma)) \\ &\qquad\qquad = p_i \end{aligned}$$

$Perception_i$ is a compound function. $Sense_i$ produces a representation of the local environment for the agent. We define Ω as the demarcation for the MAS. Ω marks out the scope agents are able to perceive in their environment. Depending on the domain, Ω can be a fixed parameter for all agents, or a function that determines the perceptual range for an agent based on his particular capabilities and / or the state of the environment[13]. $\Sigma|_\Omega$ is Σ limited according to Ω applied in the context of an agent a_i. $Interpret_i$ takes a representation of the local environment and produces a percept for the agent.

Besides perception, a situated agent is also capable to consume effects from the environment. Consuming a consumption is expressed by the $Consumption_i$ function that is typed as follows:

$$\begin{aligned} &Consumption_i : \Psi \rightarrow C_i \\ &Identify_i : \Psi \rightarrow \Psi|_Y \\ &Consume_i : \Psi|_Y \rightarrow C_i \\ &Consumption_i(\psi) = Consume_i(\psi_i) \circ Identify_i(\psi) \\ &\qquad\qquad = Consume_i(Identify_i(\psi)) \\ &\qquad\qquad = c_i \end{aligned}$$

$Consumption_i(\psi)$ is also a compound function. Since consumptions are personalized, consuming a consumption requires the mapping of the set of consumptions to the agent before this latter consumes. Therefore the $Identify_i$ function uses Y, the set of identities that uniquely distinguishes any two agents of the MAS. $\Psi|_Y$ is Ψ personalized according to Y applied in the context of an agent a_i.

The $Memorization_i$ function allows a situated agent to register knowledge. This function is typed as follows:

$$\begin{aligned} &Memorization_i : P_i \times C_i \times S_i \rightarrow S_i \\ &Memorization_i(p_i, c_i, s_i) = s_i' \end{aligned}$$

The $Memorization_i$ function takes three arguments: the last percept, the last consumption and the actual internal state. With this $Memorization_i$ produces a new internal state. To what extent the agent uses the last percept and consumption to update his internal state depends on the implementation of $Memorization_i$ and is a choice of the designer.

To enable simultaneous actions, a situated agent must be capable to set up synchronization with other agents. This is done through the $Synchronization_i$ function:

[13] Ω may e.g. specify how a region behind an obstacle is out of the scope of a perceiving agent. Whereas physical sensing naturally incorporates such constraints, in software MASs the constraints have to be modeled explicitly

$Synchronization_i : P_i \rightarrow \Upsilon$
$Synchronization_i(p_i) = \upsilon_i$

$Synchronization_i$ establishes synchronization of a_i with every agent whose identity belongs to υ_i. An important design choice concerns the way how such sets are composed. The approach of the model is to let synchronization be the *natural consequence* of situatedness of agents and not be part of the agents *decision mechanism* (see below). This is reflected in the fact that the composition of a set of synchronized agents only depends on the actual perception of the agents and the implementation of the $Synchronization_i$ module. For regional synchronization υ_i represents the set of agents to which a_i is directly synchronized. Such synchronization is achieved through the execution of a synchronization protocol, i.e. the exchange of a structured set of synchronization messages $\mu_i \subseteq M$. The protocol is implemented in the $Synchronization_i$ module. The decision to separate synchronization setup from the agent's decision making is motivated by a basic principle of situated agents: keep decision making simple, avoid expensive reasoning but select an action in a reactive fashion according to the actual situation.

An essential capability of any agent is decision making, i.e. the ability of an agents to decide what action should be executed next. This functionality is reflected in the model in the $Decision_i$ function:

$Decision_i : P_i \times C_i \times S_i \rightarrow O_i^{Ag}$
$Decision_i(p_i, c_i, s_i') = o_i$

To decide what to do next, an agent uses his most recent percept and consumption, and the updated internal state. His decision results in the selection of an operator for execution in the environment.

We can now describe the complete behavior of a situated agent:

$Behavior_i : S_i \times \Sigma \times \Psi \rightarrow S_i \times O_i^{Ag} \times \Upsilon$
$Behavior_i(s_i, \sigma, \psi) = < s_i', o_i, \upsilon_i >$

with $s_i' = Memorization_i(p_i, c_i, s_i)$
$p_i = Perception_i(\sigma)$
$c_i = Consumption_i(\psi)$
$o_i = Decision_i(p_i, c_i, s_i')$
$\upsilon_i = Synchronization_i(p_i)$

The behavior of a situated agent can be expressed as follows: based on the internal state of the agent, the perception in the environment and a consumption, a situated agent updates his internal state and selects an operator for execution simultaneously with a set of directly synchronized agents.

6. Model for ongoing activities in the environment

This section describes a model for ongoing activities in the environment. Contrary to agents that are encapsulated entities capable of performing deliberated actions, ongoing activities simply operate into the environment according to the evolving state of the environment. As for agents, ongoing activities produce influences that are subject to the modeled laws of the MAS. Although ongoing activities exist independently of any particular agent, they typically originate from triggers invoked by agents. Fore example: a ball rolls after it was kicked by an agent, or a pheromone starts evaporating after it was

dropped by an ant. But clearly, this approach should not be generalized. Sometimes ongoing activities in the environment are likely to be modeled independent of agent intervention, e.g. the evolution of environmental variables such as temperature.

To describe ongoing activities in the environment we use the following definitions:

$Ac = \{d_1, \ldots d_j, \ldots d_m\}$: the set of all possible ongoing activities in the environment
$d \in D$: the set of ongoing activities that simultaneously operate into the environment, with $D \subseteq 2^{Ac}$
$O^{Ac} \subseteq O$: the operators of the ongoing activities in the MAS
$O_j^{Ac} \subseteq O^{Ac}$: the set of operators of ongoing activity d_j, i.e. the singleton o_j

We introduce a function $Operation$ that returns the operator for a particular ongoing activity in the environment[14]:

$$Operation : Ac \rightarrow O^{Ac}$$
$$Operation(d_j) = o_j$$

Depending on the state of the environment, the application of the operator of an ongoing activity returns a set of influences. This set is empty if the operator is not applicable in the current state of the environment, otherwise the set contains one influence for each instance of that particular ongoing activity. We elaborate on the application of operators of ongoing activities in the next section.

7. Model for a situated MAS

We are now ready to describe an integral model for a situated MAS that is in accordance with the theory for a dynamical system discussed in section 4 and that integrates the model for situated agents and ongoing activities in the environment as described in the previous two sections.

Intuitively, a situated MAS is a set of situated agents which can perform simultaneous actions in a dynamic environment. Formally, we define a situated MAS as an 20-tuple:

$$< Ag, Id, Ac, Op, \Sigma, \Psi, O, \Lambda, \Omega, Y, \Phi, G, D, Exec_i, Compose^{\Phi}, Apply_j, Collect^{D}, \uplus, React, Cycle >$$

In the following subsections, successively we discuss the contributions of activity in the MAS by the agents and the contributions of the ongoing activities in the environment. Next we integrate both contributions and discuss the reaction of the environment to the simultaneously performed activity in the system. Then we present a graphical overview of the integral formal model of situated MASs. We conclude with a discussion of a number of open issues of the model.

7.1. Contribution of activity by the agents in the MAS

We call a set of simultaneously acting agents a set of regional synchronized agents, in short a *region*. A region is denoted by $f_k \in F$, with $F \subseteq 2^{Ag}$ the set of all possible regions of the MAS. f_k is an equivalence class of agents related by the relation $isSychronizedWith$. $isSychronizedWith$ is the equivalence relation over a set of agents f_k that are directly or indirectly synchronized, i.e.:

[14] For convenience we sometimes will use Op as short for $Operation$.

$\forall a_x, a_y \in f_k : \exists\{b_1, \ldots, b_p\}$ with $b_i \in f_k : (a_x \in \alpha_{b_1} \wedge b_1 \in \alpha_{b_2} \wedge \ldots \wedge b_p \in \alpha_{a_y})$
with $\alpha_{x_i} \subseteq f_k$ the agents with whom x_i is directly synchronized, thus $\forall x_j \in \alpha_{x_i} : Id(x_j) \in \upsilon_{x_i}$

To describe the activity invoked by a region we use the following definitions:

G: the set of all possible influences that can be invoked into the environment
$G^{Ag} \subseteq G$: the set of all possible influences that can be invoked by agents
$g_i \in G_i^{Ag}$: an influence invoked by agent a_i with $G_i^{Ag} \subseteq G^{Ag}$ the set of all possible influences that can be invoked by a_i
$\gamma^f \in \Gamma^F$: a set of influences invoked by a region f, i.e.
$\gamma^f = \{g_i \mid g_i \in G_i^{Ag} \ \& \ a_i \in f \ \& \ \forall a_i, a_j \in f : Id(a_i) \neq Id(a_j)\}$
and $\Gamma^F \subseteq 2^{G^{Ag}}$ the set of all sets of influences that can be invoked by regions

The $Exec_i$ function for agent a_i is typed as follows:

$$Exec_i : O_i^{Ag} \times \Upsilon \times \Sigma \rightarrow G_i^{Ag} \times \Upsilon$$
$$Exec_i(o_i, \upsilon_i, \sigma) = < g_i, \upsilon_i >$$

with $o_i = Decision_i(p_i, c_i, s_i')$ and $\upsilon_i = Synchronization_i(p_i)$

In section 4, we defined $Exec^A$ as a representation of the activity invoked by a set of simultaneously acting actors. In the model for situated MASs we have split up this functionality in two parts. $Exec_i$ denotes the part of activity invoked by a single agent, i.e. g_i the influence for a_i. However, the operator selected by a_i must be parallel composed with the operators of the other agents of the region to which a_i belongs. Moreover, the simultaneous activity of the agents in a situated MAS may originate from more than only one region. Different regions can be active at the same time, apart from the other ongoing activities in the environment we discuss in the next section. The composition of all simultaneous activity of the agents is represented in the model by the $Compose^{\Phi}$ function. $\phi_k \in \Phi$ denotes a set of regions that act at the same time into the environment, with $\Phi = 2^F$ the set of all possible subsets of regions in the MAS. We denote an agent a_i who belongs to a region $f_k \in \phi_k$ by $a_i \triangleright \phi_k$. $Compose^{\Phi}$ is typed as follows:

$$Compose^{\Phi} : 2^{G^{Ag}} \times 2^{\Upsilon} \times 2^{C} \times \Psi \rightarrow 2^{\Gamma^F} \times \Psi$$
$$Compose^{\phi_k}((g_s, \ldots, g_w), (\upsilon_s, \ldots, \upsilon_w), (c_s, \ldots, c_w), \psi) = < \{\gamma_{f_k}\}^{\phi_k}, \ \psi^I >$$

with $\forall a_x \in \{a_s, \ldots, a_w\} : a_x \triangleright \phi_k$
$\{\gamma_{f_k}\}^{\phi_k} = \{\cup^{f_k} g_i\}$ for all $a_i \in f_k$ and all $f_k \in \phi_k$
$\psi^I = \psi - \bigcup^{\phi_k}(\cup^{f_k} c_i)$ for all $a_i \triangleright \phi_k$

$Compose^{\phi_k}$ composes the influence sets[15], one for each region f_k that belongs to ϕ_k, i.e. $\{\gamma_{f_k}\}^{\phi_k}$. This composition per region is based on the identity sets υ_i of directly synchronized agents, one set for each agent $a_i \triangleright \phi_k$. For a detailed explanation of region composition we refer to [34]. Besides, the $Compose^{\phi_k}$ function removes the consumptions consumed by all agents of ϕ_k, resulting in an intermediary set of consumptions denoted by ψ^I.

[15] Strictly speaking the domains for selected influences, directly synchronized agents and consumptions should be restricted to sets for *different* agents, however for convenience we use the more general sets of possible subsets for each type.

7.2. Contributions of ongoing activities in the environment

Simultaneously with the activity invoked by the agents of the active regions a number of ongoing activities may produce influences into the environment too. To describe the activity invoked in the environment by the ongoing activities we use the following definitions:

$G^{Ac} \subseteq G$: the set of all possible influences that can be invoked by ongoing activities
$\Gamma^{Ac} = 2^{G^{Ac}}$: the set of all influence sets that can be invoked by ongoing activities
$G_j^{Ac} \subseteq G^{Ac}$: the set of influences that can be invoked by an ongoing activity d_j
$\gamma_j \in \Gamma_j$: a set of influences invoked by an ongoing activity d_j, with $\Gamma_j \subseteq 2^{G_j^{Ac}}$
$\gamma^d \in \Gamma^D$: a set of influences simultaneously invoked by a set of ongoing activities d, i.e.
$\gamma^d = \{\gamma_j \mid \gamma_j \in \Gamma_j \ \& \ d_j \in d \ \& \ \forall d_x, d_y \in d : Op(d_x) \neq Op(d_y)\}$ and
$\Gamma^D \subseteq \Gamma^{Ac}$ the set of all influence sets that simultaneously can be invoked by ongoing activities

Similar to the agents, we have split up the simultaneous activity of ongoing activities in two parts. The $Apply_j$ function expresses the activity invoked by one ongoing activity and is typed as follows:

$$Apply_j : O_j^{Ac} \times \Sigma \ \rightarrow \ \Gamma_j$$
$$Apply_j(o_j, \sigma) = \gamma_j$$

with $o_j = Operation(d_j)$

$Apply_j$ produces γ_j, i.e. a set of influences, one for each instance of the ongoing activity d_j in the environment. The composition of all simultaneously performed ongoing activities in the environment is represented by the $Collect^D$ function and is typed as follows:

$$Collect^D : 2^{\Gamma^{Ac}} \rightarrow \Gamma^D$$
$$Collect^d(\gamma_r, \ldots, \gamma_v) = \gamma^d$$

with $\forall\, d_x \in \{d_r, \ldots, d_v\} : d_x \in d$
$\gamma^d = \cup^d \gamma_j$ for all $d_j \in d$

$Collect^d$ composes one influence set γ^d for the ongoing activities d in the system, i.e. simply the union of all influence sets γ_j each of them produced by one ongoing activity $d_j \in d$.

7.3. Reaction to the simultaneously performed activity in the environment

Since in the MAS model the $Collect^d$ function is responsible for collecting the influences of ongoing activities in the environment and the $Compose^{\phi_k}$ function assembles the influence sets of the active regions, we slightly modify the $\uplus$ operator that combines the influence sets resulting from both functions:

$$(\uplus) : ((2^{\Gamma^{Ac}}) \times (2^{G^{Ag}} \times 2^{\Upsilon} \times 2^{C} \times \Psi)) \rightarrow \Gamma^D \times 2^{\Gamma^F} \times \Psi$$
$$\uplus\,((\gamma_r, .., \gamma_v), ((g_s, .., g_w), (\upsilon_s, .., \upsilon_w), (c_s, .., c_w), \psi)) \ = < \gamma^d, \{\gamma_{f_k}\}^{\phi_k}, \psi^I >$$

or $\uplus\,((\gamma_r, .., \gamma_v), ((g_s, .., g_w), (\upsilon_s, .., \upsilon_w), (c_s, .., c_w), \psi)) =$
$\cup_T \ \circ \ (Collect^d(\gamma_r, \ldots, \gamma_v), Compose^{\phi_k}((g_s, \ldots, g_w), (\upsilon_s, \ldots, \upsilon_w), (c_s, \ldots, c_w), \psi))$
with $\cup_T : (\Gamma^D) \times (2^{\Gamma^F} \times \Psi) \ \rightarrow \ \Gamma^D \times 2^{\Gamma^F} \times \Psi$

Based on the composition of influence sets defined by the $\uplus$ operator, we can now define the $React$ function for a situated MAS. The $React$ function expresses the reaction of the environment to the simultaneously performed activity, i.e. (1) the sets of influences produced by the agents of the regions of ϕ_k and (2) the influences produced by the ongoing activities d in the environment. It is important to notice that since the activity of the agents per region is by definition local, there is no interference between actions of agents of different regions. However, since ongoing activities are not associated to particular regions, each influence of the set of ongoing activities can potentially interfere with any influence of an active agent. Therefore, in practice, the reaction of the environment to the simultaneously performed activity in one cycle can be calculated per region, each of them combined with the set of influences of the ongoing activities. $React$ is typed as follows:

$$React : (2^{\Lambda}, ||) \times \Sigma \times \Gamma^{D} \times 2^{\Gamma^{F}} \times \Psi \rightarrow \Sigma \times \Psi$$
$$React((\lambda_q \mid\mid .. \mid\mid \lambda_u), \sigma, \gamma^d, \{\gamma_{f_k}\}^{\phi_k}, \psi^I) = < \sigma', \psi' >$$
$$\text{with } \psi' = \psi^I + \psi^R$$

$React$ executes the influence sets of the active regions $\{\gamma_{f_k}\}^{\phi_k}$ and the ongoing activities γ^d in the MAS, given the state of the environment σ, the set of consumptions ψ^I and the parallel composed laws of the world $(\lambda_q \mid\mid .. \mid\mid \lambda_u)$. This results in a new state of the environment σ' and a new set of consumptions, say ψ^R. This latter set is added to the intermediary set of consumptions ψ^I delivered by the $Compose^{\phi^k}$ function, resulting in the next set of consumptions denoted by ψ'.

To conclude, we express the evolution of a situated MAS in the model, based on the dynamics of agents and environment:

$$Cycle : S \times \Sigma \times \Psi \rightarrow S \times \Sigma \times \Psi$$
$$Cycle((s_1 \times \ldots \times s_n), \sigma, \psi) = < (s'_1 \times \ldots \times s'_n), React((\lambda_q \mid\mid .. \mid\mid \lambda_u), \sigma, \gamma^d, \{\gamma_{f_k}\}^{\phi_k}, \psi^I) >$$

$$\text{with } \forall a_i \triangleright \phi_k : s'_i = Memorization_i(p_i, c_i, s_i) \text{ and } \forall a_j \not\triangleright \phi_k : s'_j = s_j$$
$$\gamma^d = Collect^d(\gamma_r, \ldots, \gamma_v)$$
$$\forall d_j \in d : \gamma_j = Apply_j(o_j, \sigma)$$
$$o_j = Operation(d_j)$$
$$< \{\gamma_{f_k}\}^{\phi_k}, \psi^I > = Compose^{\phi_k}((g_s, \ldots, g_w), (\upsilon_s, \ldots, \upsilon_w), (c_s, \ldots, c_w), \psi)$$
$$\forall a_i \triangleright \phi_k : < g_i, \upsilon_i > = Exec_i(o_i, \upsilon_i, \sigma)$$
$$o_i = Decision_i(p_i, c_i, s'_i)$$
$$\upsilon_i = Synchronization_i(p_i)$$
$$p_i = Perception_i(\sigma)$$
$$c_i = Consumption_i(\psi)$$

We can now express the dynamics of a situated MAS: the evolution of a situated MAS is defined as a sequence of cycles. In each cycle the $Cycle$ function transfers the dynamical state into the next dynamical state, i.e. it produces new internal states for the agents, a new state of the environment and a new set of consumptions. The new internal states result from the *memorization* of the active agents; the new state of the environment and the new set of consumptions are the result of the *reaction* of the environment to the *composition* of the parallel *executed decisions* of the agents per active region together with the *collection* of the *applied* operators of the ongoing activities in the environment, given the previous dynamical state and a set of parallel composed laws of the world. Figure 2 gives an overview of the integral

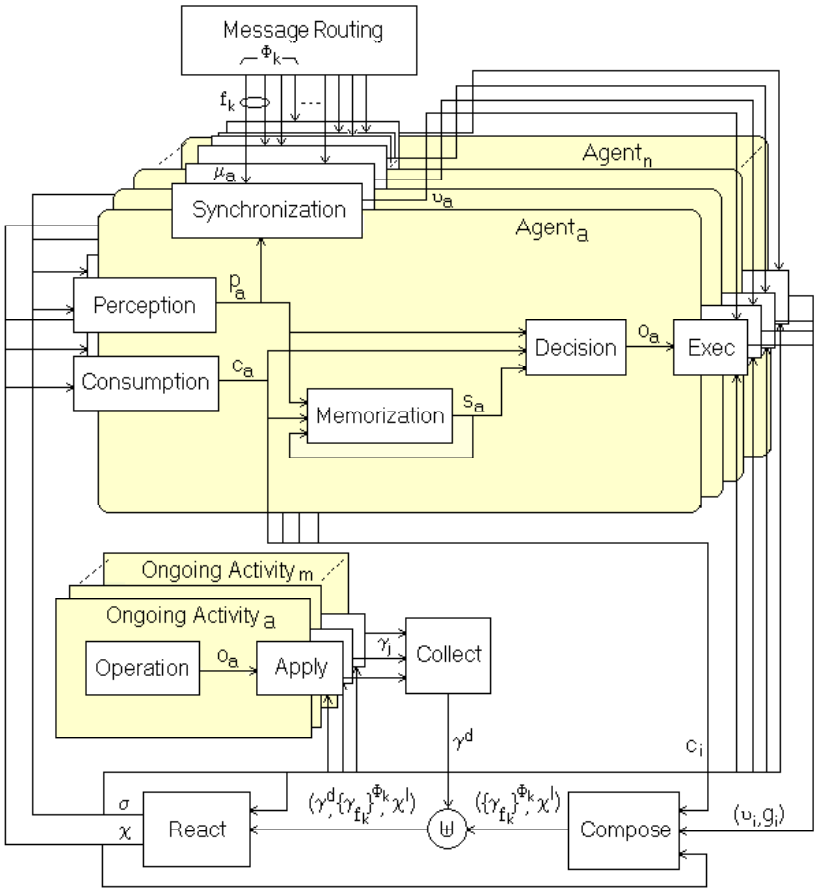

Figure 2. Graphical overview of the formal model for situated MASs.

model of situated MASs[16]. It is important to notice that the evolution of a MAS is typically non-deterministic. However, to avoid overloaded expressions we have not included non-determinism in the model. To illustrate one cause of non-determinism, we briefly look at the application of laws in the model[17]. Consider a MAS with agents that can pick up objects in the environment. Suppose now that two neighboring agents simultaneously try to pick up the same object. To resolve these concurrent actions, we further suppose that a law in the MAS determines that in such case a non-deterministic choice is made to grant the object to one of the two involved agents. As a consequence of the actions of the two agents the MAS evolves in one of two possible directions, depending on which of the agents gets the object. Based on such (and other kinds of) non-determinism we can reconsider the evolution of a MAS and say that each cycle in the evolution starts from a particular dynamical state that is transferred into one of a number (1 or more) of possible new dynamical states. As such, the dynamical state space of the MAS grows per cycle as a divergent tree and the evolution of the MAS expressed in our model as a sequence of cycles reflects one trace in this tree.

[16]The $MessageRouting$ module simply routes the synchronization messages from senders to receivers.
[17]We take this kind of non-determinism into account in the example application discussed in the next section.

7.4. Open issues

Dealing with time. In the presented model, the evolution of a situated MAS can be viewed as an abstract state machine that executes, in each cycle, the simultaneously performed activity in the environment at that point in time. This way the model ensures conceptual simultaneity, however in practice things happen mostly sequentially, e.g. on a single processor system. With respect to the practical implementation of MASs the question then arises how to reconcile conceptual simultaneous activity with physical activity. Although from the practical point of view this is an important problem, in this paper we only mention briefly some issues of it.

In the Ferber-Müller model the mapping of conceptual simultaneity with physical activity is straightforward. During each cycle all agents in the MAS have to decide about their next action. Only when all agents have invoked their influence the cycle is executed. A variant of this approach is to allow agents not to act in some cycles or let them execute a special $null\text{-}action$ when they have not yet concluded decision making. This is a way to imitate asynchronous evolution of the MAS. However, prudence is in order. An example of this approach is the RoboCup Soccer Server [18] where time is updated in discrete steps of 100 ms. All agents (players) are allowed to perform one action in each cycle. Players that have not acted in time are simply ignored. The simulator deals with interfering actions, e.g. when several players kick the ball together, all the kicks are applied to the ball and a resulting acceleration is calculated. This approach may be perfect for the RoboCup competition, however it is not a general solution for simultaneous actions. In fact, the Soccer Server only guarantees that actions that are *processed* during the physical time-slot of 100 ms are treated as if they happened together. It does not offer support for logical simultaneity of actions. When for some reason the actions of some of the simultaneously acting agents are not handled in time, e.g. because of network delays or because these agents got too little execution time on a sequential execution platform, the effects of the simultaneous actions of these agents are not taken into account.

In the model presented in this paper, the mapping of conceptual simultaneity onto physical activity is more complex. Agents act synchronously per region, however agents of different regions may be active at the same time. Therefore, in practice the mapping can be implemented in different ways. One solution is to give each agent of the MAS a fixed period of time for decision making and after that execute all *regions* that have concluded action selection. Contrary to the Robocup Soccer implementation discussed above, this solution ensures that simultaneous actions of agents are treated correctly since influences are executed on a per region basis. After all, with regional synchronization, the simultaneity of actions is established based on the locality of agents and that exactly matches with the radius of actions of different situated agents that may interfere with one another. However, ensuring that each agent gets a guaranteed slot of time for calculation requires a dedicated scheduling infrastructure. This problem is subject of current research in our working group.

Synchronization of agents and dynamics. Another remark concerns the quite complex problem of the synchronization of agent activity and other dynamics in the environment. The problem here is that in many cases the ongoing activities in the environment are associated with physical time. Pheromone evaporation for example is mostly implemented as a function of physical time [28]. However, load on a computer system typically fluctuates. As a consequence the decline of the pheromone strength does not evolve proportionally to the available computation time for the agents. These kinds of problems too are subject of ongoing research.

[18] http://sserver.sourceforge.net/downloads.html

Direct communication. The model presented in this paper does not explicitly deal with direct communication between agents. According to Ferber in [13], messages can be transmitted at the same time as the influences, and received with perception. However, we decided not to integrate communication in the model this way. The point is that the synchronization of messages with actions would pin down the model to handle actions in the environment and communicative acts between agents at one and the same pace. In [32] we demonstrated that this is not always desirable. Ongoing research intends to integrate direct communication in the model in a general way.

8. The model applied

In this section we apply the model for situated MAS to a simple example application. First we introduce the application. Next, we define a number of operators and laws. Then we follow a number of cycles in a possible evolution of the MAS. We have selected arbitrary regions to be active during the successive cycles with different cycles focusing on different kinds of interactions. For operators, laws, state representations etc., we use a simple STRIPS–like language [15], however the model is not bound to this language, any other representation language could be used.

8.1. A simple application

Fig. 3 depicts the MAS we use as a case to illustrate the model for situated MASs presented in the previous section. We take up and start following the activity in the example application from the depicted situation. The MAS has a discrete one-dimensional environment in which four agents live. The goal of the agents is to bring all the packets (black squares) from the *in* to the *out* place. Agents are able to pick up or put down a packet, but only from a neighboring place, the in and out places included. Neighboring agents can also transfer packets directly to one another. Such transfer requires a simultaneous action of the two agents, i.e. the agent who carries the packet must *pass* the packet while the accepting agent must *accept* the packet. An agent can carry at most one packet at a time. Furthermore, we allow agents that do not carry a packet to push forward a neighboring packet towards the out place. When the road is clear, the distance the packet moves depends on the applied force and the friction in the environment. When a moving packet bumps to an obstacle we take the simple approach and let the moving packet stop. Besides packets and agents, the in and out places too are assumed to be obstacles for moving packets. However, there is an exceptional case, i.e. an agent is able to catch a moving packet when he stretches his arms the moment when the packet arrives at his position. After such a catch the agent holds the packet.

The places 1 to 12 can contain at most one packet or one agent (that can carry a packet). Agents can make a step to a free neighboring place, but can not step into the in and out places. An important property of the agents is their limited view on the environment. For our example MAS, the *demarcation* Ω is fixed for all agents, and defined as a view–size of 2, illustrated for agent A in Fig. 3. The view–size marks out the range agents are able to perceive in their environment and thus also to setup regional synchronization with other agents. Such synchronization enables them to act simultaneously, e.g. to transfer a packet.

We represent state as a set of formulas of the form $p(c_1, \ldots, c_r)$, where p is a predicate and c_i are values. The initial dynamical state of the application, denote as δ_1, is defined as follows:

$$\delta_1 =< s_1, \sigma_1, \psi_1 >$$

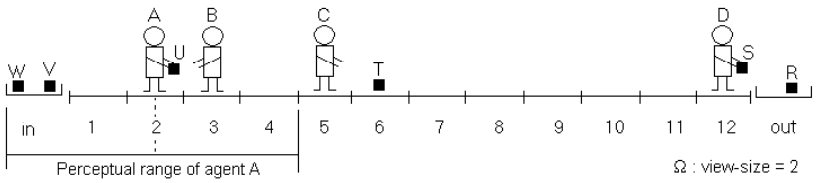

Figure 3. Initial situation of the MAS application, $f_1 = \{A, B, C\}$ active region.

with $\sigma_1 = \{loc(A,2), loc(B,3), loc(C,5), loc(D,12), at(T,6),$
$free(1), free(4), free(7), free(8), free(9), free(10), free(11),$
$carry(A), carry(D), at(V,in), at(W,in), at(R,out), friction(1)\}$
$\psi_1 = \{\}$
$s_1 = < s_A, s_B, s_C, s_D >$
$= < \{id(A), pos(2), hold(U)\}, \{id(B), pos(3)\},$
$\{id(C), pos(5)\}, \{id(D), pos(12), hold(S)\} >$

These definitions are self-explanatory. Note that the current set of consumptions ψ_1 is empty.

8.2. Operators and Laws

Before we elaborate on the evolution of the MAS, we first give an overview of the operators available for the agents and the ongoing activities in the environment, and the laws of the world that determine the effects of the performing of these operators[19].

8.2.1. Operators

According to the general definition for operators as defined in section 4, an operator is a 3-tuple:

$$o \in O : < name, conditions, influence >$$

For the applied representation language, these terms are described as follows: $name$ is an expression of the form $n(v_1, \ldots, v_s)$, with v_i variables that can appear both in $conditions$ and in $influence$. $conditions$ can be state representations or other boolean expressions with variables and values. $influence$ is a term of the form $n(p_1, \ldots, p_t)$ with p_i the parameters for the influence, i.e. a set of variables or values. For the example application, we define the following operators for the agents:

$< walk(x,dx), \{id(x), pos(lx), free(lx+dx), |dx| = 1\}, \{step(x,dx)\} >$
$< pickup(x,o), \{id(x), pos(lx), \neg hold(\_)$[20]$, at(o,lo), |lo{-}lx| = 1\}, \{pick(x,o,lo{-}lx)\} >$
$< putdown(x,dx), \{id(x), pos(lx), hold(o), |dx| = 1\}, \{put(x,o,dx)\} >$
$< push(x,o,p), \{id(x), pos(lx), at(o,lo), lo - lx = 1\}, \{press(x,o,p)\}$
$< passon(x,y,o), \{id(x), pos(lx), hold(o), loc(y,ly), |ly - lx| = 1\}, \{pass(x,y,o)\} >$
$< receive(x), \{id(x), \neg hold(\_)\}, \{accept(x)\} >$

[19] We only declare operators and laws that are relevant for the explanation of the MAS evolution in the following example.
[20] $\neg state$ denotes that $state$ does not hold, while _ denotes any possible value of a particular domain.

To clarify the operators for agents, we explain the operator $pickup(x, o)$ as an example. An agent x is able to pick up a packet o when the following conditions hold: (1) the identity of the agent is x; (2) the agent is positioned at location lx; (3) currently the agent does not hold any packet; (4) the packet o is positioned at location lo; (5) the packet and the agent are positioned next to each other, i.e. $\mid lo - lx \mid = 1$. When all these conditions hold, the agent invokes the influence $pick(x, o, lo - lx)$ into the environment, i.e. agent x performs an attempt to pick up the packet o located next to him ($lo - lx$ denotes in which direction the agent picks up the packet, i.e. the relative distance compared to his own actual location). Note that agent x can verify the conditions 1 to 3 based on his internal state, while the verification of conditions 4 and 5 includes information the agent has perceived in the environment.

For the ongoing activities in the environment, we define only a single operator:

$$< moveon(o), \{at(o, lo), speed(o, v), v > 0\}, \{move(o, v)\} >$$

Thus a packet o located at lo attempts to move on as long as it has a speed $v > 0$.

8.2.2. Laws

For the representation language, laws are defined as follows:

$$\lambda \in \Lambda : < influence\text{-}set, conditions, effects >$$

$influence\text{-}set$ is the set of influences involved in the law, i.e. a collection of possibly interfering influences originated from the execution of a set of parallel composed operators. $conditions$ can be state representations of the environment or other boolean expressions with variables and values. Every term in $conditions$ must hold to apply the $effects$, otherwise no effect at all is induced by the law. $effects$ is a set of formulas that expresses state changes and consumptions for agents. Additional effects are denoted with the keyword add, while the keyword rem refers to effects that remove existing state. The outcome of $effects$ can be a non–deterministic selection of different possibilities. Such selection is denoted as $\oplus$. For the example application, we define the following laws:

Simple Laws.

$step(x) : < \{step(x, dx)\}, \{loc(x, lx), free(lx + dx), |dx| = 1\},$
$\{rem : loc(x, lx), free(lx + dx); add : loc(x, lx + dx), free(lx)\} >$
$pick(x, o) : < \{pick(x, o, dx)\}, \{loc(x, lx), \neg carry(x), at(o, lx + dx), |dx| = 1\},$
$\{rem : at(o, lo); add : free(lo), carry(x), hold(x, o)\} >$
$import(x, o) : < \{pick(x, o, dx)\}, \{loc(x, lx), \neg carry(x), at(o, in), lx{+}dx{=}in, dx = -1\},$
$\{rem : at(o, in); add : carry(x), hold(x, o)\} >$
$put(x) : < \{put(x, o, dx)\}, \{loc(x, lx), carry(x), free(lx + dx), |dx| = 1\},$
$\{rem : carry(x), hold(x, o); add : at(o, lx + dx)\} >$
$deliver(x) : < \{put(x, o, dx)\}, \{loc(x, lx), carry(x), lx + dx = out, dx = 1\},$
$\{rem : carry(x), hold(x, o); add : at(o, out)\} >$
$press(x, o) : < \{press(x, o, p)\}, \{loc(x, lx), at(o, lo), lo - lx = 1, p > 0\},$
$\{add : speed(o, p)\} >$
$move(o) :< \{move(o, v)\}, \{at(o, lo), speed(o, v), friction(f), v > f, free(lo{+}i) \mid_{i=1}^{v}\},$
$\{rem : at(o, lo), free(lo + v), speed(o, v);$

$add : free(lo), at(o, lo + v), speed(o, v - f)\} >$[21]

$halt(o) : < \{move(o, v)\}, \{at(o, lo), speed(o, v), friction(f), v <= f, free(lo{+}i) \mid_{i=1}^{v}\},$
$\{rem : at(o, lo), free(lo + v), speed(o, v); add : free(lo), at(o, lo + v)\} >$

$bump(o) : < \{move(o, v)\},$
$\{at(o, lo), speed(o, v), r > 0, r < v, free(lo + i) \mid_{i=1}^{r}, \neg free(lo + r + 1)\},$
$\{rem : at(o, lo), free(lo + r), speed(o, v); add : free(lo), at(o, lo + r)\} >$

$block(o) :$
$< \{move(o, v)\}, \{at(o, lo), speed(o, v), \neg free(lo + 1)\}, \{rem : speed(o, v)\} >$

Joint Laws.

$transfer(x, y) :$
$< \{pass(x, y, o), accept(y)\}, \{loc(x, lx), loc(y, ly), |ly{-}lx| = 1, carry(x), \neg carry(y)\},$
$\{rem : carry(x), hold(x, o); add : carry(y), hold(y, o)\} >$

$catch(o, x) :$
$< \{move(o, v), accept(x)\},$
$\{at(o, lo), loc(x, lx), speed(o, v), dt = 1, v * dt >= lx - lo, free(i) \mid_{i=lo+1}^{lx-1}\},$
$\{rem : at(o, lo), speed(o, v); add : free(lo), carry(x), hold(x, o)\} >$

Concurrent Laws.

$clash(x, y) :$
$< \{step(x, dx), step(y, dy)\},$
$\{loc(x, lx), |dx| = 1, loc(y, ly), |dy| = 1, lx + dx = ly + dy, free(lx + dx)\},$
$\{rem : loc(x, lx), free(lx{+}dx); add : loc(x, lx{+}dx), free(lx)\}$
$\oplus \quad \{rem : loc(y, ly), free(ly{+}dy); add : loc(y, ly{+}dy), free(ly)\} >$

To clarify the laws, we explain the joint law $catch(o, x)$ as an example. In a catch two simultaneously performed influences are involved: a packet o is moving with a speed v and an agent x is prepared to accept a packet. The catch only succeeds when the following conditions hold: (1) the initial position of packet o is lo ; (2) agent x is located at lx ; (3) packet o has speed v ; (4) during a unit of time $dt = 1$, with a speed v, the packet moves at least over a distance $lx - lo$, i.e. from its original position lo to the agents position lx ; (5) all places between the initial position of the packet and the agent (i.e. from $lo + 1$ to $lx - 1$) are free. If all these conditions hold the law is applied, i.e. the packet moves from its starting position lo (this place is made free) into the hands of agent x who then holds the packet o.

Contrary to Ferber and Müller we do not impose the composition of laws to be commutative. Since the result of a law can be a non-deterministic selection of different outcomes we advocate that in general the requirement of commutativity of laws for MASs is too rigid. We propose that laws are applied in an ordered way. The ordering is based on the number of influences that are involved in a law, denoted by the notion of *level* of a law. The application of laws starts with the highest level laws. If a law can be found for which a subset of influences from the given set of simultaneously performed influences matches and for which the conditions of that law hold, that law is applied and the corresponding subset of influences is removed from the set of simultaneously performed influences. Subsequently this procedure is repeated for the remainder set of simultaneously performed influences with the laws of the same level. When

[21] $p(expr(v)) \mid_{v=start}^{stop}$ produces a set of condition terms, one for each substituted value v in $p(expr(v))$ from $start$ to $stop$. If $start > stop$, the resulting set is empty.

no more matching laws can be found at this level, the procedure is repeated for the laws at the next (lower) level, and so on. In the end, for the influences for which no matching compound law has been found the simple laws (with level 1) are applied. Note that the selection of a law at one level is made non-deterministically, even if the same set of influences is involved in different laws. This latter case typically occurs when a number of laws deal with different scenarios in the MAS. The simple laws $move(o)$, $halt(o)$, $bump(o)$ and $block(o)$ illustrate this case. In each of these laws the same influence $move(o, v)$ is involved for which, if several of these laws are applicable, one is selected non-deterministically.

As shown in the example, we classify laws according to the kind of interaction they are applicable to, with simple laws concerning independent simultaneous actions. This classification increases the readability of laws and can serve as a guideline for designers to structure complex sets of laws. Note that in the example the joint and concurrent laws all have level 2, while the simple laws have level 1.

Let us finally remark that the procedure to apply laws we put forward in this paper is only a start to tackle the problem. Further research is necessary to fully disentangle this quite complex problem.

8.3. Evolution of the MAS

Starting from the initial situation (δ_1 in section 8.1, see also Figure 3), we now follow a number of cycles in a potential evolution of the MAS. In each cycle one or more regions of agents are active, together with other possible ongoing activities in the environment.

8.3.1. Cycle 1: $< s_1, \sigma_1, \psi_1 > \overset{\phi_1, d_1}{\rightarrow} < s_2, \sigma_2, \psi_2 >$ *with* $\phi_1 = \{f_1\}, f_1 = \{A, B, C\}$ *and* $d_1 = \{\}$

In the first cycle only region f_1 is active and transfers the dynamical state. First we follow the individual decision making of each agent of the region. For a description of σ_1 and ψ_1, see section 8.1.

$Agent_A$ with $s_A = \{id(A), pos(2), hold(U)\}$
$Perception_A(\sigma_1) : p_A = \{at(V, in), at(W, in), free(1), loc(A, 2), loc(B, 3), free(4), carry(A)\}$
$Consumption_A(\psi_1) : c_A = \{\}$
$Synchronization_A(p_A) : v_A = \{A, B\}$
$Memorization_A(p_A, c_A, s_A) : s'_A = \{id(A), pos(2), hold(U)\}$
$Decision_A(p_A, c_A, s'_A) : o_A = passon(A, B, U)$

$Agent_B$ with $s_B = \{id(B), pos(3)\}$
$Perception_B(\sigma_1) : p_B = \{free(1), loc(A, 2), loc(B, 3), free(4), loc(C, 5), carry(A)\}$
$Consumption_B(\psi_1) : c_B = \{\}$
$Synchronization_B(p_B) : v_B = \{B, A, C\}$
$Memorization_B(p_B, c_B, s_B) : s'_B = \{id(B), pos(3)\}$
$Decision_B(p_B, c_B, s'_B) : o_B = receive(B)$

$Agent_C$ with $s_C = \{id(C), pos(5)\}$
$Perception_C(\sigma_1) : p_C = \{loc(B, 3), free(4), loc(C, 5), at(T, 6), free(7)\}$
$Consumption_C(\psi_1) : c_C = \{\}$
$Synchronization_C(p_C) : v_C = \{C, B\}$
$Memorization_C(p_C, c_C, s_C) : s'_C = \{id(C), pos(5)\}$
$Decision_C(p_C, c_C, s'_C) : o_C = push(C, T, 3)$

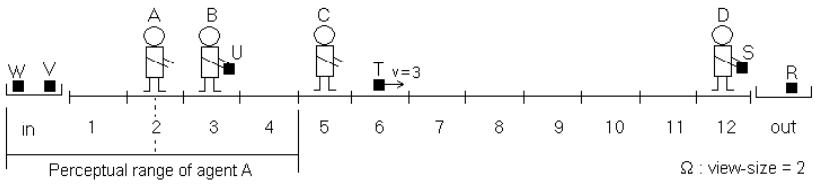

Figure 4. $f_2 = \{D\}$ active region; $d_2 = \{T\}$ ongoing activity.

So, in this cycle, $Agent_A$ decides to pass the packet he carries to his neighbor, $Agent_B$[22]. This latter is prepared to accept the packet. $Agent_C$ decides to push the packet T next to him. Thus in terms of simultaneous actions, $Agent_A$ and $Agent_B$ perform *joint actions*, while both these actions are *independent* of $Agent_C$'s action. We now can determine the reaction of the environment.

$$
\begin{aligned}
&o_{f_1} = \{passon(A,B,U), receive(B), push(C,T,3)\}\\
&Exec^{f_1}((o_{f_1}, ||), \sigma_1) = \gamma_{f_1}\\
&\qquad\qquad = \{pass(A,B,U), accept(B), press(C,T,3)\}\\
&\{\gamma_{f_1}\}^{\phi_1} = \{\{pass(A,B,U), accept(B), press(C,T,3)\}\}\\
&\psi^E = \{\}\\
&\psi^I = \psi_1 - \psi^E\\
&\quad = \{\}\\
&o_{d_1} = \{\}\\
&Apply^{d_1}(\{\}, \sigma_1) = \gamma^{d_1}\\
&\qquad\qquad = \{\}\\
&React(\lambda_1, \sigma_1, \gamma^{d_1}, \{\gamma_{f_1}\}^{\phi_1}, \psi^I) = < \sigma_2, \psi_2 >\\
&\lambda_1 = \{transfer(A,B), press(C,T)\}\\
&effects : \{rem : carry(A), hold(A,U); add : carry(B), hold(B,U), speed(T,3)\}\\
&\text{with } \psi^R = \{rem : hold(A,U); add : hold(B,U)\}
\end{aligned}
$$

$$
\begin{aligned}
&s_2 = < s'_A, s'_B, s'_C, s_D, s_E >^{23}\\
&\quad = < \{id(A), pos(2), hold(U)\}, \{id(B), pos(3)\}, \{id(C), pos(5)\}, \{id(D), pos(12),\\
&\qquad hold(S)\} >\\
&\sigma_2 = \{loc(A,2), loc(B,3), loc(C,5), loc(D,12), at(T,6), speed(T,3),\\
&\qquad free(1), free(4), free(7), free(8), free(9), free(10), free(11),\\
&\qquad carry(B), carry(D), at(V,in), at(W,in), at(R,out), friction(1)\}\\
&\psi_2 = \psi^I + \psi^R\\
&\quad = \{rem : hold(A,U); add : hold(B,U)\}
\end{aligned}
$$

The new situation of the application after the first cycle is executed is depicted in Figure 4.

[22] In fact we abstract from the implementation of the $Decision_i$ module and assume that this is the selected action.

[23] Note that the update of the internal state of the agents is based on the last percepts/consumptions. The effects of the last cycle are assimilated only after the next perception/consumption.

8.3.2. Cycle 2: $< s_2, \sigma_2, \psi_2 > \overset{\phi_2, d_2}{\rightarrow} < s_3, \sigma_3, \psi_3 >$ $with\ \phi_2 = \{f_2\}, f_2 = \{D\}\ and\ d_2 = \{T\}$

In the second cycle the region with $Agent_D$ and the moving packet T are active. $Agent_D$ acts asynchronously with respect to the other agents, because there is no other agent inside his perceptual range. $Agent_D$ decides to deliver the packet S he carries at the out place. Packet T moves with an initial speed of 3 from place 6 towards the out place. Due to the friction of the environment, during this movement, the speed of T is reduced from 3 to 2. We skip the detailed individual decision making of the agent and look how the environment reacts to the application of the active operators.

$$
\begin{aligned}
&o_{f_2} = \{putdown(D, +1)\}\\
&Exec^{f_2}((o_{f_2}, ||), \sigma_2) = \gamma_{f_2}\\
&\qquad\qquad = \{put(D, S, +1)\}\\
&\{\gamma_{f_2}\}^{\phi_2} = \{\{put(D, S, +1)\}\}\\
&\psi^E = \{\}\\
&\psi^I = \psi_2 - \psi^E\\
&\quad = \{rem : hold(A, U); add : hold(B, U)\}\\
&o_{d_2} = \{moveon(T)\}\\
&Apply^{d_2}(\{moveon(T)\}, \sigma_2) = \gamma^{d_2}\\
&\qquad\qquad = \{move(T, 3)\}\\
&React(\lambda_2, \sigma_2, \gamma^{d_2}, \{\gamma_{f_2}\}^{\phi_2}, \psi^I) = < \sigma_3, \psi_3 >\\
&\lambda_2 = \{deliver(D), move(T)\}\\
&effects : \{rem : carry(D), hold(D, S), at(T, 6), free(6 + 3), speed(T, 3);\\
&\qquad add : at(S, out), at(T, 6 + 3), free(6), speed(T, 3 - 1)\}\\
&\text{with } \psi^R = \{rem : hold(D, S)\}
\end{aligned}
$$

$$
\begin{aligned}
&s_3 = < s_A, s_B, s_C, s'_D >\\
&\quad = < \{id(A), pos(2), hold(U)\}, \{id(B), pos(3)\}, \{id(C), pos(5)\}, \{id(D), pos(12),\\
&\qquad hold(S)\} >\\
&\sigma_3 = \{loc(A, 2), loc(B, 3), loc(C, 5), loc(D, 12), at(T, 9), speed(T, 2),\\
&\qquad free(1), free(4), free(6), free(7), free(8), free(10), free(11),\\
&\qquad carry(B), at(V, in), at(W, in), at(R, out), at(S, out), friction(1)\}\\
&\psi_3 = \psi^I + \psi^R\\
&\quad = \{rem : hold(A, U), hold(D, S); add : hold(B, U)\}
\end{aligned}
$$

The simultaneous actions in this cycle are *independent actions*. There was no interference between the action of $Agent_D$ and the movement of T in the environment. The new situation of the application after cycle 2 is executed is depicted in Figure 5.

8.3.3. Cycle 3: $< s_3, \sigma_3, \psi_3 > \overset{\phi_3, d_3}{\rightarrow} < s_4, \sigma_4, \psi_4 >$ $with\ \phi_3 = \{f_3\}, f_3 = \{A, B, C\}\ and\ d_3 = \{T\}$

In the third cycle, again the region with agents A,B and C is active while packet T still moves on. Agent A decides to step towards the *in* place to pick up another packet, while agent B and C both decide to step towards each other. Clearly this latter leads to a nasty collision, but according to the laws fortunately without further consequences for the agents.

The following effects result from the agents decisions and the moving packet:

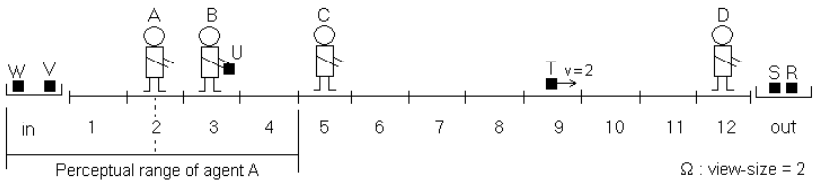

Figure 5. $f_3 = \{A, B, C\}$ active region; $d_3 = \{T\}$ ongoing activity.

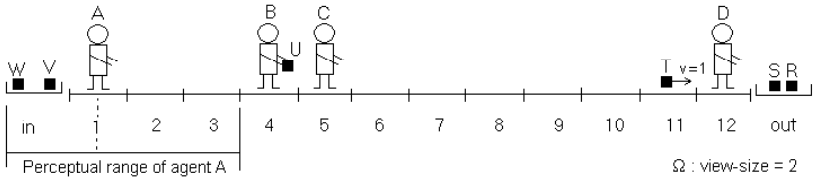

Figure 6. $f_{41} = \{A\}$ and $f_{42} = \{D\}$ active regions; $d_4 = \{T\}$ ongoing activity.

$o_{f_3} = \{walk(A,-1), walk(B,+1), walk(C,-1)\}$
$Exec^{f_3}((o_{f_3}, ||), \sigma_3) = \gamma_{f_3}$
$= \{step(A,-1), step(B,+1), step(C,-1)\}$
$\{\gamma_{f_3}\}^{\phi_3} = \{\{step(A,-1), step(B,+1), step(C,-1)\}\}$
$\psi^E = \{rem : hold(A,U); add : hold(B,U)\}$
$\psi^I = \psi_3 - \psi^E$
$= \{rem : hold(D,S)\}$
$o_{d_3} = \{moveon(T)\}$
$Apply^{d_3}(\{moveon(T)\}, \sigma_3) = \gamma^{d_3}$
$= \{move(T,2)\}$
$React(\lambda_3, \sigma_3, \gamma^{d_3}, \{\gamma_{f_3}\}^{\phi_3}, \psi^I) = < \sigma_4, \psi_4 >$
$\lambda_3 = \{clash(B,C), move(T)\}$
$effects : \{rem : loc(B,3), free(4), at(T,9), free(9+2), speed(T,2);$
$add : loc(B,4), free(3), at(T,9+2), free(9), speed(T,2-1)\}$
with $\psi^R = \{\}$

$s_4 = < s'_A, s'_B, s'_C, s_D, s_E >$
$= < \{id(A), pos(2)\}, \{id(B), pos(3), hold(U)\}, \{id(C), pos(5)\}, \{id(D), pos(12),$
$hold(S)\} >$
$\sigma_4 = \{loc(A,1), loc(B,4), loc(C,5), loc(D,12), at(T,11), speed(T,1),$
$free(2), free(3), free(6), free(7), free(8), free(9), free(10),$
$carry(B), at(V,in), at(W,in), at(R,out), at(S,out), friction(1)\}$
$\psi_4 = \psi^I + \psi^R$
$= \{rem : hold(D,S)\}$

The new situation of the application resulting of the execution of cycle 3 is depicted in Figure 6.

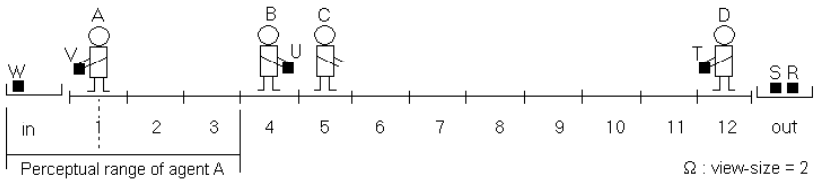

Figure 7. Resulting situation of the MAS application.

8.3.4. Cycle 4: $< s_4, \sigma_4, \psi_4 > \overset{\phi_4, d_4}{\rightarrow} < s_5, \sigma_5, \psi_5 >$ *with* $\phi_4 = \{\{A\}, \{D\}\}$ *and* $d_4 = \{T\}$

In the fourth cycle, two regions are active, the first with $Agent_A$, the second with $Agent_D$. Besides, packet T is still moving on. In this cycle, $Agent_A$ picks up a new packet form the *in* place, while $Agent_D$ catches the moving packet T that arrives at his position. These latter are *joint actions*, while $Agent_A$ acts *independently* of the interaction between $Agent_D$ and packet T. To determine the effects of the actions in the environment we perform $Exec$, $Apply$ and $React$ in the current dynamical state.

$$
\begin{aligned}
& o_{f_{41}} = \{pickup(A, V)\}; o_{f_{42}} = \{receive(D)\} \\
& Exec^{f_{41}}((o_{f_{41}}, ||), \sigma_4) = \gamma_{f_{41}} \\
& \qquad\qquad = \{pick(A, V, -1)\} \\
& Exec^{f_{42}}((o_{f_{42}}, ||), \sigma_4) = \gamma_{f_{42}} \\
& \qquad\qquad = \{accept(D)\} \\
& \{\gamma_{f_{41}}, \gamma_{f_{42}}\}^{\phi_4} = \{\{pick(A, V, -1)\}, \{receive(D)\}\} \\
& \psi^E = \{rem : hold(D, S)\} \\
& \psi^I = \psi_4 - \psi^E \\
& \quad = \{\} \\
& o_{d_4} = \{moveon(T)\} \\
& Apply^{d_4}(\{moveon(T)\}, \sigma_4) = \gamma^{d_4} \\
& \qquad\qquad = \{move(T, 1)\} \\
& React(\lambda_4, \sigma_4, \gamma^{d_4}, \{\gamma_{f_{41}}, \gamma_{f_{42}}\}^{\phi_4}, \psi^I) = < \sigma_5, \psi_5 > \\
& \lambda_4 = \{catch(D, T), import(A, V)\} \\
& effects : \{rem : at(T, 11), speed(T, 1), at(V, in); \\
& \qquad add : carry(D), hold(D, T), free(11), carry(A), hold(A, V)\} \\
& \text{with } \psi^R = \{add : hold(D, T), hold(A, V)\}
\end{aligned}
$$

$$
\begin{aligned}
& s_5 = < s'_A, s_B, s_C, s'_D > \\
& \quad = < \{id(A), pos(1)\}, \{id(B), pos(3), hold(U)\}, \{id(C), pos(5)\}, \{id(D), pos(12)\} > \\
& \sigma_5 = \{loc(A, 1), loc(B, 4), loc(C, 5), loc(D, 12), \\
& \qquad free(2), free(3), free(6), free(7), free(8), free(9), free(10), free(11) \\
& \qquad carry(A), carry(B), carry(D), at(W, in), at(R, out), at(S, out), friction(1)\} \\
& \psi_5 = \psi^I + \psi^R \\
& \quad = \{add : hold(A, V), hold(D, T)\}
\end{aligned}
$$

With the execution of this cycle, we conclude our investigation of the evolution of the example application. The resulting, but not yet finished situation of the application is depicted in Figure 7.

9. Evaluation

The model for situated MASs we presented in this paper deals with complex interactions in situated MASs, i.e. (1) interactions between agents through the environment, (2) interactions between an agent and an ongoing activity in the environment and (3) interactions between different ongoing activities in the environment. The model keeps the balance between two fundamental properties for the situated agents: autonomy for the agents to decide when to act and the ability to perform simultaneous actions.

In the model, simultaneous actions are supported through regional synchronization. Contrary to centralized synchronization, regional synchronization does not impose centralized control. With regional synchronization each agent is equipped with a local synchronizer that is responsible to setup synchronization locally with neighboring colleagues, i.e. the candidates for direct interaction. Reviewing the Execution-Reaction cycle as depicted in Fig.1 we can summarize that in the Ferber-Müller model all agents run through the execution-reaction cycle synchronously, while in our model subsets of locally synchronized agents can run through the cycle asynchronously. Both models support simultaneous actions, however the models differ in the granularity of simultaneously acting groups. In the Ferber-Müller model all agents of the MAS act simultaneously, while in our model agents act simultaneously per region. In exchange for decentralization, regional synchronization requires infrastructure and implies computational and commutative costs. In [33], we examined this matter in detail.

With respect to scalability, the cost to calculate reactions to a set of influences in the model depends on the size of regions. Apart from the influences produced by the ongoing activities in the environment, the influences of agents potentially interfere only within the same region. This reduces the average cost from $O(n^2)$ (this is the cost in the Ferber-Müller model, see section 2) to $O(n * rs)$ with n the population of agents in the MAS and rs the region-size, i.e. the average number of agents per region. So when the composition of regions is designed well, i.e. when the activity in the MAS is well localized, the model results in better performance and scalability. A side effect of regional synchronization is that the environment is responsible for composing regions of synchronized agents. In the model, this functionality is achieved by the $Compose^{\Phi}$ module.

When designing the model for an individual situated agent, we took the approach to balance between modeling fundamental aspects of situated agents and guaranteeing maximal flexibility. We integrated functionality for memorization in the agent model, however this should be seen as a generalization. For agents without memory, the $Memorization_i$ module can simply be ignored. Another topic concerns heterogeneity of the agents in the MAS. In itself, the model does not pronounce anything about the internal implementation of the separate agent modules. Besides, the model explicitly supports heterogeneity at different levels: perceptual capabilities, the kind of consumptions an agent is able to consume, the capabilities to memorize and the set of operators an agent is able to execute.

In comparison with the Ferber-Müller model, the complexity of the presented model is higher. However, this is not directly the consequence of the different approach, but rather the result of the adopted level of abstraction. In fact, numerous aspects that are implicitly present in the Ferber-Müller model are made explicit in the model presented in this paper. Examples are the explicitness of synchronization, the explicit modeling of ongoing activities in the environment, the modeling of demarcation of perception, the identification of the agents and the integration of personalized consumptions.

To conclude, we point the interested reader to [34] that elaborates on a concrete implementation of the discussed model.

10. Conclusions

In this paper we presented a model for situated MASs with regional synchronization. This model formally describes an abstract architecture for situated agents and the environment in which the agents live. The model supports simultaneous actions of agents as well as other ongoing activities in the environment that happen independently of agent intervention, e.g. the evaporation of a pheromone. The model builds upon the theory of influences and reactions to influences developed by Ferber and Müller.

The evolution of a MAS is expressed as a sequence of cycles. In each cycle the dynamical state is transferred to a new dynamical state. In the model of Ferber and Müller, dynamical state is composed of state and influences. This model takes an environment-centered point of view since it imposes all agents to produce new influences in each cycle of the MAS evolution, thus all agents act at one global pace. Contrary, the model we present takes an agent-centered point of view. In our model dynamical state is composed of state and consumptions. Now the agents themselves take the initiative to act, starting with perceiving the local environment and consuming a consumption. In order to act simultaneously, agents must synchronize. We introduced regional synchronization that enables agents to synchronize with only colleagues in their region. As such, synchronization is locally established by the agents themselves and tuned to the scope of interactions between agents. When the composition of regions is designed well, this results, in comparison to centralized synchronization, in improved autonomy of the agents and better scalability of the MAS.

In the paper, we applied the model to a simple MAS application. We showed how the model can be instantiated in practice. Then we illustrated a possible evolution of the MAS by means of different kinds of interactions, including the interaction between an agent and an ongoing activity in the environment.

We employ the model in our research group as a basis for engineering a common platform for situated MASs. Our experiences with the formalized approach point to the following advantages: (1) the model explicitly and rigorously specifies the core concepts of situated multi-agent systems ; (2) the decomposition has a precise semantics enabling further refinement towards implementation and (3) the model serves as an excellent framework for communication and discussion. A number of topics we abstract from in the presented model are subject of future work. Currently we work on a general solution to deal with the parallel composition of laws and investigate how we can integrate direct communication in the model in a general way. Other topics of research concern timing issues, such as the synchronization between agent activities and ongoing activities in the environment and the mapping of conceptual simultaneous activity to physical activity.

11. Acknowledgments

We would like to thank the members of the AgentWise task force at DistriNet labs, K.U.Leuven for the many valuable discussions that have contributed to the work presented in this paper. Also a word of appreciation goes to Frank Piessens, Wouter Joosen and Elke Steegmans for their useful comments on the paper. And last but not least, we would like to thank the anonymous reviewers for their accurate remarks on the initial version of this paper. By taking their critical comments into account we were able to improve the paper significantly.

References

[1] Allen, J.F., Ferguson, G.: *Actions and Events in Interval Temporal Logic*. Journal of Logic and Computation, Special Issue on Action and Processes, 1994.

[2] Babaoglu, O., Meling, H., Montresoret, H.: *Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems*. Proceedings of the 22th International Conference on Distributed Computing Systems, Vienna, Austria, 2002.

[3] Bonabeau, E., H'naux, F., Gu'rin, S., Snyers, D., Kuntz, P., Theraulaz, G.: *Routing in Telecommunications Networks with Ant-Like Agents*: 60-71, IATA, 1998.

[4] Bonabeau, E., Sobkowski, A., Theraulaz, G., Deneubourg, J.L.: *Adaptive Task Allocation Inspired by a Model of Division of Labour in Social Insects*. Biocomputing and Emergent Computation: 36-45, World Scientific, 1998.

[5] Boutilier, C., Brafman, R.I.: *Partial-Order Planning with Concurrent Interacting Actions*. Journal of Artificial Research 14: 105-136, Access Foundation and Morgan Kaufmann, 2001.

[6] Bratman, M.E., Israel D.J., Pollack M.E.: *Plans and resource-bounded practical reasoning*. Computational Intelligence 4: 349-355, 1988.

[7] Brooks, R. A.: *Intelligence Without Representation*, Prints of the Workshop in Foundations of Artificial Intelligence, Dedham, MA, 1987.

[8] Brooks, R. A.: *Intelligence Without Reason*, MIT AI Lab Memo No. 1293, 1991.

[9] Deneubourg, J.L., Aron, A., Goss, S., Pasteels, J.M., Duerinck, G.: *Random Behavior, Amplification Processes and Number of Participants: How they Contribute to the Foraging Properties of Ants*. In Physics 22(D): 176-186, 1986.

[10] Dorigo, M., Gambardella, L.M.: *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*. In IEEE Transactions on Evolutionary Computation, 1, 1: 53-66, 1997.

[11] Drogoul, A., Ferber, J.: *Multi-Agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies*. Decentralized A.I. 4, Elsevier North-Holland, 1992.

[12] Ferber, J.: *Un modele de l'action pour les systemes multi-agents*. Journees sur les systemes multi-agents et l'intelligence artificielle distribue, Voiron, 1994.

[13] Ferber, J.: *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*, ISBN 0-201-36048-9, Addison-Wesley, Great Britain, 1999.

[14] Ferber, J., Müller, J.P.: *Influences and Reaction: a Model of Situated Multiagent Systems*. Proceedings of the 2th International Conference on Multi-agent Systems, AAAI Press, Japan, 1996.

[15] Fikes, T., Nilsson, N.J.: *STRIPS: a New Approach to the Application of Theorem Proving to Problem-solving*. Artificial Intelligence 2 (3-4): 189-208, 1971.

[16] Genesereth, M. R., Nilsson, N.: *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, 1987.

[17] Griffiths, N., Luck, M., d'Iverno, M.: *Cooperative Plan Annotation through Trust*. P. McBurney, M. Wooldridge (eds.), Workshop Notes of UK Workshop on Multi-agent Systems, Liverpool, 2002.

[18] JASSS: *Journal of Artificial Societies and Social Simulation*. http://jasss.soc.surrey.ac.uk/JASSS.html

[19] Kowalski, F., Sadri, R.A.: *Towards a unified agent architecture that combines rationality with reactivity*. Proceedings of International Workshop on Logic in Databases, San Miniato, Italy, LNCS 1154, 1996.

[20] Macy, M., Willer, R.: *From Factors to Actors: Computational Sociology and Agent-Based Modeling*. Annual Review of Sociology 28, 2002.

[21] Maes, P.: *Modeling Adaptive Autonomous Agents*. Artificial Life Journal, 1 (1-2): 135-162, MIT Press, Cambridge, MA, 1994.

[22] Maes, P.: *Situated Agents can have Goals*. Ed. P. Maes, Designing Autonomous Agents: 49-70, MITT Press, Cambridge, MA, 1990.

[23] Parunak, V.: *Go to the Ant: Engineering Principles from Natural Agent Systems*. Annals of Operations Research 75: 69-101, 1997.

[24] Parunak, V., Baker, A.D., Clark, S.J.: *The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design*. Proceedings of Workshop on Agent-Based Manufacturing, ICAA98, Minneapolis, MN, 1998.

[25] Rao, A.S.: *AgentSpeak(L): BDI agents speak out in a logical computational language*. Proceedings of 7th European Workshop on Modeling Agents in a Multi-Agent World, LNAI volume 1038: 42-55, 1996.

[26] Rouchier, J.: *Review of Multi-agent Systems: An Introduction to Distributed Artificial Intelligence, J. Ferber, 1999*. JASSS, Volume 4, Issue 2, see [18], 2001.

[27] Sauter, J.A., Parunak, V.: *ANTS in the Supply Chain*. Proceedings of Workshop on Agent based Decision Support for Managing the Internet-Enabled Supply Chain, Agents 99, Seattle, WA, 1999.

[28] Sauter, J.A., Matthews, R., Parunak, V., Brueckner, S.: *Evolving Adaptive Pheromone Path Planning Mechanisms*. Proceedings of the First International Conference on Autonomous Agents and Multi-agent Systems: 434-441, Bologna, Italy, 2002.

[29] Schelfthout, K., Holvoet, T.: *'To do or not to do': The individual's Model for Emergent Task Allocation*. Proceedings of Adaptive Agents and Multi-Agent Systems, London, UK, 2002.

[30] Steels, L.: *Cooperation between distributed agents through self-organization*. Proceedings of the First European Workshop on Modeling Autonomous Agents in a Multi-Agent World: 175-196, Elsevier Science Publishers, Holland, 1990.

[31] Wavish, P.R., Connah, D.M.: *Representing Multi-Agent Worlds*. ABLE, Technical Note: TN2964, Philips Research Laboratories, 1990.

[32] Weyns, D., Holvoet, T.: *Look, Talk and Do: A Synchronization Scheme for Situated Multi-Agent Systems*. P. McBurney, M. Wooldridge (eds.), Workshop Notes of UK Workshop on Multi-agent Systems, Liverpool, 2002.

[33] Weyns, D., Holvoet, T.: *Regional Synchronization for Simultaneous Actions in Situated Multi-Agent Systems*. Proceedings of 3th International/Central and Eastern European Conference on Multi-Agent Systems, CEEMAS, Prague, Czech Republic, LNAI 2691: 497-511, 2003.

[34] Weyns, D., Holvoet, T.: *Model for Simultaneous Actions in Situated Multi-Agent Systems*. First German Conference on Multi-Agent System Technologies, MATES, Erfurt, Germany, LNAI 2831: 105-119, 2003.

[35] Wooldridge, M.: *An Introduction to MultiAgent Systems*, ISBN 0-471-49691-X. John Wiley and Sons, Ltd. England, 2002.

[36] Wooldridge, M., Jennings. N.R. : *Intelligent Agents: theory and practice*. The Knowledge Engineering Review, 10(2): 115-152, 1995.