A Model for Active Perception in Situated Multi-agent Systems

Danny Weyns, Elke Steegmans and Tom Holvoet

AgentWise, DistriNet, Department of Computer Science,
K.U.Leuven, B-3001 Heverlee, Belgium
{danny.weyns,elke.steegmans,tom.holvoet}@cs.kuleuven.ac.be
www.cs.kuleuven.ac.be/~danny/home.html

Abstract. In this paper we present a generic model for *active perception* in situated multi-agent systems. Active perception enables an agent to direct its perception at the most relevant aspects in the environment according to its current task. The model decomposes perception into three functionalities: sensing, interpreting and filtering.
Sensing takes place at the agent-environment interface and maps the state of the environment to a representation. This mapping depends on two factors. First the sensing agent can select a set of *foci*. Focus selection enables an agent to sense specific types of data in the environment. Second, the representation of the state is composed according to a set of *perceptual laws*. Perceptual laws enforce domain specific constraints on perception. Whereas physical sensing naturally incorporates such constraints, in software multi-agent systems the constraints have to be modeled explicitely.
The second functionality is interpreting. Agents interpret representations by means of *descriptions*. Descriptions are blueprints that map representations onto percepts. Percepts are expressions that can be understood by the internal machinery of the agent.
The third functionality of perception is filtering. By selecting a set of *filters* an agent is able to improve its perception by restricting the perceived data according to specific context relevant selection criteria.

1 Introduction

Although most researchers in the multi-agent community agree on the fact that *environment* is an essential part of any multi-agent system (MAS), most of them neglect to integrate environment as a primary abstraction in MAS. In fact most of the proposed models or tools for MAS reduce the concept of environment to a passive message delivering system, see e.g. [8][28][24][3].

Researchers working in the context of *situated MASs*[1] traditionally emphasize the importance of the environment and provide an explicit model for it, see e.g. [23][14][13][7][30]. In situated MASs, the environment embeds the agents

[1] Alternative descriptions are behavior-based agents [4], adaptive autonomous agents [11] or hysteretic agents [9][7].

and other domain objects, each on an individual position. Depending on the modeled domain, the environment can have very different topologies, such as grids, environments with non-discrete coordinates or arbitrary graph structures. In the approach of situated MASs, agents and environment constitute complementary parts of a multi-agent world that can mutually affect each other. As such an environment itself is active. It has its own processes that can change its state, independent of the actions of the embedded agents [15]. By modeling the environment explicitly, the notion of situatedness of agents gets a much richer meaning. Situatedness places an agent in a context in which it is dynamically related to other agents and objects. It is in this context that an agent is able to perceive his environment and in which it can (inter)act.

In this paper we study perception of agents in situated MASs. Perception is the ability of an agent to sense his near environment, resulting in a percept of the environment. Percepts describe the sensed environment in the form of expressions that can be understood by the agent. Agents use percepts to update their knowledge about the world or use it directly for decision making. Although perception is very common for any MAS, relatively little structured research work has been done to develop theories and generic models for perception. This is especially the case for software MASs where all aspects of perception must be modeled explicitely. The lack of attention for perception in MASs was already raised by P. Maes in the mid nineties. In the overview paper [11], Maes indicated the problem of the "narrow-minded view on the relationship between perception and action", pointing to the poor support for active or goal-driven perception[2]. Due to the poor theoretical foundations for perception in software agent systems, most of the designers of MASs take or (1) a simplistic approach to model perception, e.g. perception is based on a fixed perceptual range as in [17] or [10]; or (2) use an ad hoc approach typically bounded to the domain at hand, e.g. noise is added to sensor input (according to the properties of the domain) in a hard coded way, see e.g. [16]. This results in MASs that do not (or inadequately) take into consideration the consequences of 'real' perception; or the solutions are inflexible and hardly reuseable in other domains.

We propose a generic model for active perception in situated MASs. The model is generic in the sense that (1) it is independent of any domain or specific topology of the environment; (2) it offers reusable core abstractions for active perception in situated MASs, and (3) it offers support to model domain specific properties of perception.

The model we present fits in a generic model for situated MASs we have described in previous work, see [26]. This generic model formally describes an abstract architecture for situated MASs. On the one hand this architecture explicitely models the environment and the processing of actions that are invoked by agents and other ongoing activities such as moving objects. On the other hand it decomposes the behavior of agents into a set of functional modules. A simplified

[2] The term *active perception* is introduced by Ruzena Bajcsy in [2].

model[3] of this functional decomposition of an agent's behavior is depicted in Fig. 1.

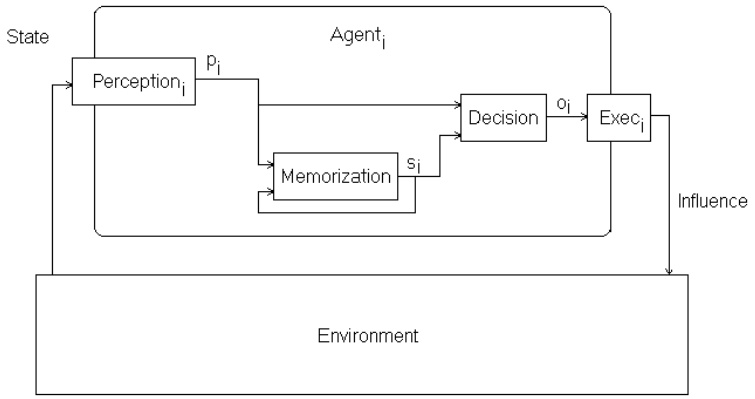

Fig. 1. Generic model for a situated agent

We touch briefly on the different modules, for more details see [26]. The Perception module takes care for the agent's perception of the environment, i.e. it maps the local state of the environment onto a percept, denoted as p_i. The Memorization module enables an agent to register information, i.e. during an update the agent uses the information of the most recent percept to adjust its internal state. The Decision module is the heart of the agent architecture, responsible for action selection. To decide about its next action, the Decision module takes the agent's most recent percept p_i, the actual knowledge s_i and selects an operator o_i for execution. The execution of this operator produces an influence into the environment. The environment collects the influences of simultaneously acting agents [27] and calculates according to a set of domain specific laws the reaction, i.e. state changes in the environment as well as effects on the acting agents (e.g. an agent receives an object passed by another agent).

In this paper we zoom in on the Perception module. Whereas perception is modelled passively in the original model of [26], in this paper we extend the model to active perception. Active perception enables an agent to direct its perception at the most relevant aspects of the environment according to its current task, facilitating better situation awareness and helps to keep processing of perceived data under control.

The remainder of this paper is structured as follows. Section 2 presents the model for active perception and illustrates it with a simple example. This section fully explains the model but rather in an intuitive manner. Section 3 gives a mathematical foundation for the model. We formally describe the model at

[3] This simplified model boils down to the very basic model for MASs as described by several authors in the literature, compare e.g. [21] or [29].

different levels of abstraction. Next, in section 4 we discuss related work. We conclude and look to future work in the final section 5.

2 Model for Active Perception

In this section we present the model for active perception in situated MASs. We start with a general description of the model. Subsequently, we illustrate the model for a simple multi-agent application.

2.1 Description of the model

Fig. 2 gives a graphical overview of the model. The model decomposes active perception into three functional modules: sensing, interpreting and filtering.

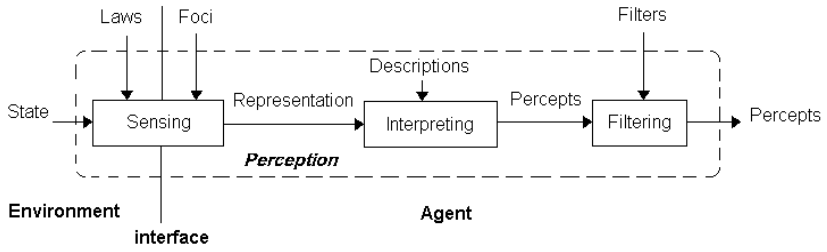

Fig. 2. Model for active perception

Sensing maps the state of the environment to a representation. We follow [1] and define a representation as a structured assembly of symbols that refer back to something in the environment, i.e. external to the agent. The mapping of state to representation depends on two factors. First the agent can select a set of *foci*. Each focus is characterized by its sensibility, but may have other properties too, such as an operating range, a resolution etc. Focus selection enables an agent to direct its perception, it allows him to sense the environment only for specific types of information. E.g., in an ant-like MAS, one agent may be interested in a 'visible' perception of his environment, while another agent may be interested in 'smelling' pheromones. To sense the desired type of information both agents have to select a different appropriate focus. Second, the representation of the state is composed according to a set of *perceptual laws*. A perceptual law is an expression that constraints the composition of a representation according to the requirements of the modeled domain. As such, perceptual laws are an instrument for the designer to model domain specific constraints on perception. Contrary to physical sensing that incorporates such constraints naturally, in software multi-agent systems we have to model the constraints explicitely. Examples are a perceptual law that specifies how an area behind an obstacle is out of the scope

of a perceiving agent or a law that under certain conditions add some noise to perception. Besides the modeling of domain specific sensing, perceptual laws also permit the designer to introduce 'synthetic' constraints on perception. E.g., for reasons of efficiency one could introduce default limits for perception in order to restrain the amount of information the agents have to process. It is important to notice that the model supports parallel sensing of the environment. Since agents can select different foci simultaneously, sensing typically results in a compound representation of the environment. This property is important to enable agents to sense their environment in a multi-mode integral manner.

The second functionality of active perception is interpreting. Interpreting maps a representation to a percept. To interpret a representation, agents use *descriptions*. Descriptions are blueprints that enable agents to extract percepts from representations. Percepts describe the sensed environment in the form of expressions that can be understood by the internal machinery of the agent. Consider e.g. a representation that contains a number of similar objects in a certain area. The agent that interprets this representation may use one description to interpret the distinguished objects and another description to interpret the group of objects as a cluster.

The third and final functionality of active perception is filtering. By selecting a set of *filters* an agent is able to select only those data items of a percept that match specific selection criteria. Each filter imposes conditions on the elements of a percept. These conditions determine whether the elements of a percept can pass the filter or not. E.g., an agent that has selected a focus to visually perceive its environment and who is currently interested in only the agents within his perceptual range can select a appropriate filter that matches only agents for his percept.

2.2 Example application

To illustrate the model for active perception, consider the simple file searching system in a peer-to-peer (P2P) network, depicted in Fig. 3. The idea of this application is to let mobile agents act on behalf of users and browse a shared distributed file system to find requested files. Each user is situated in a particular node (its base) and can send out different agents to work for him. We assume a fixed network structure, however the available bandwidth on the links between nodes depends on the current load. Agents can observe the environment, but only to a limited extent. In general, we assume a perceptual range of 2 hops from the agent's current position, however agents are not able to perceive the content of nodes along heavy loaded links, i.e. links with an actual bandwidth less then 5 KB/s. An agent can sense different types of information in the environment. First, he can sense "visible" information, i.e. agents (α, β), nodes $(A, B, \ldots, Z)$, bandwidth on heavy loaded links (bold units marked on links in KB/s), bases (B_α, B_β) and files (T_1, T_2 and T_3). Second, agents can sense "signals". Each base emits such a signal. The intensity of the signal decreases with every hop. Sensing the signal of its base enables an agent to "climb up" the gradient, i.e. move towards its base or alternatively "climb down", i.e. move away from it. In

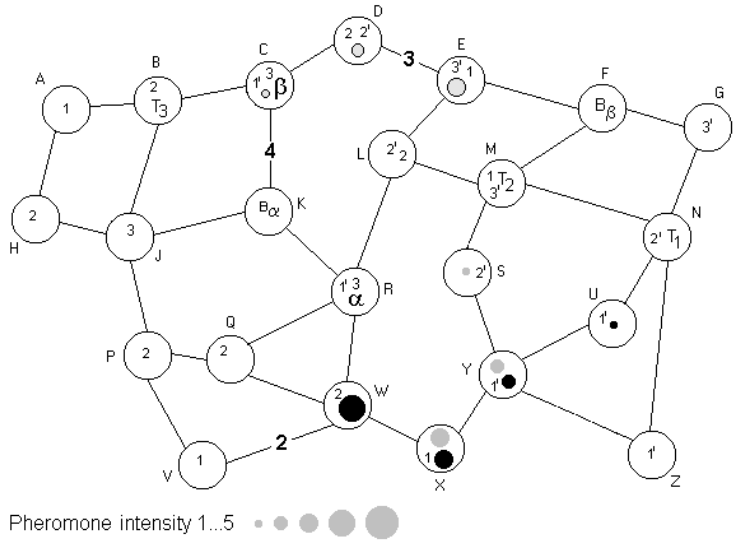

Fig. 3. Example application

Fig. 3 we indicated the intensity of the signal of base B_α by means of integers marked in the nodes and signals of base B_β by means of integers with accent-marks. Finally, agents can "smell" pheromones. An agent can drop a file-specific pheromone in the environment when he returns back to his base with a copy of a file. Such pheromone trail can not only help the agent later on when he needs a new copy of the file, it can also help other agents to find their way to that file. Pheromones tend to evaporate, thereby limiting their influence over time. This is an important property to avoid that agents are misled when a file disappears at a certain node. In the example there are currently pheromone trails to each of the files, e.g. for T_1 the trail reaches out along the nodes $U - Y - X - W$.

To sense a particular type of information the agent has to select an appropriate focus. Let us first consider agent α who is looking for file T_1 and actually is interested in smelling pheromones to guide his search (see Fig. 3). Therefore the agent selects the focus *smell*. According to the perceptual laws agent α is able to sense pheromones in the nodes W, X and E. Node C is currently unreachable since the link $K - C$ is to heavy loaded. Since agent α is interested in T_1 he further can select a filter that matches pheromones of only T_1. This result in a perception with pheromones for the target file T_1 in nodes W and X with values respectively 4 and 3. This information clearly suffice for agent α to move along the pheromone trail $W - X$ towards file T_1.

Let us now look at agent β. Suppose that this agent returns to its base with a copy of file T_3. To find his way back home, agent β selects the focus *receive*. According to the perceptual laws the agent senses signals in the nodes A, B, C, D and J. Nodes K and E are unreachable due to overloaded links. With a filter

that matches the signal for base B_β this results in a perception with signals in nodes C and D with values respectively $1'$ and $2'$. This suffices for agent β to climb up the gradient along node D towards its base.

3 Formal description of the model

In this section we give a formal description of the model for active perception. The formal notation we use is based on set theory. This notation is in accordance with the formalism used in [26]. We start from a graphical overview of the model and then introduce a number of definitions. Afterwards we discuss each module of the model in detail. Finally we reconsider the example discussed in the previous section based on the formal model.

3.1 Introduction

Fig. 4 depicts a detailed overview of the model for active perception.

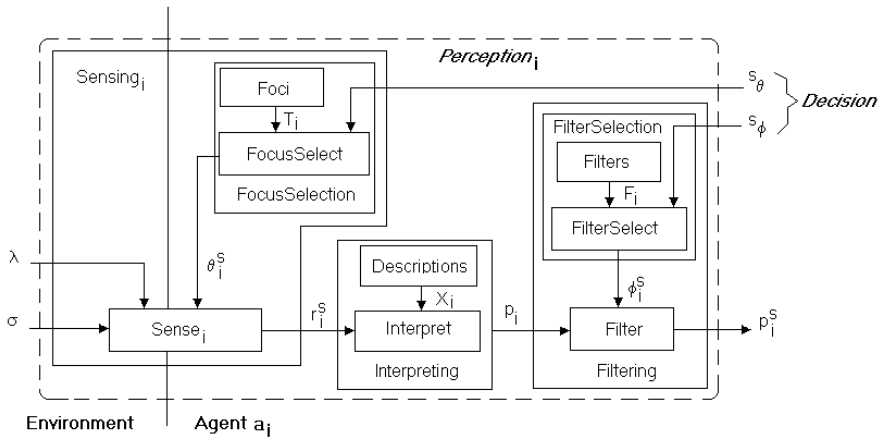

Fig. 4. Detailed model for active perception

In this model each module of the high level model described in Fig. 2 is decomposed into a number of primitive components. This decomposition allows us to precisely describe the subsequent phases of perception. Before we zoom in on each module, we introduce a number of general definitions:

$Ag = \{a_1, \ldots, a_i, \ldots, a_n\}$: the set of agents in 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. $Id(a_i) = y_i$
$\sigma \in \Sigma$: the actual state of the environment, with Σ the set of all possible

states of the environment
L: the set of perceptual laws modeling the constraints on perception of the domain
$\lambda \in \Lambda$: a set of perceptual laws, with $\Lambda = 2^L$ the set of all possible subsets of perceptual laws
$T = \{t_1, \ldots, t_u\}$: the set of foci to sense the environment of the MAS
R: the set of all perceptible representations of the environment
$X = \{x_1, \ldots, x_v\}$: the set of descriptions available in the MAS to enable agents to interpret representations
P: the set of all percepts of the environment in the MAS
$F = \{f_1, \ldots, f_w\}$: the set of filters for percepts available in the MAS

3.2 Model in detail

First we zoom in on sensing. Based on the general definition of foci T, we introduce a number of definitions with respect to agent specific sensing:

$T_i \in 2^T$: the set of foci available for a_i
Θ_i: the set of all subsets of foci for a_i, i.e. $\Theta_i = 2^{T_i}$
$s_\theta \in S_{\Theta_i}$: a focus selector of a_i, with S_{Θ_i} the set of all possible foci selectors of a_i
$\theta_i^s \in \Theta_i^s$: a subset of foci selected with s_θ and $\Theta_i^s \subseteq \Theta_i$ all possible combinations of foci that can be selected by a_i
$r_i^s \in R_i^s$: a representation of the environment sensed by a_i through the selected foci θ_i^s; $R_i^s \subseteq 2^R$ is the set of all representations that can be selected by a_i through its available foci selectors Θ_i^s

Let us now look at focus selection. *Foci* is a function that returns the set of foci available for a_i and is typed as follows:

$$Foci : Y \rightarrow 2^T$$
$$Foci(y_i) = T_i$$

To direct its perception, an agent can select a particular subset of foci from this repository:

$$FocusSelect : 2^T \times S_{\Theta_i} \rightarrow \Theta_i^s$$
$$FocusSelect(T_i, s_\theta) = \theta_i^s$$

FocusSelection integrates the *Foci* function with *FocusSelect* and is typed as follows:

$$FocusSelection : (Y \rightarrow 2^T) \times S_{\Theta_i} \rightarrow \Theta_i^s$$
$$FocusSelection(Foci(y_i), s_\theta) = \theta_i^s$$

Thus *FocusSelection* selects a set of foci θ_i^s from the repository $T_i = Foci(y_i)$ according to the focus selector s_θ selected by a_i.
The $Sense_i$ function produces representations and is typed as follows:

$Sense_i : \Sigma \times \Lambda \times \Theta_i^s \rightarrow R_i^s$
$Sense_i(\sigma, \lambda, \theta_i^s) \ = \ r_i^s$

The sense function takes the current state of the environment σ and a set of foci θ_i^s and produces, according to a set of perceptual laws λ, a representation r_i^s for a_i. Perceptual laws are defined as 3-tuples:

$l_i \in L$: $< focus, conditions, effects >$

$focus$ is a name that refers to the type of information an agent is interested in. *conditions* can be state representations 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 express how the law affects the composition of the representation. We discuss a concrete law in the example at the end of this section.

Integrating $FocusSelection$ with the $Sense_i$ function allows us to describe the integral sensing function:

$Sensing_i : \Sigma \times \Lambda \times ((Y \rightarrow 2^T) \times S_{\Theta_i} \rightarrow \Theta_i^s) \rightarrow R_i^s$
$Sensing_i(\sigma, \lambda, FocusSelection(Foci(y_i), s_\theta)) = r_i^s$

$Sensing_i$ produces a representation r_i^s for an agent a_i according to a set of applicable laws for perception λ given the current state of the environment σ and a subset of foci selected from the agent's repository $T_i = Foci(y_i)$ with a focus selector s_θ.

Next we take a closer look at interpreting. To support agent specific interpretation, we introduce the following definitions:

$X_i \in 2^X$: the set of descriptions available for a_i
$p_i \in P_i$: a percept of a_i, with $P_i \subseteq 2^P$ the set of all possible percepts that can be interpreted by a_i from R_i^s

First we type the *Descriptions* function:

$Descriptions : Y \rightarrow 2^X$
$Descriptions(y_i) = X_i$

The *Descriptions* function returns the repository of descriptions X_i available for agent a_i. Descriptions enable an agent to *Interpret* representations:

$Interpret : R_i^s \times 2^X \rightarrow P_i$
$Interpret(r_i^s, X_i) = p_i$

The integral *Interpreting* function is typed as follows:

$Interpreting : R_i^s \times (Y \rightarrow 2^X) \rightarrow P_i$
$Interpreting(r_i^s, Descriptions(y_i)) \ = \ p_i$

The *Interpreting* function interprets a representation r_i^s through the the repository of descriptions $X_i = Descriptions(y_i)$ of a_i, resulting in a percept p_i.

Next we zoom in on the filtering function. Based on the general definition of filters F, we introduce a number of definitions with respect to agent specific filtering:

$F_i \in 2^F$: the set of filters available for a_i
Φ_i: the set of all subset of filters for a_i, i.e. $\Phi_i = 2^{F_i}$
$s_\phi \in S_{\Phi_i}$: a filter selector of a_i, with S_{Φ_i} the set of all possible filter selectors of a_i
$\phi_i^s \in \Phi_i^s$: a subset of filters selected with s_ϕ and $\Phi_i^s \subseteq \Phi_i$ all possible combinations of filters that can be selected by a_i
$p_i^s \in P_i^s$: a percept selected by a_i through filtering with ϕ_i^s; P_i^s is the set of all percepts that can be selected through Φ_i^s, thus $P_i^s \subseteq P_i$

First we look at filter selection. $Filters$ is a function that returns the set of filters available for a_i with the following typing:

$$Filters : Y \rightarrow 2^F$$
$$Filters(y_i) = F_i$$

To select specific data from a percept, an agent can select a appropriate subset of filters from this repository:

$$FilterSelect : 2^F \times S_{\Phi_i} \rightarrow \Phi_i^s$$
$$FilterSelect(F_i, s_\phi) = \phi_i^s$$

$FilterSelection$ integrates the $Filters$ function with $FilterSelect$ and is typed as follows:

$$FilterSelection : (Y \rightarrow 2^F) \times S_{\Phi_i} \rightarrow \Phi_i^s$$
$$FilterSelection(Filters(y_i), s_\phi) \ = \ \phi_i^s$$

Thus $FiltersSelection$ select a set of filters ϕ_i^s from the repository $F_i = Filters(y_i)$ according to the filter selector s_ϕ selected by a_i.
Finally we come to the $Filter$ function that performs the filtering. $Filter$ is typed as follows:

$$Filter : P_i \times \Phi_i^s \rightarrow P_i^s$$
$$Filter(p_i, \phi_i^s) \ = \ p_i^s$$

$Filter$ selects from a percept p_i only the data that matches the criteria of the selected filters ϕ_i^s.
Integration of $FilterSelection$ with $Filter$ yields the integral $Filtering$ function and is typed as follows:

$$Filtering : P_i \times ((Y \rightarrow 2^F) \times S_{\Phi_i} \rightarrow \Phi_i^s) \rightarrow P_i^s$$
$$Filtering(p_i, FilterSelection(Filters(y_i), s_\phi)) \ = \ p_i^s$$

The filtering module takes a percept p_i and restricts it to p_i^s according to a set of filters selected by a_i from its repository of filters $F_i = Filters(y_i)$ through selector s_ϕ.

To conclude this section we can describe an overall function for active perception, indicated by $Perception_i$ and typed as follows:

$$Perception_i : \Sigma \times \Lambda \times S_{\Theta_i} \times S_{\Phi_i} \rightarrow P_i{}^s$$
$$Perception_i(\sigma, \lambda, s_\theta, s_\phi) = p_i^s$$

The $Perception_i$ function abstracts from the internal architecture of the perception module and describes active perception as a blackbox. $Perception_i$ takes the current state of the environment σ and produces a percept p_i^s for agent a_i according to the selectors s_θ and s_ϕ selected by that agent and the valid laws λ for perception of the multi-agent application.

Alternatively we can describe active perception as a sequence of the functions $Sensing_i$, $Interpreting$ and $Filtering$. For convenience, we introduce an operator $\prec$ to express a sequence of functions. $\prec$ has the following semantics:

$$f \prec g : A \times B \to Q \prec Q \times D \to T$$
$$f(a,b) = q \prec g(q,d) = t$$

Applied to active perception we get the following sequence of functions:

$$\begin{aligned} &Sensing_i \prec Interpreting \prec Filtering : \\ &\quad \Sigma \times \Lambda \times ((Y \to 2^T) \times S_{\Theta_i} \to \Theta_i^s) \to R_i^s \\ &\prec R_i^s \times (Y \to 2^D) \to P_i \\ &\prec P_i \times ((Y \to 2^F) \times S_{\Phi_i} \to \Phi_i^s) \to P_i^s \end{aligned}$$

$$\begin{aligned} &\quad Sensing_i(\sigma, \lambda, FocusSelection(Foci(y_i), s_\theta)) = r_i^s \\ &\prec Interpreting(r_i^s, Descriptions(y_i)) = p_i \\ &\prec Filtering(p_i, FilterSelection(Filters(y_i), s_\phi)) = p_i^s \end{aligned}$$

Putting it to words: to perceive the environment an agent senses its neighborhood through a set of selected foci. According to the current state of the environment and a set of valid laws this results in a representation. This representation is interpreted by the agent with a set of descriptions resulting in a percept. This percept is further restricted through a set of selected filters that express context relevant selection criteria.

3.3 Example revisited

To conclude this section, we revisit the example discussed in section 2.2. We represent state as a formula of the form $p(c_1, \ldots, c_r)$, where p is a predicate and c_i are values. The laws that constrain the perception of the agents in the example are defined as follows:

$$\begin{aligned} l_1 =< &smell, \{at(a_i, pos), dist(loc, pos) <= 2, bandwidth(pos, loc) > 5\}, \\ &\{add : pherom(loc, distance, file\_name, intensity)\} > \\ l_2 =< &receive, \{at(a_i, pos), dist(loc, pos) <= 2, bandwidth(pos, loc) > 5\}, \\ &\{add : signal(loc, distance, base\_name, intensity)\} > \end{aligned}$$

l_1 expresses that for a focus *smell* selected by an agent a_i all pheromones *pherom* on a location *loc* are included (*add*) in the representation that can be sensed by the agent at a distance *dist* of maximal 2 hops of its current position *pos*, assuming that the available *bandwidth* along the path from *pos* to *loc* is at least 5 KB/s. l_2 expresses in a similar way how percepts are constrainted when agents sense, through a focus *receive*, signals emitted by the bases.

We limit the illustration of the formal model to only the perception of agent α in the example[4]. If we apply l_1 for agent α in Fig. 3, this agent gets the following representation:

$$r_i^s = \{pherom(E, 2, T_3, 3), pherom(W, 1, T_1, 4), \\ pherom(X, 2, T_1, 3), pherom(X, 2, T_2, 3)\}$$

According to the example, agent a_i is only interested in smelling pheromones of T_1 and therefore he selects the following filter:

$$f_1 = pherom(\_, \_, T_1, \_)$$

Filter f_1 selects all pheromones from a percept that matches T_1 as $file\_name$. Applying filtering with f_1 results in the following percept:

$$p_i^s = \{pherom(W, 1, T_1, 4), pherom(X, 2, T_1, 3)\}$$

Based on this information it is quite simple for agent a_i to continue its search for T_1 by moving to the neighbor node W.

4 Related work

Most of the research on perception is done in the context of robotics, however work is also done in the context of software agents. Two research tracks are identified in the work on perception: the passive approaches to perception and the active approaches to perception. The generic perception model proposed in this paper is situated in the active approaches track.

In the context of robotics, the RoboCup Soccer Server [19] is a well known example. Three kinds of sensors are supported in its sensor model: the aural sensor, the visual sensor and the body sensor. These sensors correspond to foci in the generic model for active perception proposed in this paper. The CMUnited agents [12] of the RoboCup Soccer Server are capable of doing perception, cognition and action. In this agent architecture, the sensor information is used as input for the interpreter. Filtering functionality is not explicitly present in the architecture, but is included in the step after interpretation in which the state is used to select the behavior. The functionality for perception available in RoboCup Soccer maps quite well to our model, however this functionality is bounded to the specific application of the Soccer Simulator while we presented a generic model for active perception in situated MASs, independent of any particular application.

The VAP I' system as described in [5] describes interaction protocols to control vision systems with an example worked out in the robotics domain. They define a number of agents cooperating for performing a perception: a camera-control agent, a 2D-description agent, an interpretation agent and a mmi (man machine interface) agent. In the active perception model proposed in this paper,

[4] For simplicity, we abstract from interpretation, i.e. we use the same set of expressions for representations and percepts.

the sequence of steps for doing perception is similar, but are all part of one agent. While sensing in the VAP I' systems happens through a real camera, our model is intended for software agent systems and therefore explicitely models foci (for sensing) as well as perceptual laws to enforce domain specific constrains on perception.

In the context of software agents, a number of models and specific techniques for perception are proposed, most of them focusing on specific properties of perception. An example of such a specific property is [18] that describes how reliability and temporality of perception can be taken into account.

L. Ronnie [20] makes a distinction between perception management and sensor management. Perception management is defined as the generic concept of data fusion (i.e. acquiring percepts to serve information needs) without paying particular attention to details of concrete sensor devices. Sensor management on the other hand is connected with the reconfiguration of sensor devices. Sensor management is comparable to the sensing functionality in our model, while perception management is comparable to the interpreting and filtering functionality.

We conclude that most of the research done on perception is focussed on performing perception in a specific application or on one specific property of perception. However from a software engineering point of view, it is important to define generic models for perception. Such models can serve as conceptual frameworks to build infrastructure for perception in different application domains. Besides a reusable core, a generic model should also offer abstractions that allow the designer to model the domain specific properties of perception in a flexible way. The model for active perception proposed in this paper intends to contribute in this way.

5 Conclusions and Future Work

In this paper we proposed a generic model for active perception in situated multi-agent systems. The model decomposes active perception into a succession of three functionalities: sensing, interpreting and filtering. The genericity of the model is based on two complementary features. On the one hand, the model offers a reusable framework to deal with active perception in situated multi-agent systems. On the other hand, laws, foci filters are abstractions that enable the designer to model domain specific properties of perception. Perceptual laws constrain the perception of agents according to the modelled domain, while foci and filters enable agents to direct their perception at the most relevant aspects in the environment. To make an analogy with biological perception: focus selection can be viewed as choosing a particular sense to observe the environment, while filter selection is comparable to the direction of attention, both driven by the current interests.

Active perception facilitates better situation awareness of the agents and helps to keep processing of perceived data under control. However, these potential advantages do not come for free. Active perception requires the designer to identify the appropriate selection criteria for agents that provide useful inform-

ation. This points to a close relationship between an agent's behavior and its perception. Therefore we have started to integrate the generic model for active perception with the agent's decision functionality and we will be working on this in the future.

So far we have applied the model for active perception only in a couple of study applications, including the PacketWorld [25] and a simple P2P file management system [22]. Currently our research group is involved in a project with the Egemin company [6]. In one part of this cooperation we investige how the multi-agent paradigm can be applied to Automated Guided Vehicle warehouse systems. This is a promising, complex real-world application, well suited to verify our model for situated agents in general and the model for active perception in particular.

References

1. D. Andler, *Introduction aux sciences cognitives*, ISBN 97-82070325-771, Gallimard, Paris, France, 1992.
2. R. Bajcsy, *Active Perception versus Passive Perception*, in Proceedings of 3th Workshop on Computer Vision, Representations and Control, Bellair, MI, 1985.
3. F.Bellifemine, A. Poggi and G. Rimassa, *Jade, A FIPA-compliant Agent Framework*, in CSELT Internal Technical Report; also published in Proceedings of PAAM'99, pp.97-108, London, UK, 1999.
4. R. A. Brooks, *Intelligence Without Representation*, in Artificial Intelligence Journal (47), pp. 139-159, 1991.
5. Y. Demazeau, O. Boissier and J.-L. Koning, *Using Interaction Protocols to Control Vision Systems*, in Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, Texas, 1994.
6. Egemin, *Creative Technology for Total and Innovative Industrial Automation Solutions*, http://www.egemin.com/home.html
7. J. Ferber, *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, ISBN 0-201-36048-9, Great Britain, 1999.
8. FIPA, *Foundation for Intelligent Physical Agents*, http://www.fipa.org/.
9. M. R. Genesereth and N. Nilsson, *Logical Foundations of Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1987.*
10. D. Hales, Group Reputation Supports Beneficent Norms, Journal of Artificial Societies and Social Simulation vol. 5, no. 4, 2002.
11. P. Maes, *Modeling Adaptive Autonomous Agents*, in Artificial Life Journal, 1 (1-2) pp. 135-162, MIT Press, Cambridge, MA, 1994.
12. I. Noda and P. Stone, *The RoboCup Soccer Serer and CMUnited Clients: Implemented Infrastructure for MAS Research*, in Journal of Autonomous Agents and Multi-Agent Systems, vol.7(1-2), 2003.
13. J. Odell, V. Parunak, M. Fleischer and S. Brueckner, *Modeling agents and their Environment*, in Proceedings of AOSE, Workshop at AAMAS, Bologna, Italy, 2002.
14. V. Parunak, S. Brueckner, J. Sauter and R. Matthews, *Distinguishing Environmental and Agent Dynamics: A Case Study in Abstraction and Alternate Modeling Technologies*, in Proceedings of the ESAW Workshop at ECAI'00, Berlin, Germany, 2000.

15. V. Parunak, *"Go to the Ant": Engineering Principles from Natural Agent Systems*, in Annals of Operations Research 75 pp. 69-101, 1997.
16. H. Pasula, S. Russell, M. Ostland and Y. Ritov, *Tracking many objects with many sensors*, in Proceedings of IJCAI-99, Stockholm 1999.
17. M. E. Pollack and M. Ringuette, *Introducing the Tileworld: experimentally evaluating agent architectures*, in Proceedings of 8th National Conference on Artificial Intelligence, AAAI Press, Menlo Park, Cam 1900.
18. J. L. Pollock, *Taking Perception Seriously*, in Proceedings of the First International Conference on Autonomous Agents (Agents'97), ACM Press, New York, 1997.
19. RoboCup, homepage: http://www.robocup.org/; RoboCup Soccer Server: http://sserver.sourceforge.net/
20. L. Ronnie M. Johansson and N. Xiong, *Perception Management: An Emerging Concept for Information Fusion*, Information Fusion, 2003.
21. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2003.
22. K. Schelfthout and T. Holvoet, *A Pheromone-Based Coordination Mechanism Applied in Peer-to-Peer*, Agents and Peer-to-Peer Computing, AAMAS'03 Workshop, Melbourne, Australia, 2003.
23. L. Steels and R. Brooks, *The artificial life route to artificial intelligence: Building Situated Embodied Agents*, New Haven, Lawrence Erlbaum Ass, 1992.
24. K. Sycara, M. Paolucci, M. van Velsen and J. Giampapaand, *The Retsina MAS Infrastructure*, TR CMU-RI-TR-01-05, Robotics Institute, Carnegie Mellon University, March, 2001.
25. D. Weyns and T. Holvoet, *The Packet–World as a Case to Investigate Sociality in Multi-agent Systems*, Demo presented at the Conference of Autonomous Agents and Multi-Agent Systems, AAMAS 2002, Bolgona, 2002.
26. D. Weyns and T. Holvoet, *A Formal Model for Situated Multi-agent Systems*, in Formal Approaches for Multi-Agent Systems, Special Issue of Fundamenta Informaticae, Eds. B. Keplicz and R. Verbrugge, to appear, 2003.
27. D. Weyns and T. Holvoet, *Regional Synchronization for Situated Multi-agent Systems*, 3rd International/Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic, in LNCS Vol. 2691, pp. 497- 511, 2003.
28. M. Wooldridge, N.R. Jennings and D. Kinny, *The Gai Methodology for Agent-Oriented Analysis and Design*, Autonomous Agents and Multi-Agent Systems, 3, pp. 285-312, Kluwer Academic Publishers, The Netherlands, 2000.
29. M. Wooldridge, *An Introduction to MultiAgent Systems*, ISBN 0-471-49691-X. John Wiley and Sons, Ltd. England, 2002.
30. M. Mamei, F. Zambonelli, L. Leonardi, *Self-Organization in Multi Agent Systems: a Middleware Approach*, in Proceedings of First International Workshop on Engineering Self-Organizing Applications, pp. 1-10, AAMAS Workshops, 2003.