International Journal of Computer & Information Science, Vol. 5, No. 1, March 2004

Towards a Unified Model of Sociality in Multiagent Systems

Matthias Nickles and Michael Rovatsos and Wilfried Brauer and Gerhard Weiß
Department of Informatics, Technical University Munich

Abstract

This paper presents *communication systems* (CS) as the first available unified model of socially intelligent systems defined in terms of communication structures. It combines the empirical analysis of communication in a social system with logical processing of social information to provide a general framework for computational components that exploit communication processes in multiagent systems. The two main contributions offered by this paper are as follows: First, a formal model of CS that is based on an improved version of expectation networks and their processing is presented. This formal model is based on a novel approach to the semantics of agent communication languages which contrasts with traditional approaches. Second, a number of CS-based applications are described which illustrate the enormous potential and impact of a CS-based perspective of socially intelligent systems.

Keywords: Artificial Agents, Multiagent Systems, Agent Communication Languages, Agent-Oriented Software Engineering, Socionics.

1. Introduction

The crucial property of artificial agents is their *autonomy*, and since *communication* is the only autonomy-preserving way for agents to interact, it can be argued [1, 2] that *any* kind of social relationship among agents (constituted through e.g. virtual organizations, interaction protocols, social norms, common ontologies...) can be described in form of communication structures. Traditional attempts to model the semantics of agent communication languages (ACLs), which constitute communication structures, are mostly based on describing *mental states* of communicating agents [3, 4, 5, 6] or on publicly available (usually commitment-based) *social states* [7, 8, 9]. The theoretical advantages of the former approach are that it could be able to fully expose the whole mechanism of utterance and utterance understanding (provided that the agents are equipped with social intelligence). But it has two obvious shortcomings, which eventually led to the development of the latter "objectivist" approach: At least in open multiagent systems, agents appear more or less as autonomous black boxes, which makes it impossible in general to impose and verify a semantics described in terms of cognition. Furthermore, the description of interaction scenarios in terms of the cognition of individuals tends to become extremely complicated and intractable for large sets of agents, even if one could in fact "look inside the agents' heads". This is not so much the case due to the complexity of communication processes themselves, but due to the subjective, limited perspective of the involved individual agents, which makes it hard to conclude a concise picture of supra-individual processes. Current mentalistic approaches either lack a concept for preventing such complexity at all, or they make simplifying but unrealistic assumptions, for example that all interacting agents were benevolent and sincere. The "objectivist" semantics, in contrast, is fully verifiable, and it achieves a big deal of complexity reduction by limiting itself to a small set of normative semantical rules. Therefore, this approach has been a big step ahead. But it oversimplifies social processes neglecting pragmatics in favor of traditional sentence semantics, and it does not have a sufficient concept of meaning dynamics and generalization, and social rationality like argumentation and sanctioning [10]. Communication always *has* an implicit social semantics prior to any normative definition, and this semantics needs to be exploited. In general, we doubt that the predominately normative, static and definite concepts of most of the current ACL research, borrowed from the study of programming languages and protocol semantics, are really adequate to cope with concepts like agent autonomy, agent opaqueness and the emergence and vagueness of socially constructed meaning, which communication is in fact about. Therefore, both these traditional views fail to recognize that communication semantics evolve during operation of a multiagent system (MAS), and that they always depend on the view of an observer who is tracking the communicative processes of black- or gray-box agents in the system [10, 11]. Yet communication dynamics and observer-dependency are crucial aspects of inter-agent communication, especially in the context of open systems in which a pre-determined semantics cannot be assumed, let alone the compliance of agents' behavior with it.
In response to these issues, in [1, 12] we have – influenced by sociological systems-theory [13] – introduced *expectations* regarding observable *communications* as a universal means for the modelling of emergent sociality in multiagent systems, and in [11, 10, 14], we have presented – influenced by socio-cognitive and socio-systems theories [15, 16, 13] – formal frameworks for the semantics of communicative action that is *empirical*, *constructivist* and *consequentialist* in nature and analyzed the implications of this model on social reasoning both from an agent (bottom-up) and a systemic (top-down) perspective.

Thereby, we have suggested that recording observations of message exchange among agents in a multiagent system (MAS) empirically is the only feasible way to capture the meaning of communication, if no *a priori* assumptions about this meaning can be made. Being empirical about meaning naturally implies that the resulting model very much depends on the observer's perspective, and that the semantics would always be the semantics "assigned" to the utterances by that observer, hence this view is inherently constructivist. Since, ultimately, no more can be said about the meaning of a message than that it lies in the expected consequences that this message has, we also adopt a consequentialist outlook on meaning.

In this paper, we present a framework for the formal description of socially intelligent multiagent systems based on the universal, systems-theoretical concept of *communication systems* (CS). Following sociological systems theory, communication systems (also called *social systems*) are systems that consist of interrelated communications which describe their environment [13]. We use this term to denote computational models of such systems that process empirical information about observed communication which takes place within a MAS of artificial agents (either with the CS as an MAS-external MAS-observer or as a component of an agent which participates in the observed communication himself). Their distinct features are (i) that they only use data about communication for building models of social processes, the underlying assumption being that all relevant aspects of agent interaction are eventually revealed through communication, and (ii) that, if the respective CS is part of an agent, the results of the observations are suitable to take action in the MAS to influence its behavior; in other words, there might be a feedback loop between observation and action, so that an CS-based agent becomes an autonomous component in the overall MAS.

CSs might be (part of) socially intelligent software agents. Note, however, that this is not necessarily the case. Although their autonomy presumes some agentification in the traditional sense, their objectives need not be tied to achieving certain goals in the physical (or virtual simulation) world as is the case with "ordinary" agents. Thus, CS are best characterized in a abstract fashion as components used to (trans)form expectations (regardless of how these expectations are employed by agents in their reasoning) and are autonomous with respect to how they perform this generation of expectations.

Thereby, the "semantics" aspect mentioned above plays a crucial role, because the meaning of agent communication lies entirely in the total of communicative expectations in a system [1], and CS capture precisely these expectations and how they evolve.

The remaining sections are structured as follows: We start by introducing expectation networks in section 2, which constitute our formal model for describing communicative expectations. Then, we formally define communication systems and their semantics (section 3). Section 4 discusses applications and extensions of the CS, and section 5 concludes.

2. Expectation Networks

It is widely accepted in Distributed Artificial Intelligence that the most important property of intelligent agents is their *autonomy*. The major consequence of the autonomous behavior of agents is that a certain agent appears to other agents and observers more or less as a *black box* which cannot fully be predicted and controlled. This obscurity and uncontrollability is particularly salient in the open multiagent systems we are focussing on. Because only the actions of the agent in its environment can be observed, while its mental state keeps obscure, *beliefs* and *demands* directed to the respective agent can basically be stylized as *action expectations*, which are fulfilled or disappointed in future agent actions[1]. To overcome the situation of mutual indeterminism among black-box agents (the so-called situation of *double contingency* [13]), i.e. to determine the respective other agent and to achieve reasonable coordination (including "reasonable" conflicts), the agents need to *communicate*. A single communication is the whole of a message act as a certain way of telling (not necessarily via speech, but also e.g. as a demonstrative gesture or other semiotically relevant manipulations of the "physical" domain), plus a communicated information, plus the understanding of the communication attempt. Communication is observable as a course of related agent interactions. Because communications are the only way to overcome the problem of double contingency (i.e. the isolation of single agents), they are the basic constituents of sociality, and the relation of subsequent communications forms the *social system* for the participating agents. And if action expectations are related to message acts as parts of communications (i.e. sociality), *social structures* can be modeled as *expectation structures* [13].
By processing existing expectations, agents determine their own actions, which, then, influence the existing expectations in turn. So communication is not only structured by individual agent goals and intentions, but also by expectations, and the necessity to test, learn and adapt expectations from observed interactions and in order to optimize future communications.
Expectations regarding agent behavior can be formed not only by peer agents (as an aspect of their mental state), but also by observers with a global view of the multiagent system. Such *system-level expectations* are called *emergent* if they are formed solely from the statistical evaluation of the observed communications. In contrast, e.g. the system designer forms expectations not only from her knowledge about the existing multiagent system "under construction", but also from her design goals (in 4.2 we will present a design method for multiagent system based on this principle). In either case, interrelated expectations regarding future agent

[1]This view of expectations and sociality follows the *Theory of Social Systems* ("systems theory") of the sociologist *Niklas Luhmann* [13] and is described in detail in [1].

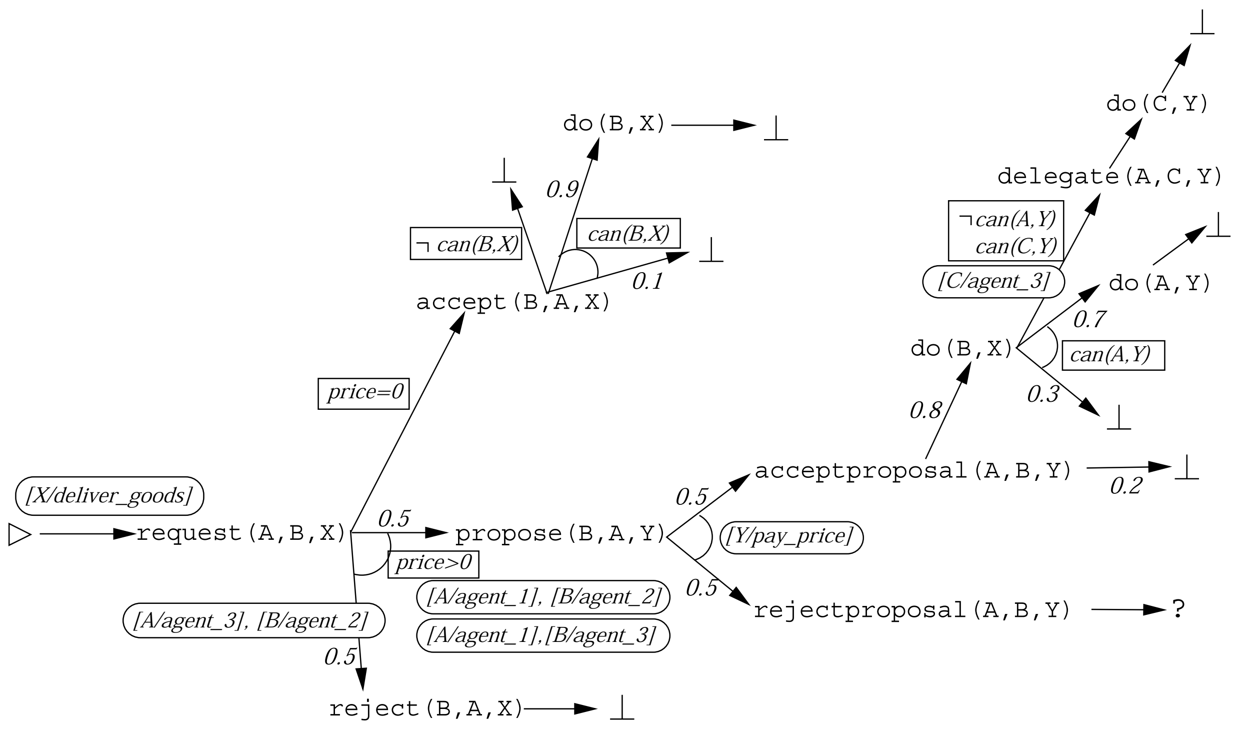

Figure 1.
An expectation network. Nodes are labelled with message templates in `typewriter` *font and the special symbols* $\triangleright$, $\perp$ *and* $?$*; they are connected by (solid) edges labelled with numerical expectabilities in italic font. Substitution lists/conditions belonging to edges appear in rounded/edged boxes near the edge. If neighbouring edges share a condition this is indicated by a drawn angle between these edges. This example network shows a short communication course of three agent roles, A, B and C, which are instantiated with three agents through substitution lists ($A = agent\_1$, $B = agent\_2$, $C = agent\_3$). The provision of such substitution lists is optional. The EN starts with a request to do action X ($deliver\_goods$) of A directed to B. In case condition $price = 0$ is fulfilled, B is expected to always accept this request and to perform the requested action X in case he is able to do it (condition $can(B, X)$ is true). Otherwise, with probability 0.5 the request will be rejected and the communication ends ($\perp$). With probability 0.5, B answers with a proposal Y (where $Y = pay\_price$), which is accepted by A with probability 0.5. After rejection, the further course of communication is unknown ($?$), whereas the acceptance leads to the fulfillment of X through B ($do(B, X)$). In the latter case, A in turn does Y (i.e., pays the price for X), or delegates this to C if he is not able to pay ($can(A, Y)$ is false).*

behavior are the *only* modeling means that are universally suitable for the description of black-box agents' interaction behavior from an external observers point of view as our computational models of communication systems.

Expectation networks (ENs) [1] are the graphical data structures used in this work for the formal representation of such social expectations. They capture the regularities in the flow of communication between agents in a system by interconnecting message templates (nodes) that stand for utterances via links (edges) which are labelled with (i) probabilistic weights called *expectabilities*, (ii) a logical condition and (iii) lists of variable substitutions. Roughly speaking, the semantics of such a weighted edge is as follows: If the variables in the messages have any of the values in the (optional) substitution lists, and the logical condition is currently satisfied, then the weight of this edge reflects the probability with which a message matching the label of the target node is expected to follow the utterance of a message matching the label of the source node of the edge. Before presenting a full definition of ENs, we have to introduce some basic notions and notation we use, and to make certain auxiliary definitions and assumptions. The example network in figure 2 will be used throughout the discussion of ENs to illustrate the purpose of definitions and assumptions.

2.1 Basics

A central assumption that is made in ENs is that observed messages may be categorised as *continuations* of other communications, or may be considered the start of a new interaction that is not related to previous experience. So an edge leading from message m to message m' is thought to reflect the probability of communication being "continued" from the observer's point of view. Usually, continuation depends on temporal and spatial proximity between messages, but it might also be identified through a connection about "subject", or, for example, through the use of the same communication medium (m' was shown on TV after m was shown some time earlier on).

Apart from "ordinary" node labels denoting messages, we use three distinct symbols "$\rhd$", "$\perp$", and "?". "$\rhd$" is the label occurring only at the root node of the EN. Whenever a message is considered a start of a new conversation instead of continuing previous sequences, it is appended to this "$\rhd$"-node. Nodes labelled with "$\perp$" denote that a course of communications is expected to end with the predecessor of this node. The label "?", finally, indicates that there exists no expectation regarding future messages at this node. Nodes with such "don't know" semantics are usually messages that occur for the first time – the observer knows nothing about what will happen after them being uttered.

To define the syntactic details of EN, we introduce formal languages $\mathcal{L}$ and $\mathcal{M}$ used for predicate-logical expressions and for message templates. $\mathcal{L}$ is a simple logical language consisting of propositions *Statement* potentially containing (implicitly universally quantified) variables and of the usual connectives $\vee$, $\wedge$, $\Rightarrow$ and $\neg$, the logical constants "true" and "false", and braces () for grouping subexpressions together (the language is formally given by the grammar in table 1). Given the set of all possible interpretations $\mathcal{I} = \{I : \mathit{Statement} \to \{\text{true}, \text{false}\}\}$ we define the relation $\models \subseteq \mathcal{I} \times \mathcal{L}$ in the usual way by induction over formulae $\varphi \in \mathcal{L}$ and interpretations $I \in \mathcal{I}$:

$$\begin{aligned} I \models \varphi \quad &\text{iff} \quad \varphi \in \mathit{Statement} \text{ and } I(\varphi) = \text{true} \\ I \models \varphi \quad &\text{iff} \quad \exists \vartheta : \varphi'\vartheta = \varphi \text{ and } I \models \varphi' \end{aligned}$$

$$\begin{aligned} I \models \neg\varphi \quad &\text{iff} \quad I \not\models \varphi \\ I \models \varphi \vee q \quad &\text{iff} \quad I \models \varphi \text{ or } I \models q \end{aligned}$$

where $\vartheta = \langle [v_1/t_1], \ldots, [v_k/t_k] \rangle$ is a variable substitution. As usually, $\wedge$ and $\Rightarrow$ can be defined as abbreviations through the other operators. Also, we write $\models \varphi$ if φ is a tautology that is satisfied by any $I \in \mathcal{I}$. A *knowledge base* $KB \in 2^{\mathcal{L}}$ can be any finite set of formulae from $\mathcal{L}$. For simplicity, we will often write $KB \models \varphi$ to express $\models (\bigwedge_{\varphi' \in KB} \varphi' \Rightarrow \varphi)$.

As for $\mathcal{M}$, this is a formal language that defines the message patterns used for labelling nodes in expectation networks. Its syntax is given by the grammar in table 1. Messages observed in the system (we write $\mathcal{M}_c$ for the language of these *concrete* messages) can be either physical messages of the format $\mathtt{do}(a, ac)$ where a is the executing agent and ac is a symbol used for a physical action, or a non-physical message $\mathit{performative}(a, b, c)$ sent from a to b with content c. (Note that the symbols used in the *Agent* and *PhysicalAction* rules might be domain-dependent symbols the existence of which we take for granted.) The node

Var	$\to$	$X \mid Y \mid Z \mid \ldots$
AgentVar	$\to$	$A_1 \mid A_2 \mid \ldots$
PhysicalActVar	$\to$	$X_1 \mid X_2 \mid \ldots$
Expect	$\in$	$[0;1]$
Agent	$\to$	*agent_1* $\mid \ldots \mid$ *agent_n*
Head	$\to$	*it_rains* $\mid$ *loves* $\mid \ldots$
Performative	$\to$	*accept* $\mid$ *propose* $\mid$ *reject* $\mid$ *inform*
		$\mid \ldots$
PhysicalAction	$\to$	*move_object* $\mid$ *pay_price*
		$\mid$ *deliver_goods* $\mid \ldots$
Message	$\to$	*Performative*(*Agent*, *Agent*, *LogicalExpr*)
		$\mid$ *do*(*Agent*, *Agent*, *PhysicalAction*)
MsgPattern	$\to$	*Performative*(*AgentTerm*, *AgentTerm*,
		LogicalExpr)
		$\mid$ *do*(*AgentTerm*, *AgentTerm*,
		PhysicalActTerm)
		$\mid \rhd \mid \perp \mid$?
PhysicalActTerm	$\to$	*PhysicalActVar* $\mid$ *PhysicalAction*
AgentTerm	$\to$	*AgentVar* $\mid$ *Agent*
LogicalExpr	$\to$	(*LogicalExpr* $\Rightarrow$ *LogicalExpr*)
		$\mid$ (*LogicalExpr* $\vee$ *LogicalExpr*)
		$\mid$ (*LogicalExpr* $\wedge$ *LogicalExpr*)
		$\mid \neg$*LogicalExpr*
		$\mid$ *Statement*
Statement	$\to$	*Head* $\mid$ *Head*(*TermList*) $\mid$ true
		$\mid$ false
TermList	$\to$	*TermList*, *Term* $\mid$ *Term*
Term	$\to$	*Var* $\mid$ *AgentTerm* $\mid$ *MsgPattern*
		$\mid$ *Graph*
EdgeList	$\to$	(*MsgPattern*, *Expect*, *MsgPattern*,
		LogicalExpr, *SubstList*) *EdgeList* $\mid \varepsilon$
Graph	$\to$	$\langle$*EdgeList*$\rangle$
SubstList'	$\to$	*SubstList'* *Subst* $\mid \varepsilon$
SubstList	$\to$	$\langle$*SubstList'*$\rangle$
Subst	$\to$	$[\mathit{AgentVar}/\mathit{Agent}]$
		$\mid [\mathit{PhysicalActVar}/\mathit{PhysicalAction}]$
		$\mid [\mathit{Var}/\mathit{Term}]$

Table 1.
A grammar for messages, generating the languages $\mathcal{M}$ (the language of message patterns, using MsgPattern as starting symbol), $\mathcal{M}_c$ (the language of concrete messages, using Message as starting symbol) and the logical language $\mathcal{L}$ (using LogicalExpr as starting symbol).

labels (type $MsgPattern$) used in the expectation networks may also contain variables for agents and physical actions (though not for performatives). Following the concept of agent roles introduced in [1, 12], the variables for agents are called (agent) *roles*. These variables are useful to generalize over different observed messages, and can optionally be further specified by adding variable substitution lists. The content c of a non-physical action, finally, is given by type $LogicalExpr$. It can either be (i) an atomic proposition, a (ii) message term or physical action term, (iii) an expectation network[2], or (iv) a logical formula containing these elements according to table 1. Syntactically, expectation networks are here represented as lists of edges (m, p, n, c, l) where m and n are message terms, p is a transition probability (expectability) from m to n, c is a logical condition, l is a list of variable substitutions. We use functions $in : V \to 2^C$, $out : V \to 2^C$, $source : C \to V$ and $target : C \to V$ which return the ingoing and outgoing edges of a node and the source and target node of an edge, respectively, in the usual sense. C is the set of all edges, V the set of all nodes in the EN. $cond : C \to \mathcal{L}$ returns the conditions of edges, $subst : C \to SubstList$ (with $SubstList$ as in table 1) returns the edges' substitution lists. Edges denote correlations in observed communication sequences. Each cognitive edge is associated with an expectability (returned by $Expect : C \to [0; 1]$) which reflects the probability of $target(e)$ occurring shortly after $source(e)$ in the same communicative context (i.e. in spatial proximity, between the same agents, etc.).

The full meaning of these ingredients will be further clarified once the full definition of expectation networks has been presented. Note that according to table 1 expectation networks are allowed to be contained within message terms of expectation network nodes themselves to allow the modelling of the communication of complex expectation structures among agents.

2.2 Edge Conditions

As a final ingredient to network edges, we briefly discuss edge conditions. The idea is that these conditions should further define the scope of validity to cases in which a formula can be shown to hold using the observer's knowledge base. So, if $\varphi = cond(e)$, then e is only relevant iff $KB \models \varphi$.

Because all conditions for outgoing edges of a certain node should be mutually exclusive to ensure that later the semantics of a certain message trajectory can be calculated unambiguously, we want the sum of expectabilities of all out-edges of a node to be one for a certain knowledge base content[3]. In other words, the condition

$$\forall v \sum_{e \in out(v), KB \models cond(e)} Expect(e) = 1$$

should hold.

This can be ensured, for example, by guaranteeing that the following condition holds through appropriate construction rules for the EN. Assume the outgoing links $out(V)$ of every node v are partitioned into sets $O_1, O_2, \ldots O_k$ where the links' expectabilities in each O_i are non-negative and sum up to one[4]. Now let all edges in O_i share the same edge condition, i.e. $\forall i \exists \varphi \forall o \in O_i.(cond(o) = \varphi)$ and define $cond(O_i)$ as precisely this shared condition φ. (The O_i sets are precisely those sub-sets of $out(v)$ connected by a drawn angle in figure 2.)

If we make sure that the outgoing links of every node are partitioned in this way, we can assign mutually exclusive conditions to them, i.e. ensure that

$$\begin{aligned} \forall i \neq j. cond(O_i) \wedge cond(O_j) &\equiv \text{false} \\ \text{and} \quad \vee_i \ cond(O_i) &\equiv \text{true} \end{aligned}$$

This way, it is not only guaranteed that we can derive unambiguous probabilities directly from the $Expect$ values, but also that we can do so for *any* knowledge base contents (cf. 2.4)[5].

2.3 Formal Definition

Having discussed all the prerequisites, we can now define ENs formally:

Definition 1. An *expectation network* is a structure

$$EN = (V, C, \mathcal{M}, \mathcal{L}, H, mesg, cond, subst, Expect)$$

where

- V with $|V| > 1$ is the set of nodes,
- $C \subseteq V \times V$ are the *cognitive* edges (or edges for short) of EN. (V, C) is a tree called *expectation tree*.
- $\mathcal{M}$ is a *message term language*[6], $\mathcal{L}$ is a logical language, $cond : C \to \mathcal{L}$ returns the conditions of edges,
- $mesg : V \to \mathcal{M}$ is the *message label* function for nodes such that
  - $mesg(v) = \triangleright$ exactly for the root node of (V, C),

[2] Such expectation networks within messages are useful to replace performatives of agent communication languages. We have to refer to [10, 14] for details of this concept.

[3] From a probabilistic point of view, it would be sufficient to demand a sum lower or equal one, but a sum of exactly one (which is practically always feasibly through insertion of a "dummy" '?'-edge) formally ensures the exhaustiveness of the set of outgoing edges.

[4] Formally, $out(v) = \cup_{1 \leq i \leq k} O_i$ and $\forall 1 \leq i < j \leq k. O_i \cap O_j = \emptyset$, and $\forall i \leq k.(\forall o \in O_i.Expect(o) \geq 0 \wedge \sum_{o \in O_i} Expect(o) = 1)$.

[5] This comes at the price of having to insert redundant edges in some situations. For example, insertion of a new edge e with $cond(e) = \varphi$ if $out(v) = \emptyset$ necessitates insertion of another edge e' with $cond(e) = \neg\varphi$.

[6] All languages as defined in the previous sections.

  - $\forall v \in V. \forall e, f \in out(v).$ $\neg unify(mesg(target(e)), mesg(target(f)))$ (where $unify$ shall be $true$ iff its arguments are syntactically unifiable, i.e., target node labels of outgoing links never match),

- $H \in \mathbb{N}$ is a finite *communication horizon*,
- $Expect : C \rightarrow [0; 1]$ returns the edges' expectabilities,
- $subst : C \rightarrow SubstList$ (with $SubstList$ as in table 1) returns the edges' substitution list.

Our full formal framework [17] also defines so-called *normative* edges, which have been omitted here for lack of space. In contrast to cognitve edges, the expectabilities of normative edges are only seldomly adapted by the communication system according to newly observed messages, and under very specific circumstances. In the tradition of systems theory, here the term "cognitive" means "adaptable through cognition about observations".

The only element of this definition that has not been discussed so far is the communication horizon H, which denotes the scope of maximal message sequence length for which the EN is relevant. It is necessary for defining the semantics of the EN, and will be discussed in detail in the following section.

2.4 Formal Semantics of Message Sequences

The purpose of an EN is to provide a semantics for messages. For an arbitrary set S, let $\Delta(S)$ be the set of all (discrete) probability distributions over S with finite support. We define the semantics $I_{EN}(KB, w)$ of an observed message sequence w in a network EN as a mapping from knowledge base states and current message sequence prefixes to the posterior probability distributions over all possible postfixes (conclusions) of the communication. Formally,

$$I_{EN}(KB, w) = f_w, \quad f_w \in \Delta(\mathcal{M}_c^*) \tag{1}$$

where

$$f_w(w') = \frac{g_w(w'\bot)}{\sum_{v \in \mathcal{M}_c^*} g_w(v\bot)} \tag{2}$$

is defined as the normalized value of $g_w(w'\bot)$. $g_w(w'\bot)$ represents the probability that w will be concluded by message sequence w', for any $w, w' \in \mathcal{M}^*$. We compute the probability for $w'\bot$ to make sure w' is followed by a node with label $\bot$ in the network, because the probability of w' is the probability with which the communication sequence will *end* after $w'_{|w'|}$ (and not that w' will simply be the prefix of some longer sequence). Also note that the sum in the denominator is not, as it may seem, infinite, because f_w has finite support and the length of the considered message sequences is limited by means of the communication horizon H (see below), and that the semantics of w depends on KB, because only those edges which have conditions that are true according to KB are used for calculating the semantics of w.

Informally, the probability of w' should be inferred from multiplying all the expectability weights along the path that matches w' (if any). Before presenting the top-level formula for $g_w(w')$, we need some auxiliary definitions:

Firstly, we need to determine the node in a network EN that corresponds to a word w, which we denote by $mesg^{-1}$:

$$mesg^{-1}(\varepsilon) = v \quad :\Leftrightarrow \quad mesg(v) = \triangleright$$

$$mesg^{-1}(wm) = \begin{cases} v' & \text{if } \exists (v, v') \in C(KB). \\ & \exists \vartheta \in subst((v, v')). \\ & (mesg(v') \cdot subst(w)\vartheta = m \\ & \wedge mesg^{-1}(w) = v) \\ \bot & \text{if no such } v' \text{ exists} \end{cases} \tag{3}$$

if $w \in \mathcal{M}_c^*, m \in \mathcal{M}_c$[7]. The first case states that the node corresponding to the empty sequence ε is the unique root node of (V, C) labelled with $\triangleright$. According to the second case, we obtain the node v' that corresponds to a sequence wm if we take v' to be the successor of v (the node reached after w) whose label matches m under the following condition:

There has to be a substitution $\vartheta \in subst((v, v'))$ which, when composed with the substitution $subst(w)$ applied so far to obtain the messages in w_1 to $w_{|w|}$ from the respective nodes in EN, will yield m if applied to $mesg(v')$. This is expressed by $mesg(v') \cdot subst(w)\vartheta = m$. In other words, there is at least one combined (and non-contradictory) variable substitution that will make the node labels along the path $mesg^{-1}(wm)$ yield wm if it is applied to them (concatenating substitutions is performed in a standard fashion). Thereby, the following inductive definition can be used to derive the substitution $subst(w)$ for an entire word w:

$$\begin{aligned} w = \varepsilon : \quad & subst(w) = \langle\rangle \\ w = w'm : \quad & subst(w) = \\ subst(w') \cdot unifier(&mesg(mesg^{-1}(wm)), m) \end{aligned}$$

where $\cdot$ is a concatenation operator for lists and $unifier(\cdot, \cdot)$ returns the most general unifier for two terms (in a standard fashion). Thus, $subst(w)$ can be obtained by recursively appending the unifying substitution of the message label of each node encountered on the path w to the overall substitution. With all this, we are able to compute $g_w(w')$ as follows:

$$g_w(w') = \begin{cases} |\cup_{i=1}^{H} \mathcal{M}_c^i|^{-1} \\ \text{if } \exists v \in out(mesg^{-1}(w)).mesg(v) = ? \\ \prod_i \left(\sum_{e \in pred(ww', i)} S(e) \right) & \text{else} \end{cases} \tag{4}$$

which distinguishes between two cases: if the path to node $mesg^{-1}(w)$ whose labels match w (and which is unique, because the labels of sibling nodes in the EN never unify) ends

[7]For convenience, let $C(KB)$ be the set of nodes within the sub-network of expectation network EN where the edge set is reduced to those edges whose conditions are satisfied under KB.

in a "?" label, the probability of a w' is simply one over the size of all words with length up to the communication horizon H (hence its name). This is because the semantics of "?" nodes is "don't know", so that all possible conclusions to w are uniformly distributed. Note that this case actually only occurs when new paths are generated and it is not known where they will lead, and also that if an outgoing link of a node points to a node with label "?", then this node will have no other outgoing links.

In the second case, i.e. if there is no "?" label on the path p from $mesg^{-1}(w)$ to $mesg^{-1}(ww')$, then the probability of w' is the product of weights $S(e)$ of all edges e on p. Thereby, $S(e)$ is just a generalized notation for expectability or normative force depending on the typed edge, i.e. $S(e) = Exp(e)$ for $e \in C$. The sum of these S-values is computed for all ingoing edges $pred(ww', i)$ of the node that represents the the ith element of w', formally defined as

$$\forall w \in \mathcal{M}_c^*.pred(w,i) = \begin{cases} in(mesg^{-1}(w_1 \cdots w_i)) \\ \text{if } mesg^{-1}(w_1 \cdots w_i) \neq \bot \\ \emptyset & \text{else} \end{cases} \quad (5)$$

3. Communication Systems

A communication system can be seen as a description of the social dynamics of a multiagent system. The two main purposes of a CS are i) to capture the social expectations (represented as an EN) in the current state of a multiagent system under observation, and ii) to capture changes to these expectation structures. Whereas the EN models the current meaning of communicative action sequences (i.e., their expected, generalized continuations in a certain context of previous message utterances), the CS models the way the EN is build up, and, if necessary, adapted according to new statistical observations. As already mentioned in section 1, in contrast to agents who reason about expectations (such as *InFFrA* agents [18]), a CS need not necessarily be an active agent who takes action in the MAS itself.

Describing how communication systems work should involve (at least) clarifying:

- which communicative actions to select for inclusion in an EN,
- where to add them and with which expectability (in particular, when to consider them as "non-continuations" that directly follow "$\rhd$"),
- when to delete existing nodes and edges (e.g. to "forget" obsolete structures), and how to ensure integrity constraints regarding the remaining EN.

A formal framework for specifying the details of the above is given by the following, very general, definition:

Definition 2. A *communication system* at time t is a structure

$$CS_t = (\mathcal{L}, \mathcal{M}, f, \varpi_t, \kappa)$$

where

- $\mathcal{L}$, $\mathcal{M}$ are the formal languages used for logical expressions and messages (according to table 1),
- $f : \mathcal{EN}(\mathcal{L},\mathcal{M}) \times \mathcal{M}_c \to \mathcal{EN}(\mathcal{L},\mathcal{M})$ is the *expectation structures update function* that transforms any expectation network EN to a new network upon experience of a message $m \in \mathcal{M}_c$,
- $\varpi_t = m_0 m_1 ... m_t \in \mathcal{M}_c^*$ is the list of all messages observed until time t. The subindexes of the m_i impose a linear order on the messages corresponding to the times they have been observed[8].
- $\kappa : 2^{\mathcal{L}} \times \mathcal{M}_c \to 2^{\mathcal{L}}$ is a *knowledge base update function* that transforms knowledge base contents after a message accordingly,

and $\mathcal{EN}(\mathcal{L},\mathcal{M})$ is the set of all possible expectation networks over $\mathcal{L}$ and $\mathcal{M}$. The intuition is that a communication system can be characterized by how it would update a given knowledge base and an existing expectation network upon newly observed messages $m \in \mathcal{M}_c$. The EN within CS_t can thus be computed through the sequential application of the expectation structures update function f for each message within ϖ, starting with an empty expectation network (i.e., an EN which contains only the start node). ϖ_{t-1} is called the *context* of the message observed at time t, and $I_{EN}(KB, \varpi_t)$ computes the semantics of this message within this context.

This definition of CS is very general, as it does not prescribe how the EN is modified by the CS. However, some assumptions are reasonable to make, although not obligatory (see [14] for a concrete EN-/CS-learning algorithm):

- If KB is the current knowledge base, $\kappa(KB, m) \models KB(m)$ should hold, so that all facts resulting from execution of m are consistent with the result of the κ-function.
- An EN should predict the future of the respective observable communication sequences as accurately as possible. Although there is no canonical method a CS should use to construct and update ENs, we propose the following very general heuristic: if any message sequence w' has occurred with frequency $\Pr(ww')$ as a continuation of w in the past, and EN' is the same as EN, $I_{EN'}(KB, w)(w') = \Pr(ww')$ should be the case, i.e. the expectabilities along a certain path within the expectation tree shall reflect the frequencies with which the respective message sequences have been occurred.

[8]For simplicity, we assume a discrete time scale with $t \in \mathbb{N}$, and that no pair of messages can be uttered at the same time.

In addition to these basic assumptions, we propose the following functionality a CS shall provide to be of practical use:

3.1 Message Filtering and Syntax Recognition.

Depending on its goals and the application domain, the CS as an autonomous observer might not be interested in all observable messages. Since ENs may not provide for *a priori* expectations, the discarding of such "uninteresting" messages can only take place *after* the semantics (i.e., the expected outcome) of the respective messages has already been derived from previous observation. Because discarding messages bears the risk that these messages become interesting afterwards, as a rule of thumb, message filtering should be reduced to a minimum. More particularly, messages should only be filtered out in cases of more or less settled expectations. Paying attention to every message and filtering uninteresting or obsolete information later by means of structure reweighting and filtering (cf. below) is presumably the more robust approach.

3.2 Structure Expansion and Generalization.

Structure expansion is concerned with the growth of an EN in case a message sequence is observed which has no semantics defined by this EN yet. In order to do so, we could start with an empty EN and incrementally add a node for each newly observed message. But this would be not smart enough, because it does not take advantage of generalizable message sequences, i.e. different sequences that have approximately the same meaning. In general, such a generalization requires a relation which comprises "similar" sequences. The properties of this relation of course depends on domain- and observer-specific factors. A quite simple way of generalizing is to group messages which can be unified syntactically, using the message patterns introduced in table 1.

In theory, the expansion of the EN would never be necessary if we could a-priori generate a *complete EN*, i.e. an EN which contains dedicated paths for *all* possible message sequences. In this case, the CS would just have to keep track of the perceived messages using ϖ and to identify this sequence within the EN to derive its semantics, with no need for f. For obvious reasons, such a complete EN cannot be constructed in practice.

3.3 Pruning the EN.

Several further methods of EN processing can be conceived of that aid in keeping the computation of (approximate) EN semantics tractable. This can be achieved by continuously modifying expectation structures using certain meta-rules, for example:

1. "fading out" old observations by levelling their edge weights;

2. replacing large sets of sibling edges with (approximately) uniformly distributed expectabilities with single edges leading to "?" nodes;

3. removal of "?"s that are not leafs. Such nodes can occur as outcome of the previous measure.

4. keeping the EN depth constant through removal of one old node for each new node to save space and remove obsolete structures (e.g. using the communication horizon H as maximum EN depth);

5. removal of edges with very low expectabilities. In case this results in cut-off branches, these have to be connected with the start node subsequently.

Since these modifications are highly application-dependent, we don't provide exact criteria for their practical application here.

4. Applications and Extensions

The modelling of social structures on the basis of expectation networks and communication systems allows for novel approaches to a variety of challenging issues in multiagent system technology. In the following, we review three of these issues, namely, (i) identification of ontologies for inter-agent communication and – closely related – the finding of verifiable and flexible semantics for agent communication languages; (ii) *mirror holons* as a new model for holonic theories of agency and software engineering methods based on expectation-oriented modelling and analysis of multiagent systems; (iii) the agent-level social reasoning architecture *InFFra*.

4.1 Social Ontologies

In Distributed Artificial Intelligence (DAI), an ontology is a set of definitions as a means to provide a common ground in the conceptual description of a domain for communication purposes. Ontologies are usually represented as graphical hierarchies or networks of concepts, topics or classes, and either top-down imposed on the agents or set up bottom-up by means of ontology negotiation. In a similar way, expectation networks are descriptions of the social world in which the agents exist. But ENs do not only describe social (i.e. communication) structures, but indirectly also the communication-external environment the message content informs about. Thus, communication systems can be used, in principle, for an incremental collection of ontological descriptions from different autonomous sources, resulting in stochastically weighted, possibly conflicting, competitive and revisable propositions about environmental objects [21]. The crucial difference to traditional mechanisms is that such a *social ontology* (also called *Open Ontology* [21, 20, 19]) includes probabilistic expectations about how a certain domain object or event is described, constructed and used *socially by multiple autonomous knowledge sources* (which do not necessarily have consistent views and which are not necessarily cooperative, reliable or trustable), i.e. in the course of communication processes using communicative means like approval, contradiction, conflict, argumentation or specification. This opposes somewhat the traditional understanding of ontologies, where an ontology provides an *a priori* grounding for communication only instead of taking into consideration that ontologies are *evolving, possibly inconsistent results* of communication also, and it makes this social approach appear particularly suitable for the Semantic Web, open multiagent systems, Peer2Peer systems and *communities of practice* with a highly dynamic environment, where a commonly agreed, homogenous domain perception of distributed, autonomous knowledge sources cannot be assumed. Also, it is appropriate whenever distributed domain descriptions are influenced by individual preferences such that a consensus cannot be achieved (think, e.g., about "politically" biased resource descriptions in the context of Peer2Peer systems or the Semantic Web [22]). In the following, we'll sketch two approaches for extracting social ontologies from expectation networks (for more information about the underlying approach called *Social Reification* see [21, 20, 19]).

4.1.1 Extraction of Speech Act Types.

The current version of FIPA-ACL [23] provides an extensible set of speech-act performative types with semantics defined in a mentalistic fashion. In our approach, we can imagine a special CS variant as a MAS component (e.g., a so-called multiagent system *mirror* [2, 1], cf. 4.2) that provides the agents with a set of performatives *without* any predefined semantics and wait for the semantics of such "blank" performatives to emerge. To become predictable, it is rational for an agent to stick to the meaning (i.e., the consequences) of performatives, at least to a certain extent. This meaning has been previously (more or less arbitrarily) "suggested" for a certain performative by some agent performing demonstrative actions after uttering it.

Of course, a single agent is usually not able to define a precise and stable public meaning for these performatives, but at least the intentional attitude associated with the respective performative needs to become common ground for communication to facilitate a non-nonsensical, non-entropic discourse [11, 22, 14]. A particular performative usually appears at multiple nodes within the EN, with different consequences at each position, depending on context (especially on the preceding path), message content and involved sender and receiver. To build up an ontology consisting of performative types, we have to continually identify and combine the occurrences of a certain performative within the current EN to obtain a general meaning for this performative (i.e., a "type" meaning). Afterwards, we can communicate this meaning to all agents using some technical facility within the multiagent system, like a MAS mirror or an "ACL semantics server". Of course, such a facility cannot impose meaning in a normative way as the agents are still free to use or ignore public meaning as they like, but it can help to spread language data like a dictionary or a grammar does for natural languages. The criteria for the identification and extraction of performative meaning from ENs are basically the same as the criteria we proposed in 3 for the generalization over message sequences.

4.1.2 Extraction of Domain Descriptions.

While a set of emergent speech act types constitutes a social ontology for communication events, classical ontologies provide a description of an application domain. To obtain a social version of this sort of ontology from an EN, two different approaches appear to be reasonable: (1) Inclusion of environment events within the EN and (2) probabilistic weighting of assertions. The former approach, which is introduced in [22, 14], treats "physical" events basically as utterances. Similar to the communicative reflection of agent actions by means of do, a special performative happen($event$) would allow EN nodes that reflect events occurring in the environment. These events will be put in the EN either by a special CS which is able to perceive the agents' common environment, or by the agents themselves as a communicative reflection of their own perceptions. A subset of $event$ is assumed to denote events with consensual semantics (think of physical laws), i.e., the agents are not free to perform an arbitrary course of action after such an event has occurred, whereas the remainder of $event$ consists of event tags with open semantics that has to be derived empirically from communications observation just as for "normal" utterances. If such an event appears for the first time, the CS does not know its meaning in terms of its consequences. Its meaning has thus to be derived a-posteriori from the communicational reflection of how the agents react to its occurrence. In contrast, approach (2), which we proposed for the agent-based competitive rating of web resources [22], exploits the propositional attitude of utterances. The idea is to interpret certain terms within $LogicalExpr$ as domain descriptions and to weight these descriptions according to the amount of consent/dissent (using predefined performatives like $Assert$ and $Deny$). The weighted propositions are collected within a knowledge base (e.g., KB as defined before) and are communicated to the agents in the same way as the emergent speech act types before. Unlike approach (1), ontologies are constructed "by description" not "by doing" in this way. The advantage of approach (1) lies in its seamless integration of "physical" events into the EN, whereas (2) is probably more easy to apply in practice.

4.2 The *Mirror* concept

In [1, 12], we have introduced the *social system mirror* architecture for open MAS. The main component of this architecture is a so-called *social system mirror* (or "mirror" for short), an intelligent system component containing a CS which continually observes communications, empirically derives emergent expectation structures (represented as an ENs, which might also contain *normative structures*, which are given by the designer instead of being learned from statistical observations) from these observations, and "reflects" these structures back to the agents. In addition to the recording of empirical expectation structures described in this work, the mirror is a goal-directed agent within the MAS. Its goals are to influence agent behavior by means of system-wide propagation of social structures and norms to achieve quicker structure evolution (catalysis) and higher coherence of social structures without restricting agent autonomy, and the provision of a representation of a dynamic communication system for the MAS designer.

Technically, a mirror can be thought of as a knowledge base which derives system-level expectation structures from communications and makes them available as information for both the participating agents and the designer of the multiagent system. It can be implemented as an actual software component (e.g. a middle agent), but it might as well be used without a physical implementation as a purely theoretical concept. The mirror has three major purposes:

1. monitoring agent communication processes,

2. deriving emergent system-level expectation structures from these observations, and calculating the deviation of actual agent behaviour from normative expectations,

3. making expectation structures visible for the agents and the designer (the so-called *reflection effect* of the mirror).

The published structures are not necessarily the emergent structures derived from the system observation - the mirror can be pre-structured, i.e. used to "reflect" manually designed ("manipulated"), non-emergent expectation structures as well. In both cases, the agents can query the mirror very much like a database and actively use the otherwise latent social structures which are made explicit to them through the mirror as a guide for their decision making and their interaction behaviour.
For example, agents can instantiate themselves in social programs which seem to be useful to them, or refrain from a certain behaviour if the mirror tells them that it would violate a norm. If the agents make (further) use of reflected structures, the structures become stronger, otherwise weaker (the degree of these changes depending on the respective normativity). Thus, the mirror reflects a model of a social system to the agents and by means of this influences the agents – very much like mass media do in human society. Conversely, it continualy observes the actual multiagent system and adopts the expectation structures in its database in accordance with the agent interactions. In doing so, the mirror never restricts the autonomy of the agents. Its influence is solely by means of information, not through the exertion of control.

4.2.1 Expectation-Oriented Software Development

As it has been recognized that due to new requirements arising from the complex and distributed nature of modern software systems the modularity and flexibility provided by object orientation is often inadequate and that there is a need for encapsulation of robust functionality at the level of software components, agent-oriented approaches are expected to offer interesting prospectives in this respect, because they introduce interaction and autonomy as the primary abstractions the developer deals with.

However, although interaction among autonomous agents offers great flexibility, it also brings with it contingencies in behavior. In the most general case, neither peer agents nor the MAS designer can "read the mind" of an autonomous agent, let alone change it. While the usual strategy to cope with this problem is to restrict oneself to closed systems, this means loosing the power of autonomous decentralized control in favour of a top-down imposition of social regulation to ensure predictable behavior. The EXPAND method (Expectation-oriented Analysis and Design) [12] follows a different approach. EXPAND is based on expectation networks as a primary modelling abstraction which both system designer and agents use to manage the social level of their activities. This novel abstraction level is made available to them through a special version of the social system mirror (4.2), i.e., a special CS, very similar to a CASE tool. For the designer, this mirror acts as an interface he uses to propagate his desired expectations regarding agent interaction to the agents and as a means for monitoring runtime agent activity and deviance from expected behavior. For agents, this mirror represents a valuable "system resource" they can use to reduce contingency about each other's behavior. EXPAND also describes an evolutionary process for MAS development which consists of multiples cycles: the modelling of the system level, the derivation of appropriate expectation structures, the monitoring of expectation structure evolution and the refinement of expectation structures given the observations made in the system.
Within the EXPAND process, a social system mirror is used in a cyclic process of two alternating mirror operations which are the core of our design process:

1. It makes the system-level expectations the software designer has derived from her design goals explicit and known to the agents (i.e. it makes them the "expectations of expectation" described in the previous section).

2. It monitors the system-level expectation structures which emerge from the communications of the running multiagent system.

Together, both operations allow the designer to control and influence if and how her design specifications are realized and adopted by the agents. This approach borrows its underlying concept from the "evolutionary software engineering method", and will now be sketched from the designers' point of view.

- Phase I: Modelling the system level

In the first phase of the process, the software designer models the system level of the multiagent system according to her design goals in the form of design specifications which focus on "social behaviour" (i.e. desired courses of agent interaction) and "social functionality" (i.e. functionality which is achieved as a "product" of agent interaction, such as cooperative problem solving) in the widest sense (we don't take into account "second-order" design goals like high execution speed or low memory consumption). For this task, the usual specification methods and formalisms can be used, e.g. the specification of desired environment states, constraints, social plans etc. In addition or as a replacement, the specification can be done in terms of system-level expectation structures, like social programs.

- Phase II: Deriving appropriate expectation structures

In the second phase, the designer models and derives system-level expectation structures from the design specifications and stores them in the social system mirror. If the design specifications from phase I are not already expectation structures (e.g. they might be given as rules of the form "Agent X must never do Y"), they have to be transformed appropriately. While social behaviour specifications are expectation structures *per se*, social functionalities (for instance: "Agents in the system must work out a solution for problem X together") possibly need to be transformed, most likely into social programs. Sometimes a full equivalent transformation will not be feasible. In this case, the designer models expectation structures which cover as much design requirements as possible.

System-level specifications can be modelled as adaptable or normative expectations. The former can be used for the establishing of *hints* for the agents which are able to adapt during the structure evolution, the latter for the transformation of *constraints* and other "hard" design requirements into expectations[9].

Figure 2 shows the spectrum of system-level specifications and expectation structures that result from this phase of the analysis and design process.

- Phase III: Monitoring structure evolution

After the designer has finished the expectation modelling, she makes them visible for the agents via the social system

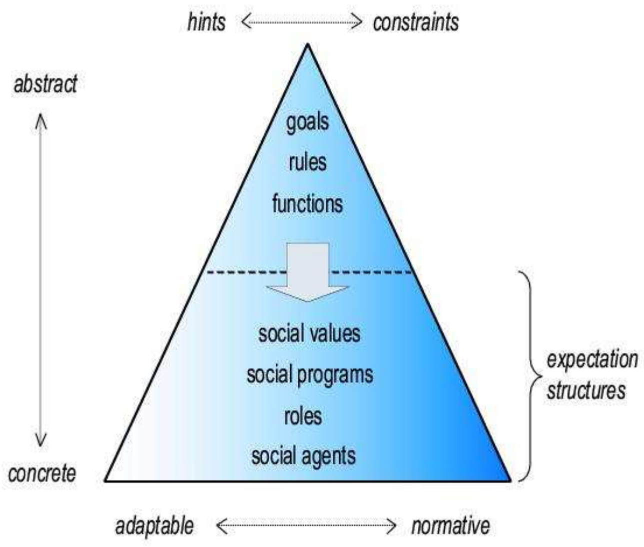

Figure 2. System-level specification

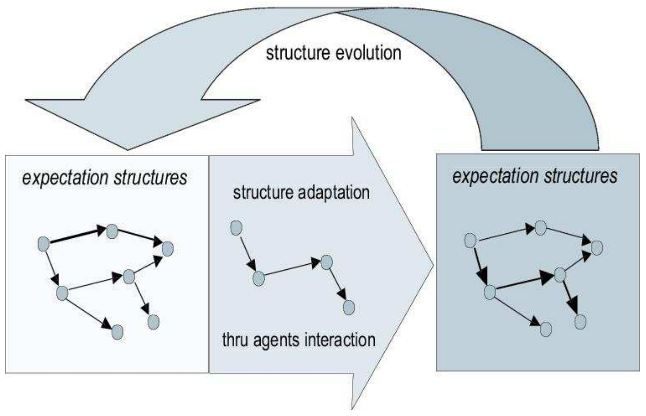

Figure 3. Evolution of expectation structures

[9]It should be kept in mind that a norm derived from a constraint does not force the agents to behave conforming to the rule, since it is "only" an expectation. In some cases, where expectations seem too "soft" a modelling tool, direct programming of agent goals and behaviour might be necessary. This is certainly the case for agents that are *built* by the same designer (which is not the default case in open systems), and that would be underspecified if their reasoning capabilities are not defined more rigidly.

mirror and puts the multiagent system into operation (if it is not already running). In the third phase of the design and analysis process, it is up to the designer to observe and analyse the evolution of expectation structures which becomes visible to her through the mirror (Figure 3). In particular, she has to pay attention to the relationship of the continuously adapted system-level expectation structures and her design specifications from phase I, which means that she analyses the expectation structures with regard to the fulfilment of norms established by the designer and the achievement of the desired social functionality. Because the mirror is only intended to show expectation structures, it could be necessary to support the mirror with a software for the (semi-)automatical "re-translation" of expectation structures into more abstract design specifications like social goals.
As long as the expectations structures develop in a positive way (i.e. they match the design goals) or no emergent structures can be identified that deserve being made explicit to improve system performance, the designer does not intervene. Otherwise she proceeds with phase IV.

- Phase IV: Refinement of expectation structures

In the last phase, the designer uses her knowledge about the positive or negative emergent properties of the multiagent system to improve the system-level expectation structures. Usually, this is achieved by removing "bad" expectation structures from the mirror database, and, if necessary, the introduction of new expectation structures as described at phases I and II. In addition, expectation structures which have proved to be useful can be actively supported by e.g. increasing their expectation strength and/or their normativity. The process proceeds with phase III until all design goals are achieved or no further improvement seems probable.

For lack of space, we have to refer the interested reader to [12] for further details.

4.2.2 *Mirror Holons*: Multi-Stage Observation, Reflection and Enactment of Communication Structures

While a social system mirror only models a single communication system, and, except for the propagation of expectations, does not take action itself, the successor architecture *HoloMAS* [2] is able to model multiple communication systems at the same time through multiple *mirror holons* in order to model large, heterogenous systems. In addition, a mirror holon can take action himself by means of the execution of social programs which are generated from emergent expectation structures. "Ordinary agents" (and other mirror holons) can optionally be involved in this execution process as *effectors*, which realize holon commands within their physical or virtual application domain (unless they deny the respective command). In any case they can influence the social programs within a mirror holon through the irritation of expectation structures by means of communication. A mirror holon thus represents and (at least to some extent) replaces the functionality of the ordinary agents that contribute to the emergence of the respective expectation structures, but it does not disregard the autonomy of his adjoint actors. Another difference between mirror holons and traditional agent holons is that a mirror holon does not represent or contain groups of agents, but instead a certain functionality which is identified in form of regularities in the observed communications. This functionality is extracted and continually adopted from dynamic expectation structures regarding criteria like consistency, coherence and stability, corresponding to the criteria sociological systems theory ascribes to social programs [13]. Mirror holons pave the way for applications in which agent autonomy should not (or cannot) be restricted on the one hand, while reliable, time-critical system behavior is desired. They can also be used as representatives for entire communication systems (e.g., virtual organizations) that behave smoothly towards third parties whenever the communication system itself lacks coherence due to, for example, inner conflicts.

4.3 Social reasoning with InFFrA

The Interaction Frame and Framing Architecture InFFrA [18] is a social reasoning architecture in which so-called *interaction frames* are used to represent patterns of social interaction and strategically employed by socially intelligent agents to guide their interaction and communication behaviour. This is achieved by agents deriving models of frames from observation of encounters and applying the most appropriate patterns in future interactions (this process is called *framing*). The concepts of frame and framing are based on Erving Goffman's micro-social analyses of everyday life [16].

In the conceptual (abstract) architecture, a frame is a data structure that contains information about

- the possible courses of interaction (so-called *trajectories*) characteristic for a particular frame,
- *roles and relationships* between the parties involved in an interaction of this class,
- *contexts* within which the interaction may take place, and
- *beliefs*, i.e. epistemic states of the interacting parties.

In computational terms, the trajectory model is usually a representation of a set of admissible message and action sequences, while the latter three elements can be collapsed into a single set of logical constraints which then have to be verified using the agent's internal belief state (usually represented by the contents of a knowledge base).

InFFrA uses the following data structures for reasoning with frames:

- the *active frame*, the unique frame currently activated to describe the expected course of events,
- the *perceived frame*, an interpretation of the currently observed state of affairs,

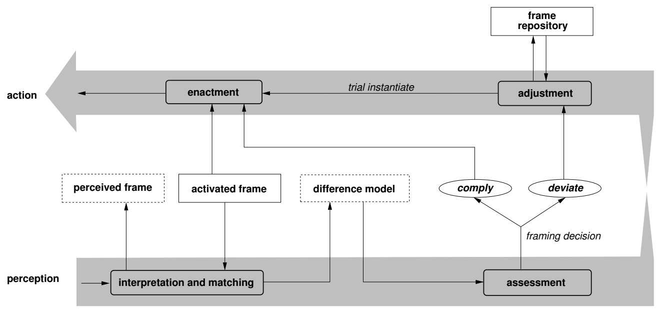

Figure 4. Overview of the framing process

- the *difference model* that contains the differences between perceived frame and active frame,
- the *trial frame*, used when alternatives to the current frame are sought for,
- and the *frame repository*, in which the agent locally stores its frame knowledge.

Using these data structures, the InFFrA framing cycle (as shown in a simplified fashion in figure 4) consists of the folowing reasoning steps:

1. *Interpretation & Matching:* Update the perceived frame and compare it with the active frame.
2. *Assessment:* Assess the usability of the active frame in terms of
   (i) adequacy (compliance with the conditions of the active frame),
   (ii) validity (the degree to which the active frame's trajectory matches the perceived encounter) and
   (iii) desirability (depending on whether the implications of the frame correspond to the agent's private goals).
3. *Framing decision:* If the active frame seems appropriate, continue with 5. Else, proceed with 4 to find suitable alternatives.
4. *Adjustment/Re-framing:* Search the frame repository for better frames. "Mock-activate" them as trial frames iteratively and go back to 1; if no suitable frame is found, end the encounter.
5. *Enactment:* Derive action decisions by applying the active frame.

From a CS perspective, InFFrA is nothing but an *agent-centric* interpretation of our concepts. Instead of defining a general observer of communication, InFFrA exclusively deals with agent observers, and rather than observing general communication processes, InFFrA agents only observe "face-to-face" interaction processes (mostly those they are personally involved in).

In other words, interaction frames in InFFrA are micro-models of communicative expectations that encode knowledge about communication processes from the standpoint of an agent observer. They contain information about the surface structure of conversations together with logical conditions in the same way as general ENs but as a collection of "manageable chunks" of dialogue traces rather than a huge network of global correlations. This allows for handling expectations in a computationally tractable fashion and with this the concept of frames makes the CS approach accessible for the design of social reasoning methods.

What makes InFFrA an interesting extension of the general CS framework is the fact that agents actually have to strategically decide which utterances to generate in accordance with their current model of the CS to achieve their goals. In [11] we have suggested entropy-based methods for reconciling the utility-based preferences of InFFrA agents with long-term considerations about the effect of their decisions on the overall CS in decision-theoretic terms. There, the interesting question was how agents can achieve a trade-off between their current pursuit for high utility and the modifications to the CS that will result from their current decision.

In [17], we have shown how InFFrA frames can be formally converted to general expectation networks. Of course, some problems occur when attempting to transform general expectation networks to interaction frames, since the full expressiveness of expectation networks is not available in the formal model of InFFrA for reasons of practicability.

Based on the empirical semantics approach, a formal model of InFFrA has been developed for a particular, concrete instance of the abstract architecture [24]. This model called m^2InFFrA is based on viewing frames as *policy abstractions* in the sense of Markov Decision Processes (MDPs) [25].

More specifically, in m^2InFFrA, a frame describes a set of two-party, discrete, turn-taking interaction *encounters* which can be thought of as conversations between two agents. The trajectory is given in the form of a sequence of message patterns that defines the surface structure of the encounters described by the frame, while a list of *substitutions* captures the values of variables in the trajectory in previously experienced interactions. Each substitution also corresponds to a set of logical *conditions* that were required for and/or precipitated by execution of the trajectory in the respective encounter. Finally, *trajectory occurrence* and *substitution occurrence* counters record the frequency with which the frame has occurred in the past.

In terms of MDP theory, frames can be seen as "macro"-actions can be invoked as MDP decisions and then executed until certain conditions apply (typically, until they are considered undesirable according to some heuristics, until their context conditions are no more fulfilled, or until the other agent has uttered a message that does not match the trajectory of the frame).

This allows us to combine the principles of InFFrA with the *options* framework [26] for hierarchical reinforcement learn-

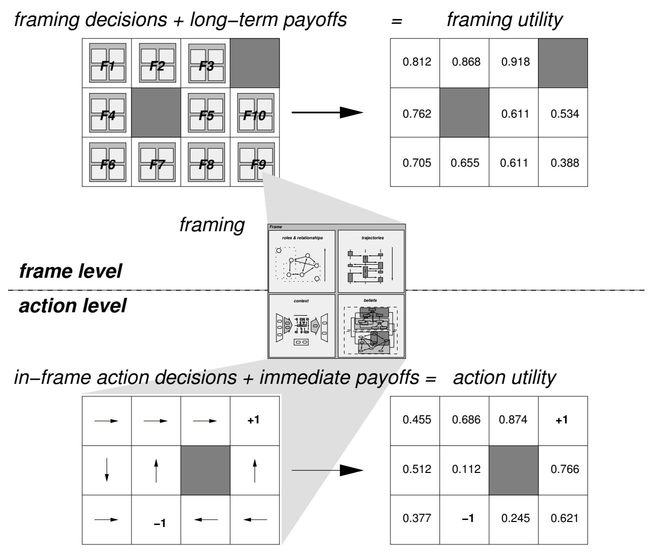

Figure 5. Frame-based hierarchical view of communication MDPs.

ing (RL; for an introduction, see e.g. [27]). This framework is based on augmenting the sets of admissible "primitive" actions by sets of so-called "options", where an option is a triple consisting of an input set of states, a policy (that is admissible once a state in the input set is entered), and a termination condition that determines when an option will be exited and a new one has to be selected.

Combining the options framework with m^{2}InFFrA, we obtain a two-level ocial reasoning and learning view as shown in figure 5. According to this view, what we obtain is a two-level MDP:

- At the *frame level*, the agent chooses a frame as a communication policy that may be used over an extended period of time, depending on whether it can be successfully completed. We employ Q-learning [28] to learn a long-term "framing" utility function that enables us to derive optimal strategies for frame selection from experience. The states of this "upper" MDP are abstract representations of the goal of a conversation.

- At the *action level*, we have to determine which concrete instance of a frame to select so as to optimise the outcome of a conversation. Remembering that frames contain message patterns that may allow for additional choices (e.g. regarding which argument to use in an argumentation dialogue), we use *adversarial* search to maximise expected utility considering the other's potential reactions.

This not only allows for combining the theory of empirical communication semantics and communication systems with the decision-theoretic principles of MDP theory, it also nicely illustrates that we can apply hierarchical methods to cope with the complexity of the usually huge spectrum of possible communicative behaviour. Finally, and maybe most importantly, it allows for the construction of agents that are able to adapt to a particular communication system and to use it according to their own needs.

In future work, we are going to investigate how CS that have been observed by global entities can be used by agents in the system to improve their interaction behaviour.

5. Conclusion

This paper presented communication systems as a unified model for socially intelligent systems based on recording and transforming communicative expectations. We presented formalisms for describing expectations in terms of expectation networks, the formal semantics of these networks, and a general framework for transforming them with incoming observation. Then, a number of important applications of CS were discussed, some of which have already been addressed by our past research, while others are currently being worked on.

While a lot of work still lies ahead, we strongly believe that, by virtue of their general character, CS have the potential of becoming a unified model for speaking about methods and applications relevant to the improvement of multiagent systems using sociological theories [29]. Also, we hope that they can contribute to bringing key insights of this new research direction to the attention of the mainstream DAI audience, as they put emphasis on certain aspects of MAS that are often neglected in traditional approaches.

Acknowledgements
This work has partially been supported by Deutsche Forschungsgemeinschaft under contracts no. BR 609/11-2 and MA 759/4-2.

6. References

[1] K. F. Lorentzen and M. Nickles. Ordnung aus Chaos – Prolegomena zu einer Luhmann'schen Modellierung deentropisierender Strukturbildung in Multiagentensystemen. In T. Kron, editor, *Luhmann modelliert. Ansätze zur Simulation von Kommunikationssystemen.* Leske & Budrich, 2001.

[2] Matthias Nickles and Gerhard Weiss. Multiagent Systems without Agents – Mirror-Holons for the Compilation and Enactment of Communication Structures. In K. Fischer and M. Florian, editors, *Socionics: Its Contributions to the Scalability of Complex Social Systems*, LNCS. Springer-Verlag, Berlin, Germany, 2004. To appear.

[3] P. R. Cohen and H. J. Levesque. Performatives in a Rationally Based Speech Act Theory. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 79–88, 1990.

[4] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In *Proceedings of the*

First International Conference on Multi-Agent Systems (ICMAS-95), pages 65–72, 1995.

[5] Y. Labrou and T. Finin. Semantics and conversations for an agent communication language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 584–591, 1997.

[6] M. P. Singh. A semantics for speech acts. *Annals of Mathematics and Artificial Intelligence*, 8(1–2):47–71, 1993.

[7] Frank Guerin and Jeremy Pitt. Denotational Semantics for Agent Communication Languages. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents-01)*, pages 497–504. ACM Press, 2001.

[8] J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[9] M.P. Singh. A social semantics for agent communication languages. In *Proceedings of the IJCAI Workshop on Agent Communication Languages*, 2000.

[10] M. Nickles and G. Weiss. Empirical Semantics of Agent Communication in Open Systems. In *Proceedings of the Second International Workshop on Challenges in Open Agent Environments at AAMAS-2003*, 2003.

[11] M. Rovatsos, M. Nickles, and G. Weiß. Interaction is Meaning: A New Model for Communication in Open Systems. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, Melbourne, Australia, 2003.

[12] W. Brauer, M. Nickles, M. Rovatsos, G. Weiß, and K. F. Lorentzen. Expectation-Oriented Analysis and Design. In *Proceedings of the 2nd Workshop on Agent-Oriented Software Engineering (AOSE-2001) at the Autonomous Agents 2001 Conference*, volume 2222 of *LNAI*, Montreal, Canada, May 29 2001. Springer-Verlag, Berlin.

[13] N. Luhmann. *Social Systems*. Stanford University Press, Palo Alto, CA, 1995. translated by J. Bednarz, Jr. and D. Baecker.

[14] M. Nickles, M. Rovatsos, and G. Weiss. Empirical-Rational Semantics of Agent Communication. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, New York, NY, 2004.

[15] G. H. Mead. *Mind, Self, and Society*. University of Chicago Press, Chicago, IL, 1934.

[16] E. Goffman. *Frame Analysis: An Essay on the Organisation of Experience*. Harper and Row, New York, NY, 1974. Reprinted 1990 by Northeastern University Press.

[17] M. Nickles and M. Rovatsos. Communication Systems: A Unified Model of Socially Intelligent Systems. In K. Fischer and M. Florian, editors, *Socionics: Its Contributions to the Scalability of Complex Social Systems*, LNCS. Springer-Verlag, Berlin, Germany, 2004. To appear.

[18] M. Rovatsos, G. Weiß, and M. Wolf. An Approach to the Analysis and Design of Multiagent Systems based on Interaction Frames. In Maria Gini, Toru Ishida, Cristiano Castelfranchi, and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, Bologna, Italy, 2002. ACM Press.

[19] T. Froehner, M. Nickles, and G. Weiß. Open Ontologies – The Need for Modeling Heterogeneous Knowledge. In *Proceedings of The 2004 International Conference on Information and Knowledge Engineering (IKE 2004)*, 2004.

[20] T. Froehner, M. Nickles, and G. Weiß. Towards Modeling the Social Layer of Emergent Knowledge Using Open Ontologies. In *Proceedings of The ECAI 2004 Workshop on Agent-Mediated Knowledge Management (AMKM-04)*, Valencia, Spain, 2004.

[21] M. Nickles and T. Froehner. Social Reification for the Semantic Web. Research Report FKI-24x-04, AI/Cognition Group, Department of Informatics, Technical University Munich, 2004.

[22] M. Nickles and G. Weiß. A Framework for the Social Description of Resources in Open Environments. In *Proceedings of the 7th International Workshop on Cooperative Information Agents (CIA-2003)*, volume 2782 of *Lecture Notes in Computer Science*, Berlin, Germany, 2003. Springer-Verlag.

[23] FIPA. FIPA (Foundation for Intelligent Agents), http://www.fipa.org, 1999.

[24] F. Fischer. Frame-Based Learning and Generalisation for Multiagent Communication. Diploma Thesis, Department of Informatics, Technical University Munich, Munich, Germany, December 2003.

[25] M. L. Puterman. *Markov Decision Problems*. John Wiley & Sons, New York City, NY, 1994.

[26] Andrew Barto and Sridhar Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Special Issue on Reinforcement Learning, Discrete Event Systems*, 13:41–77, 2003.

[27] R.S. Sutton and A.G. Barto. *Reinforcement Learning. An Introduction*. MIT Press/A Bradford Book, Cambridge, MA, 1998.

[28] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[29] Th. Malsch. Naming the Unnamable: Socionics or the Sociological Turn of/to Distributed Artificial Intelligence. *Autonomous Agents and Multi-Agent Systems*, 4(3):155–186, 2001.

Matthias Nickles studied Computer Sciences (minor Psychology) at the University of Munich, where he graduated with a Diploma degree. After graduation, he has been a scientific software developer and a research assistant for a research project in Theoretical Linguistics, and worked as a developer of web software for a commercial internet access provider. Currently, he finishes his PhD thesis on agent communication at the Department of Informatics, Technical University Munich, where he is a research associate and member of the Artificial Intelligence/Cognition group. His research interests include agent communication languages, multiagent systems, formal ontologies and knowledge modeling, functional programming languages, and computational linguistics.

Michael Rovatsos is a research associate and PhD student at the Department of Informatics, Technical University Munich, Germany. His research interests include models of interaction in multiagent systems, multiagent learning, agent communication languages, and agent-oriented software engineering. Michael received his Diploma in Computer Science from the University of Saarbruecken in 1999, where he also worked for the DFKI multiagent systems group. Before coming to Munich, he worked for Knowbotic Systems, a Frankfurt-based AI startup. He is currently completing his PhD thesis on ”Computational Interaction Frames”.

Prof. Dr. Dr. h.c. Wilfried Brauer - Full professor of Informatics since 1971 (first at the University of Hamburg, since 1985 at Technical University Munich)
- ordinary member of the Bavarian Academy of Sciences and of the Academia Europaea, London
- former president of the European Association for Theoretical Informatics (EATCS); former vice president of International Federation for Information Processing (IFIP)
- third honorary member (after Konrad Zuse and F.L. Bauer - number 4 is G. Hotz) of Gesellschaft für Informatik; recipient of the Werner Heisenberg Medal of the Alexander von Humboldt Foundation and of the Felix Hausdorff Memorial Award from the University of Bonn. (Honorary doctoral degree from Hamburg)
- Research and teaching in theoretical informatics and foundations of Artificial Intelligence
- particularly interested in interdisciplinary research; close cooperation with industry

Dr. Gerhard Weiß leads the AI/Cognition Group at the chair for Theoretical Informatics and Foundations of Artificial Intelligence (headed by Prof Dr W. Brauer) of Technical University Munich TUM since September 1997. His main interests have been in computational intelligent and autonomous systems in general, and in the foundations and application of agent and multiagent technology in particular. Gerhard Weiß co/authored a number of articles and co/edited ten books in his areas of interest. He is the editor-in-chief of the international book series on Multiagent Systems, Artificial Societies, and Simulated Organizations, and he serves as an editorial board member of several international journals. Since 2000 he is speaker of the official German interest group on distributed artificial intelligence within the German Informatics association (GI); since 2002 he is a member of the scientific board of the European Network of Excellence for Complex Systems (Exystence); from 1998 until 2001 he was a member of the management board of the European Network of Excellence for Agent-based Computing (Agentlink); and since 1999 he is in the Board of Directors of the International Foundation for Multiagent Systems (IFMAS).