Expectation-oriented Modeling*

Matthias Nickles*, Michael Rovatsos†, Gerhard Weiss*

*Department of Informatics
Technical University of Munich
D-85748 Garching, Germany
{nickles,weissg}@cs.tum.edu

†School of Informatics
University of Edinburgh
Edinburgh EH8 9LE, UK
mrovatso@inf.ed.ac.uk

Abstract

This work introduces *Expectation-oriented Modeling* (EOM) as a conceptual and formal framework for the modeling and influencing of black- or gray-box agents and agent interaction from the viewpoint of modelers like artificial agents and application designers. EOM is unique in that autonomous agent behavior is not restricted in advance, but only if it turns out to be necessary at runtime, and does so exploiting a seamless combination of evolving probabilistic and normative behavioral *expectations* as the key modeling abstraction and as the primary level of analysis and influence. Expectations are attitudes which allow for the relation of observed and predicted actions and other events to the modeler's intentions and desires on the one hand and her beliefs on the other in an integrated, adaptive manner. In this regard, this work introduces a formal framework for the representation and the semantics of expectations embedded in social contexts. We see the applicability of EOM especially in open domains with a priori unknown and possibly unreliable and insincere actors, where the modeler can not rely on cooperation or pursue her goals through the exertion of strictly normative power, e.g. the development and assertion of flexible interaction policies for trading platforms in the Internet, as illustrated in a case study.

To our knowledge, EOM is the first approach to the specification, prediction, analysis and influencing of social interaction that aims at tackling the level of expectations explicitly and systematically, and allow for representing the beliefs *and* the intentions of agents in terms of empirical and desired predictions.

Keywords: Open Environments, Computational Autonomy and Trust, Agent Communication, Policy and Protocol Specification, Agent-oriented Software Engineering, Social AI

1 Introduction

A key focus of contemporary agent-oriented research and engineering is on *open* multiagent systems [15] composed of interacting truly autonomous agents [17, 3, 18, 21, 33]. This poses new challenges, as entities in open systems are usually more or less mentally opaque (e.g., possibly insincere), and can come and go at their will. Thus, interactions among such black- or gray-box entities usually imply heavy contingencies in behavior - in the most general case, neither a peer agent nor the system designer can be guaranteed to know what goes on inside a truly autonomous agent, with what observable result. These contingencies can cause improved performance at run-time (e.g., by increased system adaptivity and flexibility, and the emergence of unforeseen problem solutions), but can also be a source of potential unpredictability and undesirable behavior at the social level.

Practically, this means that even for the most powerful participants and controlling instances in open systems, it is neither realistic to assume that full control over autonomous entities being parts of the system (temporarily or permanently) can be guaranteed under all circumstances, nor that the other agents can ever be fully predicted. Even if it would be possible to control a MAS

*To appear in EAAI Vol. 18. Preliminary version.

fully, this would obviously mean loosing desirable system abilities which are based on autonomous decentralized control, such as the abilities to self-organize, to self-manage and to self-structure.

Taking autonomy seriously means to accept that any "strictly *normative*" (in the sense of inflexible action-prescribing) social-level exertion of control must be abandoned – instead, desired or persistent interaction patterns can only be modeled as revisable, basically uncertain descriptions of possible or desired behavior which might or might not occur in actual operation. Agents can thus only use models of interaction as *expected* courses of social action that are always hypothetical unless when actually enacted by them, their co-actors, and designer-controlled entities (e.g., case tools). A combination of *normative* and *deliberative* motives in agents' actions (the former resulting from previous system behavior, the latter from agents' autonomy) [7] makes certainty about future interactions impossible.

Starting from these observations, this article identifies a novel level of modeling agent interactions and thus of analyzing, designing and influencing agent behavior and multiagent systems: the *expectation level*. *Expectation-Oriented Modeling* (EOM) is introduced as a conceptual and formal framework for the modeling and influencing social interactions using agent *expectations* [20, 23] as the primary modeling abstraction. The intention behind EOM is to equip pro-active, autonomous entities in a MAS (ordinary agents as well as the system designer) with the means to represent the social (i.e., communication) level of the MAS (since, in the most general sense, expectations can be seen as the most fundamental representation of communication structures), and to take action on this level towards their goals in a "minimally-invasive" but effective manner.

To this end, we aim at making expectation-level knowledge about the anticipated, dynamic interactional behavior of agents *explicit* and *available* to agents as well as to the system analyst and designer. This offers the possibility for modeling and influencing interaction structures and autonomous entities which can not be controlled completely and which show a high degree of behavioral dynamics, and allows to retain a high degree of autonomy by using expectations as valuable knowledge for reducing contingency about each other's behavior and (public) goals.
Our approach comprises i) the introduction of expectations for the passive and active modeling of agents and social interaction (in strong demarcation from the usual modeling of agents using assumed mental attitudes of the modeled agents), ii) more specifically, the definition of expectation as a dynamic mental attitude that is *both* obtained empirically from observation *and* subject to deliberate manipulation (in order to represent and communicate desirable events), and iii) the presentation of principles and techniques for analyzing and setting up expectations in an evolutionary process of modeling and influencing agent interaction.
Doing so, we cover a wide range of aims and types of expectation-holders: from the optimization of the social behavior of "ordinary", self-interested artificial agents that make use of EOM internally for their cognition and planing, up to the design of a whole MAS or important constituents like behavior policies and protocols from the standpoint of a human designer.

Concretely, we present[1]

- a formal framework for *adaptive*, *empirical*, *normative* and *adaptive-normative* expectations within dynamic *social contexts*, and specifically the annotation of expected events with their respective *degree of expectedness*, degree of *normativeness*, and *deviancy* (divergence of expected and actual events). This allows to represent both empirically obtained probabilities (agent belief in terms of predictions) and to specify *desired probabilities* in an integrated, adjustable manner.

- formal representation languages for context-sensitive and correlated expectations in form of *Expectation Networks* (ENs).

  ENs allow to represent interaction patterns (e.g., protocols and policies) with gradual *flexibility* and *adaptivity*, and to measure the actual *adherence* of autonomous entities to behavioral specifications (predicted as well as desired) at runtime.

[1] For lack of space, some details can be found in [23] only.

Another particular feature of ENs is that they are accompanied with algorithms for their learning and incremental revision from ongoing experiences the MA makes, apart from the possibility to set them up manually,

- means for the learning and revision of expectations from selectively-overheard agent communications, and for the enactment and communication of expectations facing other agents (in form of so-called Social *Mirrors* [20, 26] and *Expectation Engines*).

At this, expectation-oriented modeling is performed from the viewpoint of modelers, so-called *Modeling Agents* (MAs). A MA observes agents and interactions, and maintains and revises expectations obtained from these observations and previous beliefs, intentions and desires, in order to effectively model and influence his social environment. An MA can be the MAS developer or an artificial agent that acts on behalf of the designer, but also a self-interested "ordinary" agent situated in a social environment.

By representing even the designer of an agent-based application as an agent conceptually, we constitute a novel paradigm in agent-oriented software engineering in so far as this suggests that the designer of open MAS should not and can not be granted the omniscient, almighty position as with ordinary software. Rather, we see her in the role of a *primus inter pares* among other agents, that, although equipped with more power than "real" agents, should aim for her goals socially (i.e., communicatively) in interaction with the other agents as far as possible. In addition, the openness of open MAS suggests that the development of such systems can only be done in an *evolutionary* manner, with the need to monitor the system and to improve its model even after deployment during runtime. A way to put the conceptualization of system designers as agents into practice in a semi-automatic manner is to assign the designer an intelligent, agent-like case tool, as proposed in [5] and in section 3.1.

As far as we know, EOM is the first approach to the specification, prediction, analysis and influencing of social interaction that aims at tackling the level of expectations explicitly and systematically. EOM adopts the concept of agent expectation from [20, 23] and is also strongly influenced by the *EXPAND* methodology (*Expectation-Oriented Analysis and Design*) [5] and the concept of *Mirror-Holons*[26][2][3]. EOM also possesses a strong sociological background; more specifically, its underlying view of sociality is quite close to Luhmann's *Social Systems Theory* [22], as it has been adapted to artificial agency.

The remainder of this article is structured as follows. The next section presents the generic conceptualization of expectations and so-called *Expectation Networks* as representation means for interaction structures. Section 3 describes EOM, and shows how a feasible and adequate incremental process can be derived that exploits the importance of the expectation level. This is followed by an exemplification of the usefulness of our approach in a case study based on a "car-trading platform" application scenario in Section 4. Finally, Section 5 provides more general considerations on the challenge of the modeling of autonomous systems and explicates relationships to other methods and approaches.

2 Expectations

An expectation can both express how much a future event *will* happen and/or *should* happen. The difference of both is represented by the degree of adaptivity (or inversely: normativity) of the expectation: The expectation of a desired or intended repeatable event will change less in case of

[2] Mirror-Holons are higher-level agents which represent all their beliefs and goals in form of expectations, and can each execute entire social programs emergent from communication processes.

[3] In contrast, EOM is not directly related to the approach by *Tran* [32], which is based on a different meaning of the term "expectation", and deals primarily with the perception layer of agents. Both approaches might be complementary to each other, but further research would be necessary to make out the exact relationship.

a disappointment of the expectation.
Expectations are related to the occurrence of anticipated behavioral or other events ("event" in a broad sense, e.g., "agent x utters message m to agent y", "agent y performs action a" or "the alarm bell is ringing"). A major consequence of the autonomous behavior of agents is that a certain agent appears to agent and non-agent observers more or less as a *black box* which cannot fully be predicted and controlled. This obscurity and uncontrollability is particularly salient in open multi-agent systems (open MASs). Because only the actions of an autonomous agent in its environment are known to an observer, while its mental state remains obscure, *beliefs* and *demands* directed to the respective other agent can basically be stylized *only* as mutable behavioral *expectations* which are *fulfilled* or *disappointed* in future events. In the case of disappointment, an expectation can either be revised in order to consider the new perception accurately (so-called *fully-adaptive* expectations), or the expecter decides to keep this expectation even contra-factually (so-called *normative* expectations), or to revise (resp. maintain) it only to a certain degree (*adaptive-normative* expectations). In the two latter cases, the expectation holder likely also decides to take action in order to make further disappointments of this expectation less probable (by, e.g., sanctioning unexpected - so-called *deviant* - behavior). And in any case, the expectation can be strengthened/weakened if an expected repeatable event turns out to be useful/useless afterwards.
Thus, expectations can not only express how the respective other agent *will* likely behave, but also how he *should* behave from the viewpoint of the expectation-holder. In addition, expectations can address the behavior of the expecter *himself* also, which can be useful for the expecter in order to model his self-commitments, and to communicate them to other agents in form of uttered expectations. As we will see in 2.4.6, expectation can also be used as an event *semantics*, expressing the meaning of observed or expected events (especially of agent communications) in terms of other observed or expected events (*empirical semantics*).

Expectations are called *empirical* (or *emergent*, when alluding to their newness from the expectation-holders viewpoint) if they are formed empirically from observations of events. In contrast, an MA can form expectations not only from her previous or empirical knowledge about her social environment, but also from her individual *intentions* and *desires*, resulting in so-called *normative* or *adaptive-normative* expectations[4]. In case the MA is or represents the MAS designer, these kinds of expectations can represent for example social norms, obligatory policies, protocols and agent communication language semantics, and organizational structures, or foster the maximization of social welfare and system coherence.
The dichotomy of adaptivity vs. normativity reflects the ambiguous meaning of the term "expectation" in natural language, which comprises both the anticipation of *probable* as well as of *desired* or *planned* behavior, with an adjustable transition of both stylizations. This makes expectations especially appropriate to the modeling of autonomous systems, where an adjustment of the MA's goals and constraints on the one hand, and the deliberate allowance, unavoidability or unpredictability of autonomous behavior on the other has to be found.
Since empirical expectations are usually adaptive also, and adaptive expectation become empirical during the course of observation, the attributes adaptive and empirical are more or less exchangeable in practice.

For the purpose of this paper, we use the terms agent "goals" and "desires" colloquially, and found the formal approach in the agent's "beliefs" and "intentions" only, the later thus indirectly also comprising goals and desires in a quite broad, flexible understanding of the term intention. At this, "intentions" are used to model states or events an agent commits himself to reach, *including* such states/events he cannot bring about himself directly. Therefor, the commitments of other agents show up indirectly as self-commitments of the MA, i.e., if the (possibly insincere) other agent is somehow committed to perform some action in favor of the MA, the MA might be self-committed to actually bring about this action indirectly by influencing the other agent. "Agent

[4]Since an expectation might be hold purely subjectively and hidden, and even normative expectations have no legislative power *per se*, we do not identify normative expectations with *social norms* [22], except from the case the expecter represents the MAS or is some entity with normative power. But adaptive-normative and normative expectations are a way to represent norms, of course.

a_1 intends that agent a_2 performs action x " is thus to read as "a_1 intends to get agent a_2 to realize a possible state in which *done*(agent a_2 performs action x) is true. We feel that refraining from a formal use of "desire" and "goal" is reasonable in view of this usage of "intention", since at least persistent and consistent goals are covered this way [8]. Note that having an intention in this sense does not imply knowing how to act concretely in order to make the intended state true (but to assume that there is *some* way).

All in all, from the viewpoint of the MA, expectations are formed *retrospectively* from utterances of observed agents in the social context of the MA, other observed events (e.g., "physical" agent actions like "Closing the window") and previous knowledge, goals and intentions, and held in order to *anticipatorily*

- represent her environment in terms of predictions (fully-adaptive and adaptive-normative expectations),
- represent intentions and goals (adaptive-normative and normative expectations) in terms of desired predictions,
- communicate desires and assertions directed to other agents, in order to influence their behavior (communicated expectations),
- filter out undesired (untypical, temporary...) effects, or conversely emphasize desired effects (adaptive-normative and normative expectations) (cf. 2.4.2).

A major feature of Expectation-oriented Modeling is thus that in form of expectations a relatively large spectrum of attitudes (mental and communicative) can be directly related to social events, using a single notion. This allows for the MA's cognition, belief acquisition and revision, and planing directly on the level of social interaction, in contrast especially to the reasoning about the hidden mental states of the modeled agents, suggesting that it is justified, and even inevitable, to integrate expectations as a modeling abstraction into the reasoning, analysis and design processes of agents, multiagent systems and autonomous software systems in general. This is not to say that EOM should replace common models like BDI (Belief-Desire-Intention). Rather, EOM should be seen as an additional means for the modeling and influencing of social interaction.

It is very important in this regard to see that we intend expectations obtained from observation to be also the primary means for modeling *single* agents (i.e., even aside from its embeddedness in social relationships, which are our main concern) from the MA's point of view. What can be expected from a black-box agent is not just additional knowledge, but in a way, from an observers perspective, the other agent *is* at a point of time what can be expected from him in the respective social situation, probably enriched with previous knowledge and presumptions, like that this agent is rational) (please refer to [23] for details). Therefor, we capture even agents in their entirety as expectation structures, including what other agents want the MA to expect and the limitations of such expected expectations. Please note also that focussing (but not restricting) ourselves on action expectations does not mean to neglect propositional information, since on the expectation level, such knowledge can be captured indirectly via expected assertive communication acts in the form of, e.g., "agent x asserts that p holds".

2.1 Sociality, Communication and Expectation Structures

Because we are focusing on systems with multiple inter-operating agents, we are primarily interested in expectations which constitute *sociality*: if it comes to an encounter of two or more agents, the described situation of mutual indeterminism is called *double contingency* [22]. To overcome this situation, that is, to determine the respective other agent and to achieve coordination (including the capability of conflictive behavior), the agents need to *communicate*. A single communication is the whole of a message act as a certain way of telling (e.g., via speech or gesture), plus a communicated information, plus the understanding of the communication attempt. Communication

is indirectly observable as a course of interrelated symbolic agent actions (i.e., messages in a agent communication language, or demonstrative behavior). 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 they form the *social system* in which the communicating entities are embedded [22]. EOM adopts this view, and assigns communication a key role in systems composed of interacting software agents.
Inter alia, one important practical consequence from this viewpoint is that in contrast to most other modeling and design methods for multiagent systems and organizations, in the center of EOM are *interaction processes* rather than fully-exposed agents, roles and groups. In fact, an agent role and even an artificial agent participating in social interaction is, from the MA's viewpoint, no more (and no less) than the sum of the behavioral expectations triggered by its observed previous behavior. Our modeling of agent roles in 2.4.5 reflects this process-oriented paradigm.

Another important point is that communication and thus the structures of MASs as modeled using EOM need not to be collaborative. In fact, even from conflicts stable and useful structures can emerge.

As action expectations are related to communications and thus to sociality, *social structures* (including, e.g., organizational structures) can be modeled as *expectation structures* [20, 24, 25]. Basically, expectation structures are interrelated expectations regarding a specific set of events (e.g., the behavior or a certain agent). Expectation structures can be tailored to local agent environments and topical communication domains.

We distinguish four types of expectation structures: (i) *social agents* as sets of all current behavioral expectations regarding single agents (i.e., a social agent abstracts from the actual agent with its opaque mental properties, and rather represent the interaction-related, externally-ascribed "public intentional stances" of actual agents [23][5]); (ii) *roles* as placeholders that are associated with certain kinds of expected behavior and that can be instantiated by different agents; (iii) *social programs* as flexible interaction schemes for multiple interacting social agents and/or roles; and (iv) *social values* as ratings of expected generalized behavior (e.g., "Conflictive behavior is always bad"). The focus of EOM is on *social programs* [22] with social agents and roles, since these are particularly suited for describing processes that occur between agents. Focusing on social values, in contrast, would suggest a rule-based approach.

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 adopt expectations for the use with future communications. The process of continuous expectation structure adaption by means of agent interaction (or communication, especially) *and* incremental, deliberative modifications of expectation structures by the MA is called *expectation structure evolution*. As described in section 3, this kind of evolution plays a key role in EOM.

2.2 Making Expectations Expected

"Expectations of expectations" [22] are necessary if expectations are formed in order to expect what others expect. We do not treat the MA's expectations of expectations explicitly (only implicit as they show up in the expected behavior of the other agents), but we have to deal with the fact that some of the expectations held by the MA need to be *expected* themselves by the other agents to be able to have any influence on the system.

The establishment of such *"expectations of expectation"* can be achieved through the communication of the MA's expectations to the agents and/or through the publishing of the expectations via an appropriate agent-external instance within the multiagent system. Once achieved, agents

[5] The public identity the modeled agent *would like* to present via communication is an important constituent of the social agent, but of course usually not identical with it.

can "expect" what "is expected". As described in section 3, EOM technically realizes this through so-called "Mirrors" and "Expectation Engines", which are also responsible for the acquisition of empirical expectations from observed MAS communications.
Expectations are communicated to the agents by the MA mainly for the following reasons:

- To inform the agents about actual social structures and processes they would otherwise not be aware of (e.g., because they evolved outside their local interaction environment, but are nevertheless relevant). This is especially important if the MA holds a higher position than these agents, and overhears a large part of the MAS communication (e.g., being the system designer, or a middle agent, or a manager agent in an organizational MAS).
- To inform the agents about the MA's goals and intentions, and possibly about the expectable consequences in case of acting against them or refusing collaboration.
- Even to deliberatively pretend actually disbelieved knowledge about social structures in order to influence agent behavior.

2.3 The Scope of Expectation-oriented Modeling

In this work, we define and use expectations for the purpose of modeling and influencing social states of a MAS (i.e., the interactional behavior of other agents), from the standpoint of MAs.
This leads to the following EOM tasks from the perspective of an MA, which will be described in detail later.

a. Modeling planned or desired events/event courses The MA encodes all or some of these as (adaptive-)normative expectations, denoting desired or intended event courses. Typically, such expectations are directed to the behavior of other agents in order to influence them, but they can also be in regard to the own behavior of the MA, or any other events.

b. Modeling empirical events This is done using fully-adaptive and also adaptive-normative expectations, as a part of the belief of the agent.
This task and the previous task are usually tightly interwoven, since the gradual blend of empirical and intentional expectations, and the run-time determination and minimization of the difference of both expectations is a special feature of EOM.

c. Overhearing and monitoring of MAS communications The MA observes agent interactions and categorizes them as desired, undesired and unassessed events.

d. Adaption of expectation structures, if necessary. Fully-adaptive expectation structures are adapted if they have been disappointed, and (adaptive-)normative expectation structures might need to be modified if they turned out to be not realizable, or not useful in order to reach the MA's desires.

e. Optionally, taking action in order to influence the MAS, by:

- Communication of expectations to other agents At this, the MA communicates selected expectations to other agents, making them "expectations of expectations", for the other agents, for the reasons listed in 2.2. These information do not have to be intentionally accurate or correct, nevertheless, but can be *ostensible* (cf. Figure 4). In [23] more information can be found about the enactment of expectations.

- Positive/negative sanctioning of deviant behavior, argumentation, negotiation Additional tasks which cannot be separated from the communication of expectations in general.

2.4 Key Aspects of Expectations

2.4.1 Strength, Normativity, Deviancy

Expectations can be weighted in two complementary ways, namely, w.r.t. their *strength* and w.r.t. their *normativity* (or inversely, their *adaptability*). The strength of an expectation indicates its "degree of expectedness" (also called *expectability*): the weaker (stronger) the expectation is, the less likely is or should be its expected fulfilment (violation). Against that, the normativity of an expectation (*both* weak *and* strong expectations) indicates its intentional "degree of changeability": the more normative (adaptive) an expectation is, the smaller (greater) is the change in its strength when being contradicted by unpredicted actual actions. With that, the strength of a lowly normative expectation tends to change faster, whereas the strength of a highly normative expectation is maintained in the longer term even if it is obviously inconsistent with reality (i.e., with the observed agents' actual activities), whereat the term "adaptive-normative expectation" denotes an expectation with normativity greater zero and lower one, and "fully-adaptive" ("(fully-)normative") means normativity zero (one). Fully-normative expectations ignore the actual occurrences of their modeled events completely, as long as they are not adapted "manually" by the MA. The idea of expectation weighting based on strengths and normativity is adopted by EOM, and it is also assumed that there is a continuous transition from weak to strong strength and from low to high normativity. The difference between the probability and the expectability (normativity-biased probability) of a certain event is called *deviancy* (cf. below). So, if EOM is used to model social norms, these can be both gradual and, to some degree, auto-adaptive - in contrast to, e.g., binary modalities like obligation and permissibility as in deontic logic.

Here are some examples of quite extreme combinations of expectation strength and normativity, mostly related to deontic modalities: *rules that govern criminal law* (strong/non-adaptable: even hundreds of actual murders will not alter the respective laws, and most people think of murder as a rather exceptional event); *habits* (strong/adaptable: before the times of fast food, people took full service in restaurants for granted, but as fast food became popular, they were willing to abandon this expectation); *public parking regulations* (strong/hardly adaptable: almost everyone violates them even if they are, in principle, rigid); and *shop clerk friendliness* (weak/adaptable: most people expect bad service but are willing to change their view once encountering friendly staff).

Thus, the term "expectation" is inherently ambiguous, as it deliberatively combines subjective, demanding expectations (reflecting the goals and intentions of the expecting agent) and the empirical likeliness of events (desired or not). In this regard it is worth to state that even the strengths of fully-adaptive expectations are not necessarily probabilities (from a frequentist point of view), because expectations are maintained ("expected") as a part of the belief a subjective observer has, and do not necessarily take into account enough "real world" facts to determine expectation strengths objectively when he sets up his expectations. So, not only (adaptive-)normative, but also fully-adaptable expectations could theoretically be used to represent individual, contra-factual preferences ("desired probabilities", so to say) instead of likelihoods. But such contra-factual yet non-normative expectations converge immediately to probabilities, since they are "willing to learn", so to say.

2.4.2 The Semantics of Expectation

Computational expectations have two dimensions: What the MA expects at a certain time, and how she will adapt this expectation in the course of time, in case the expected event is repeatable. The latter is treated in 2.4.3. So, what does "to expect an event" at the current time point mean exactly? So far, we've characterized the meaning of "expectation" intuitively as a graded blend of actual event probability and "desired probability". (Adaptive-)normative expectations are maintained even contra-factual for some time (i.e., if they *deviate* from knowledge). In this case, the only way for a rational and "non-ignorant" agent to reach the intended belief is to change the reality, i.e., to act in order to minimize the deviancy. An other interpretation would be to

understand a normativity as the degree of distrust in a probability, thus a kind of higher-order probability.
Starting from these observations, we define the semantics of an expectation held by the MA *agent* as his intention to make (or keep) his *gradual* belief regarding the occurrence of a certain event identical with the expectability of this expectation. This intention is weaker than to intent a certain probability of the event, but as we will see later, in the most common case we actually get by with defining normative and adaptive-normative expectations as the intention to make the reality (in form of a probability corresponding to the frequency of some event) conforming to the expected state (in form of an expectability).
At this, "intending a probability" can be understood as either aiming at bringing about a certain frequency of a repeatable event, or as the will to provide occurrence conditions for the event that make it probable to a certain degree.
Formally, an agent's expectation (denoted as $Expect$) is a mental attitude, represented as a logic modality, and defined as follows:

Definition 1.

$$Expect(agent, event|context, \psi) = e :\Leftrightarrow \begin{cases} Bel(agent, event|context) = e \\ \quad \vee Int(agent, Bel(agent, event|context) = e)) \\ \text{if } \psi > 0 \\ Bel(agent, event|context) = e \text{ otherwise} \end{cases}$$

Hereby, e is the expectability, $\psi \in [0; 1]$ is the normativity of the expectation,
$Bel(agent, event|context) = b$ denotes that *agent* believes that *event* occurs with probability b in *context* [6][7], and $Int(agent, p)$ denotes that agent intends p to become true (if *agent* is not capable to bring about the desired fact or action directly by herself, this shall include the intention to make other agents bring about p etc., i.e., to use them like a tool)
We write $Expect(agent, event|context)$ as an abbreviation of $Expect(agent, event|context, 0)$, and $Expect_t$ for $Expect$, when the time point t at which the expectation is held matters and can not be derived from the context (for ψ, Int and Bel analogously). Note that t is not the time point at which the event (should) occur(-s). If we would like to express that some event will or should happen at a certain time, we would have to encode this time within *context*.

The exact normativity (except from distinguishing if it is above zero or not) is not used in the definition above, because the normativity prescribes how an expectability auto-evolves *in the course of time* with new information, if the expectability it is not set "manually". If the normativity is zero, the expectation is set equal to the belief of the MA immediately. Otherwise, the expectability adopts gradually to the belief when both differ, with a "learning rate" of the expectation inverse to the normativity. Cf. 2.4.3 for details.

Our definition of expectation is build straightforwardly upon probabilistic versions of the KD45 and Belief-Intention axioms usually used for multi-modal logics of mental attitudes (e.g. [14]), and is related to Sadek's *want* attitude [31].

Given the agent's belief (e.g., obtainable from an expectation via the so-called *deviancy*, cf. below), the following proposition obviously holds, with $e = Expect(agent, event|context, \psi)$:

Proposition 1.

[6] We can use this syntax also to denote *expected expectations*: $Expect(agent_1, Expect(agent_2, ...)...)$.
[7] *context* here has, in general, to be distinguished from the empirical "context" the MA has been used to *obtain* the expectability, although *context could* have been a course of perceived events leading to conclude $Bel(agent, event|context) = e$. It is in general also not the context in which the agent *holds* the expectation.

$Int(agent, Bel(event|context) = e)$ if $(\psi > 0 \wedge Bel(agent, event|context) \neq e)$

$Bel(agent, event|context) = e$ otherwise

To the common axioms, we add the following bridge axiom (adopted from $Rel_{IntBel2}$ in [14]):

Axiom 1.

$Int(agent, Bel(agent, event|context) = e)$
$\wedge Bel(agent, event|context) \neq e \Rightarrow Int(agent, occurs(event|context, e))$

Axiom 1 denotes that disbelief in the occurrence of an event with probability e while intending to belief the event occurs with this probability forces the agent to intend the event to occur with probability e (denoted as $Int(agent, occurs(event|context, e))$). This also expresses that in case the agent has no particular belief regarding the occurrence of this event, she can bring about her introspective intention to belief in the event even without intending the event itself (e.g., by exploring new perceptions, or by improving her reasoning process).

If we would either drop the usual $Bel(p) \rightarrow \neg Int(p)$ axiom in Belief-Intention logics, or introduce alternatively *maintenance intentions* [4] (denoted as Int^M), definition 1 would change to

Definition 1-M.

$$Expect(agent, event|context, \psi) = e :\Leftrightarrow \begin{cases} Int^M(agent, Bel(event|context) = e)) \text{ if } \psi > 0 \\ Bel(agent, event|context) = e \text{ otherwise} \end{cases}$$

The agent can achieve the intention to change his belief in several ways, which can also be pursuit concurrently.

i. Change the world This is considered to be the usual way to enforce adaptive-normative and normative expectations, either by execution of the expected events directed to the MA himself, or by bringing about the intended events indirectly (e.g., by asking other agents).

ii. Explore The agent can try to obtain new perceptions in order to change his belief by exploration. Here, the (adaptive-)normative expectation serves as a kind of hypotheses, and the agents strives after new evidence in order to support or refute it.

iii. Wait This is actually not covered by the original intention at time t, but is a way to automatically decrease the "strength" of the intention (i.e., the degree and duration of the self-commitment) in consecutive time steps instead: If the normativity is below 1, in the longer term the expectation *learns* (i.e., adapts to the current probability), provided the probabilities of a certain event remain stable enough to be learnable (cf. 2.4.3). Practically, this happens if the expectation holder failed to decrease the deviancy actively (due to insufficient social power, for example). The adaptation of the expectability to the probability in this case can nevertheless be desired, and it can even be a prerequisite for the enforcement of less flexible and thus likely more important expectations.

iv. Ignore the deviation Here, the agents simply believes in the expectation, possibly ignoring reality thereby:
$Bel(agent, event|context) = Expect(agent, event|context, \psi)$ holds *in any case* then.
Such deliberative ignorance appears to be irrational for intelligent agents, but is a common attitude of human agents and obviously somewhat functional for them. In any case, the identification of certain expectations with beliefs regardless of deviance might be reasonable for artificial agents in case the event belief is obtained from an unreliable source.
A less debatable use for such deliberative ignorance is to set the normativity greater zero

in order to filter out ("flatten") temporal and insignificant fluctuations of probabilities. In 2.4.3, a concrete way to adapt expectabilities is shown which flattens a graph depicting the changing probabilities of some event.

In all cases except from iv., we assume that the expectability is equal to the probability (in case the normativity is zero).

Note that even for the cases i.-iii. so far no assumptions have been made on how e has been obtained - the MA is basically free to hold any expectabilities she likes / is interested in from her subjective and possibly irrational viewpoint.

Definition 2.

The *deviancy* Δ of an event regarding a certain expectation (or vice versa of an expectation regarding an event) is defined with

$$\Delta(event, context) = Expect(agent, event|context, \psi) - Bel(agent, event|context).$$

The deviancy can intuitively be seen as an indicator of the effort that would be required to make a normatively expected event happen, and as a measure for the compliance of the event-generating agent with the expectation, whereas the normativity is intuitively a kind of "stamina" of the intention (the strength of a self-commitment. Please remember in this regard, that we allow intentions also to be denoted as desired behavior of other agents).

There is also a conjunction with the *utilities* of events: If the normativity is larger zero, the utility for the MA to reach the specified probability is certainly larger zero also. The expectability *might* correspond to the utility of the event in this case (but this is to state a heuristic only, suggesting further research).

Proposition 2.

Except from the case iv. above (belief despite ignorance of event occurrences)

$$Int(agent, \forall t_i, t \leq t_i \leq t + h : \Delta_{t+i}(event, context) = 0)$$

holds at time step t. At this, h is a possibly infinite intention horizon which determines how long the expectation is maintained.

Finally, we want to further simply the semantics in case the probability of an intended event is irrelevant:

Proposition 3.

$$(Expect(agent, event|context, \psi) = e \wedge Bel(agent, event|context) < e) \rightarrow Int(agent, event)$$

To sum up, our notion of expectation is (to our knowledge) the first computational means for a coverage of both agent belief and intention using a *single* attitude, with the possibility of a gradual adjustment of the emphasis of either aspect. This corresponds to the double-faced common-sense meaning of expectation in natural language, and to the meaning of this term introduced in [20]. Apart from having single "points of attack", each allowing to express how much a believed event is intended or an intended event is believed, and how strong the commitments directed to intended events (i.e., to reduce the deviances of (adaptive-)normative expectations) should be.

The described semantics of expectation of course only applies in case the expectations are held as mental attitudes by the MA. In case the expectation is used to be *communicated* to other agents (to make it an "expectation of expectation") instead, its semantics changes, cf. 3.2 and [23].

2.4.3 Unattended Adaption of Expectations

For lack of space, the empirical derivation of fully-adaptive expectation (their expectabilities, resp.) and the probabilistic part of adaptive-normative expectations is omitted in this paper, please refer to [25]. We describe the adaption of adaptive-normative expectations here, though.
After the expectabilities and normativities of adaptive-normative expectations have been obtained from goals and intentions, they are exposed to reality, so to say. The following shows how such an expectation can be adapted automatically, depending from its normativity (degree of commitment). The following definition covers expectations with normativity zero and one also.

To this end, it is assumed that for an event $event|context$ corresponding to a certain EN node an initial expectation strength $\theta(event, context) = P_0(event|context)$ exists. Analogously to $Bel_t()$, $P_t(event|context)$ denotes a probability stated at time t, not the probability of an event happening at time t. Given a normativity ψ_t and a probability $P_t(event|context)$ obtained empirically at time step t, the expectation strength at this time step can be calculated recursively as follows. This way to calculate $Expect_t$ is not to be seen as canonical, other definitions for the adaption of adaptive-normative expectations might be reasonable too, depending from the concrete application also.

Definition 3.

$$Expect_t(agent, event|context, \psi_t) =$$

$$\begin{cases} \theta(event, context) \text{ if } t < 1 \\ Expect'_{t+1}(agent, event|context, \psi_t) \text{ otherwise} \end{cases}$$

with $Expect'_t(agent, event|context, \psi_t) =$

$$\begin{cases} Expect'_{t-1}(agent, event|context, \psi_t) \\ \quad -\Delta'_{t-1}(event, context)(1 - \psi_t) \\ \quad \text{if } t > 0 \\ \theta(event, context) \text{ otherwise} \end{cases}$$

$\Delta'(event, t)$ is calculated as
$Expect'_t(agent, event|context, \psi_t) - P_t(event|context)$[8].

This (non-mandatory) way to calculate $Expect_t$ reminds of the econometrics technique of *Exponential Smoothing* used for the smoothing and extrapolation of non-linear time series. It calculates a flattened version (with a flattening degree depending on the normativity) of the graph of $P_t(event|context)$, and lets $Expect_t(agent, event|context, \psi_t)$ converge to $P_t(event|context)$ at least if $P_t(event|context)$ remains constant with increasing t, and ψ_t remains constant also. The normativity (i.e., the expectation adaption rate) itself does not change.

If, e.g., $\psi_t = 1$, the expectation strength
$Expect_t(agent, event|context, \psi_t) = \theta(contex, event)$ will remain constant, whatever the empirical evidence is. In contrast, if $\psi_t(agent, contex, event) = 0$,
$Expect_t(agent, event|context, \psi_t) = P_t(event|context)$ applies at all time steps.

Example: Figure 1 shows the time and normativity dependent expectabilities of an event a, with $\psi_{0..20} = 0.95$ and $\theta(a, context) = 0.4$. Being a fictive event, the potential effect the announcement of these values to the event generator (a communication partner of the MA, for example) would have, is not considered. The *agent* parameter has been omitted.

[8] Calculating $Expect_t(...)$ using $Expect'_{t+1}(...)$ is done just in order to get rid of the delay of one time step in the adaption of $Expect_t(...)$ to $P_t(...)$ that would exist otherwise.

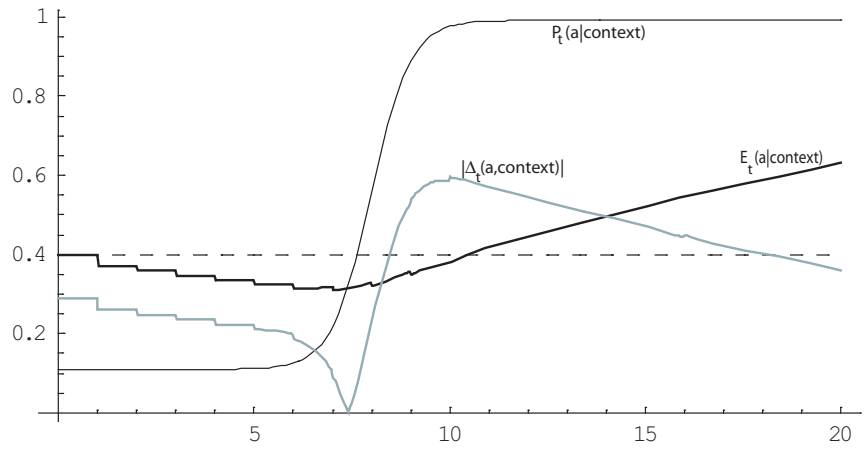

Figure 1: Unattended adaption of an expectability ($t \rightarrow$)

2.4.4 Computational Representation of Expectations and Interaction Processes

Settling on particular representation formalisms naturally affects the level of abstraction and with it the scope of expectation structures. Here, we focus on non-deterministic social programmes (with regular protocols and policies as special cases), for which a graphical notation – so-called *Expectation Networks* (ENs) [20, 25, 24] – is presented [9]. The main characteristics of ENs are that they represent expectations embedded within explicit *social contexts*, and model probabilistic event courses (i.e., beliefs regarding events), action intentions and normative protocols in a seamlessly integrated manner, and (in contrast to *Hidden Markov Models* (HMMs)) interrelate stochastic events instead of stochastic states. At this, ENs are intuitive (as we believe), and can be set into straightforward relation to several other formalisms, especially HMMs and *stochastic automata*, *Dynamic Logic*, *Dooley Graphs* [28] and *Interaction Frames* [30].

Expectation Networks can be given a formal semantics as described in 2.4.2, and also induce a so-called *Empirical Semantics* of events and event processes [25]. Informally, this empirical semantics assigns an event a meaning in terms of its likely *consequences*, as represented by EN branches. E.g., the empirical semantics of some message from some agent communication language would be the expected effect the utterance of this message has, cf. 2.4.6.

In regard to EOM, the use of ENs is not mandatory, although ENs are probably the most suitable representation formalism. In principle, other formalisms could be used also, as long as they are capable to make explicit the consecutive states of agent interaction (e.g. Interaction Frames [30]).

Figure 2 depicts a very simple EN (in a notation called *DG-EN* which allows for cycles in the graph). The nodes correspond to contextualized events (especially agent message acts and other agent actions for our purpose, but also "physical" events perceived in the agents' environment) that are uttered and addressed to/by agents, probably acting as instances of roles (r_i). Time stamps could be part of node labels, but for our purpose, we want nodes to represent repeatable events (i.e., we form expectations about events like "Agent x performs action y in the future" instead of singular events like "Agent x performs action y at time step 100"). Events are always contextualized, i.e., the same event can occur multiple times within a certain EN, probably with

[9] The significant differences of the EN data structure presented in this work (based on [20]) and older realizations of ENs (e.g. [24]) are the treatment of normativity, generalizations and variables. The notation in this work is downwards-compatible with [25].

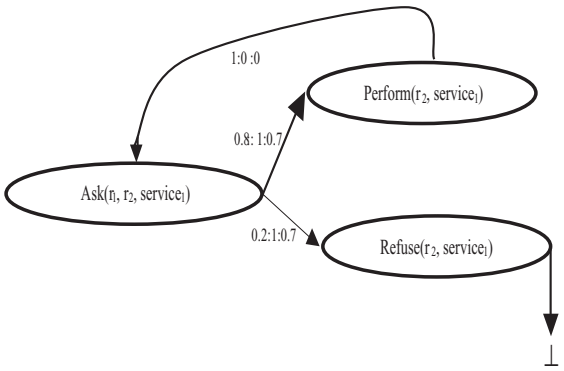

Figure 2: *A social program as DG-EN*

different expectation strengths depending on the respective context (in this regard, the relationship of Expectation Networks and HMMs and stochastic automata becomes obvious when paths in the network are associated with states). Technically, an (event) context is the EN path up to and including the parent of the node that is annotated with the event label. The directed edges represent the expectation that a certain event is followed by a certain subsequent event. Each edge is labeled by a triple *s:n:d* of real values, where s (ranging between 0 and 1) denotes its expectability (strength of the expectation), summing up to 1 for edges leading to sibling nodes. n (ranging between 0 and 1 also) denotes the normativity. The deviancy is denoted as d, ranging between -1 and 1 (informally, the difference of s and the strength this expectation would have in case of $n = 0$). n and/or d can be omitted if they are not of interest in a specific application context. Note that outgoing edges of a node always do have the same normativity, because the degree of expectability-change represented by the normativity is the same for all sibling nodes. $\perp$ denotes the expected end of the communication process.

In tree ENs, an expectability e at the incoming edge of a node v within an EN en_t maintained by an MA at time step t means that at this time step, the MA expects the event represented by v with expectability e assuming he already has or *would have observed* the course of all events along the path leading to v. This path up to the parent of v is called the *context* of the event. In case the expectation holder makes new observations, he probably needs to modify this expectability. The context thereby allows partially to foresee the effect optional observation would have: If, e.g., you expect today that the sun will shine tomorrow afternoon *if* it is foggy tomorrow morning, the EN determines the exact expectability you'll have tomorrow morning in regard to sunny weather in the afternoon (in case you keep this EN until then). On the other hand, the expectabilities within an EN are of course not fully determined by the EN itself: If you experience heavy rain today, you might want to decrease the EN's expectability for sunny weather tomorrow, even in case of fogginess. It this regard, it is important to see that the context might be only a part of the probabilistic condition for the *probability* $P(event_v|condition)$ an expectation represents (i.e., the expectability minus its deviance, cf. below). In fact, all relevant observations and influences experienced/known until time t have to be included additionally to form a proper probability condition[10].

We will introduce the following kinds of EN representations, in order to allow the tailoring of the representation formalism to different needs in different application contexts. All of them can be reduced to so-called *ground Tree-ENs* in order to define their semantics.

***Ground Tree-ENs*:** Graphical trees without variables. If we refer to an "EN" in this work, we

[10] It is in principle also possible to use ENs for modeling the *past* as a course of already observed or otherwise believed events, each with expectability 1.

denote a ground Tree-EN if not stated otherwise.

***(full) Tree-ENs*:** As before, but optionally with variables for nodes, sub-branches, and node and edge labels.

***KB-conditioned Tree-ENs*:** Tree-ENs where graph edges can be annotated with logical conditions from a knowledge base KB. Please refer to [24] for these kinds of EN.

***DG-ENs*:** Directed graphs, allowing for nodes with multiple incoming edges (denoting events with different contexts but the same empirical semantics) and cycles

***Textual ENs*:** Representing ENs using the formal language (*ENL*)

***Probabilistic ENs*:** ENs with normativity 0 for all edges.

***EN sets*:** A set of multiple ENs can be stated either extensionally (by enumeration), or intentionally by way of placing free variables within an EN, such that different instantiations result in different ENs, whereat the variables can be existentially quantified (e.g., to be used to query other ENs), or all-quantified. EN sets can be used to represent concurrent event sequences. They are not considered in this work for lack of space.

For the understandability of EOM, it is sufficient to know Tree-ENs as far as explained above. Thus, readers not interested in details about ENs and empirical semantics can safely jump to section 3 now.

Formally, a so-called *ground Tree-* Expectation Network is defined as follows:

Definition 4. An ground Tree-*Expectation Network* $en \in \mathcal{EN}(\mathcal{E})$ over a certain event term language $\mathcal{E}$ is a (possibly infinitely deep or broad) tree

$$(V, C, \mathcal{E}, \mathit{Expectability}, \mathit{Normativity}, \mathit{Deviancy}, \mathit{nodelabel}, \mathit{edgelabel})$$

where

- V, $|V| > 1$ is the set of nodes,
- $C \subseteq V \times V$ are the edges of EN.
- $\mathcal{E}$ is the *event term language* (a speech-act-oriented agent communication language, for example, to denote agent messages. But terms denoting non-symbolic events are allowed also, e.g. "physical" agent actions, or events like "the sun is shining").
- $\mathit{Expectability} : C \to [0;1]$ returns the expectation strength (*expectability*) for the following event, with $\sum \mathit{Expectability}((v_{parent}, v_i)) = 1$ for each set of sibling nodes v_i,
- $\mathit{Normativity} : C \to [0;1]$ returns the normativity of the corresponding expectability,
- $\mathit{Deviancy} : C \to [-1;1]$ yields the corresponding deviancy,
- $\mathit{nodelabel} : V \to \mathcal{E} \cup \{\bot, \rhd\}$ is the *label* function for nodes. The labels of sibling nodes must be mutually different (i.e., denoting different events).

  Optionally, $\bot$ denotes the expected end of a conversation, and $\rhd$ denotes "no action" (nothing happens). As we will see later, the latter is useful as a dummy root node representing the start of a conversation.
- $\mathit{edgelabel} : C \to \{"s : n : d" : s \in [0;1], n \in [0;1], d \in [-1;1]\}$ is the *label* function for edges, with $\mathit{edgelabel}(c) = \mathit{Expectability}(c) : \mathit{Normativity}(c) : \mathit{Deviancy}(c)$. Edge labels are usually omitted if the edges' expectability and normativity are both 1. Normativity and deviancy can also be omitted if these are not of interest.

$\mathcal{EN}(L)$ shall denote the set of all ENs over an event term languages L.

If the normativity for every edge within an EN is 0, we speak of a *probabilistic EN*, useful for denoting adaptive stochastic protocols.

Given an Expectation Network en_t maintained at time t, with

$en_t = (V, C, \mathcal{T}, Expectability, Normativity, Deviancy, nodelabel, edgelabel)$,

we set $Expectability(c) = Expect_t(agent, event|path, Normativity(c))$, and
$Deviancy(c) = \Delta_t(path, action)$ for each edge $c \in C$ leading to the node corresponding to the event *event* reached from the EN's root node following *path*. *agent* is the MA who holds the expectations/the EN. The set of contexts *Context* is provided as the set of paths within the EN, denoted unambiguously (since the labels of the children of each EN node have to be mutually different) as linear lists of consecutive event labels $(event_1 \sqcup ... \sqcup event_n)$.

Ground Tree-ENs can be extended to (full) Tree-ENs with variables by allowing placeholders for agents (in order to allow for agent *roles*), and for other message term constituents and even branches. Such variables can be used with all other kinds of ENs also, by allowing them in place of nodes, edge labels and within node labels. This extension is shown only for textual ENs below, since it is completely straightforward for other EN types.

Obviously, an EN can be represented in a compressed way without loss of expressibility as a directed graph by merging of identical branches. This also reflects that the empirical semantics of EN paths lays in their continuations expressed by subsequent EN-branches, and because of this, two paths with the same continuation have the same meaning (cf. 2.4.6). These so-called *DG-EN*s also allow graph cycles, as an abbreviation for infinitely deep tree branches. Since the enhancement of ground Tree-ENs to DG-ENs is trivial, and DG-ENs can be mapped to ground Tree-ENs fully preserving their semantics (by manifolding branches with more than one parent node), we omit a formal definition of DG-ENs here.

Since an EN might be quite cumbersome to draw betimes, we now introduce the following EN representation language ENL.

ENL	→	Branch
Branch	→	Event
	|	[Event Children]
Event	→	'$\mathcal{E}$pattern' | Variable | ?Variable | ?
Children	→	[Weight Branch] Children | ϵ
Weight	→	Expectability
	|	(Expectability, Normativity)
	|	(Expectability, Normativity, Deviancy)
Expectability	→	$\{e : e \in \mathbb{R}, 0 \leq e \leq 1\}$ | Variable | ?Variable | ?
Normativity	→	$\{n : n \in \mathbb{R}, 0 \leq n \leq 1\}$ | Variable
Deviancy	→	$\{d : d \in \mathbb{R}, -1 \leq d \leq 1\}$ | Variable
$\mathcal{E}$pattern	→	Performative(Agent, Content)
	|	PhysicalAction(Agent, ActionDescription)
	|	UnintendedEvent(...)
Agent	→	$agent_1$ | $agent_2$ | ... | Variable | ?Variable
Performative	→	*request* | *deny* | *accept* | *assert* | ... | Variable | ?Variable
ActionDescription	→	*turnLeft* | *closeDoor* | *leaveRoom* | ... | Variable | ?Variable
Content	→	LogicalStatement | Variable | ?Variable

...

This deliberatively incomplete syntax of $\mathcal{E}$pattern (corresponding to node labels) is just for example - basically, any message or action language can be used.

For simplicity, we will use certain non-terminals of the grammar in place of the sets of the respective produced words. Since every event (term) and every branch in a textual EN corresponds to a node (branch) in the equivalent ground Tree-EN, we refer to event (terms) within a textual EN sometimes as "nodes".

To demarcate variables (alphanumerical terminals starting with big letters or "?"), we usually use small letters for non-variable parts of $\mathcal{E}$pattern. Variables starting with "?" are called *ignorance variables*.
Words in ENL and words in $\mathcal{E}$pattern are called *ground* iff they do not contain variables. Other variables are either bounded using a substitution list (with each variable possibly having more than one instance simultaneously), or free in order to retrieve sets of ENs (as existentially quantified query variables, or as universally quantified variables). Free (non-ignorance) variables are not considered in this work for lack of space.

To reduce ENs with variables to a known type of EN we provide a mapping o ground Tree-ENs. Mapping a textual EN $ten \in$ ENL does not yield, as one might expect, a set of ENs in case *ten* contains variables. Rather, the capability of ENs to represent non-deterministic alternative actions is used to "inflate" *ten* to a single, possibly infinite ground EN (2.4.5 describes the opposite task of "deflation" by introducing new variables). This proceeding has the advantage that the semantics and handling of ENs can be defined in a clear cut way without the necessity to account for variables later (and thus to deal with instantiations, unification etc.). In addition, focusing on single ENs avoids handling multiple ENs possibly denoting mutually inconsistent probability distributions and conflicting normativities (every single EN is inherently consistent as it can, by definition, not represent mutually inconsistent beliefs or "norms"). Of course, a complete inflation is in general not meant to be performed actually, if only for the reason that the resulting ground EN might be infinitely large.

Definition 5. A *textual EN* $ten \in TEN(\mathcal{E})$ over the event term language $\mathcal{E}$ is defined as a structure $(enl, \vartheta, Inflate)$ where

- $enl \in ENL$ is a word from the language defined using the grammar above, with $\mathcal{E}$ being the subset of all ground terms in $\mathcal{E}$pattern.

  For sibling branches constraints analogous to those defined for graphical ENs apply (i.e., their expectabilities have to sum up to 1, and two or more siblings with the same ground event term are forbidden (but not those with same event terms as long as these transform to different ground terms by means of substituting the same variable with different values, which is allowed)). But for convenience, in case the sum of sibling expectabilities is below 1, we assume an implicitly given additional sibling "?" (denoting "unknown additional events expected here"), completing the expectability gap.

- $\vartheta : ENL \to \theta$ resulting in the *environment* of a branch,
  with θ being the set of all lists of the form
  $\langle\langle Variable/inst, Variable/inst, ...\rangle, \langle Variable/inst, Variable/inst, ...\rangle ...\rangle$. The environment of a branch is thus a list of variable substitution lists, to be applied from the position in *enl* on determined by a $path \in ENL$, *path* being a prefix in *enl*, until the next closing]-bracket that has no accompanying opening bracket counted from *path* on (i.e., substitutions extend over branches).
  A certain variable, if not a variable for edge weights, can occur in multiple sublists within

$\vartheta(path)$ at the same time for the same path (denoting non-deterministic instantiation), but not within the same sublist. E.g., in
$\langle\langle Variable_1/inst_{1_1}, Variable_2/inst_{2_1}, ...\rangle, \langle Variable_3/inst_{3_1}, Variable_2/inst_{2_2}, ...\rangle\rangle, Variable_2$
is bounded to $inst_{2_1}$ *and* to $inst_{2_2}$ simultaneously.
For convenience, we insert the ϑ directly at their proper positions, e.g.,
$\underline{[^{\langle\langle Role_1/\texttt{agent}_1, Role_2/\texttt{agent}_8\rangle\langle Role_1/\texttt{agent}_2\rangle\rangle}\ \texttt{'request}(Role_1, \texttt{service})\texttt{'}}$
$\underline{[0.3\ \texttt{'}deny(\texttt{agent}_3, \texttt{service})\texttt{'}][0.7\ [\texttt{'accept}(\texttt{agent}_2, \texttt{service})\texttt{'}}$
$[_{\langle\langle Role_1/\texttt{agent}_3, ProbabilityPay/0.8\rangle\rangle}\underline{(ProbabilityPay, 0, 0)\ \texttt{'pay}(Role_1, \texttt{service})\texttt{'}]]]]}$.
The extent of the respective substitutions is hinted by under-/overlines. Note that in this example, the last role substitution ($Role_1/\texttt{agent}_3$) is never applied, because at this position, $Role_1$ is already bound by either $agent_1$ or $agent_2$.

The precise semantics of ϑ is defined via *Inflate* (cf. below). Please observe that both this semantic and the syntax is different to those proposed earlier for similar looking variable substitutions in [24].

- *Inflate* : $TEN(\mathcal{E})$ yields a semantically equivalent textual EN. The required definition of *Inflate* is given below.
  If the result of *Inflate* is ground (i.e., all variables except from ignorance variables are bound by ϑ, and a repeated application of these substitutions results in ground values eventually), then resulting EN is equivalent to a ground Tree-EN.

Term variables

Informally, an 'event term pattern' containing a variable in place of Agent / Performative / Content... is eventually *inflated* to a set of sibling nodes, each for one of all possible instantiations of this variable. To this end an environment ϑ can provide more than one instance value of each variable, whereat the instances can be any (not necessarily ground) terms syntactically allowed in place of the variable. The children of each generated sibling are those of the un-inflated event term pattern, whereat the variable is bounded exclusively to the respective chosen instance within the whole branch rooting in this sibling. In case the variable is an ignorance variable, their implicit set of instances is the set of *all* syntactically appropriate substrings within the ground subset of $\mathcal{E}$pattern.

E.g., in a MAS with three agents, ['Ask($agent_1$, a)' [1 'Reply($?Agent_x$, b)']] is inflated to ['Ask($agent_1$, a)' [0.33 'Reply($agent_1$, b)'] [0.33 'Reply($agent_2$, b)']
[0.33 'Reply($agent_3$, b)']], the $\texttt{agent}_i$ (the addresses of the speech acts are omitted for simplicity).

In case of a finite set of resulting sibling nodes (e.g., using a propositional event term language), the expectability of each sibling resulting from such an inflation is the expectability for the un-inflated branch, divided by the number of generated siblings (i.e., denoting uniform distribution of the siblings). For infinite sets, explicit numerical expectabilities cannot be stated for this uniform distribution.

Expectability and normativity variables

Using "?"s in place of expectabilities denotes uniform distribution of the respective children (i.e., all siblings have the same expectability). Following a Bayesian viewpoint, uniform distribution stands for "Don't know", whereby possibly the respective parts of the EN become less entropic in the course of the revision of the EN by learning. Named variables can also be used in place of expectabilities, normativities and deviancies, but multiple values for the same variable are not allowed in this special case, because otherwise, the inflation would result in identical events with different expectabilities, violating the semantics of ground Tree-ENs. Ignorance variables for expectabilities are treated like "?", but will be bound to the resulting numerical value in the whole

subsequent branch.

Sub-EN variables

In order to support modularization, variables can also occur in place of a whole node (respectively "Event" for textual ENs). They are be replaced with "sub-ENs" (not just single nodes), more specifically, they are inflated to a set of sibling branches each, analogous to the inflation of variables in event term patterns (but note that the result is in general not the same as of the inflation of a variable in place of an event *term*, as described before). Such a variable can be bounded by ϑ, whereby the instances are ENs to be included in place of the variable (denoted in ϑ as words from ENL, not necessarily ground), or be an ignorance variable, standing for the (possibly infinite) set of *all* ENs over $\mathcal{E}$ ($TEN(\mathcal{E})$). So, "?" in place of a node can be interpreted as an "unknown course of events". An complete graphical or textual representation of the resulting EN is of course not feasible in the general case.
A "?" in place of a node / event stands for an unnamed ignorance variable.

DG-ENs are especially suitable for the "folded" graphical representation of ENs containing variables. If, e.g., the same (non-ignorance) sub-EN variable appears as a leaf node multiple times, the inflated leafs can be merged graphically, using multiple edges leading to the same node that resulted from the former leafs.

Precisely, variables are inflated as follows:

Definition 6. *Inflate* : $TEN(\mathcal{E})$

$Inflate = inflate^k(\texttt{[}\rhd\ \texttt{[1 enl]]}, \vartheta, \langle\rangle)$[11]

At this, we choose $k :\Leftrightarrow inflate^k(enl, \vartheta, \langle\rangle) = inflate^{k+1}(enl, \vartheta, \langle\rangle)$ (i.e. such that $inflate^k(enl, \vartheta, \langle\rangle)$ is a fixpoint of *inflate*), and define

$inflate : ENL \times \theta^* \times \theta \to ENL$

$inflate(event, \vartheta, \vartheta') = event,$

$inflate(\texttt{[}father\ child_1\ ...\ child_n\texttt{]}, \vartheta, \vartheta') =$
$\texttt{[}father$ merge$(child_1\ ...\ child_{n-1}\ child'_{n_1}\ ...\ child'_{n_m})\texttt{]}$

at which $child_n = \texttt{[}(expect_n, ...)\ \texttt{[}deflatedEvent_n\ grandchild_{n_1}...grandchild_{n_g}\texttt{]]}$, and the $child'_{n_i}$ being defined with

$$child'_{n_i} = \begin{cases} \texttt{[}(expect_{n_i}, ...)\ inflate(instanciated'_{n_i}, \vartheta, \vartheta''_{n_i} \sqcup \vartheta')\texttt{]} \\ \quad \text{with } instanciated'_{n_i} = instanciated_{n_i} + (grandchild_{n_1}\ ...\ grandchild_{n_g}) \\ \text{if } instanciated_{n_i} \in ENL \\ \\ \texttt{[}(expect_{n_i}, ...)\ inflate(\texttt{[}instanciated_{n_i}\ grandchild_{n_1}\ ...\ grandchild_{n_g}\texttt{]}, \vartheta, \\ \quad \vartheta''_{n_i} \sqcup \vartheta')\texttt{]}, expect_{n_i} = \frac{expect_n}{|\{rootEvent(instanciated'_{n_i}) : 1 \le i \le m\}|} \\ \text{otherwise} \end{cases}$$

[11] For convenience, we denote the resulting graphical EN as a ground textual EN.
Prepending the empty action $\rhd$ to *enl* here is required because otherwise it would not be possible to inflate the root node of *enl*.

using [12]

$$instanciated_{n_i} = (\vartheta' \sqcup \vartheta''_{n_i})(deflatedEvent_n), \vartheta''_{n_i} :\Leftrightarrow \{\vartheta''_{n_i} : 1 \leq i \leq m\} = \vartheta''$$

$$\vartheta'' = \vartheta(path_{child_n}) \sqcup \begin{cases} \langle\langle ?v_r/e\rangle : e \in \mathcal{E}, ?v_r \Subset deflatedEvent_n\rangle \\ \quad \text{if } deflatedEvent_n \in \mathcal{E}pattern \\ \langle\langle ?v_r/e\rangle : e \in TEN(\mathcal{E}\}), ?v_r \Subset deflatedEvent_n\rangle \\ \quad \text{otherwise} \end{cases}$$

At this, the functional application of a local environment ϑ' is defined as the application of the substitution lists within ϑ' in turn. E.g.,
$\langle\langle Role_x/\texttt{agent}_3, Obligation_y/\texttt{task}_7\rangle, \langle Role_x/\texttt{agent}_9, Obligation_y/\texttt{task}_7\rangle\rangle$(`'request` $(Role_x, Obligation_y)$`'`) results in the two siblings
`'request(`$\texttt{agent}_3, \texttt{task}_7$`)'`, `'request(`$\texttt{agent}_9, \texttt{task}_7$`)'`.
Since substitution lists range to the end of the whole branch from their position, and add to previous substitution lists, we could have abbreviated these substitutions as
$\langle\langle Obligation_y/\texttt{task}_7\rangle\rangle\langle\langle Role_x/\texttt{agent}_3\rangle, \langle Role_x/\texttt{agent}_9\rangle\rangle$.

$\sqcup$ denotes list (string) concatenation, $s_x \Subset s_y$ yields true iff s_x is a sub-list (a substring) of s_y.
$\vartheta_1 \sqcup \vartheta_2$ concatenates two substitution lists. If a substitution list resulting from such an operation is applied, and it contains two substitutions for the same variable, only the first substitution is used. Applying $(\vartheta' \sqcup \vartheta''_{n_i})(deflatedEvent_n)$ thus ensures that variables bound by previous calls of *inflate* ranging over the current branch cannot be rebounded by ϑ''_{n_i}.
$\vartheta(path_{child_n})$ yields the list of substitution lists next to the position of $child_n$ within enl.

$merge(child_1 \ ... \ child_n)$ obtains the argument, but with only one among those children branches that start with the same event, thereby keeping only the largest (in terms of string length) of these doublet children. In case the argument results from substituting a bounded variable in ϑ, such doublets could have been avoided manually. Against that, if a ignorance variable is EOMed, it inflates to *all* possible instances syntactically allowed, including those that are already present as siblings, making a merging necessary. If the instance variable inflates to the elements of $TEN(\mathcal{E})$, *merge* keeps for each subset of $TEN(\mathcal{E})$ with elements having the same root node only the largest one.

In addition, $merge(\texttt{[}(expect_1...)\ child_1\texttt{]}\ ...\ \texttt{[}(expect_n...)\ child_n\texttt{]})$ also replaces "?"s at the positions of expectabilities (denoting "expectability unknown"), obtaining $expect'_i$ for each "?":

$$expect'_i = \frac{1 - \sum \{expect_j : expect_j \neq \text{"?"}\}}{|\{expect_j : expect_j = \text{"?"}\}|}$$

The operator + in $instanciated_{n_i} + (grandchild_{n_1} ... grandchild_{n_m})$ "adds" the branches $grandchild_{j_1}$ to the textual EN $instanciated_{n_i}$ multiple times by adhering to every leaf of $instanciated_{n_i}$ the set of all grandchildren as children (avoiding doublet siblings using $merge()$ as described above).

Again, it is important to see that the syntactical transformations done in *Inflate* are not generally intended or possible to be actually performed globally for a whole EN.

2.4.5 Generalization and Role Emergence by EN Deflation

Likewise Expectation Networks can be inflated by variable instantiation, they can be *deflated* also by the merging of multiple branches resulting in single branches. Deflating an EN can be useful in order to *compress* ENs for better manageability, to calculate the *entropy* of an EN (which can be informally characterized as the reciprocal of the size of the smallest semantically equivalent

[12] We show the inflation of variables for whole event terms, whole events, and of "?" for expectabilities. Variables for parts of event term patterns, like for agent identifiers, for normativities and deviancies, and ignorance variables for expectabilities inflate analogously in a straightforward manner.

textual EN), and - most important - to derive interaction patterns and agent roles from a set of concrete communication processes. If a set of multiple interaction processes is described using a *single* interaction pattern (represented as a textual EN with variables), this pattern is called a *generalization* of the processes. Analogously, agent roles (represented by agent variables) describing the temporary behavior of multiple, not necessarily specified, agents are generalizations of single agents.

For these purposes, we define a recursive function *generalize* based on a method of agent role formation in ENs introduced in [20]. It is a special case of finding the *Least General Generalization* [29]. *generalize* operates on a list of ground textual EN branches.

Definition 7. $generalize : ENL^+ \times \theta \rightarrow ENL$

$$
\begin{array}{l}
generalize((branch_i : 1 \leq i \leq m), \vartheta') = \\
\quad [headGeneralization\ \vartheta''^{13} \\
\quad\quad [\frac{\sum expects(part_1)}{\sum_{p=1}^{q} \sum expects(part_p)}\ generalize(part_1, \vartheta'')] \\
\quad ... \ [\frac{\sum expects(part_q)}{\sum_{p=1}^{q} \sum expects(part_p)}\ generalize(part_q, \vartheta'')]], \text{ with} \\
\\
\quad\quad (headGeneralization, \vartheta'') = generalize'((head_i : 1 \leq i \leq m), \vartheta'), \text{ with} \\
\quad\quad branch_i = [head_i[weight_{i_1}\ c_{i_1}]\ ...\ [weight_{i_{n_i}}\ c_{i_{n_i}}]], \\
\quad\quad head_i =' performative_i(agent_i, content_i)', \\
\quad\quad (part_1,\ ...,\ part_q) = subPartition(cs), \text{ such that } cs \supseteq \biguplus_{i=1}^{q} part_i, \text{ with} \\
\quad\quad\quad cs = \{c_{k_j} : 1 \leq j \leq n_k, 1 \leq k \leq m\} \\
\\
\quad\quad \text{(cf. below for an exemplarily } subPartition \text{ function.) } expects(part_k) \text{ yields} \\
\quad\quad \text{the set of expectabilities of all nodes within } part_k. \\
\\
generalize'(('performative_i(agent_i, content_i)' : 1 \leq i \leq n), \vartheta') = \\
('PerformativeGeneralization(\\
\quad AgentGeneralization, ContentGeneralization), \vartheta' \sqcup \vartheta p \sqcup \vartheta a \sqcup \vartheta c), \text{ with}^{14}
\end{array}
$$

$$
\begin{cases}
AgentGeneralization = agent_1, \vartheta a = \langle\rangle \\
\quad \text{if } \forall i, 1 \leq i \leq n-1 : agent_i = agent_{i+1} \\
\begin{cases}
AgentGeneralization = var, \vartheta a = \langle\rangle \\
\quad \text{if } \exists var, \langle\langle var/agent_i\rangle : 1 \leq i \leq n\rangle \subseteq \vartheta' \\
AgentGeneralization = Var_{new}, \\
\quad \vartheta a = \langle\langle Var_{new}/agent_i\rangle : 1 \leq i \leq n\rangle \text{ otherwise}
\end{cases} & \text{otherwise}
\end{cases}
$$

(*PerformativeGeneralization* and *ContentGeneralization* defined analogous.)
At this, Var_{new} denotes a new variable.

In case the set of branches $branch_i$ corresponds to a full set of sibling branch-roots, a replacement by their resulting generalization would yield a semantically equivalent result (just add up the expectabilities of the original branches). Otherwise, we have to manifold the resulting branch and link it at *different* locations with different expectabilities to different parent nodes. The latter does yield a transformation of the original EN which is in general *not* semantically equivalent to the original EN.
Note that repeated application of *generalize* on a *single* branch can "flatten" this branch until eventually the branch becomes a linear list.

[13] We include ϑ'' here to denote that from here on the environment ϑ'' applies.
[14] Of course, event terms other than speech act-like message terms could be generalized analogously.

The list of substitutions that could be applied in order to retrieve a branch $branch_j$ back from the generalization can be calculated as
$mgu(branch_j, generalize((branch_i : 1 \leq i \leq m)))$, with mgu yielding a most general unifier of the branch and its generalization. But note that this re-transformation can be lossy, i.e., does not necessarily retrieve the original branch.

If we would replace the variables yielded by a generalization by *ignorance variables* (i.e., $?Var$ instead of Var), the generalization would apply to *all* possible instantiations (e.g., *all* syntactically possible agent identifiers), which is useful in order to derive stochastic protocols for open systems (i.e., with a fluctuating set of participants) from a set of example interaction courses. Reasoning on such a generalized EN allows for *conclusions by analogy*, which is also the basis of, e.g., *case based reasoning*, with the argument branches of *generalize* corresponding to the cases here (loosely speaking). A that way resulting generalized EN would of course not be semantically equivalent to its non-generalized predecessor.

As a rule of thumb, branches could be good candidates for being merged via generalization, if the resulting generalization i) does not contain variables for performatives, ii) relatively many agents are replaced by new role variables during generalization, iii) but relatively few new variables are introduced over all, and iv) the expectabilities of merged children do not differ too much.

The following exemplarily partition function realizes i) and iv): $subPartition(s) = p_1 \uplus ... \uplus p_q$ such that

$$\forall k, 1 \leq k \leq q : \forall c_i, c_j \in p_k : Performative(c_i) = Performative(c_j) \wedge \sigma(p_k) < \varepsilon.$$

($\sigma(p_k)$ denoting the standard deviation of the elements in p_k. ε is some tolerance constant.)

Variables in a generalization generated this way with agents as instances reflect characteristic courses of behavior that can be used to constitute agent *roles*, corresponding to these variables. An example using this partition function is shown as Figure 3 (with variables given descriptive names). The generalization step yields from the three argument sequences ("Enter shop...") inter alia that "frequent buyers" (as a role) also normally pay for their goods, which is not the case for "infrequent buyers" here. Note that the underlined expectabilities 0.2 and 1 on the left side have not the same meaning as the corresponding expectabilities above, since the generalized child of node x (y, respectively) is defined for other agents than the original child, even if we would apply restricting environments for the generalized child (e.g.,
$\langle\langle Prospects/a_1, Sellers/a_4, FrequentBuyers/a_1\rangle\rangle$ between "x" and "Prospects : Enter shop"). Information loss is thus a possible side effect of this kind of generalization. The figure depicts a DG-EN. Textually, the generalizing branch could be represented as an sub-EN variable appearing at different positions.

2.4.6 Event and Communication Process Semantics

So far, we have provided a semantic for single expectations. Starting from there, we can ask for the semantic of the expected events in terms of expectations. Informally, the semantics of the event course (an event in its context of preceding events) represented by an EN path is given as its expected continuation (thus we commit ourself to a dedicated pragmatical viewpoint, in linguistics terms). Formally:

Definition 8. The EN-related *Empirical Semantics* of an event e_t in a context $e_0 \sqcup ... \sqcup e_{t-1}$ (a sequence of previous events, with $\sqcup$ denoting timely succession as represented by EN paths) is defined as the probability distribution $\Upsilon_{en, e_0 \sqcup ... \sqcup e_t}$. The distribution is defined using

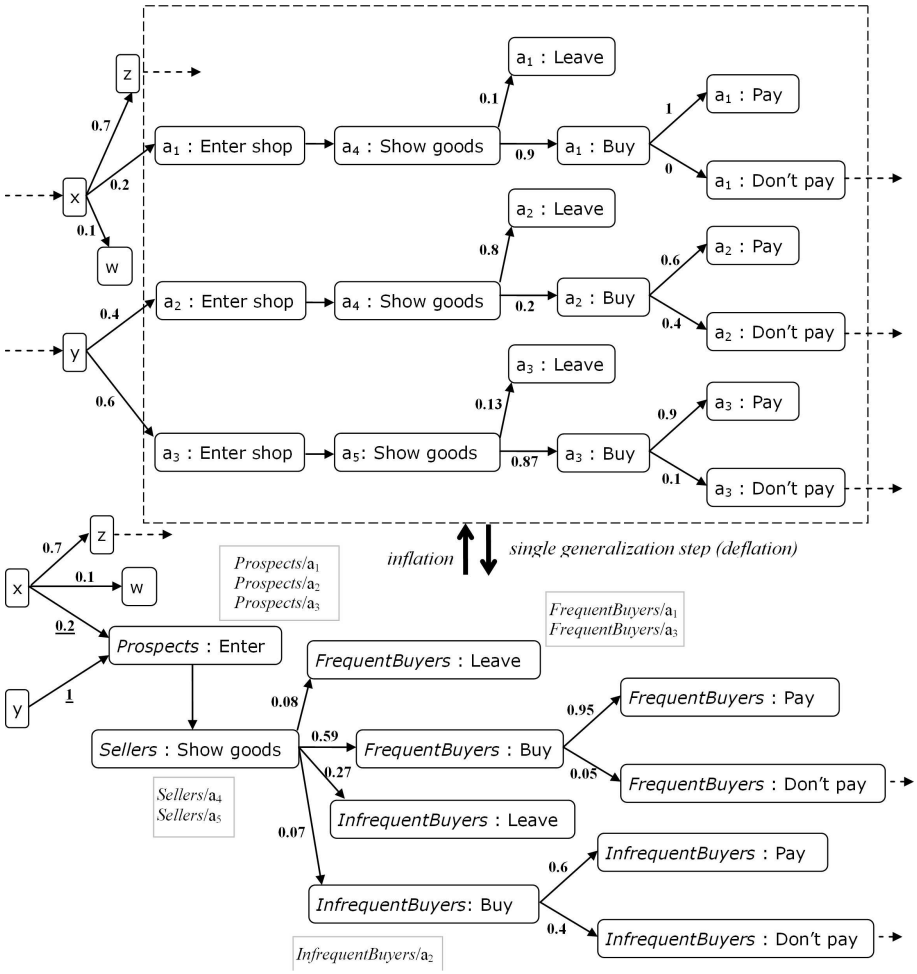

Figure 3: *A generalization step transforming a probabilistic Tree-EN into an DG-EN, creating role variables, The step into the opposite direction would be the* inflation *of the DG-EN.*

$$\Upsilon_{en,e_0 \sqcup ... \sqcup e_t}(w') = \frac{\prod\limits_{i,1 \leq i \leq |w'|} P(w'_i | e_0 \sqcup ... \sqcup e_t \sqcup w'_1 \sqcup ... \sqcup w'_{i-1})}{\sum\limits_{e' \in \mathcal{E}^+} \prod\limits_{i,1 \leq i \leq |e'|} P(e'_i | e_0 \sqcup ... \sqcup e_t \sqcup e'_1 \sqcup ... \sqcup e'_{i-1})},$$

for any finite $w' \in \mathcal{E}^+, w' = w'_1 \sqcup w'_2 \sqcup ...$, and $e' = e'_1 \sqcup e'_2 \sqcup ....$ The numerator thus represents the probability that the sequence $e_0 \sqcup ... \sqcup e_t$ for which we calculate the semantics is concluded by a sequence w', and the denominator is used to normalize this value.
Intuitively the semantics of an event sequence is thus depicted by the branch starting with the node corresponding to that event [20, 24, 25].

A practical approach to empirical semantics using ENs in form of a concrete way to learn adaptive expectations and thus Υ from observed agent interactions is described in [25].

By replacing in this definition the P with the expectability, we gain an "*expected Semantics*" consequently. In case the normativity of at least one edge is greater than zero, the "expected semantics" might deviate from the empirical semantics, being a desired empirical semantics instead:

$$\Upsilon^{Expect}_{en,e_0 \sqcup ... \sqcup e_t}(w') = \frac{\prod\limits_{i,1 \leq i \leq |w'|} \mathit{Expectability}(e_0 \sqcup ... \sqcup e_t \sqcup w'_1 \sqcup ... \sqcup w'_{i-1} \sqcup w'_i)}{\sum\limits_{e' \in \mathcal{E}^+} \prod\limits_{i,1 \leq i \leq |e'|} \mathit{Expectability}(e_0 \sqcup ... \sqcup e_t \sqcup e'_1 \sqcup ... \sqcup e'_{i-1} \sqcup e'_i)}$$

The arguments of *Expectability* denote edges represented as event sequences (e_0 corresponding to the EN's root node).

3 The EOM process

Based on the given description of computational expectations, this section presents the tasks of EOM to be performed by the MA in detail. EOM is tailored to general active and passive modeling tasks, from the viewpoint of both software agents and human designers.
The activities of identifying, evaluating, adapting and propagating social-level expectations in an evolutionary, cyclic process are crucial to EOM. EOM supports these activities by two means: so-called i) *Social Mirrors*, henceforth briefly called *Mirrors*, and ii) agent-internal *Expectation Engines*.

3.1 The *Mirror* concept

Mirrors are software components within the MAS, with the tasks to observe communications, derive expectations structures, and "reflect" modified/enriched versions of them back to the observed agents in form of new communications, all on behalf of MAs. Thus a Mirror functions a bit like a (possibly distorting) real mirror, with communications instead of light beams. Mirrors are rather passive coordination media, and do not take action pro-actively. They are thus meant to support the acquisition and enactment of expectations on behalf of MAs, and being rather un-intelligent themselves, they are especially suited as tools for human MAs (a MAS designer, for example). In this case, a Mirror is corresponding to an EOM-specific CASE tool.

Technically, a Mirror is to its main part a "social knowledge" base with observation capability which empirically derives social-level expectation structures from communications and makes them pro-actively available to both the participating agents and the MA. A Mirror has three major purposes:

1. monitoring agent communication processes,
2. deriving emergent social-level expectation structures from these observations, and

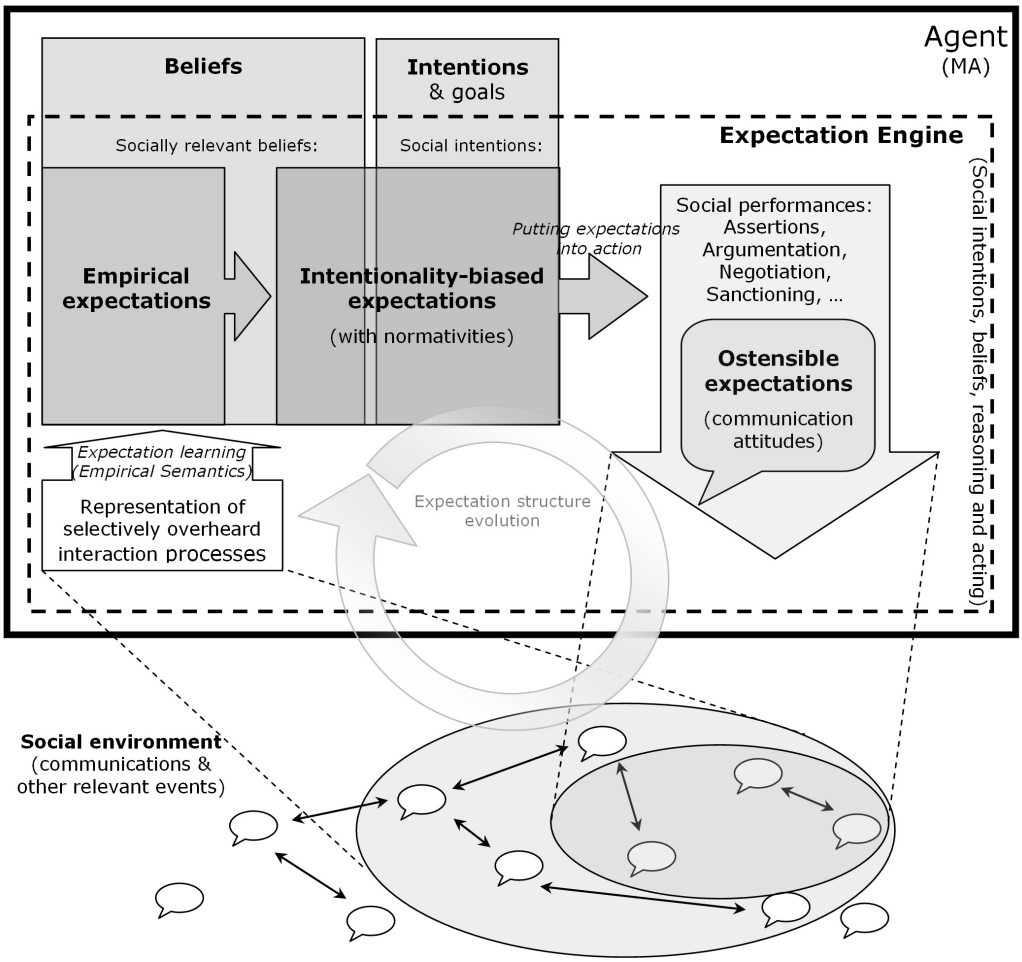

Figure 4: Dataflow in a MA with Expectation Engine

3. making expectation structures visible for the agents and the MA (the former is the so-called *reflection effect* of the Mirror, enabling the self-observation of the system).

It is important to see that not all structures that are made visible to the agents need to be emergent and derived through empirical system observation. Rather, the Mirror can also be structured by the MA to "reflect" deliberatively designed, non-empirical expectation structures as well. In both cases, the agents can access the Mirror very much like a database and actively use the expectation structures provided by it as "guidelines" influencing their reasoning and interactivity. Without the help from a Mirror, even empirically derived expectation structures are considered to be likely often hidden to single agents due to the agent's engagement in locally bounded interaction contexts, their observability restrictions and their limited social reasoning capabilities.

For example, agents can participate in social programs which seem to be useful to them, or refrain from a certain behavior if the Mirror tells them that participation would violate some (adaptive-)normative expectation. Social programs (or structures in general) in which agents continue to participate become stronger, otherwise weaker. (The degree of change in strength depends on the respective normativity.) Thus, the Mirror reflects a model of a social system and propagates it to the agents. As a consequence, the Mirror *influences* the agents – very much like *mass media* do in human society. Conversely, the Mirror continually observes the actual interactions among the agents and adopts the announced expectation structures in its database accordingly. In doing so, the Mirror never restricts the autonomy of the agents. Its influence is solely by means of providing information (possibly about expectable sanctions and norms, though),

and not through the exertion of control[15].

The Mirror, and thus EOM, realizes the principle of *evolutionary* software engineering [1]. More precisely, within the overall EOM process (i.e., within the EOM phases described below) two Mirror-specific operations are continuously applied in a *cyclic* way:

1. it makes the social-level expectations derived by the MA from his goals explicit and known to the agents; and
2. it monitors the social-level expectation structures which emerge from the communications among the software agents, and makes them explicit and known to both the MA and the other agents.

These two operations constitute the core of the overall EOM process, and together they allow an MA to control and to influence the agents' realization and adoption of her specifications. For EOM, the term "evolution" thus applies to expectation structure changes caused both "top-down" by the MA's interventions and "bottom-up" by autonomous variations in the observed agents' behavior.
Further details on Mirrors are provided in [20, 26].

3.2 Expectation Engines

Expectation Engines are MA-internal modules with a functionality similar to that of Mirrors. They serve as a complement of common agent facilities for belief acquisition, planing, and acting. Their tasks are the recording, revision and enactment of expectations, contributing a distinct level for the modeling and influencing of the social behavior and the social environment of the agent modeled as expectations. An Expectation Engine maintains three ENs (or alternative data structures for the representation of expectations):

1. As a part of the MA's belief, an EN for empirical expectations learned from overhearing agent communication and previous knowledge.
2. As a part of the MA's belief and intentions an intentionality-biased EN. It is generated from the empirical EN, plus intentions in form of normativities. It represents those beliefs and intentions of the MA which relate to social activities, and usually contains thus not only fully-adaptive expectations, but adaptive-normative and fully-normative expectations also.
3. An EN which represents the *ostensible* beliefs and intentions of the MA in form of expectations. This EN is communicated to the other agents. It represents what the MA wants other agents to belief about his beliefs and intentions (his communication attitudes, so to say). It is important to see that realizing this EN (actively aiming at making the communication attitudes credible, and pursuing the ostensible intentions) might be only pretended, and might be of course only one means among others aiming at the real goals of the MA. For the special case the MA is sincere, this third EN could of course be identical with the second EN.
   Please find details about ostensible mental attitudes in [23].

The architecture of an MA with an Expectation Engine is depicted in Figure 4.

3.3 The EOM Phases

3.3.1 Phase I: Specifying social-level goals

In the first phase, the MA models the social level of a part of or the whole multiagent system according to her goals in the form of specifications which focus on "social behavior" (i.e., desired

[15] An explicit notion of sanctions in terms of expectations is omitted here (cf., e.g., [34, 10] for approaches to the deontic or contract-based regulation and sanctioning of autonomous agents): In general, Expectation Networks can incorporate information about every kind of treatment of agents as long as it can be represented as a (sequence of) events, i.e. nodes of the expectation network.

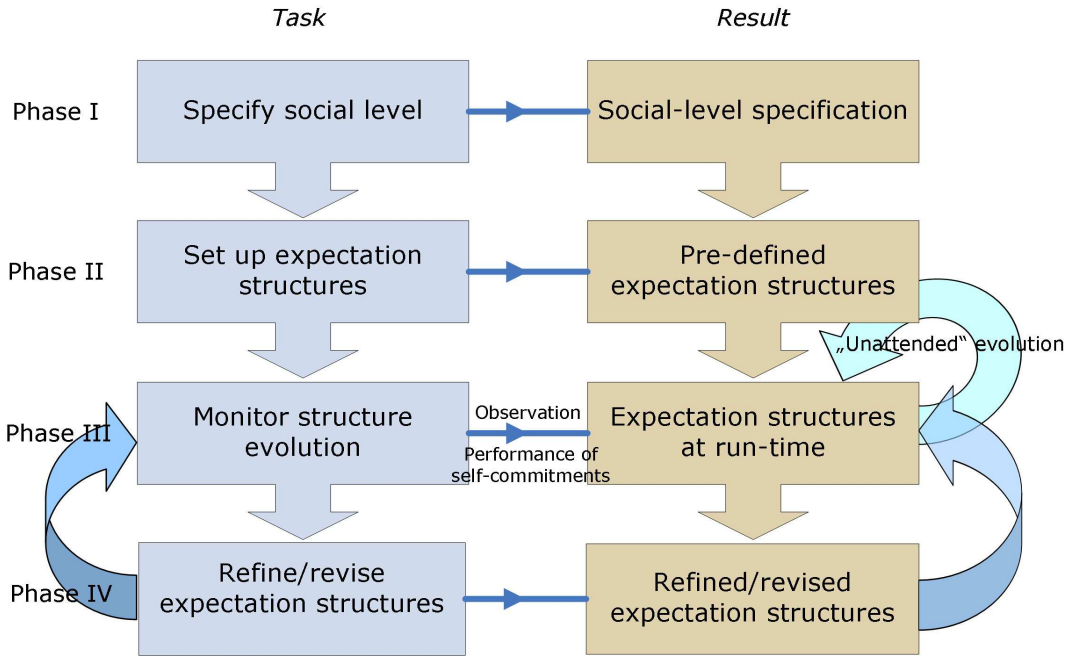

Figure 5: *The EOM phases*

courses of agent interaction) and "social functionality" (i.e., functionality which is achieved as a "product" of agent interaction, such as interactive problem solving) in the widest sense. For this task, the usual specification methods and formalisms might be used, for instance, the specification of desired environment states, policies, constraints, social plans, protocols etc. Of course, this specification could be done directly in terms of social-level expectation structures, like social programs.

3.3.2 Phase II: Setting up and enacting appropriate expectation structures

In the second phase, the MA models and derives social-level expectation structures from the specifications and stores them in the Mirror/Expectation Engine. If the 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 behavior 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 MA models expectation structures which cover as much requirements as possible.

Social-level specifications can be modeled as fully-adaptable or adaptive-normative expectations. The former can be used for the establishing of *hints* for the other agents which are able to adapt during the structure evolution, the latter for the transformation of *constraints* and other more or less "hard" requirements into expectations. It should be kept in mind that even a fully-normative expectation derived from a constraint does by itself not force the agents to behave conforming to the rule, since it is "only" an expectation. If a (adaptive-)normative expectation is constantly violated by the agents (i.e., the deviancy of the expectation rises), the MA can either decide to try to argue in favor of the (adaptive-)normative expectation, or to enforce it (introducing sanctions and propagate them with associated additional sanction expectations), or to drop it (change the normativity). If the normativity is lower 1, the Mirror/Expectation Engine also changes the expectability of the adaptive-normative expectation at least in the long term.

After the MA has finished the expectation modeling, she communicates them (either sincerely, or in form of ostensible expectations) to the other agents via the Mirror/Expectation Engine.
Whereby EOM does not prescribe or provide an explicit notion of sanctions or argumentation, we can use the fact that (adaptive-)normative expectations need to be communicated to the agents to achieve a semi-automatic enforcement of (adaptive-)normative expectations using the *EMPRAT* algorithm. This way, we make (adaptive-)normative expectations pro-active, so to say. For lack of space details had to be omitted here, please refer to [23].

3.3.3 Phase III: Monitoring structure evolution

In the third phase of the EOM process, it is up to the MA to observe and evaluate the evolution of expectation structures which becomes visible to her through the Mirror/Expectation Engine. In particular, she has to pay attention to the relationship of the continuously adapted social-level expectation structures and her objectives from phase I, which means that she analyzes the expectation structures with regard to the fulfilment of (adaptive-)normative expectations established by the MA and the achievement of her goals. Because the Mirror/Expectation Engine is only intended to obtain and deploy expectation structures, it could be necessary to support it with a software for the (semi-)automatical "re-translation" of expectation structures into other forms of specification like rules, and vice versa.
As long as the expectations structures develop in a positive way (i.e., they match the MA's goals, deviancies are sufficiently low) or no emergent structures can be identified that deserve being made explicit to improve system performance, the MA does not intervene. Otherwise she proceeds with phase IV.

3.3.4 Phase IV: Refinement of expectations

In the last phase, the MA uses her knowledge about the positive or negative emergent properties of the interaction system to improve the social-level expectation structures. Usually, this is achieved by setting up expectation which discourage "bad" events, 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 relevant MA goals are achieved or no further improvement seems probable at least for the moment (per definition, open systems never settle on a final equilibrium while active).

4 Case Study: The Internet Car Trading Platform

4.1 Scenario Overview

In the following we present an example for EOM, with the MA being the MAS designer. Imagine a web site that brings together car dealers, private pre-owned car sellers and potential buyers who trade cars online (cf. `www.imotors.com` , `www.autoweb.com`, `www.autointernet.com`, `www.autotradecenter.com`). There is an "offers" section in which sellers can display images, technical details and prices of cars for sale. In the "requests" area, buyers can post requests for cars that they would be interested in. A forum is available, in which inquiries can be placed, discussions, bargaining and negotiations may take place publicly or privately (as forum users wish), etc.

4.2 Making Top-Level Design Decisions

Having made a decision on taking an agent-based approach, the MA (the website designer in this case) must develop a top-level description of the system which will, to the least, include decisions regarding infrastructure, interaction environment and, above all, participating agents (or agent types).
Here, we will assume that the designer of the platform is designing a semi-open system: on the

one hand, the system offers user interface agents that monitor the platform on behalf of users, profile users to derive interests/needs and draw their attention to interesting information on the platform. A second, pre-built type of agents are search agents that constantly re-organize the platform's database and can search it efficiently. These can be contacted by user interface agents as well as by humans for search purposes. We assume that all interactions with these search agents are benevolent, since they are not truly autonomous (they simply execute others' requests). On the other hand, there is a number of agent types that have not been designed by the designer of the platform. There can (and should) exist human and non-human agents representing individuals or organizations that interact with the platform in a "socially" unprescribed way (only restricted by implementation-level protocols and standards, e.g. FIPA compliance). Generally, these agents are black-boxes for the system designer.

Further refinement of these initial design decisions will require looking at a multitude of issues, ranging from communication facilities and standards and capabilities of in-built profiling and search agents to database models etc. For our purposes, we can restrict this identification of requirements to social level characteristics of the platform since these are the subject of the EOM process.

4.3 Identifying Social Level Requirements

As social-level goals, we consider the following motives of a car trading platform (CTP) provider:

1. Maximum quality of service should be provided: the range of offered and requested cars has to be broad and their specifications must relate to their prices; the reliability of transactions must be high; trust between buyers and sellers and between all users and the platform must be at a reasonable level.

2. Transaction turnover should be maximized, because it indicates (in our example) high return on investment for the CTP provider stakeholders.

3. Traffic on the platform must be maximized, to ensure high advertisement returns.

In the following, we sketch how the EOM process model can be applied in the analysis and design of such a system.

The dilemma in designing the social level of such a platform is obvious: system behavior should meet the design goals and at the same time it shouldn't compromise participating external agents' private goals by being overtly restrictive. An expectation-level model of social structures is needed to cope with this situation. We next sketch the application of the suggested analysis and design process to the CTP.

4.4 Implementing the EOM Process

4.4.1 Phase I: Specifying the social level

In the first step the social structures are modeled in the form of (formal or informal) design specifications. They might include the following (we use natural language for convenience and concentrate only on a few design issues for lack of space):

1. Agents committing themselves to purchase/sell actions towards other must fulfil all resulting obligations (deliver, pay, invoice etc.)

2. Unreliable behavior induces reluctance to enter business relationships on the side of others. Fraudulence leads to exclusion from the platform.

3. Interest in offers and requests must be shown by others in order to provide motivations to keep up the use of the platform.

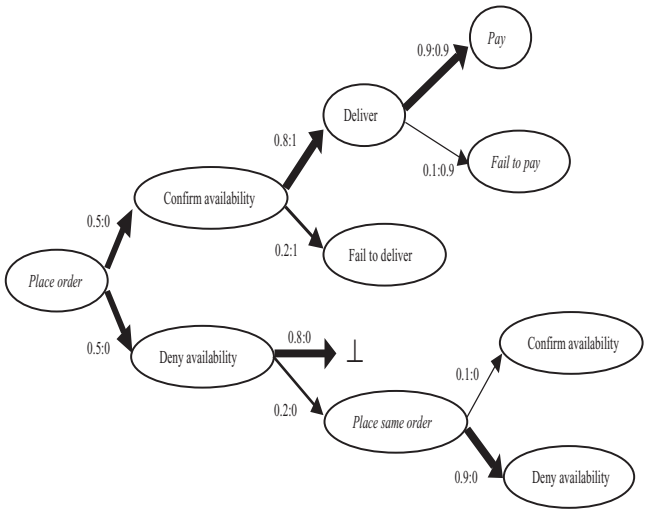

Figure 6: *Social program "order-deliver-pay" (buyer actions are shown in italic font, seller actions in plain face, as in all following figures, speech act arguments are omitted for lack of space): expectations about availability are balanced; in the "available" case, dealers are expected to deliver and customers are expected to pay. In the "not available" case, dealers are expected to confirm their prior statement if asked a second time (even though the probability of such a second request is low).*

The first specification is very important in order to foster trust among agents in such a platform. If communication were only inducing a bunch of loose pseudo-commitments that are never kept, the CTP risks becoming a playground instead of a serious, efficient marketplace. This principle is refined by item 2: the "must" in the first rule can obviously not be deontically enforced on autonomous agents, so it has to be replaced by a "softer" expression of obligation: by specifying that unreliable behavior decreases the probability of others interacting with the unreliable individual in the future, we provide an interpretation of the former rule in terms of "consequences". Also, we distinguish "sloppy" from "illegal" behavior and punish the latter with exclusion from the platform, a centralized sanction that the platform may impose. The third specification is somewhat more subtle: it is based on the assumption that agents will stop posting offers and requests, if they don't receive enough feedback. Since we have to ensure both a broad range of offers as well as reasonable traffic on the site, we want to make agents believe that their participation is honored by others so that they keep on participating (for private buyers this might be irrelevant, since they buy a car once every 5 years, but it is surely important to have plenty of professional dealers frequent the site).

The process of specifying such possible societal behaviors should be iterated on the basis of "scenarios" for all courses of communication that are of interest and seem possible, so as to yield requirements for the social system that is to be implemented.

4.4.2 Phase II: Deriving and enacting appropriate expectation structures

Clearly, the three requirements above can be analyzed in terms of expectations, that is, as variedly normative, possibly volatile rules that are made known to agents and evolve with observed interaction. The second phase of the EOM process consists of making these abstract requirements concrete as expected communication structures. Two such expectation structures derived from the above requirements 1. and 3. are shown in Figures 6 and 7.

The first example depicts an expectation structure of an order-deliver-pay-procedure in the CTP. It encapsulates high delivery and payment expectations (i.e., high transaction reliability),

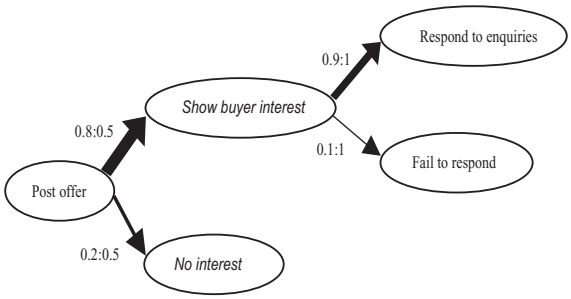

Figure 7: *"Initiatives are honored" program: it is expected that dealers receive some response to their offers by potential customers, and that they react to enquiries themselves.*

but also a more specific expectation as concerns availability statements that are made by dealers: although it is equally probable that the requested car will be available upon a first order, it is highly unexpected that a car that had not been available is suddenly available upon a second, identical order (in our model, responses to communication are supposed to occur in time-spans that are much shorter than those needed to change stock). Thus, the first response is given much more weight, and a notion of "honesty" in responding to orders is assumed.

The second example is closely related to design goal 3 introduced above. Here, the expectation structure is used to express that few posted offers go unanswered by interested customers, and that the enquiries of such customers are responded to with high probability. By using such a structure, the designer can reassure both dealers and customers that it is *worthwhile* posting orders and enquiries to orders. If followed by the users of the CTP, such a structure would imply that postings will be answered even if the other party is not *actually* interested in the offer/question, and is just replying out of a sense of "politeness", to the end of making everyone feel that their contributions are honored. Associated with such conventions would be the designer's goal to keep the CTP frequented, by presenting the social structures as open and rich.

These simple examples given, we can return to our EOM design process model. We have shown how two social structure specifications were turned into concrete expectation structures (phases I and II). For lack of space, we have concentrated on *social programmes* and neglected roles, social agents and values. Preassuming that the CTP is implemented and observed during operation, we can now proceed to phase III.

4.4.3 Phase III: Monitoring structure evolution

Unlike phases I and II, this phase focuses on *observation* of the system in operation in order to further refine expectation structures and their processing. It is essential to keep in mind that the systemic expectation "Mirror" (as a software component) leaves plenty of choices not only as concerns the *choice* of employed expectation structures, but also with respect to how these structures are *processed*, that is, how they *evolve* through monitored agent behavior in system operation. To stress this second aspect, we concentrate on this processing of expectations in the following examples.

Suppose, first, that we observe that actual behavior largely deviates from that assumed in Figure 6 in that there are many fraudulent customers who do not comply with their obligation to pay once the car has been delivered unless the dealer threatens with legal consequences several times. Obviously, identifying such a problem preassumes that interaction is tracked and that interaction patterns are statistically analyzed and evaluated with respect to existing system goals. Therefore, the software engineer's primary duty is, at this stage, to spot interesting behaviors (both desirable and undesirable ones). Once realized, we are faced with a problem. By default, even though payment was designed as a norm, the "expectation Mirror" would in show a high

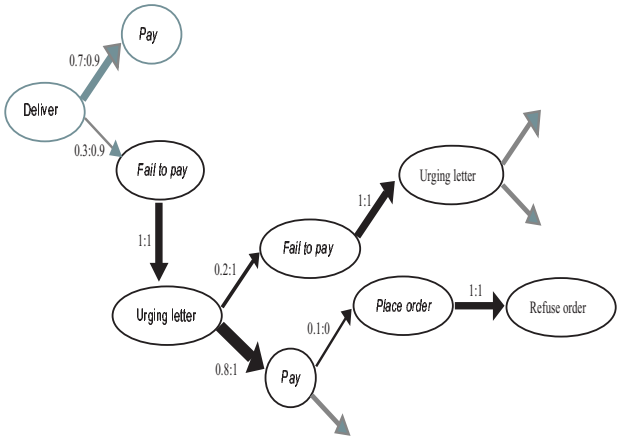

Figure 8: *Specifying a new functionality.*

deviancy (and since the normativity of "pay" is lower 1, it would in the long term even "truthfully" adapt the expectation strengths of this expectation so that the strength of "fail to pay" increases). This would mean that an emergent, hidden structure would be made explicit in the system, but, unfortunately, this would be a structure that embodies a functionality which does not serve the system goals (even though it has been "selected" through actual interaction) because it would make future dealers doubt the reliability of the system.

As a second example, suppose that the expectation structure in Figure 7 corresponds to the actual system behavior, but not because of some "polite" policy of customers to show interest in *any* dealer posting – instead, demand in cars is simply (temporarily) so high (and maybe the CTP is for some other reasons very attractive for customers) that almost no offer posting goes unanswered. Assume further, that our initial design was to enforce "politeness" by insinuating that it was a convention of the platform, even if customers would not have been polite at all, that is, we had implemented this expectation structure as rather immutable (normativity of 0.5/1) regardless of the agents' behavior.

In both cases, we have identified emergent (positive and negative) properties of the system that must be dealt with in phase IV.

4.4.4 Phase IV: Refinement of expectation structures

As designers of the platform, we can react to such emergent properties in different ways. To give a flavor for the kind of decisions designers have to make when refining expectation models, we discuss the two examples mentioned above.

In the case of the "spreading fraudulent customers", the most straightforward solution would be to impose sanctions on the fraudulent behavior observed (i.e., to add new expectation structures). Let us assume, however, that an analysis has shown that it is too costly to verify customers' solvency and payment reserves (e.g., by inquiring other E-commerce platforms about them). On the other hand, ignoring the changes by keeping the old expectation structure (and asserting a high payment reliability in a "propaganda" way) might result in future inconsistencies: if too many individuals realize that it does not correspond to the actual social structure, they will use it less, and the "social design" level will provide lesser possibilities to influence system behavior for the designer.

Obviously, a trade-off has to be found. One possible solution would be to extend the structure in the way suggested by Figure 8, such that failure to pay results in reluctance of dealers to accept future orders from the unreliable customer. So, in phase IV we can specify a new functionality that feeds into the system in the next cycle. As concerns the second, "positive" emergent property, we might consider lifting the constraint of presenting an "immutable" politeness convention, in order

to allow for optimization on the agents' side: making the rule normative implies that it wouldn't change, even if, for example, dealers' offers changed over time – hence, there is little pressure for dealers to actively try to meet customer demand. Thus, if we allowed this expectation to *adapt* to the actual interest shown in offers (e.g., by updating expectation strengths as *real* probabilities, which can be achieved by decreasing the normativity value shown in Figure 7), dealer agents would start noticing which of their postings are good (ones which increase the rate of customer inquiry) and which aren't. (After all, maximizing market efficiency in this way might help maximizing CTP profits, which also depend on gross trade turnover.) We therefore decide to increase the adaptivity of this expectation structure.

Performing such modifications to the expectation level design of a system nicely illustrates how rather restrictive social structures can give way to more emergent phenomena in "safe" non-risky situations as the one depicted here when optimization is the prominent issue, and not the reduction of chaos.

These simple examples underpin the usefulness of explicit modeling of social structures in the proposed EOM process model. In particular, they show how both designing social *structures* and designing the *processing* of such structures plays an important role in the open systems we envisage. Also, they illustrate the evolutionary intuition behind our design process: agents select social structures through their interaction, and designers select them through design.

5 Conclusion

EOM is thought to be applicable in all fields of agent-oriented research and engineering, where an entity (a MA in our case) needs to model and maybe influence the behavior of autonomous black- or gray-box agents. Besides the possibility to implement EOM concepts within "ordinary" agents in order to improve their social cognition and interaction abilities, as for the future specification and extension of EOM, we aim especially for the area of agent-orient software engineering and programming. Engineering agent-oriented software while at the same time taking autonomy as a key feature of agency seriously is a great challenge. On the one hand, it is (among other things) autonomy that makes the concept of an agent powerful and particularly useful, and that makes agent orientation significantly distinct from standard object orientation. There is an obvious and rapidly growing need for autonomous software systems capable of running in open application environments, given the increasing inter-operability and inter-connectedness among computers and computing platforms. On the other hand, autonomy in behavior may result in "chaotic" overall system properties such as uncontrollability that are most undesirable from the point of view of software engineering and industrial application. In fact, it is one of the major driving forces of standard software engineering to avoid exactly such properties. To come up to each of these two contradictory aspects – the urgent need for autonomous software systems on the one hand and the problem of undesirable system properties induced by autonomous behavior on the other – must be a core concern of agent-oriented software engineering, and is the basic motivation underlying the work described here. A number of agent-oriented software engineering methods (see [13, 16] for surveys) as well as agent-oriented autonomy and organizational structure specification formalisms (e.g., [2, 9, 11, 35, 27, 19, 12, 34]) are now available. Like EOM, all these methods and formalisms aim at supporting a structured development of "non-chaotic" autonomous software. However, they do so in a fundamentally different way, even compared to the most elaborated of these frameworks which grant the actors a high degree of autonomy (e.g. *OperA* [10]): Besides the possibility to specify social structures deliberatively, EOM also *learns and revises* social structures empirically from observed agent interactions at run-time, resulting in a structure-level model of the multiagent system, and restricts autonomous behavior only if this turns out to be necessary *retrospectively* during the evolutionary development process, with as few as possible precognition and pre-structuring required. Against that, most of the other methods and formalisms show a clear tendency toward (seriously) restricting or even excluding the agents' autonomy a priori. Different mechanisms for achieving autonomy restrictions have been proposed, including e.g. the hardwiring of organizational structures, the rigid predefinition of when and how an agent has to

interact with whom, and the minimization of the individual agents' range of alternative actions. As a consequence, methods based on such mechanisms run the risk to create software agents that eventually are not very distinct from ordinary objects as considered in standard object oriented software engineering since many years. EOM aims at avoiding this risk by accepting autonomy as a necessary characteristic of agency that must not be ruled out headily (and sometimes even can not be ruled out at all, as it is typical for truly open multiagent systems). With that, EOM is in full accordance with Jennings' claim to search for other solutions than the above mentioned restrictive mechanisms [17, p. 290]. Moreover, EOM with its grounding on Luhmann's theory of social systems precisely is in the line of Castelfranchi's view according to which a socially oriented perspective of engineering social order in agent systems is needed and most effective [6]. In addition to that, and more generally, this thorough sociological grounding also makes EOM different from other approaches that apply sociological concepts and terminology in a comparatively superficial and ad hoc manner. On these grounds, we hope that taking computational expectations as a level of social reasoning, analysis and design opens a qualitatively new perspective of agent-oriented software.

Acknowledgements. This work has been partially supported by DFG under contracts no. Br609/11-1 and Br609/11-2.
Many thanks to the reviewers for their very useful comments on a draft of this article.

References

[1] L. Arthur, editor. *Evolutionary development: Requirements, prototyping & software creation.* John Wiley & Sons, 1991.

[2] M. Barbuceanu, T. Gray, and S. Mankovski. The role of obligations in multiagent coordination. *Journal of Applied Artificial Intelligence*, 13(2/3):11–38, 1999.

[3] F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors. *Methodologies and software engineering for agent systems.* Kluwer Academic Press, Boston et al., 2004.

[4] M. Bratman. *Intentions, Plans and Practical Reasoning.* Harvard University Press, Cambridge, MA, 1987.

[5] W. Brauer, M. Nickles, M. Rovatsos, G. Weiss, and K. Lorentzen. Expectation-oriented analysis and design. In M. Wooldridge, G. Weiss, and P. Ciancarini, editors, *Agent-oriented software engineering. Proceedings of the Second International Workshop (AOSE-2001)*, Lecture Notes in Artificial Intelligence, Vol. 2222, pages 226–244. Springer-Verlag, 2002.

[6] C. Castelfranchi. Engineering social order. In *Working Notes of the First International Workshop on Engineering Societies in the Agents' World (ESAW-00)*, 2000.

[7] C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberate normative agents: Principles and architecture. In *Proceedings of the 6th International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, 1999.

[8] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence, 42(3)*, 1990.

[9] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7:69–79, 1999.

[10] V. Dignum. *A model for organizational interaction: based on agents, founded in logic.* PhD thesis, University of Utrecht, 2003.

[11] V. Dignum, J. Vazquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In R. H. Bordini, M. Dastani, and J. Dix, editors, *Programming Multi-Agent Systems: Second International Workshop ProMAS 2004*, Lecture Notes in Artificial Intelligence Vol. 3346. Springer-Verlag, 2005.

[12] M. Esteva, D. de la Cruz, and C. Sierra. Islander: an electronic institutions editor. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS 2002)*, 2002.

[13] J. Gomez-Sanz, M.-P. Gervais, and G. Weiss. A survey onf agent-oriented software engineering research. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and software engineering for agent systems*. Kluwer Academic Press, Boston et al., 2004.

[14] A. Herzig and D. Longin. A logic of intention with cooperation principles and with assertive speech acts as communication primitives. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS 2002)*, 2002.

[15] C. Hewitt. Offices are open systems. *ACM Transactions on Office Information Systems*, 4(3):271–287, 1986.

[16] C. Iglesias, M. Garijo, and J. Gonzales. A survey of agent-oriented methodologies. In J. Müller, M. Singh, and A. Rao, editors, *Intelligent Agents V. Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, Lecture Notes in Artificial Intelligence Vol. 1555, pages 317–330. Springer-Verlag, 1999.

[17] N. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.

[18] N. Jennings and M. Wooldridge. Agent-oriented software engineering. In J. Bradshaw, editor, *Handbook of Agent Technology*. AAAI/MIT Press, 2002.

[19] F. Lopez y Lopez, M. Luck, and M. d'Inverno. Constraining autonomy through norms. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, 2002.

[20] K. Lorentzen and M. Nickles. Ordnung aus Chaos - Prolegomena zu einer Luhmann'schen Modellierung deentropisierender Strukturbildung in Multiagentensystemen. In T. Kron, editor, *Luhmann modelliert. Ansaetze zur Simulation von Kommunikationssystemen*. Verlag Leske & Budrich, 2002.

[21] M. Luck, R. Ashri, and M. D'Inverno. *Agent-based software development*. Artech House, Inc, Norwood, MA, 2004.

[22] N. Luhmann. *Social systems*. Stanford University Press, Palo Alto, CA, 1995. Originally published in 1984.

[23] M. Nickles. Exposing the communication level of open systems: Expectations, ostensible attitudes and multi-source assertions. Research Report FKI-249-05, Technical University of Munich, 2005.

[24] M. Nickles, M. Rovatsos, W. Brauer, and G. Weiss. Towards a unified model of sociality in multiagent systems. *International Journal of Computer and Information Science*, 5(2):73–88, 2004.

[25] 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 2004)*, pages 94–101, 2004.

[26] M. Nickles and G. Weiss. Multiagent systems without agents: Mirror-holons for the compilation and enactment of communication structures. In K. Fischer, M. Florian, and T. Malsch, editors, *Socionics: Its Contributions to the Scalability of Complex Social Systems*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004. To appear.

[27] O. Pacheco and J. Carmo. A role based model for the normative specification of organized collective agency and agents interaction. *Autonomous Agents and Multi-Agent Systems*, 2002.

[28] H. V. D. Parunak. Visualizing agent conversations: Using enhanced dooley graphs for agent design and analysis. In *Proceedings 2nd Intl. Conf. Multiagent Systems*, 1996.

[29] G. Plotkin. A note on inductive generalization. *Machine Intelligence, 5*, 1971.

[30] M. Rovatsos, G. Weiss, and M. Wolf. An approach to the analysis and design of multiagent systems based on interaction frames. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS 2002)*, 2002.

[31] M. Sadek. A study in the logic of intention. In *Proceedings of the third international conference on principles of knowledge representation and reasoning (KR'92)*, 1992.

[32] B. Tran, J. Harland, and M. Hamilton. Observation expectation reasoning in agent systems. In *Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*, 2004.

[33] G. Weiss. Agent orientation in software engineering. *Knowledge Engineering Review*, 16(4):349–373, 2002.

[34] G. Weiss, M. Rovatsos, M. Nickles, and C. Meinl. Capturing agent autonomy in roles and XML. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 105–112, 2003.

[35] M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems 3(3)*, 2000.