Journal of Experimental & Theoretical Artificial Intelligence
Vol. 20, No. 4, December 2008, 277–317

Taylor & Francis
Taylor & Francis Group

Learning communicative actions of conflicting human agents

Boris A. Galitsky[a]* and Sergei O. Kuznetsov[b]

[a]*School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK;* [b]*All-Russian Institute for Scientific and Technical Information (VINITI), Usievicha 20, Moscow 125190, Russia*

(*Received September 2006; final version received August 2007*)

One of the main problems to be solved while assisting inter-human conflict resolution is how to reuse the previous experience with similar agents. A machine learning technique for handling scenarios of interaction between conflicting human agents is proposed. Scenarios are represented by directed graphs with labelled vertices (for communicative actions) and arcs (for temporal and causal relationships between these actions and their parameters). For illustrative purposes, classification of a scenario is computed by comparing partial matching of its graph with graphs of positive and negative examples. *Nearest Neighbour* learning is followed by the JSM-based learning which minimised the number of false negatives and takes advantage of a more accurate way of matching sequences of communicative actions. Developed scenario representation and comparative analysis techniques are applied to the classification of textual customer complaints. It is shown that analysing the structure of communicative actions without context information is frequently sufficient to estimate complaint validity. Therefore, being domain-independent, proposed machine learning technique is a good compliment to a wide range of customer relation management applications where formal treatment of inter-human interactions is required in a decision-support mode.

Keywords: behaviour of human agents; multi-agent conflict; communicative actions; machine learning

1. Introduction: Reasoning with conflict scenarios

Scenarios of interaction between agents are an important subject of study in artificial intelligence. An extensive body of literature addresses the problem of logical simulation of behaviour of autonomous agents, taking into account their beliefs, desires and intentions (Bratman 1987). A substantial advancement has been achieved in building the scenarios of multi-agent interaction, given the properties of agents, including their attitudes. Recent work in agent communications has been in argumentation (Rahwan et al. 2003), in dialogue games (Boella, Hulstijn and van der Torre 2004), formal models of dialogue (Johnson, McBurney and Parsons 2005), in conversation policies (Nodine and Unruh 2000), and in social semantics (Carley 1997; Singh 2005). A number of current approaches to the multiagent systems are based on logical deduction (Fagin, Halpern, Moses and

*Corresponding author. Email: bgalitsky@uptake.com

ISSN 0952–813X print/ISSN 1362–3079 online

DOI: 10.1080/09528130701650318
http://www.informaworld.com

Vardi 1996; Shanahan 1997), and simulation (Muller and Dieng 2000; Galitsky 2004). However, means of automated comparative analysis for interaction scenarios for human agents are still lacking (Guerra-Hernandez 2004).

In the case of deduction, the sequence of mental states of agents is deduced from their initial mental states and initial attitudes. Deductive reasoning about actions and the logic with agents' attitudes as modalities are the most popular means to yield sequences of mental states of agents (Wooldridge 2000). In the case of simulation, the system imitates the decision-making of human agents, choosing the best action for each agent at each step, taking into account its current intentions, beliefs and desires, as well as those of others. Having the preference relation on the set of resultant states, each agent selects an action that is expected to lead to the most desired state (Galitsky et al. 2005). Deduction works fairly well for automated agents, simulation helps to predict typical behaviour, but learning from previous experience is essential if the behaviours deviate from rational or normal, or are specific to current application domain.

In our previous studies we analysed the roles of deduction, simulation and learning in application to human agents (Galitsky 2006a, 2006b). In the current paper we build the representation machinery and develop a machine learning technique for operating with scenarios which include a sequence of communicative actions. We propose a framework for classifying scenarios of inter-human conflicts. This framework has been implemented in a stand-alone mode or used in combination with deductive reasoning or simulation to be a part of a decision support system.

In spite of the advances in modelling conflicts and negotiations between autonomous agents and its deployment in a number of domains, a general framework to reuse the experience on conflict resolutions accumulated in earlier cases has not been developed. For effective building and predicting of interaction between autonomous agents, it is helpful to augment reasoning and/or simulation with machine learning (Weiss and Sen 1996; Olivia, Chang, Enguix and Ghose 1999; Stone and Veloso 2000). In case of human agents, an adequate behavioural model that gives a plausible data structure for machine learning is essential as well. It would reduce the number of possible agents' actions at each step, taking into account how these agents acted in previous cases. Obviously, formalising human behaviour is a much more complex task than that of 'a communication protocol' of autonomous agents. Hence we restrict ourselves to communicative actions (and causal links between them) of human agents in the course of interaction (conflict) as a way to describe their behaviour.

Recently, the issue of providing BDI (belief-desire-intention) agents (Bratman 1987) with machine learning capabilities attracted interest; an application domain such as agents for intelligent information access was considered in Stone and Veloso (2000). Nevertheless, a BDI-based machine learning framework for operating with scenarios of inter-human interactions has not been developed yet. A number of case-based reasoning approaches have been suggested to treat the scenarios of interaction between BDI agents (Olivia et al. 1999; Laza and Corchado 2002); however, description of agents' attitude is reduced to their beliefs, desires and intentions in these studies. Indeed, behaviour of real-world conflicting agents is described in a richer language using a wide number of mental entities including *pretending*, *deceiving*, *offending*, *forgiving*, *trust*, and others.

The importance of learning in negotiation has been recognised in the game research community as fundamental for understanding human behaviour as well as for developing new solution concepts (Harsanyi and Selten 1972; Osborne and Rubinstein 1994).

Jordan (1992) studied the impact of Bayesian learning processes for finite-strategy normal form games. Kalai and Lehrer (1993) analysed infinitely repeated games in which players try to subjectively maximise their utility, learning to predict future strategies of opponents. These theoretical results, however, are available only for the simplest settings which can be represented in the game theoretic language and valid only under very restrictive assumptions, such as only a subset of possible negotiation strategies are allowed. Also, it is hard to apply the developed machinery to a practical conflict resolution system: it lacks handling mental states of participants and assumed domain-specific knowledge is available and can be subject to formalisation.

Much distributed AI and game theoretic research (Rosenschein and Zlotkin 1994; Kraus and Subrahmanian 1995) deals with co-ordination and negotiation issues by giving pre-computed solutions to specific problems. Mor, Goldman and Rosenschein (1995) discussed multi-agent learning as a means to reach equilibrium. The author modelled agents as finite automata and analysed the computational complexity of certain classes of learning strategies based on this automaton model. Also, there has been much research reported on developing theoretical models in which learning plays a key role, especially in the area of adaptive dynamics of games (e.g. mentioned above (Jordan 1992; Kalai and Lehrer 1993)). However, to build a negotiation assistant and conflict resolution system, it is necessary to improve their negotiation competence based on learning from previous experience of interactions among human agents. Learning in inter-human setting is closely related to the issue of how to model the overall interactions process in terms of communicative actions, i.e. what negotiation protocols are adopted. Zeng and Sycara (1997) simulate a sequential decision-making protocol for negotiation between autonomous agents which are able to learn in a rational manner using Bayesian approach. In the current paper we use deterministic machine learning because explicit motivations for the decisions might be more important than the decision itself, when decision support is provided.

Formalised inter-human conflict is a special case of formal scenario where the agents have inconsistent and dynamic goals; a negotiation procedure is required to achieve a compromise (Muller and Dieng 2000). In this paper we discover that following the logical structure of how negotiations are represented in a scenario (represented as a text or in a structured way), it is possible to judge about consistency of this scenario. We take advantage of this possibility and propose an interactive form where the required parameters of communicative actions are specified from the viewpoint of a given agent.

We believe that a useful machine learning framework to operate with scenarios of inter-human interactions should exhibit the following characteristics:

- It should be capable of relating a scenario to a class given a number of classes specified for a given domain by experts.
- It should be based on a concise yet compact and effective model to represent inter-human interactions, operating with a rich set of communicative actions.
- It should be domain-independent and therefore reusable from one domain to another; it should also allow avoiding building domain-specific ontologies.
- It should provide motivations for the classification decisions, being a component of a decision support system for such industry sector as customer relation management.

Inter-human interaction scenarios suggest the usage of complex data structure. In this paper we employ labelled directed acyclic graphs with arcs for describing interaction of two parties in a conflict, thus being within the standard concept graph representation

(Sowa 1984). A learning model needs to be focused on a specific graph representation for these conflicts. The learning strategies used here are based on ideas similar to that of *Nearest Neighbour* (see, for example, Mitchell 1997), case-based (Kolodner 1993) and concept-based learning (Kuznetsov 1999; Ganter and Kuznetsov 2001) or the JSM-method (Finn 1991). Having defined scenarios and the operation of finding common sub-scenarios, we use *Nearest Neighbour* as a simple illustration of our approach to relate a scenario to the class of 'valid' or 'invalid' scenarios. We then proceed to JSM-based learning to assure avoidance of false positives in as much degree as possible. JSM-based learning delivers the most cautious approach to classification of a human behaviour and attitude to comply with ethical and legal norms.

The paper is organised as follows. The introduction of the domain of conflict scenarios is followed by a formal treatment of mental actions and defining a conflict scenario as a graph with vertices labelled by mental actions. Having defined the similarity operation on graphs (finding maximal common subgraphs), we present the procedure of relating a scenario to a class using the *Nearest Neighbour* approach. To improve the classification accuracy and to adjust the machine learning technique to real-world requirements, we use the logic programming system *Jasmine* which is based on JSM learning. The procedure of finding similarities between scenarios is then described, taking into account aggregation of communicative actions with the same subject. The paper concludes with the evaluation of the proposed technique in the domain of banking complaints. The Appendix includes the code snapshots for computing the communicative action-based similarity between scenarios.

2. The domain of conflict scenarios

In this section we present our model of a conflict scenario oriented to use in a machine learning setting. Here we develop a knowledge representation methodology based on an approximation of a natural language description of a conflict (Galitsky 2003).

When modelling scenarios of inter-human conflict, it is worth distinguishing communicative (mental) and non-mental states and actions. The former include *knowing*, *pretending* (states) and *informing* or *asking* (actions); the latter are related, for example, to *location*, *energy* and *account balance* (physical states), as well as *moving*, *heating* and *withdrawal* (physical actions). To form a data structure for machine learning, we approximate an inter-human interaction scenario as a sequence of communicative actions, ordered in time, with a causal relation between certain communicative actions. Our approximation follows the style of situation calculus; scenarios are simplified to allow for effective matching by means of graphs. Only mental actions remain as a most important component to express similarities between scenarios. Each vertex corresponds to a mental action, which is performed by either *proponent*, or *opponent*, the latter are called agents (here we consider two-agent systems; however, the model is easily extended to involve multiple agents). An arc (oriented edge) denotes a sequence of two actions.

In our model mental actions have two parameters: agent name and subject (information transmitted, a cause addressed, a reason explained, an object described, etc.). Representing scenarios as graphs, we take into account both parameters. Arc types bear information whether the subject stays the same. Thick arcs link vertices that

correspond to mental actions with the same subject, thin arcs link vertices that correspond to mental actions with different subject.

The curve arcs denote a causal link between the arguments of mental actions, e.g. *service is not as advertised* $\Rightarrow$ *there are particular failures in a service contract, ask* $\sim >$ *confirm*.

Let us consider an example of a scenario and its graph (Figures 1a and 1b).

Note that first two sentences (and the respective subgraph comprising two vertices) are about the current transaction, three sentences after (and the respective subgraph comprising three vertices) address the *unfair charge*, and the last sentence is probably related to both issues above. Hence the vertices of two respective subgraphs are linked with thick arcs (*explain-accept*) and (*remind-deny-disagree*).

In formal conflict scenarios extracted from text there can be multiple mental actions per step, for example *I disagreed . . . and suggested . . .*. The former mental action describes how an agent receives a message (*accept*, *agree*, *reject*, etc.) from an opponent, and the latter one describes the attitude of this agent initiating a request (*suggest*, *explain*, etc.), or reaction to the opponent's action. This division into *passive* (response) mode and *active* (request) mode is represented in the second attribute of mental actions specified in the second column of Table 2. Sometimes, either of the above actions is omitted in textual description of conflicts. Frequently, a mental action, which is assumed but not mentioned explicitly, can be deduced. In this paper for the sake of simplicity we will consider single action per step, performing the comparative analysis of scenarios.

There is a commonsense causal link between 'being charged an unfair fee' and 'intention to have this amount of money back' which is expressed by the arc between *remind* and *disagree*. Semantically, arcs with causal labels between vertices for mental actions express the causal links between the arguments of mental actions rather than between the mental actions themselves.

How would one handle commonsense reasoning patterns in our domain? Commonsense ontology could help; in particular, OpenMind (Singh and Barry 2003), which has a few relevant statements like 'You would bring suit because you were unfairly

(a)

I ***explained*** *that my check bounced (I wrote it after I made a deposit).*
A customer service representative ***accepted*** *that it usually takes some time to process the deposit.*
I ***reminded*** *that I was unfairly charged an overdraft fee a month ago in a similar situation.*
They ***denied*** *that it was unfair because the overdraft fee was disclosed in my account information.*
I ***disagreed*** *with their fee and wanted this fee deposited back to my account.*
They ***explained*** *that nothing can be done at this point and that I need to look into the account rules closer.*

(b)

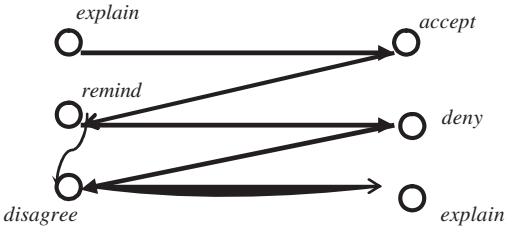

Figure 1. (a) A scenario which includes communicative actions of a proponent and an opponent. (b) The graph for approximated scenario.

wronged by another party'; 'A person doesn't want to be treated unfairly'. However, we need more (complaint-) specific commonsense knowledge to link such statements as 'unfair fee' with 'deposit back'. An ontology which would give us sufficient knowledge is not available and it would be extremely hard and expensive to build for a variety of complaint domains. Therefore, our data structure for machine learning just includes causal links (and not background knowledge). Causal links (Riloff 1996) can be extracted from text; however, to increase the accuracy, we will be using a form instead, where a complainant specifies causal links (Section 8).

One of the most important tasks in assisting negotiations and resolving inter-human conflicts is the 'validity' assessment. On the one hand, a scenario (in particular, a complaint) is 'valid' if it is plausible, internally consistent, and also consistent with available domain-specific knowledge. On the other hand, a complaint scenario is 'invalid' if there are inconsistencies in the communication discourse, so that there is a doubt whether a problem with a product (mentioned in this complaint scenario) has actually occurred. In case of inter-human conflicts or negotiations, such domain-specific knowledge is frequently unavailable. In this paper we demonstrate that a scenario can be assigned to a class valid or invalid based on communicative actions only with the accuracy sufficient for deployment in decision-support systems. In we will show how to relate this scenario (denoted as *U*) to the class of negative (unjustified) complaints.

Why do we relate this scenario to a class of invalid complaints? First of all, having the background knowledge about banking, it is clear that the customer wrongly assumed that the funds become available immediately after a deposit is made. However, it is not really viable to store this information in a generic complaint management system; therefore, we further research into the legitimacy of the observed sequence of communicative actions. 'Being in an attack mode (*reminding*) after a previous attack (*explaining*) was *accepted*' does not look like a co-operative mood. Moreover, continuing to disagree concerning the subject which has just being *denied* (speaking more precisely, a commonsense implication of this subject) is not an adequate negotiation strategy. On the other hand, if a similar scenario (in terms of the structure of communicative actions) has been assigned by a domain expert as invalid, we would want the machine learning system to relate the scenario in Figure 1 to the same class even if there are no explicit reasons.

Hence our analysis of the domain of customer complaints shows that to relate a scenario to a class without domain-specific knowledge, there is a need to analyse a sequence of communicative actions and certain relations between their subjects. Otherwise, one would have to code all relevant domain knowledge which is well-known to be an extremely hard problem and non-feasible for a practical application. Our next step is then to formalise communicative actions in a way suitable for learning scenarios.

3. Semantics of communicative actions

Before we proceed to the formal treatment of communicative actions, we introduce the Theory of Speech Acts which serves as a starting point of our approach to computing similarities between communicative actions. Negotiation, conflict dispute are forms of interactions between human agents. Elements of the language which expresses these interactions are referred to as locutions, speech acts (Bach and Harnish 1979), utterances, or communicative actions (we are going to keep using the last term).

The foundation of the current theory of speech acts was developed by Austin (1962). In his essay 'Performative Utterances' he explores the performative utterances, aiming to prove that when people speak, they are doing more than simply conveying information – they act. A speech act is essentially a theory that asserts the claim that in saying something, we perform something. It is an action that is performed by means of language. An example from the domain of customer complaints would be a performative act of a judge during a hearing when she or he says: 'I now pronounce that the complaint is solved'. Due to Austin's designation of speech acts, sentences like this adopt a notion of action. The judge's sentence is not a report of the action; it *is* the action indeed.

However, every sentence does not take on the same linguistic action. Austin distinguishes between three types of linguistic acts: the act of saying something, what one does in saying it, and what one does by saying it. He labels them *Locutionary*, *Illocutionary*, and *Perlocutionary*, respectively (Figure 2a, Farrell 2006). A locutionary act is simply saying something about the world, e.g. a declarative sentence such as: 'The product does not work.' This sentence is not posing a question, promising, or commanding anything. It simply states something about the world, containing purely propositional content. This type of act is the most basic, and does not require much more explanation.

The illocutionary act includes promising, questioning, admitting, hypothesising, etc. The locutionary act was simply the act of saying something, while the illocutionary act is performed in saying something. For example, 'A company promises to support the product after it is sold' asserts more than simply stating a sentence about the world. It includes an assertion that is performative in nature. Illocutionary acts are very prominent in language, and are frequently in use in complaint scenarios.

The third type of linguistic acts are perlocutionary ones. These are non-conventional sentences that cause a natural condition or state in a person. These acts de-emphasise the actual intentions, and focus on the effects on the hearer. Acts of frightening or convincing depend on the response of another person. If a perlocutionary act is successful, then it seems safe to say that an illocutionary act has successfully taken place.

Austin's speech act theory has been fairly influential since its inception. There have been certain improvements and clarifications made to speech acts that are worth noting; in particular, development upon Austin's insistence such acts cannot perform two different ways (Searle 1979). Searle shows that illocutionary acts can act in two different ways.

As an example from the domain of customer complaints, let us consider the following. By describing a situation of strong dissatisfaction with particular product features (locutionary component) in a writing style that is designed to have the force of a warning (illocutionary component), the complainant may actually frighten the customer support representative into providing a compensation for a faulty product (perlocutionary component). It is important to analyse whether a complainant presents communicative actions of herself and an opponent consistently in terms of these components, which we are going to do by means of machine learning.

Approximating scenarios of multi-agent interactions, we follow along the lines of communicative actions' division into constatives and performatives.

- Constatives describe or report some state of affairs such that it is possible to assess whether they are false or true.
- Performatives, on the other hand, are fortunate or unfortunate, sincere or insincere, realistic or unrealistic, and, finally valid or invalid, which is the focus of the

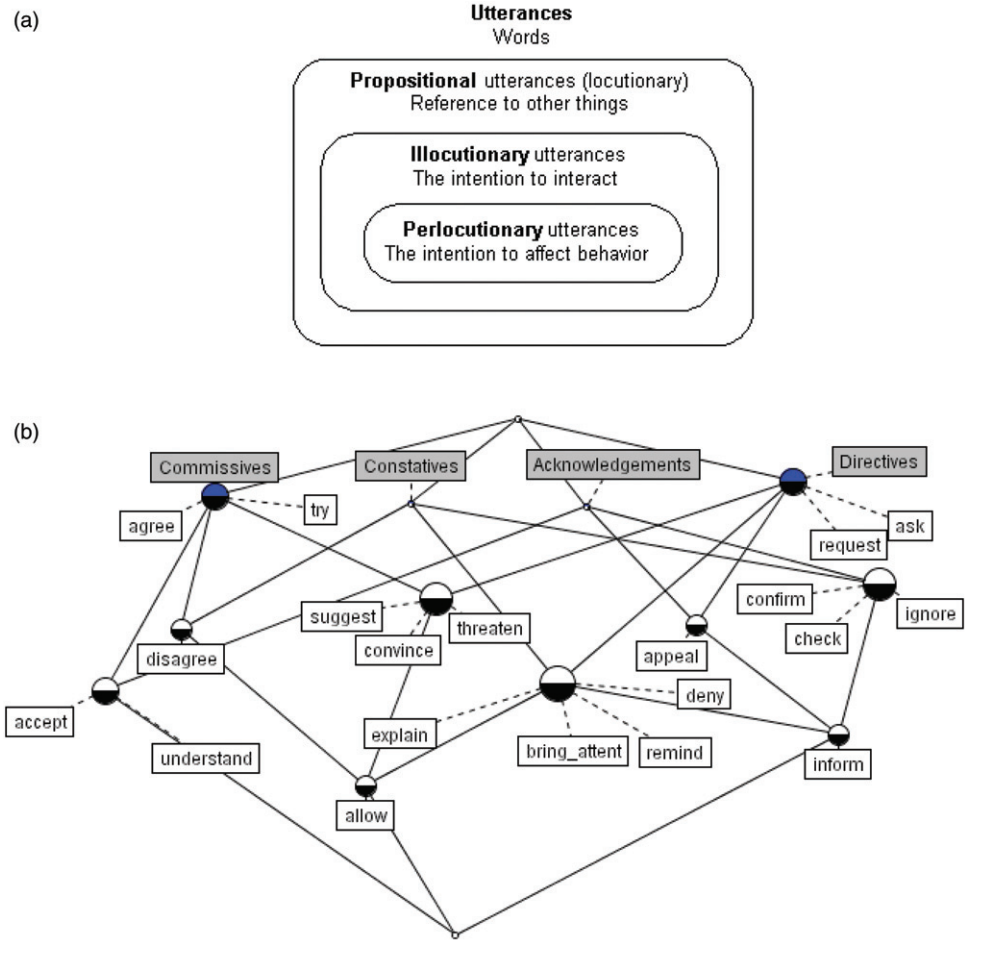

Figure 2. (a) Categories of speech acts. (b) The concept lattice for communicative actions adapting speech act theory to our domain. Each communicative action does not have a unique set of attributes: calculation of similarity might be inadequate.

current study. Performatives address the attitude of the agent performing the linguistic act, including his thoughts, feelings, and intentions.

It turns out that it is much more efficient to automatically analyse the group of performatives than that of constatives, because the former is domain-independent; in case of complaints there is always a lack of information to judge on constatives.

To choose communicative actions to adequately represent an inter-human conflict, we have selected the most frequently used ones from our structured database of complaints (Table 1, Galitsky et al. 2005).

A number of computational approaches have attempted to discover and categorise how the agents' attitudes and communicative actions are related to each other in the case of computational simulation of human agents (Searle 1969; Cohen and Levesque 1990). As we have mentioned above, applying machine learning to the attitudes and

Table 1. The set of communicative actions from a typical complaint.

Agree, explain, suggest, bring company's attention, remind, allow, try, request, understand, inform, confirm ask, check, ignore, convince disagree, appeal, deny, threaten	Agree, explain, suggest, remind, allow, try, request, understand, inform, confirm, ask, check, ignore, convince, disagree, appeal, deny, threaten, bring to customer's attention, accept complaint, accept/deny responsibilities, encourage, cheat

communicative actions, we are primarily concerned with how these approaches can provide a unified and robust framework for finding a similarity between the communicative actions. The theory of speech acts seems to be one of the most promising approaches to categorising mental actions in terms of their roles. Following Bach and Harnish (1979), we consider four categories of illocutionary communicative actions with major representatives 'stating', 'requesting', 'promising' and 'apologising'. Each speech act is related to a single category only in the framework of the speech act theory. For our purpose, each speech act is extracted from text automatically, or is selected from a list by a user as a word, and may belong to multiple categories.

Now we can calculate the similarity between communicative actions as a set (an overlap) of the speech act categories they belong to. To estimate how fruitful the speech act-theoretical approach is for calculating the similarities between communicative actions, we build a concept lattice (Ganter and Wille 1999) for communicative actions as objects and speech act categories as their features (constatives, directives, commissives, and acknowledgements). In the concept lattice, each node is assigned a set of features and a set of objects. For each node, all features assigned to nodes, assessable when navigating the lattice upwards, are satisfied by the objects assigned to this node. In Figure 2, we show either features or objects for each node. For example, let us consider the node assigned with the object *allow*. Navigating the edges up, we access *disagree* and then *commissives* and *constatives* node, triple *suggest-convince-threaten* node and then *commissives* and *directives*, and four-tuple *explain-bring_attention-remind-deny* and then *directives*. Hence the lattice is showing that the object *allow* satisfies three out of four *features*; *commissives*, *directives*, and *constatives* (as we have specified in the Table 2, tinted row).

As the reader can see, this direct speech act-theoretical approach is inadequate for uniform coverage of communicative actions in conflict scenarios. Some communicative actions (e.g. *agree*, *try*) are described by the selected features more accurately, whereas *suggest-convince-threaten* and tuple *explain-bring_attention-remind-deny* cannot be distinguished under this categorisation at all. Hence four features of the speech act theory are insufficient to differentiate between 20 communicative actions which have been evaluated to be a minimal set to express an inter-human conflict (Galitsky et al. 2005). Hence more attributes are needed to be taken into account to find an adequate means to compute similarities (Figure 2b) between communicative actions.

Formalising the semantics of communicative actions, it is also important to take into account the characterisation of their pre- and post mental states in terms of knowledge, belief and intentions, following (Cohen and Levesque 1990). We express the set of selected meanings for each mental action using an expression in the *want-know-believe* basis (Figure 3, Galitsky 2006a), extending the BDI model (Bratman 1987; Wooldridge 2000). Note that clauses may be embedded as arguments for communicative actions (as meta-predicates). We refer the reader to Galitsky (2004) for further details on defining mental

Table 2. Selected attributes of communicative actions, adapting speech act theory to our domain. The attributes for *allow* are highlighted (mentioned in the example below).

Speech acts	Constatives	Directives	Commissives	Acknowledgements
agree	0	0	1	0
accept	0	0	1	1
explain	1	1	0	0
suggest	0	1	1	0
bring_attent	1	1	0	0
remind	1	1	0	0
allow	1	1	1	0
try	0	0	1	0
request	0	1	0	0
understand	0	0	1	1
inform	1	1	0	1
confirm	1	0	0	1
ask	0	1	0	0
check	1	0	0	1
ignore	1	0	0	1
convince	0	1	1	0
disagree	1	0	1	0
appeal	0	1	0	1
deny	1	1	0	0
threaten	0	1	1	0

actions and mental states in the above basis. For example, various meanings of mental action *inform* are expressed in Figure 3a.

These definitions for communicative actions are useful if one intends to express the similarities between them as their pre-and post-conditions (Figure 3b). However, the above clauses cannot be represented using attributes without giving up most of the details, and in the current study we will not take into account this kind of semantic differences between communicative actions.

We proceed to the solution which turned out to be most robust and plausible. To extend the speech act-based means of expressing similarity between communicative actions, we introduce five attributes each of which reflects a particular semantic parameter for communicative activity (Table 3):

- *Positive/negative attitude* expresses whether a communicative action is a cooperative (friendly, helpful) move (1), uncooperative (unfriendly, unhelpful) move (−1), neither or both (hard to tell, 0).
- *Request/respond mode* specifies whether a communicative action is expected to be followed by a reaction (1), constitutes a response (follows) a previous request, neither or both (hard to tell, 0).
- *Info supply/no info supply* tells if a communicative action brings in an additional data about the conflict (1), does not bring any information (−1), 0; does not occur here.
- *High/low confidence* specifies the confidence of the preceding mental state so that a particular communicative action is chosen, high knowledge/confidence (1), lack of knowledge/confidence (−1), neither or both is possible (0).

(a)

```
inform(Who, Whom, What ) :- want(Who, know(Whom, What)),
   believe(Who, not know(Whom, What)),
   believe(Who, want(Whom, know(Whom, What))). The most general definition.
inform(Who, Whom, What ) :- believe(Who, know(Whom, What)),
want(Who, believe(Whom, know(Who,What))). To inform Whom that not only Whom but Who knows What.
inform(Who, Whom, What) :- ask(Whom, Who, What),
     want(Who, know(Whom, What)). Informing as answering.
inform(Who, Whom, What) :- ask(SomeOne, Who, believe(Whom, What)),
      want(Who, know(Whom, What). Following SomeOne's request for informing.
```

(b)

```
disagree(A,B,W) :- inform(A,B,W), not believe(B,W), inform(B,A, not W).
A informs B about W, which B does not believe (trust), and informs A (somehow) about it
agree(A,B,W) :- inform(A,B,W), believe(B,W), inform(B,A,W).
A informs B about W, which B does believe, and informs A about it
 explain(A,B, W) :- believe(A, (W :- V)), not know(B,W), inform(A,B,V),
            inform(A,B,(W :- V)), believe(B,W).
A believes that W is implied by V, B does not know W, then B is informed about V, and about the fact that W is
implied by V, and then believes that W.
confirm(A,B,W) :- inform(A,B,W), know(A, believe(B,W)), inform(A,B, know(A, believe(B,W))).
A informs B about W, A knows that B believes in W, and informs B about this knowledge
bring_attention(A,B,W) :- want(A, believe(B, know(A, W))), inform(A, want(A, believe(B, know(A, W))).
A intends B to believe that A knows W, and A informs B about his intention
remind(A,B,W) :- believe(A, believe(B,W)),
            inform(A,B,W), want(A, know(B, know(A,W))).
A believes that B believes W, but still informs B about W, with intention that B knows that A believes B
understand(A,W) :- inform(B,A,W), believe(B, believe(A,V)), believe(B, not believe(A, (W :- V))),
  want(B, believe(A, (W :- V))), inform(B,A,(W :- V)), believe(A,(W :- V),
            believe(A, W).
B informs A about W, B believes that A believes that V, B believes that A does not possess the implication of
W by V, B wants A to believe in this implication, informs A about it, so that A now possesses this implication
and finally A believes that W.
acceptResp(A,W) :- want(B, not W), believe(B, (W :-do(A,W1))),
        want(A, know(B, believe(A, (W :-do(A,W1)) ))), inform(A,B, (W :-do(A,W1)) ).
```

Figure 3. (a) The clauses for various meanings of the entity *inform*. (b) The clauses for the selected mental entities from Table 2.

- *Intense/relaxed mode* says about the potential emotional load: high (1), low (−1), neutral (0) emotional loads are possible.

Note that out of the set of meanings for each communicative action, we merge its subset into a single meaning, similar to what was done in Figure 3b. This merge is performed, taking into account relations between the meanings of the given communicative actions and those of the other ones (Galitsky 2004).

To represent the hierarchy of communicative actions by a concept lattice, we scale nominally the first and second attributes (i.e. the attribute values −1, 0, and 1 are considered as completely dissimilar). The third, fourth, and fifth attributes are already two-valued. Thus, the scaled context has seven attributes and the resulting concept lattice is presented in Figure 4. *ConExp* (Yevtushenko 2005) software was used to construct and visualise the concept lattice (Ganter and Wille 1999) of communicative actions (as objects) and their attributes.

The concept lattice illustrates the semantics of communicative actions; it shows how the choice of attribute-based expressions covers the totality of possible meanings in the knowledge domain of interaction between human agents.

A number of knowledge representation languages have been proposed for multi-agent communication protocols, including KQML (Knowledge Query and Manipulation Language), a language and protocol for knowledge and information exchange

Table 3. Augmented attributes of communicative actions.

	Attributes				
Communicative action	Positive/negative attitude	Request/respond mode	Info supply/no info supply	High/low confidence	Intense/relaxed mode
agree	1	−1	−1	1	−1
accept	1	−1	−1	1	1
explain	0	−1	1	1	−1
suggest	1	0	1	−1	−1
bring_attention	1	1	1	1	1
remind	−1	0	1	1	1
allow	1	−1	−1	−1	−1
try	1	0	−1	−1	−1
request	0	1	−1	1	1
understand	0	−1	−1	1	−1
inform	0	0	1	1	−1
confirm	0	−1	1	1	1
ask	0	1	−1	−1	−1
check	−1	1	−1	−1	1
ignore	−1	−1	−1	−1	1
convince	0	1	1	1	−1
disagree	−1	−1	−1	1	−1
appeal	−1	1	1	1	1
deny	−1	−1	−1	1	1
threaten	−1	1	−1	1	1

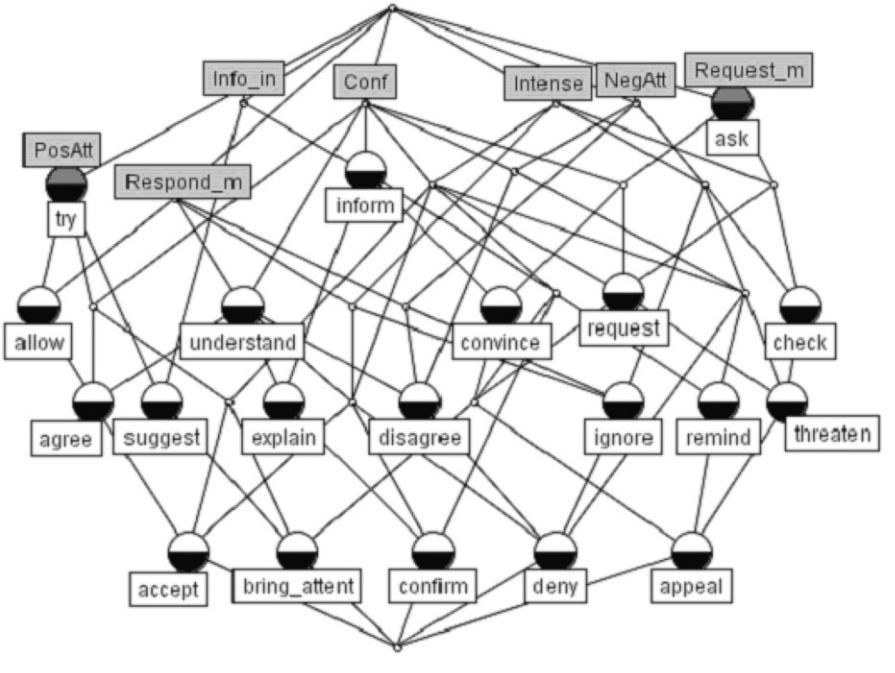

Figure 4. The concept lattice for communicative actions. Each communicative action has a unique set of attributes.

(Finin, Fritzson, McKay and McEntire 1994; Hendler and McGuinness 2000), and FIPA ACL, the language that allows agents to communicate via messages. KQML was developed within the ARPA Knowledge Sharing Effort aimed at developing techniques and methodology for building large-scale sharable and reusable knowledge bases in the early 1990s. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents. KQML can be used as a language for an application program to interact with an intelligent system or for two or more intelligent systems to share knowledge in support of co-operative problem solving. KQML focuses on an extensible set of performatives, which defines the permissible operations that agents may attempt on each other's knowledge and goal stores. The performatives comprise a substrate on which to develop higher-level models of inter-agent interaction such as contract nets and negotiation.

The intention of FIPA ACL (2006) is to provide conversational logic to agents, attempting to improve the semantic level of agent communication. In order to achieve this, each of the FIPA ACL communication primitives is given a precise semantics by providing pre- and post-conditions, expressed in a first-order modal logic. With such semantics, the agent is able to express his personal attitude (e.g. belief, uncertainty, choice, intention) towards its achieved knowledge. Based on this underlying semantic model, the agent can compile sensible options for its next move. An alternative approach for setting up intelligent conversations is to identify certain repeatedly used conversation patterns called 'interaction protocols' by examining typical software agent application areas, e.g. an auction. In our companion work we extend the formal semantics of communicative actions along the lines of Figure 3b, closely following their natural language representation (Galitsky 2006a). Furthermore, in Galitsky (2006b) we apply reasoning to interactive protocols and observe how repetitive/robust they are for the purposes of prediction of scenario outcomes.

Regrettably, for the purpose of the current study, neither KQML nor FIPA ACL are well-suited to formally represent communication between human agents, which require a rather agile treatment of communication discourse and natural language ambiguities (as we demonstrated above in Figures 3a and 3b). Moreover, mental states, associated with inter-human communication (in particular, while resolving conflicts), are much more complex than mental states of software agents. Hence instead of building an analogue of KQML for human agents (which we have attempted in Galitsky 2006a), in this study we focus on the set of features of communicative actions which are crucial for machine learning.

Before we proceed to the formal model of scenarios in terms of graphs, we define a conflict scenario as a sequence of communicative actions, each of which is a reaction to the previous communicative actions of opponents. This reaction is constrained by interaction protocols by means of enumeration of valid scenarios where this protocol is assumed to be correct. Multi-agent conflict is a scenario where agents have inconsistent intentions (about states):

$$want\,(Agent\ For, State), want\,(Agent\ \ Against, not\ State).$$

In a complaint scenario, we distinguish two selected states; one follows another. *The pre-conflict state* includes the deviation of the expected (quality) from the actual (quality) of the received product or service:

$$\text{pre} - \text{complaint} : expect\,(Customer, feature\,(Product)),$$
$$believe\,(Customer, not\ feature\,(Product)).$$

To grow into a complaint, initial dissatisfaction must be fed with the further desperation connected with an interaction with customer support representatives and other opponents. Let us consider a generic example of a complaint scenario, presented as a sequence of mental states rather than communicative actions.

ask(*Customer, fix*(*Customer Support, not feature*(*Product*))),
customer wants the product to be fixed in some way to meet her expectations
not fix(*Customer Support, not feature*(*Product*)),
customer support/company cannot/does not do it
believe(*Customer*, (*not fix*(*Customer Support, not feature*(*Product*)):- *Reason*)),
customer believes that there is a certain *Reason* for not doing it
not know(*Customer, Reason*).
but customer does not know what is the *Reason*; *not* here has the linguistic meaning (not the 'negation as failure' meaning)
upset(*Customer*).

Note that there is a wide variety of ways a customer expresses his/her dissatisfaction initially, the customer support responds to his/her dissatisfaction, etc. However, the common step in the majority of complaint scenarios is *Customers*' confusion over why they were treated in such a way, the *Reason* that the products was not fixed, and over other issues.

The scenario is defined as a sequence of communicative actions. Usually, if the sequence of communicative actions of customer support is 'adequate', a complaint does not arise. Therefore we define a typical complaint as a scenario with the following conditions:

- a conflict of intentions concerning the physical state of a product/service (pre-complaint), and
- a conflict of intentions concerning the mental and physical actions of customer support and resultant state of their satisfaction.

Indeed, these conditions are the ones for the subjects of communicative actions. The conflict is defined as a logical inconsistency. Our definition of complaint scenario includes inconsistencies in both mental and physical spaces. In the section to follow, complaint scenario will be defined formally as a graph.

4. Defining scenarios as graphs

We proceed with the description of our scenario dataset. This dataset contains two sets of complaint scenarios: showing a good attitude of a complainant (consistent plot with proper argumentation, a valid complaint) on the left, and a bad attitude of a complainant (inconsistent plot with certain flaws, implausible or irrational scenarios, an invalid complaint) on the right (Figure 5).

Each scenario includes 2-6 interaction *steps*, each consisting of communicative actions with the alternating first attribute {*request – respond – additional request or other follow up*}. A step comprises one or more consequent actions with the same subject. Within a step, vertices for communicative actions with common argument are linked with thick arcs.

For example, *suggest* from scenario V2 (Figure 5) is linked by a thin arc to communicative action *ignore*, whose argument is not logically linked to the argument of *suggest* (the subject of suggestion). The first step of V2 includes *ignore-deny-ignore-threaten*; these communicative actions have the same subject (it is not specified in the graph of conflict scenario). The vertices of these communicative actions with the same argument are linked by the *thick* arcs. For example, it could be ***ignored*** *refund because of a wrong mailing address,* ***deny*** *the reason that the refund has been ignored* [*because of a wrong mailing address*], ***ignore*** *the denial* [*. . . concerning a wrong mailing address*], *and* ***threatening*** *for that ignorant behaviour* [*. . . concerning a wrong mailing address*]. We have *wrong mailing address* as the common subject S of communicative actions *ignore-deny-ignore-threaten* which we approximate as

$$ignore(A1, S)\ \&\ deny(A2, S)\ \&\ ignore(A1, S)\ \&\ threaten(A2, S),\ keeping\ in\ mind\ the\ scenario\ graph.$$

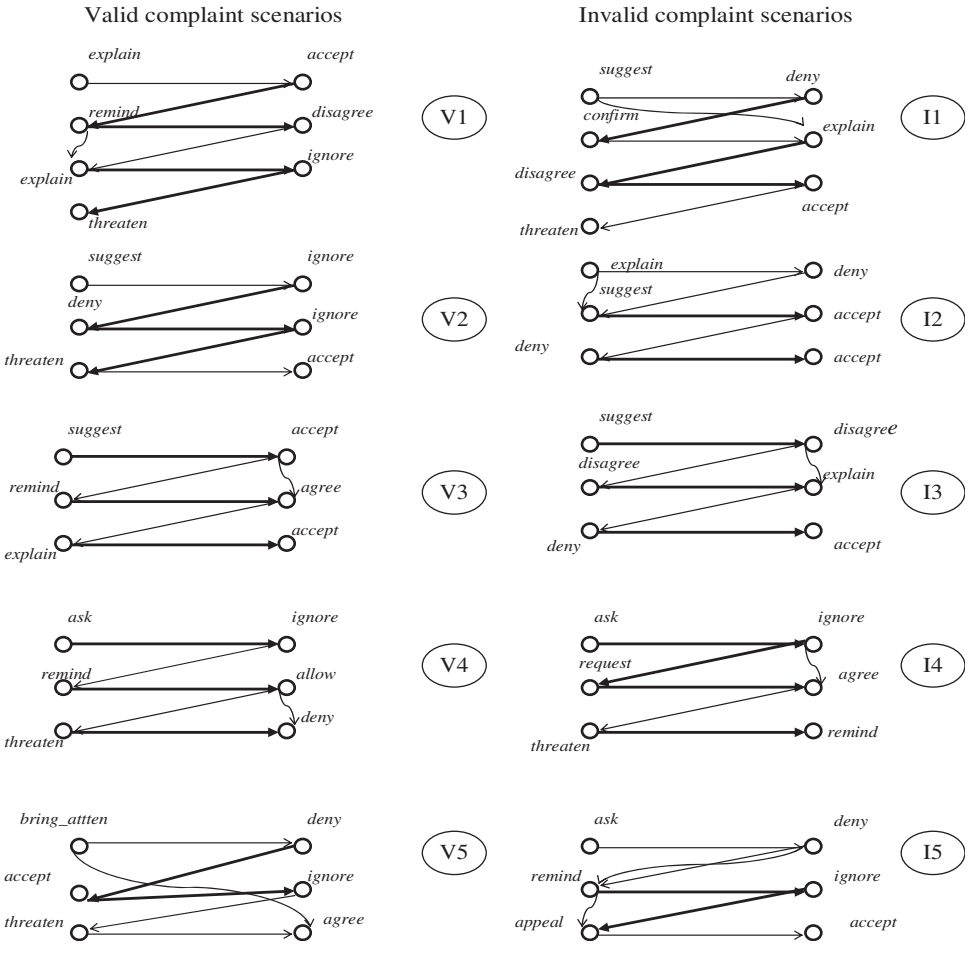

Figure 5. The training set of scenarios.

In such approximation we write *deny*(*A2, S*) for the fact that *A2 denied the reason that the refund has been ignored because of S*. Indeed, *ignore*(*A1, S*) & *deny*(*A2, S*) & *ignore*(*A1, S*) & *threaten*(*A2, S*). Without a scenario graph, the best representation of the above in our language would be

ignore(*A1, S*) & *deny*(*A2, ignore*(*A1, S*)) & *ignore*(*A1, deny*(*A2, ignore*(*A1, S*)))
& *threaten*(*A2, ignore*(*A1, deny*(*A2, ignore*(*A1, S*)))).

Let us enumerate the constraints for the scenario graph:

1. All vertices are fully ordered by the temporal sequence (earlier-later).
2. Each vertex has a special label relating it either to the proponent (drawn on the left side in Figure 1b) or to the opponent (drawn on the right side).
3. Vertices denote actions either of the proponent or of the opponent.
4. The arcs of the graph are oriented from earlier vertices to later ones.
5. Thin and thick arcs point from a vertex to the subsequent one in the temporal sequence (from the proponent to the opponent or vice versa).
6. Curly arcs, staying for causal links, can jump over several vertices.

Hence, we obtained the formal definition of a conflict scenario as a graph.

Similarity between scenarios is defined by means of maximal common sub-scenarios. Since we describe scenarios by means of labelled graphs, first we consider formal definitions of labelled graphs and domination relation on them (see, for example, Ganter and Kuznetsov 2001).

We have an ordered set G of graphs (V, E) with vertex- and edge-labels from the sets $(\mathcal{L}_V, \leq)$ and $(\mathcal{L}_\mathcal{E}, \leq)$. A labelled graph Γ from G is a quadruple of the form $((V, l), (E, b))$, where V is a set of vertices, E is a set of edges, $l\colon V \to \mathcal{L}_V$, is a function assigning labels to vertices, and $b\colon E \to \mathcal{L}_\mathcal{E}$, is a function assigning labels to edges. We do not distinguish isomorphic graphs with identical labelling.

The order is defined as follows: For two graphs $\Gamma_1 := ((V_1, l_1), (E_1, b_1))$ and $\Gamma_2 := ((V_2, l_2), (E_2, b_2))$ from G we say that Γ_1 **dominates** Γ_2 or $\Gamma_2 \leq \Gamma_1$ (or Γ_2 is a **subgraph** of Γ_1) if there exists a one-to-one mapping $\varphi\colon V_2 \to V_1$ such that it

- respects edges: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
- fits under labels: $l_2(v) \leq l_1(\varphi(v))$, $(v, w) \in E_2 \Rightarrow b_2(v, w) \leq b_1(\varphi(v), \varphi(w))$.

Note that this definition allows generalisation ('weakening') of labels of matched vertices when passing from the 'larger' graph G_1 to 'smaller' graph G_2.

Now, generalisation Z of a pair of scenario graphs X and Y (or their similarity), denoted by $X \sqcap Y = Z$, is the set of all inclusion-maximal common subgraphs of X and Y, each of them satisfying the following additional conditions:

- To be matched, two vertices from graphs X and Y must denote communicative actions of the same agent.
- Each common subgraph from Z contains at least one thick arc.

This definition is easily extended to finding generalisations of several graphs (e.g. see Kuznetsov 1999; Ganter and Kuznetsov 2001). The subsumption order μ on pairs of graph sets X and Y is naturally defined as $X \sqsubseteq Y := X \sqcap Y = X$.

Computing relation $\Gamma_2 \leq \Gamma_1$ for arbitrary graphs Γ_2 and Γ_1 is an NP-complete problem (since it is a generalisation of the subgraph isomorphism problem from Garey and Johnson 1979). Finding $X \sqcap Y = Z$ for arbitrary X, Y, and Z is generally an NP-hard problem.

In Ganter and Kuznetsov (2001) a method based on so-called projections was proposed, which allows one to establish a trade-off between accuracy of representation by labelled graphs and complexity of computations with them. In particular, for a fixed size of projections, the worst-case time complexity of computing operation and testing relation $\leq$ becomes constant. Application of projections was tested in various experiments with chemical (molecular) graphs (Kuznetsov and Samokhin 2005) and conflict graphs (Galitsky, Kuznetsov and Samokhin 2005).

After scaling the many-valued context of communicative actions, descriptions of communicative action are given by 9-tuples of attributes, ordered in the usual way. Thus, vertex labels of generalisations of scenario graphs are given by intents of the scaled context of communicative actions (see Figure 4).

If the conditions above cannot be met then the common subgraph does not exist.

5. Nearest Neighbour classification

Here we propose two schemes for classifying scenarios, given examples from positive and negative classes (see an example of a training sample in Figure 5).

The following conditions hold when a scenario graph U is assigned to a class (we consider positive classification, i.e. to valid complaints, the classification to invalid complaints is made similarly):

1. U is similar to (has a non-empty common scenario subgraph of) a positive example R^+.
2. For any negative example R^-, if U is similar to R^- (i.e. $U \sqcap R^- \neq \emptyset$) then $U \sqcap R^- \sqsubseteq U \sqcap R^+$. This condition introduces the measure of similarity and says that to be assigned to a class, the similarity between the unknown graph U and the closest scenario from the positive class should be higher than the similarity between U and each negative example (i.e. representative of the class of invalid complaints).

Condition (2) implies that there is a positive example R^+ such that for no R^- one has $U \sqcap R^+ \sqsubseteq R^-$, i.e. there is no counterexample to this generalisation of positive examples.

Let us now proceed to the example of a particular U in Figure 6 on the top. The task is to determine whether U belongs to the class of valid complaints (on the left of Figure 4) or to the classes of invalid complaints (on the right); these classes are mutually exclusive.

We observe that V_4 is the graph of the highest similarity with U among all graphs from the set $\{V_1, \ldots, V_5\}$ and find the common subscenario $U \sqcap \mathrm{V}_4$. Its only thick arc is derived from the thick arc between vertices with labels *remind* and *deny* of U and the thick arc between vertices with labels *remind* and *allow* of V_4. The first vertex of this thick arc of $U \sqcap V_4$ is *remind* $\wedge$ *remind* = *remind*, the second is *allow* $\wedge$ *deny* = $\langle 0\ 0\ 0\ 1\ 0\ 0\ 0 \rangle$ ($U \sqcap V_4$ is calculated at the left bottom). Other arcs of $U \sqcap V_4$ are as follows: that from the vertex with the label *remind* to the vertex with the label $\langle 0\ 0\ 0\ 1\ 0\ 0\ 0 \rangle$; the arc from the vertex with the label $\langle 0\ 0\ 0\ 1\ 0\ 0\ 1 \rangle$ to the vertex with the label *remind*; the arc from the vertex with the label $\langle 0\ 0\ 0\ 1\ 0\ 0\ 0 \rangle$ the vertex with the label $\langle 0\ 1\ 0\ 0\ 0\ 1\ 0 \rangle$. These arcs are thin, unless both respective arcs of $U \sqcap V_4$ are thick (the latter is not the case here). Naturally, common subscenarios may contain multiple steps, each of them may result in the satisfaction of conditions (1) and (2) for the class assignment above.

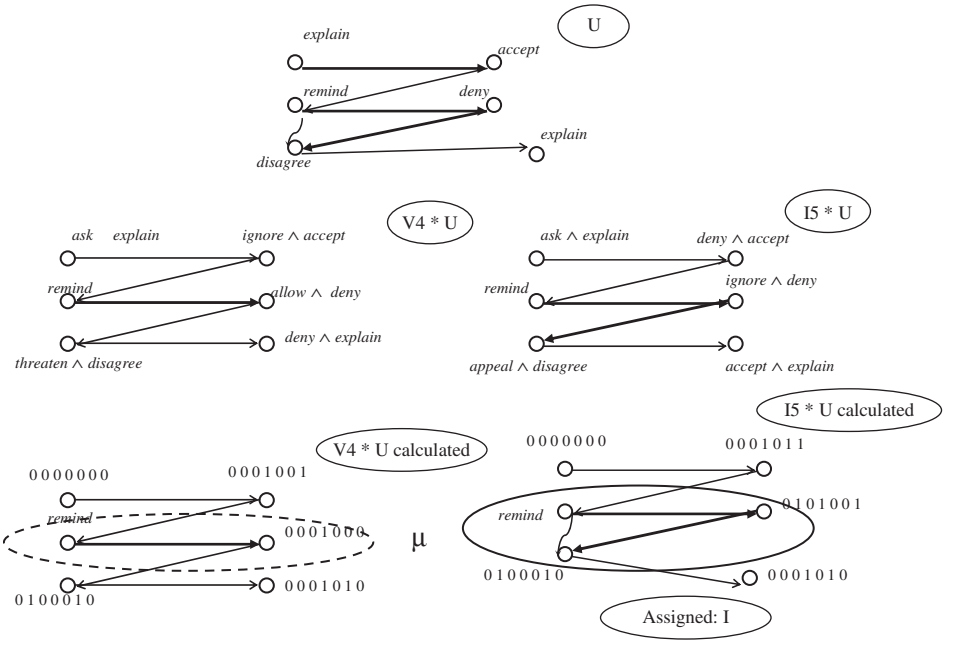

Figure 6. A scenario with unassigned complaint status and the procedure of relating this scenario to a class.

Similarly, we build the common sub-scenario $U \sqcap I_5$; I_5 delivers the largest subgraph (two thick arcs) in comparison with I_1, I_2, I_3, I_4. Moreover, $U \sqcap V_4 \sqsubseteq U \sqcap I_5$, this inclusion is highlighted by the ovals around the steps. Condition 2 is satisfied. Therefore, U is an invalid complaint as having the highest similarity to invalid complaint I_5.

Let us estimate the worst-case time complexity of *Nearest Neighbour* classification of this sort. Denote by A the worst-case time complexity of computing operation $\sqcap$ and denote by B the worst-case complexity of testing relation $\leq$ (complexities A and B were discussed in Section 4). By $|+|$ and $|-|$ we denote the numbers of positive and negative examples, respectively. First we estimate the time complexity of trying to classify a scenario positively. We need to match the scenario with a positive example (complexity A) and test whether the result is dominated by any of negative examples (complexity $B\ |+|$). To do this for every positive example, we obtain complexity $|+|$ $(A + B|-|)$. Similar for complexity of trying to classify a scenario negatively, we obtain $|-|\ (A + B\ |+|)$. Thus, the total time complexity of *Nearest Neighbour* classification does not exceed $A(|+|+|-|) + B\ |-|\ |+|$.

In Ganter and Kuznetsov (1999) and Kuznetsov (1999) we considered a learning model from Finn (1991) formulated in FCA terms. As applied to scenarios, this model is described as follows. Given similarity (meet) operation on $\sqcap$ pairs of scenarios that defines a semi-lattice, sets of positive and negative examples, a (+)-hypothesis is defined as similarity of several positive examples which does not cover any negative example (for the lack of space we refer to the above publications for exact definitions). (−)-hypotheses are defined similarly. Now an undetermined scenario is classified positively, if it contains (in terms of $\sqsubseteq$) a positive hypothesis and does not contain any negative hypothesis.

Having shown how a scenario can be related to class using Nearest Neighbour, we proceed to more cautious classification framework which minimises false negatives: it would rather refuse to classify than provide a borderline classification. This feature is crucial for the conflict resolution domain where a solution offered to the parties must have an unambiguous and concise explanation and background. Moreover, an approach to finding similarities between scenarios which is more sensitive to peculiarities of communicative actions and conflict scenarios would deliver higher classification accuracy in our domain. In the following section we present a logic programming-based machine learning framework *Jasmine* adjusted to operate with conflict scenarios.

6. Jasmine reasoning schema

Jasmine is based on a learning model called JSM-method (in honour of John Stuart Mill, the English philosopher who proposed schemes of inductive reasoning in the nineteenth century). JSM-method to be presented in this section implements Mill's idea that similar effects are likely to follow common causes (Mill 1843).

The *Jasmine* framework consists of features (communicative actions), objects (scenarios) and targets (features to be predicted: classes of scenarios). Within a first-order language, objects are atoms; features and effects (targets) are terms which include these atoms. For a target, there are four groups of objects with respect to the evidence they provide for this target: *Positive – Negative – Inconsistent – Unknown*.

An inference to obtain a target feature (satisfied or not) can be represented as one in a respective four-valued logic. The predictive machinery is based on building hypotheses, $target(S)$:- $feature_1(S, \ldots), \ldots, feature_n(S, \ldots)$, that separate scenarios S, where *target* is to be predicted, and $features_1, \ldots, feature_n \in features$ are the mental actions/attitudes which are considered causes of the target.

Desired separation is based on the similarity of objects in terms of features they satisfy. Usually, such similarity is domain-dependent. However, building the general framework of inductive-based prediction, we use the anti-unification of formulas that express the totality of features of the given and other objects (our features do not have to be unary predicates; they are expressed by arbitrary first- or second-order terms).

JSM-prediction is based on the notion of similarity between objects. Similarity between a pair of objects is a hypothetical object which obeys the common features of this pair of objects. In handling similarity JSM is close to formal concept analysis (Ganter and Wille 1999), where similarity is the meet operation of a lattice (called concept lattice). In this work we choose anti-unification of formulas expressing features of the pair of objects to derive a formula for similarity sub-object (Finn 1991). Anti-unification, in the finite term case, was studied as the least upper bound operation in a lattice of terms. Below we will be using the predicate *similar*(*Object*1*, Object*2*, CommonSubObject*) which yields the third argument given the first and the second arguments and provide its definition in Section 7.

We start with an abstract example of *Jasmine* setting, based on attitudes of candidates for a business partner. Our introductory example of JSM settings for unary predicate is as follows in Figure 7 (from now on we use the conventional PROLOG notations for variables and constants).

The problem is formulated as follows: building a rule of acceptance/rejection candidates for a position based on their five features. An expert human resource agent has made the decisions in five cased below, and *Jasmine*'s task is to resolve other possible cases.

```
features([
i, %intelligent
e, %easy makes connections
c, %critiques others
r, %risky
s, %straight-forward and honest% ]).
objects([o1, o2, o3, o4, o5, o6, o7, o8]).
targets([good_fit]).
     %% Beginning of knowledge base
i(o1). e(o1). c(o1).                 good_fit(o1).
i(o2). e(o2). c(o2).        s(o2). good_fit(o2).
i(o3).                r(o3).        good_fit(o3).
i(o4). e(o4).                s(o4). good_fit(o4).
i(o5). e(o5). c(o5). r(o5).
i(o6).        c(o6). r(o6).
              c(o7). r(o7).
     %% Prediction settings
c(o8). r(o8). s(o8). unknown(good_fit(o8)).
```

Figure 7. A sample knowledge base for mining attitudes and forming criteria for the optimal candidate (good_fit).

We start our presentation of reasoning procedure with the chart (Figure 8), followed by the logic program representation. Let us build a framework for predicting the target feature V of objects set by the formulas X expressing their features: *unknown*(X, V). We are going to predict whether $V(x_1, \ldots, x_n)$ holds or not, where $x_1, \ldots, x_n$ are variables of the formulas X (in our example, $X = [i(\mathrm{o1}), e(\mathrm{o1}), c(\mathrm{o1})], x_1 = \mathrm{o1}$).

We start with the raw data, positive and negative examples, *rawPos*(X, V) and *rawNeg*(X, V), for the target V, where X range over formulas expressing features of objects. We form the totality of intersections for these examples (positive ones, U, that satisfy *iPos*(U, V), and negative ones, W, that satisfy *iNeg*(W, V), not shown):

$$\begin{aligned} &iPos(U, V) : -rawPos(X1, V), rawPos(X2, V), X1 \setminus = X2, similar(X1, X2, U), U \setminus = [\,]. \\ &iPos(U, V) : -iPos(U1, V), rawPos(X1, V), similar(X1, U1, U), U \setminus = [\,]. \end{aligned} \quad (1)$$

Above are the recursive definitions of the intersections. As the logic program clauses which actually construct the lattice for the totality of intersections for positive and negative examples, we introduce the third argument to accumulate the currently obtained intersections (the negative case is analogous):

$$\begin{aligned} &iPos(U, V) : -iPos(U, V, \_). \\ &iPos(U, V, Accums) : -rawPos(X1, V), rawPos(X2, V), X1 \setminus = X2, similar(X1, X2, U), \\ &\qquad Accums = [X1, X2], U \setminus = [\,]. \end{aligned}$$

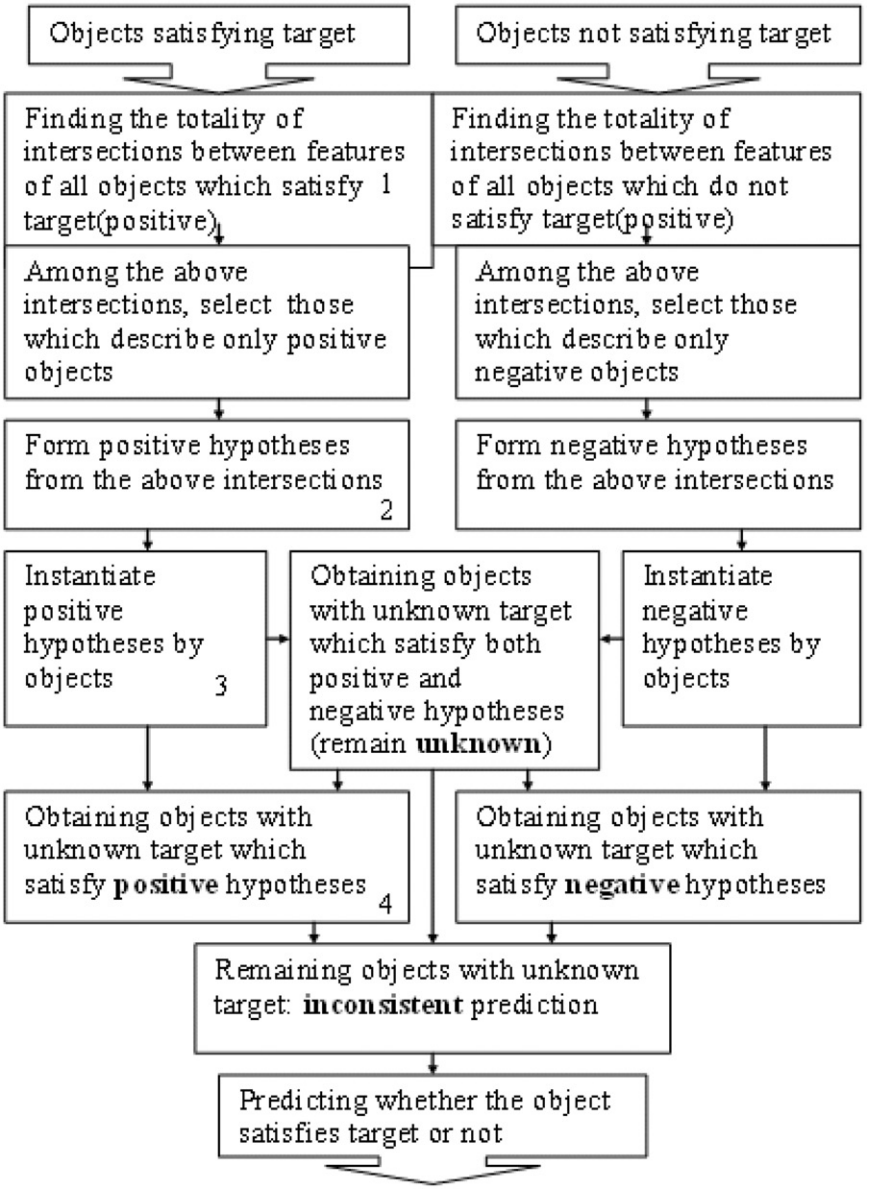

Figure 8. The chart for reasoning procedure of *Jasmine*. Note that the prediction schema is oriented to discover which mental actions/attitudes cause the target and how they do it, rather than just searching for common features for the target which would be much simpler (six units on the top). The respective clauses (1–4) are spelled out below, and sample results for each numbered unit (1–4) are presented in Figure 9.

$$iPos(U, V, Accums\ X1) : -iPos(U1, V, Accums), !, rawPos(X1, V), \\ not\ member(X1, Accums),\ similar(X1, U1, U),\ U \backslash = [], \\ append(Accums, [X1], AccumsX1).$$

To obtain the actual positive *posHyp* and negative *negHyp* hypotheses from the intersections derived above, we filter out the inconsistent hypotheses which belong to both positive and negative intersections *inconsHyp*(*U, V*):

$$\begin{aligned} &inconsHyp(U, V) : -iPos(U, V), iNeg(U, V). \\ &posHyp(U, V) : -iPos(U, V), not\ inconsHyp(U, V). \\ &negHyp(U, V) : -iNeg(U, V), not\ inconsHyp(U, V). \end{aligned} \tag{2}$$

Here *U* is the formula expressing the features of objects. It serves as a body of clauses for hypotheses *V*:- *U*.

The following clauses deliver the totality of objects so that the features expressed by the hypotheses are included in the features of these objects. We derive positive and negative hypotheses *reprObjectsPos*(*X, V*) and *reprObjectsNeg*(*X, V*) where X is instantiated with objects where V is positive and negative respectively. The last clause (with the head *reprObjectsIncons*(*X, V*)) implements the search for the objects to be predicted so that the features expressed by both the positive and negative hypotheses are included in the features of these objects.

$$\begin{aligned} &reprObjectsPos(X, V) : -rawPos(X, V), posHyp(U, V), similar(X, U, U). \\ &reprObjectsNeg(X, V) : -rawNeg(X, V), negHyp(U, V), similar(X, U, U). \\ &reprObjectsIncons(X, V) : -unknown(X, V), posHyp(U1, V), negHyp(U2, V), \\ &\qquad similar(X, U1, U1), similar(X, U2, U2). \end{aligned} \tag{3}$$

Finally, we approach the clauses for prediction. Two clauses above (top and middle) do not participate in prediction directly; their role is to indicate which objects deliver what kind of prediction. For the objects with unknown targets, the system predicts that they either satisfy these targets, do not satisfy these targets, or that the fact of satisfaction is inconsistent with the raw facts. To deliver *V*, a positive hypothesis has to be found so that a set of features *X* of an object has to include the features expressed by this hypothesis and X is not from *reprObjectsIncons*(*X, V*). To deliver $\neg V$, a negative hypothesis has to be found so that a set of features *X* of an object has to include the features expressed by this hypothesis and *X* is not from *reprObjectsIncons*(*X, V*). No prediction can be made for the objects with features expressed by *X* from the third clause, *predictIncons*(*X,V*).

$$\begin{aligned} &\boldsymbol{predictPos(X, V) : -unknown(X, V), posHyp(U, V), similar(X, U, U),} \\ &\qquad \boldsymbol{not\ reprObjectsIncons(X, V).} \\ &predictNeg(X, V) : -unknown(X, V), negHyp(U, V), similar(X, U, U), \\ &\qquad not\ reprObjectsIncons(X, V). \\ &predictIncons(X, V) : -unknown(X, V), not\ predictPos(X, V), \\ &\qquad not\ predictNeg(X, V), not\ reprObjectsIncons(X, V). \end{aligned} \tag{4}$$

The first clause above (shown in bold) will serve as an entry point to predict (choose) an effect of given features from the generated list of possible effects that can be obtained for the current state. The clause below is an entry point to *Jasmine* if it is integrated with other applications and/or reasoning components

$$predict\_effect\_by\_learning(EffectToBePredicted, S) : -$$
$$findAllPossibleEffects\ (S, As),\ loadRequiredSamples(As),$$
$$member(EffectToBePedicted, As),\ predictPos(X, EffectToBePredicted), !, X \backslash = [].$$

Predicate *loadRequiredSamples*(*As*) above forms the training dataset. If for a given dataset a prediction is inconsistent, it is worth eliminating the cases from the dataset which deliver this inconsistency. Conversely, if there are an insufficient number of positive or negative cases, additional ones are included in the dataset. A number of iterations may be required to obtain a prediction, however the iteration procedure is deterministic: the source of inconsistency/insufficient data cases are explicitly indicated at the step where predicates *reprObjectsPos* and *reprObjectsNeg* introduced above are satisfied.

For example, for the knowledge base above, we have the protocol and results in Figure 9.

Hence good_fit(o8)) does not hold: candidate o8 is not so good.

Now we show how to classify the totality of candidates' features into 'good fit', 'not a good fit' or inconsistent prediction. Table 4 presents an exhaustive solution to the problem.

Although we use unary predicates (communicative actions) as our features in our examples, the problem of scenario classification cannot be reduced to attribute value learning. Besides a non-trivial approach to finding common sub-scenarios (other than set-theoretic intersection, as we have shown in Section 5) a feature of a scenario may include relationships between scenarios in *Jasmine* prediction settings.

Towards the end of this section we enumerate the predicates we have used in our presentation of *Jasmine* reasoning (Table 5). As to the above variables, X, U, $U1$, $U2$ range over formulas containing objects, and V ranges over targets.

7. Finding associative similarity between scenarios as ordered communicative actions

Naturally, a quality of scenario classification and, in particular *Jasmine*-based prediction, is dramatically dependent on how the similarity between scenarios is defined. Usually high prediction accuracy can be achieved if the measure of similarity is sensitive to object features which determine the target (explicitly or implicitly). Since most of the time it is unclear in advance which features affect the target, the similarity measure should take into account all available features. If the totality of selected features describing each object is expressed by formulas, a reasonable expression of similarity between a pair of objects is a formula which is the least common generalisation of the formulas for both objects, which is anti-unification. Anti-unification is the inverse operation to the unification of formulas in logic programming. Unification is the basic operation which finds the least general (instantiated) formula (if it exists), given a pair of formulas.

For example, for two formulas *reconcile* (*a, X, f*(*X*)) and *reconcile*(*Y, U, f*(*f*(*b*))) their anti-unification (least general generalisation) is $p(Z1, Z2, f(Z2))$. Anti-unification was used in Plotkin (1970) as a method of generalisation and later this work was extended to form

a theory of inductive generalisation and hypothesis formation. Conversely, unification of these formulas, *reconcile*(*a, X, f*(*X*)) = *reconcile*(*Y, U, f*(*f*(*b*))) will be *p*(*a, C, f*(*f*(*b*))). Our logic programming implementation of anti-unification for a pair of conjunctions, which will be customised to our domain, is presented in Figure 10.

1. Intersections

Positive: `[[i(_),e(_),c(_)],[i(_)],[i(_),e(_)],[i(_),e(_),s(_)]]`

Negative: `[[i(_),c(_),r(_)],[c(_),r(_)]]`

Unassigned examples: `[[c(o8),r(o8),s(o8)]]`

2. Hypotheses

Positive: `[[i(_),e(_),c(_)],[i(_)],[i(_),e(_)],[i(_),e(_),s(_)]]`

Negative: `[[i(_),c(_),r(_)],[c(_),r(_)]]`

Contradicting hypotheses: `[]`

(Note that all intersections are turned into hypotheses because there is no overlap between positive and negative ones.)

3. Background (positive and negative objects with respect to the target)

Positive: `[[io1),e(o1),c(o1)],[i(o2),e(o2),c(o2),s(o2)],[i(o3),r(o3)],`
`[i(o4),e(o4),s(o4)]]`

Negative: `[[i(o5),e(o5),c(o5),r(o5)],[i(o6),c(o6),r(o6)],[c(o7),r(o7]]`

Inconsistent: `[]`

(Note that the negative component acquired the third member `[i(o5),e(o5),c(o5),r(o5)]` compared with the hypotheses (step 2).)

4. Prediction for `good_fit` (object `o8`)

Positive: `[]`

Negative: `[[c(o8),r(o8),s(o8)]]`

Inconsistent: `[]`

Figure 9. The *Jasmine* prediction protocol. Steps are numbered in accordance to the units in Figure 8.

Table 4. Division of the totality of the combinations of features into positive, negative and inconsistent class of predictions.

Good fit	Not a good fit	Inconsistent prediction (some other means of assessment are required)
`[i(o8)]`	`[c(o8), r(o8), s(o8)]`	`[e(o8)]`
`[i(o8),e(o8)]`	`[c(o8), r(o8)]`	`[c(o8)]`
`[i(o8),e(o8), c(o8)]`	`[e(o8),c(o8),r(o8)]`	`[e(o8), c(o8)]`
`[i(o8),r(o8), s(o8)]`		`[s(o8)]`
`[i(o8), e(o8), r(o8), s(o8)]`		`[r(o8),s(o8)]`
		`[i(o8),c(o8),r(o8),s(o8)]`
		`[r(o8)]`
		`[e(o8),r(o8),s(o8)]`
		`[i(o8),e(o8),c(o8),r(o8)]`
		`[i(o8),e(o8),c(o8),r(o8),s(o8)]`

Although the issue of implementation of the anti-unification has been addressed in the literature (e.g. Delcher and Kasif 1990), we present the full code to show how it can be adjusted to our domain. To adjust the anti-unification to a particular domain, additional constraints on terms can be enforced to express a domain-specific similarity. Particularly, certain arguments can be treated differently (should not be allowed to change if very important, or should form a special kind of constant). However, these measures are insufficient to adequately express similarity between scenarios represented as ordered sequences of communicative actions.

Using JSM-based learning requires associativity in finding similarity between scenarios: $(S_1 \cap S_2) \cap S_3 = S_1 \cap (S_2 \cap S_3)$. Applying anti-unification to scenarios as ordered

Table 5. Enumeration of predicates used in *Jasmine* reasoning procedure.

Positive	Negative	Other	Meaning
rawPos	*rawNeg*	*unknown*	Raw data
iPos	*iNeg*		Intersections
posHyp	*negHyp*	*inconsHyp*, inconsistent	Hypotheses
reprObjectsPos	*reprObjectsNeg*	*reprObjectsIncons*, both positive and negative, prediction: unknown	Instantiations by objects
predictPos	*predictNeg*	*predictIncons*, prediction: inconsistent	Predictions


```
similar(F1, F2, F):- antiUnifyFormulas(F1, F2, F).

antiUnifyFormulas(F1, F2, F):- clause_list(F1, F1s), clause_list(F2, F2s),
   findall( Fm, (member(T1, F1s), member(T2, F2s),
        antiUnifyTerms(T1, T2, Fm)), Fms), %finding pairs
   %Now it is necessary to sort out formulas which are not most general within the list
   findall( Fmost, (member(Fmost, Fms),
      not ( member(Fcover, Fms), Fcover \= Fmost,
            antiUnifyTerms(Fmost, Fcover, Fcover)) ), Fss),
     clause_list(F, Fss). % converting back to clause

antiUnifyTerms(Term1, Term2, Term):-
  Term1=..[Pred0|Args1],len(Args1, LA),% make sure predicates
  Term2=..[Pred0|Args2],len(Args2, LA),% have the same arity
  findall( Var, (  member(N,  [0,1,2,3,4,5,6,7,8,9,10 ]), % not more than 10 arguments
    [! sublist(N, 1, Args1, [VarN1]), %loop through arguments
      sublist(N, 1, Args2, [VarN2]),
      string_term(Nstr,N), VarN1=..[Name|_],    string_term(Tstr,Name),
      concat(['z',Nstr,Tstr],ZNstr),    atom_string(ZN, ZNstr) !],
  % building a canonical argument to create a variable as a result of anti-unification   ifthenelse( not
(VarN1=VarN2),
  ifthenelse(( VarN1=..[Pred,_|_],VarN2=..[Pred,_|_]),
      ifthenelse( antiUnifyConst(VarN1, VarN2, VarN12),
       %going deeper into a subterm when an argument is a term
        (Var=VarN12),     Var=ZNstr) ),
          OR domain-specific code here for special treatment of certain arguments
    % various cases: variable vs variable, or vs constant, or constant vs constant
            Var=ZNstr),Var=VarN1)          ), Args),          Term=..[Pred0|Args].
```

Figure 10. The clauses for logic program for generic anti-unification (least general generalisation) of two formulas (conjunctions of terms). Predicate *antiUnify*(*T*1, *T*2, *Tv*) inputs two formulas (scenarios in our case) and outputs a resultant anti-unification.

lists of expressions for communicative actions does not obey the associativity. A naive solution here would be to ignore the order of communicative actions and just consider conjunctions of expressions for these actions. This would lead to ignorance of essential information about scenarios.

To overcome this problem we represent the ordered set of formulas for communicative actions as the unordered set of these action plus multiple instances of the binary predicate *after* (*Action*1*, Action*2). Using this predicate allows retaining information about the order of communicative actions in scenario and obeying the associativity at the same time. To find a common scenario formula for two scenarios, which are represented by formulas above, we separately match predicates for actions, predicates for their order of actions *after* and predicates for other binary relations on predicates (*causal*).

Moreover, the role for scenario classification of a sequence of communicative actions forming a step is more important than the role of a single communicative action. Therefore, computing similarity between scenarios, one primarily needs to find matching steps occurring in a pair of scenarios and then search for matching individual actions.

The chart for the algorithm of finding similarity between scenarios is depicted in Figure 11. The clauses for each unit are in the Appendix.

Figure 12 presents an example of recognising scenarios, and Figure 13 outlines the reasoning protocol. In our evaluation settings this is one of the simple possible scenarios described in a customer complaint which contains enough data to be classified.

8. Evaluation and deployment

In this section we present the evaluation results of our classification model. We present the results of different evaluation settings for both *Nearest Neighbour* and *Jasmine*:

1. Estimating the accuracy for our extended illustration dataset (Figure 5).
2. Evaluation of how the complaint validity is determined for a number of customer complaints for banking services. A common training dataset allows performing a comparative analysis of the customer support quality for these banks.
3. Evaluation of the accuracy of representation means.

Firstly, for each of 10 scenarios, we set its class as unknown and verify if this scenario can be related to its class properly, building common sub-scenarios with four representatives of its class and five foreign scenarios. Only scenarios I2 and V3 (two out of 10) can be neither assigned to the proper class nor to a foreign class; the rest of scenarios were properly assigned.

Secondly, for the banking complaints, we formed the training dataset randomly selecting half of the available complaints for each bank. The other half of complaints for each bank was used for evaluation of accuracy on the one hand, and for assessment of relative quality of customer support for each bank on the other hand. The complaints we used were downloaded from the public website PlanetFeedback.com for the 3 months starting from March 2004. Each complaint was manually coded as a sequence of communicative actions and assigned a status by us. Thirdly, the usability and adequacy of our formalism was evaluated on the basis of a team of individuals divided into three classes: complainants, company representatives and judges.

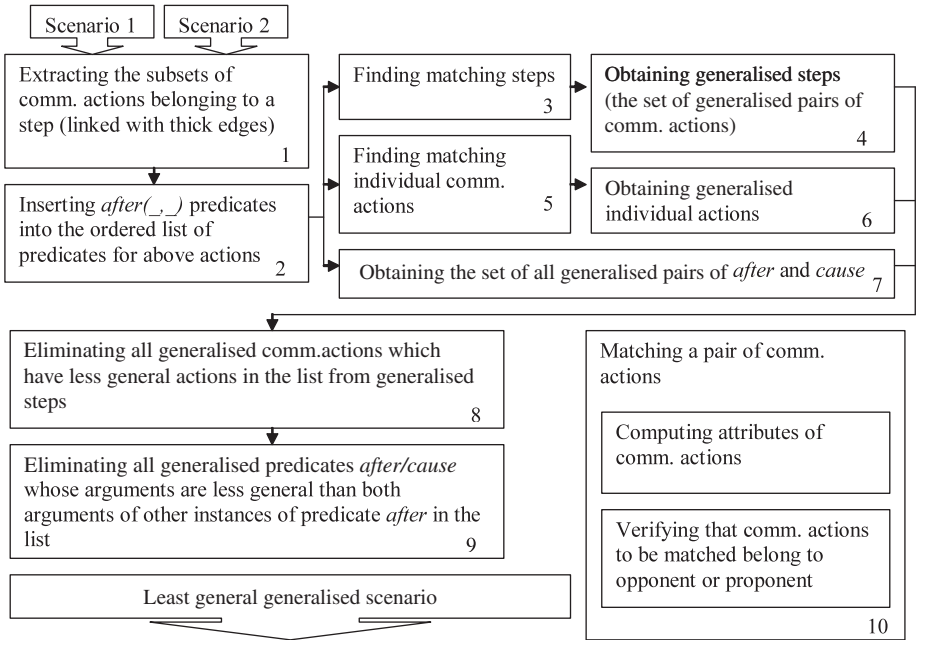

Figure 11. The chart of the algorithm for finding similarity between scenarios.

```
features([agrees, explaint, suggests, remindt, allows, …]). % see table 3
objects([o1,o2,o3,o4,o5,o6,o7,o8]).  % scenarios
targets([class1]). % valid or invalid


agrees(o1, da2). explaint(o1, da2). suggests(o1, da2).
class1(o1).
agrees(o2, da2). explaint(o2, da2). suggests(o2, da2).       allows(o2, da2).
class1(o2).
agrees(o3, da2).                              remindt(o3, da2).
class1(o3).
agrees(o4, da2). explaint(o4, da2).                      allows(o4, da2).
class1(o4).
agrees(o5, da25). explaint(o5, da21). suggests(o5, da2). remindt(o5, da21).
agrees(o6, da26).                     suggests(o6, da2). remindt(o6, da2).
                                      suggests(o7, da2). remindt(o7, da2).
agrees(o8, da2). explaint(o8, da2). suggests(o8, da2).  remindt(o8, da4).
unknown(class1(o8)).
```

Figure 12. A sample knowledge base for scenario classification: determining if it belongs to class 1.

Not all complaints submitted by upset customers to consumer advocacy websites can be assessed with respect to validity. If a complaint just mentions a failure in a product or a service without describing interaction with customer support, its validity is linked with respective product/service-related knowledge and cannot be assessed using the proposed technique. As we have revealed, this is not a typical complaint: usually, complaints

2. Hypotheses (we skip step 1)

Positive:

```
[[agrees(O1,D),explaint(O1,D),suggests(O1,D),af(agrees(O1,D),explaint(O1,D)),af(explaint
(O1,D),suggests(O1,D))],[agrees(O1,D),mentactt([x,x,1,1,x],[D]),af(agrees(O1,D),explaint
(O1,D)),
af(explaint(O1,D),suggests(O1,D))],[agrees(O1,D),explaint(O1,D),mentacts([1,x,x,-
1,1],[D]),af(agrees(O1,D),explaint(O1,D)),af(explaint(O1,D),suggests(O1,D))],
[agrees(O2,D),explaint(O2,D), suggests(O2,D),af(agrees(O2,D),explaint(O2,D)),
af(explaint(O2,D),suggests(O2,D))], [agrees(O2,D),mentactt([x,x,1,1,x],[D]),
af(agrees(O2,D),explaint(O2,D)),af(explaint(O2,D),suggests(O2,D))],
[agrees(O2,D),explaint(O2,D),allows(O2,D),af(agrees(O2,D),explaint(O2,D)),
af(explaint(O2,D),suggests(O2,D))],[agrees(O3,D),mentactt([x,x,1,1,x],[D]),
af(agrees(O3,D),remindt(O3,D))],[agrees(O4,D),explaint(O4,D),mentacts([1,x,x,-1,-
1],[D]), af(agrees(O4,D),explaint(O4,D)),af(explaint(O4,D),allows(O4,D))],
[agrees(O4,D),explaint(O4,D),allows(O4,D),
af(agrees(O4,D),explaint(O4,D)),af(explaint(O4,D),allows(O4,D))],
[agrees(O4,D),mentactt([x,x,1,1,x],[D]),
af(agrees(O4,D),explaint(O4,D)),af(explaint(O4,D),allows(O4,D))]] ,
```

Negative:

```
[agrees(O5,D5),suggests(O5,D),remindt(O5,D55)],[suggests(O5,D),remindt(O5,D5)],
[agrees(O6,D6),suggests(O6,D),remindt(O6,D6)],[suggests(O6,D),remindt(O6,D6)],
[suggests(O7,D),remindt(O7,D7)]]
```

3′. Cases/intersections close to the given

Positive:

```
[[agrees(o1,da2),explaint(o1,da2),suggests(o1,da2),af(agrees(o1,da2),explaint(o1,da2)),a
f(explaint(o1,da2),suggests(o1,da2))],
[agrees(o1,da2),mentactt([x,x,1,1,x],[da2]),af(agrees(o1,da2),explaint(o1,da2)),af(expla
int(o1,da2),suggests(o1,da2))], [agrees(o1,da2),explaint(o1,da2),mentacts([1,x,x,-1,-
1],[da2]),af(agrees(o1,da2), explaint(o1,da2)),af(explaint(o1,da2),suggests(o1,da2))],
[agrees(o2,da2),explaint(o2,da2),suggests(o2,da2), af(agrees(o2,da2),explaint(o2,da2)),
af(explaint(o2,da2),suggests(o2,da2))], [agrees(o2,da2),mentactt([x,x,1,1,x],[da2]),
af(agrees(o2,da2),explaint(o2,da2)),af(explaint(o2,da2),suggests(o2,da2))]].
```

Negative:

```
[agrees(o5,da25),suggests(o5,da2),remindt(o5,da2555)],
[suggests(o5,da2),remindt(o5,da2555)],

[agrees(o6,da26),suggests(o6,da2),remindt(o6,da26)],
[suggests(o6,da2),remindt(o6,da26)],

[suggests(o7,da2),remindt(o7,da27)]].
```

4. Prediction for `class1` (object `o8`)

Positive: `[]`. Negative: `[]`. Inconsistent: `[]` (because close to positive and negative scenarios).

Figure 13. The *Jasmine* prediction protocol for the scenario classification problem. Steps are numbered in accordance to the units at Figure 8. Binary predicate *af* is inserted where appropriate. Note some naming conventions: a communicative action of a proponent predicate name ends with 's', and opponent – with 't'. Also, note the usage of mentacts ([1,x,x,−1,−1], [da2]) – a communicative action of a proponent, from step D; its semantics is close to *suggest*, *allow*, *try*.

are caused by both product failure and customer support failure. Also, we did not include complaints with too simple or too complex a structure of interaction scenarios (the former case is similar to the above, and the latter case is subject to manual evaluation in the decision-support settings).

We used the data for 14 banks, 20 complaints for training and 20 complaints for evaluation. Firstly, the consistency of each training dataset is evaluated when complaint validity for each complaint is assumed unknown and classification is performed. After that the numbers of false positives, false negatives and correct classification results are obtained. The classification accuracy is then attempted to be increased by means of temporary eliminating cases from the training dataset which leads to inconsistent classification (refusal to classify). Finally, the numbers of false positives and negatives cases among previously unclassified cases are obtained.

Table 6 which contains the results of validity assessment is organised as follows:

- The first four columns contain bank number, dataset volumes, and the numbers of valid/invalid complaints as manually assessed by the authors (two lightly tinted areas on the left).
- Self-evaluation of training dataset column (light-tinted area) shows the percentage of complaints from the training dataset for each bank which were classified wrongly or not classified at all.
- Middle column (Classifications) gives the number of complaints which were classified correctly and incorrectly (false positives and false negatives).
- The tinted columns on the right show the results of classifying the inconsistent classifications columns when the inconsistencies are attempted to be overcome by removing certain complaints from the training dataset in the course of classifying a given one.
- The rightmost two columns, Overall classification accuracy (*Jasmine*/*Nearest Neighbour*), give the number of correctly assigned complaints as a percentage to the total number in the evaluation dataset.

Rather low data self-evaluation results of 77% show that the domain is quite complex; and the scenarios in the training dataset are quite diverse to cover all plausible conflict protocols. There is a low deviation between the self-evaluation accuracies for different banks: it means that the coverage of the totality of scenario possibilities is similar (although incomplete).

Hence the resultant recognition accuracy is 70.4% for cautious classification by *Jasmine* and 72.5% for *Nearest Neighbour*. Being quite low in accordance to pattern recognition standards in such domains as speech and visual object recognition, this accuracy is believed to be satisfactory for the decision-support settings where the number of complaints which have to be re-assessed manually is relatively low. Obtained classification accuracy cannot be compared with 50% that one would get by a random prediction because our prediction setting requires a highest accuracy providing an explanation for the decision. *Nearest Neighbour* gives just 2% better accuracy than *Jasmine* and does not guarantee to deliver the best explanations for the categorisation decision; therefore, *Jasmine* is a preferred solution for decision support applications. Since the training dataset self-evaluation (by *Jasmine*) accuracy is just 6.9% higher than that of the evaluation dataset, one can conclude that the complaint scenarios are rather diverse and accumulating of a larger dataset might increase the accuracy (and probably worsen the performance).

For evaluation we used a form which allows specifying a graph similar to Figure 1.

We proceed to the evaluation of the adequacy of representation language. Complainants had a task to read a textual complaint and input it into the form (Figure 14a) so that another team member (a company representative) could comprehend it (and briefly sketch the plot as a text). A third team member (judge) then compared the original complaint and the one written by the company representative as perceived from the form. The result of this comparison was the judgment on whether the scenario structure has been dramatically distorted in respect to the validity of a given complaint. Table 7 shows the results of representation adequateness. We observe that less than 15% of complaints were hard to capture by means of communicative actions. We also observed that about a third of complaints lost important details and could not be adequately restored (although they might still be properly related to a class).

Table 6. Complaint classification results for fourteen banks.

		As assigned by experts			Classification results					Cases are removed from the training dataset				
Bank	Number of complaints Training/ evaluation	Valid	Invalid	Self-evaluation of training dataset,%	Valid	Invalid	Classified as valid but invalid (false positives)	Classified as invalid but valid (false neg.)	Inconsistent classification (refuse to classify)	Valid	Invalid	Inconsistent or wrong classification	Overall classification accuracy, *Jasmine*,%	Overall classification accuracy, nearest neighbour,%
Bank 1	20/20	8	12	80	6	8	1	1	4	1	0	3	75	75
Bank 2	20/20	6	14	75	6	9	2	0	3	0	0	3	75	65
Bank 3	20/20	7	13	80	5	8	2	1	4	0	1	3	70	75
Bank 4	20/20	5	15	75	3	9	2	2	4	0	1	3	65	70
Bank 5	20/20	8	12	80	5	7	3	2	3	1	0	2	65	75
Bank 6	20/20	8	12	65	4	8	2	2	4	1	0	3	65	65
Bank 7	20/20	11	9	75	6	6	1	3	4	0	0	4	60	70
Bank 8	20/20	8	12	80	6	8	1	1	4	0	1	3	75	75
Bank 9	20/20	7	13	75	4	8	1	2	5	1	1	3	70	75
Bank 10	20/20	9	11	80	6	8	3	1	2	0	0	2	70	65
Bank 11	20/20	10	10	85	6	7	2	2	3	1	1	1	75	80
Bank 12	20/20	5	15	75	2	11	1	2	4	1	0	3	70	75
Bank 13	20/20	10	10	75	6	4	2	1	7	2	1	4	65	65
Bank 14	20/20	8	12	80	7	10	0	1	2	0	0	2	85	85
Average		7.9	**12.1**	**77.1**	**5.1**	**7.9**	**1.6**	**1.5**	**3.8**	**0.6**	**0.4**	**2.8**	**70.4**	**72.5**
Percentage		39.3%	**60.7%**		**25.7%**	**39.6%**	**8.2%**	**7.5%**	**18.9%**	**2.9%**	**2.1%**	**13.9%**		

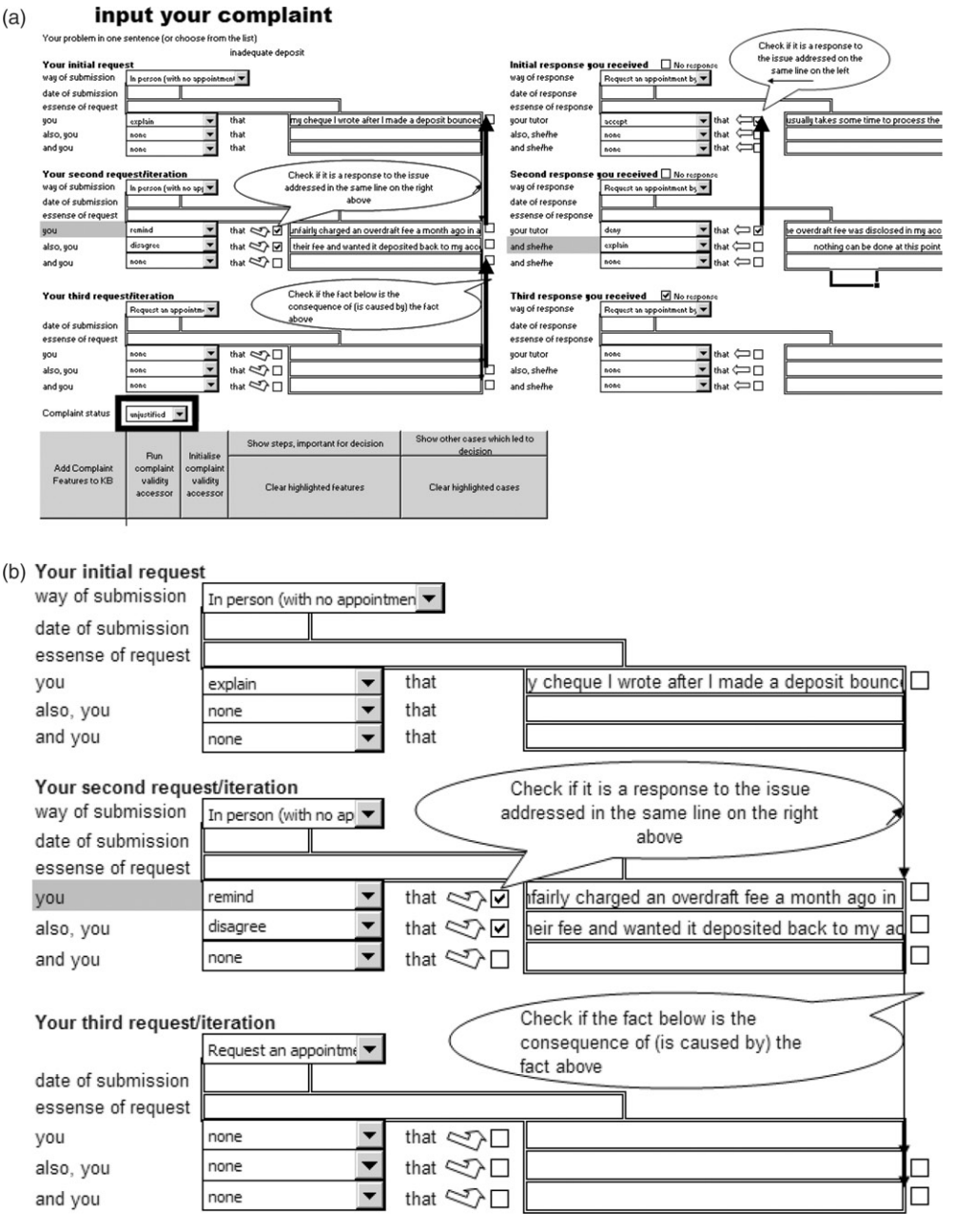

Figure 14. (a) The screen-shot of the interactive complaint form where the complaint scenario U from Figure 6 is specified. (b) The left pane of the interactive complaint form. Here a complainant specifies her communicative actions and their parameters. (c) The right pane of the interactive complaint form. Here a complainant specifies the communicative actions and their parameters of his opponent (a company).

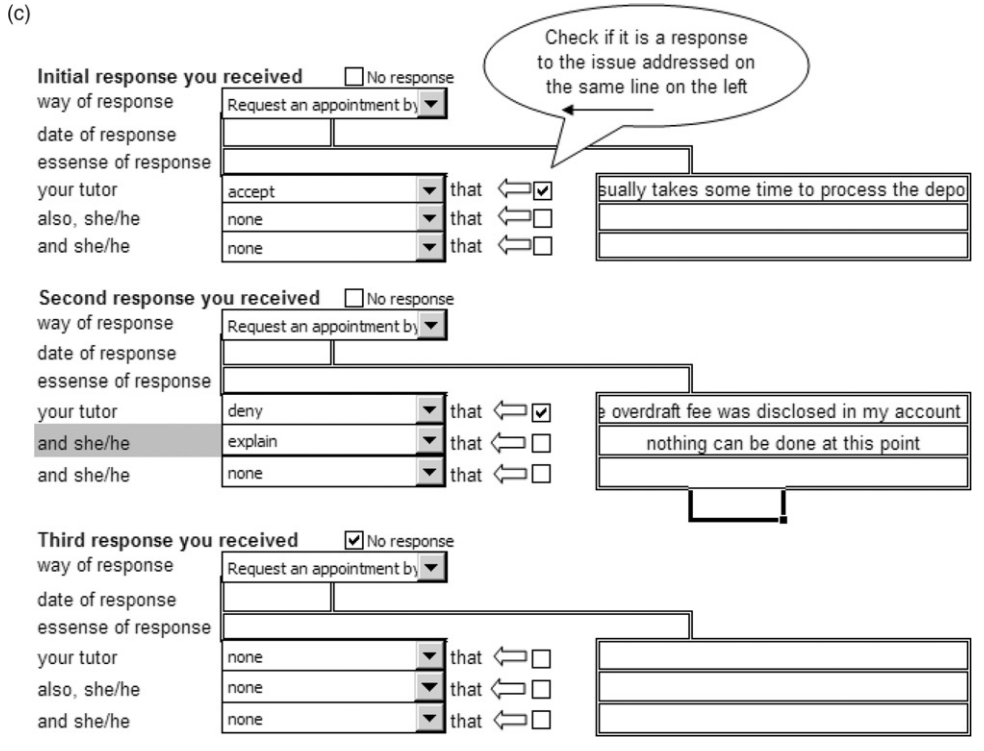

Figure 14. Continued.

Table 7. Evaluation of the complaint validity assessment.

Dataset	Percentage of complaints which were understood and (somehow) specified in the Form	Percentage of complaints which were (at least partially) reconstructed from the Form	Percentage of complaints which were properly specified and reconstructed
Bank 1	85	75	65
Bank 2	80	75	60
Bank 3	95	85	75
Bank 4	90	85	75
Bank 5	90	80	75
Bank 6	80	75	70
Bank 7	85	75	65
Bank 8	90	85	75
Bank 9	85	75	65
Bank 10	85	80	70
Bank 11	90	80	65
Bank 12	85	75	65
Bank 13	80	70	60
Bank 14	90	85	75
Average	86.4	78.6	68.6

Nevertheless, one can see that the proposed representation mechanism is adequate for representing so complex and ambiguous structures as textual complaints in most cases.

As to the performance of the *Jasmine* classification, it turns out to be a minor issue due to a low number of communicative actions per scenarios and their diversity.

Our further evaluation involved an improvement of existing software for processing customer complaints, called ComplaintEngine (Google: complaint + engine). Five attributes of communicative actions, selected for the model presented in this paper, helped to improve the accuracy of scenario recognition, given the particular set of complaints from our database of formalised complaints. Our database primarily originates from the data on the financial sector, obtained from the website of publicly available textual complaints, PlanetFeedback.com.

Currently, *ComplaintEngine* uses anti-unification procedure to find a similarity between scenarios. Machine learning of *ComplaintEngine* uses the JSM-type plausible reasoning (Finn 1999) augmented with situation calculus, reasoning about communicative states and other reasoning domains. *ComplaintEngine* applies domain-independent anti-unification to formulas that include enumeration of communicative actions in time.

As expected, the employed machine learning technique allowed noticeable improvement of complaint recognition accuracy. Judging by the restricted dataset of 80 banking complaints (40 complaints make the training set and 40 complaints have to be classified), the performance of *ComplaintEngine* was improved by 6% to achieve the resultant recognition accuracy of 89%. Relating a scenario to a class, *ComplaintEngine* is capable of explaining its decision by enumeration of similar and dissimilar scenarios, as well as particular communicative actions which led to its decision.

Since such accuracy was achieved by manual adjustment of the model of multi-agent scenario, we expect it to be much lower for other complaint domains. However, we believe that the role of improved machine learning technique for functioning in a new complaint domain will be substantial.

We would like to briefly introduce *ComplaintEngine* (Galitsky 2006b), the integrated complaint management component of a customer response management infrastructure. The user interface to specify a complaint scenario is shown at Figure 14a. Figures 14b and 14c depict the fragments of this form where complainant selects his communicative actions and communicative actions of his opponent (a company) respectively. Communicative actions are selected from the list of 20 or more, depending on the industry sector of a complaint. The parameters of communicative actions are specified as text in the Interactive Form; however they are not present in the formal graph-based scenario representation. Causal links between the parameters (subjects) of communicative actions are specified in pairs of check boxes (shown by vertical arrows).

Having performed the justification of complaint validity, *ComplaintEngine* sets the list box for complaint status at 'unjustified'. *ComplaintEngine* provides the explanation of its decision, highlighting the cases which are similar to *U* (unjustified), and which are different from *U* (justified). Moreover, *ComplaintEngine* indicates the communicative actions (steps) that are common for *U* and other unjustified complaints to further back up its decision.

A similar form to Figure 6 is used for a complainant to file a complaint, and for a company to store complaints, analyse them, determine validity, explain how the decision has been made, and finally advise on a strategy for complaint resolution.

A complainant has a choice to use the above form or to input complaint as a text so that the linguistic processor processes the complaint automatically and fills the form for

her. Using the form encourages complainants to enforce a logical structure on a complaint and to provide a sound argumentation. After a complaint is partially or fully specified, the user evaluates its consistency. *ComplaintEngine* indicates whether the current complaint (its mental component) is consistent or not, it may issue a warning and an advice concerning improvement of the logical structure of this complaint.

When the complainant is satisfied with the response of *ComplaintEngine*, he submits the completed form. The other option is if a user observes that it is not possible to file a reasonable complaint, it may be dismissed at this early stage by the complainant.

9. Discussion and conclusions

Based on our experience with complaint resolution environments, we outline two major types in respect to how the involved agents operate with background knowledge:

1. Competent environment, where domain knowledge is used in the decision of involved agents concerning their communicative actions. Agents on the company side may also follow the domain-specific policies developed by senior company representatives.
2. Incompetent environment, where customer support agents act in accordance to common dialogue rather than company policy rules, follow their emotions, take into account uninstantiated beliefs, and have miscommunication among each other.

A special class of conflicts (non-co-operation dialogues) has been considered in the competent environment (Gabbay and Woods 2001). Gabbay and Woods observed that dialogue participants need to co-operate with one another in a manner sufficient to continue the dialogue itself. Customer complaints present as non-cooperative dialogue: customers are usually demanding and frequently express a bad mood, and customer support representatives conduct the dialogue because of the company's contractual obligation to provide customer care, and not because of their own intentions. The precise treatment of co-operation was conducted, where agents co-operate only to support the dialogue but hold their intention not to satisfy those of an opponent.

A sample complaint is presented by the above authors, where a customer wants to have their faulty toaster replaced. The company representative proposes that the toaster can be repaired by a manufacturer, rather then replaced, and backs up his proposal by the reference to company policy. All requests by the complainant and their motivations are ignored. Company representatives are instructed not to be involved in a discussion of the merits of customer circumstances which are not covered by the company's commercial policies. Gabbay and Woods outline two reasons for company's dialogue policy: it shortens the dialogue time and is believed to lead to customer acquiescence. In accordance to the model of these authors, customer support representatives follow two following modes of participation in a dialogue:

1. *Mind-closed*: they never accept customers' positions, in spite of valid arguments from the customer side. Because of this customer support's position is not semantically or epistemically privileged; instead, it is supported by the company's dialogue policy. To express this mode as a communicative action, we would use *insist* which is subject-independent (does not take into account subjects of the communicative actions of the opponent).

2. *No-engage*: customer support's utterances are unresponsive to opponent's utterances (communicative actions ignore).

The above authors argue that the customer service representative should have explained to the customer that, although his demands are reasonable and justifiable, the company could not afford to accede to them. Following the company officers who build the complaint policy, such an approach would make a dialogue longer to complete and decrease the likelihood of customer's acquiescence.

Consulting our database of customer complaints, we observe that most complaints concern incompetent environment. This is partially due to the fact that a competent environment yields fewer complaints; therefore it is less likely that such a complaint is submitted to a public database. It is worth mentioning that both competent and incompetent environments share similar attitudes of company representatives, including mind-closeness and no-engaging.

In this paper we proposed a machine learning approach to relate a formalised conflict scenario to valid and invalid complaint scenarios. The representation language is that of labelled directed acyclic graphs with a generalisation operator. For the purpose of machine learning, the scenarios are represented as a sequence of communicative actions attached to agents; the communicative actions are grouped by subjects, and the order of communicative actions is retained using binary predicates *after*. We considered the concept lattice of communicative actions and showed how the procedure of relating a complaint to a class can be implemented as a *Nearest Neighbour* and JSM (Finn 1999) learning machinery. This is believed to be an innovative approach to learn scenarios of inter-human interactions encoded as sequences of communicative actions.

Building a framework for comparative analysis of formal scenarios, one way or another way to express the similarity between the main entities has to be employed. In our earlier studies (Galitsky 2003) we approximated the meanings of mental entities using their definitions via the basis of *want-know-believe*; however we observed that this approach would be too coarse for recognising complaints. In this study we extend the speech act theory-based set of attributes to build an adequate concept lattice for communicative actions which was found to be more suitable than our earlier approaches to define a concept lattice for scenarios themselves while learning them.

Also, we have suggested a novel approach to building a semantic network between linguistic entities on the basis of selected attributes. The choice of attributes in this study is motivated by the task of scenario comparison; these attributes may vary from domain to domain. Twenty selected communicative entities are roughly at the same level of generality – there are 'horizontal' semantic relations between them. In this respect we have established a link between the theory of concept structures and speech act theory, which has been discussed (Turoff et al. 1999; Priss 2004) but not subject to computational analysis. This work sheds light on what kind of conceptual structures the communicative actions are; a necessity to extend the traditional set of attributes of communicative actions for the purposes of machine learning has been demonstrated.

We believe the current work is one of the first that has targeted machine learning in such a domain as multi-agent interactions described in natural language. A number of studies have shown how to enable BDI-agents with learning in a particular domain (e.g. information retrieval). In BDI settings the description of agents' attitudes is quite limited: only their beliefs, desires and intentions are involved. Moreover, just the automated (software) agents are addressed. In this paper we significantly extended the expressiveness

of representation language for agents' attitudes, using 20 communicative actions linked by a concept lattice. The suggested machinery can be applied to an arbitrary domain including inter-human conflicts, obviously characterised in natural language.

The evaluation of our model shows it is an adequate technique to handle such complex objects (both in terms of knowledge representation and reasoning) as communicative actions of scenarios of multi-agent interactions. The *Nearest Neighbour* approach was found suitable to relate an inter-human conflict scenario to a class. Evaluation using our limited dataset, as well as the dataset of formalised real-world complaints, showed a satisfactory performance.

The proposed method for formal representation of conflict scenarios allows their classification as well as an efficient user interface for complaint submission. The suggested approach to assessment complaint validity is appropriate for deployment in decision support settings: most typical complaints are subject to automated processing, and atypical cases are handled manually.

Further steps of the research along the line of machine learning technique for formal scenarios as graphs will be as follows:

- Developing more precise representation languages for scenarios of multi-agent interactions; adding more features to scenario representation in addition to temporal and causal links.
- In terms of applications, proceeding beyond the domain of customer complaints.
- Performing comparison with other classification techniques.

Acknowledgements

The second author was supported by the project 'Automated cognitive reasoning for data analysis in intelligent systems' of the Russian Academy of Sciences and Russian Foundation for Humanities, project no. 02-03-18166a.

References

Austin, J.L. (1962), in *How to Do Things with Words*, ed. J.O. Urmson, Oxford: Clarendon.

Bach, K., and Harnish, R.M. (1979), *Linguistic Communication and Speech Acts*, Cambridge, MA: MIT Press.

Boella, G., Hulstijn,V., and van der Torre, L. (2004), 'Persuasion Strategies in Dialogue,' in *Proceedings of the ECAI Workshop on Computational Models of Natural Argument* (*CMNA*'04), Valencia, eds. F. Grasso and C. Reed.

Bratman, M.E. (1987), *Intention, Plans and Practical Reason*, Cambridge, MA: Harvard University Press.

Carley, K. (1997), 'Extracting Team Mental Models Through Textual Analysis,' *Journal of Organisational Behaviour*, 18, 533–538.

Cohen, P.R., and Levesque, H.J. (1990), 'Performatives in a Rationally Based Speech Act Theory,' in *Proceedings of the 28th Conference on Association for Computational Linguistics*, pp. 79–88.

Delcher, A.L., and Kasif, S. (1990), ''Efficient Parallel Term Matching and Anti-unification,'' *Logic Programming*, Cambridge, MA: MIT Press, pp. 355–369.

Fagin, R., Halpern, J.Y., Moses, Y., and Vardi, M.Y. (1995), *Reasoning About Knowledge*, Cambridge, MA/London, England: MIT Press.

Farrel, J. (2006). 'Philosophy of Language.' *http://www.farrell.blogs.com/Writings/Speech-Acts.pdf*

Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1992), 'KQML – A Language and Protocol for Knowledge and Information Exchange,' Technical Report CS-94-02, Computer Science Department, University of Maryland, UMBC, Baltimore, MD.

Finn, V.K. (1991), 'Plausible Reasoning in Systems of JSM-type,' Itogi Nauki I Techniki,' *Seriya Iformatika*, 15, 54–101 , [in Russian].

Foundation for Intelligent Physical Agents. 'FIPA Communicative Act Library Specification, Version H.' *http://www.fipa.org/specs/fipa00037/*

Gabbay, D., and Woods, J. (2001), 'More on Non-Cooperation in Dialogue Logic,' *Logic Journal of Interest Group in Pure and Applied Logic*, 9(2), 305–323.

Galitsky, B. (2003), *Natural Language Question Answering System Technique of Semantic Headers. Advanced Knowledge International*, Adelaide, Australia: Magill.

———. (2004), 'A Library of Behaviours: Implementing Commonsense Reasoning about Mental World,' in *8th International Conference on Knowledge-Based Intelligent Information Systems*.

———. (2006a), 'Reasoning About Mental Attitudes of Complaining Customers,' Knowledge-Based Systems, 19 (2), 592–561.

———. (2006b), 'Merging deductive and inductive reasoning for processing textual descriptions of inter-human conflicts,' *Journal of Intelligent Information Systems*, 27(1), 21–48.

Galitsky, B., Kuznetsov, S., and Samokhin, M. (2005), 'Analyzing Conflicts with Concept-Based Learning,' in *International Conference on Concept Structures*.

Ganter, B., and Kuznetsov, S. (2001), 'Pattern Structures and Their Projections,' in *Proceedings of the 9th International Conference on Conceptual Structures*, ICCS'01, ed. G. Stumme and H. Delugach, *Lecture Notes in Artificial Intelligence*, 2120, 129–142.

Ganter, B., and Wille, R. (1999), ''Formal Concept Analysis,'' *Mathematical Foundations*, Berlin, Germany: Springer.

Garey, M.R., and Johnson, D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: Freeman.

Guerra-Hernandez, A.I, Fallah-Seghrouchni, A.E., and Soldano, H. (2004), ''Learning in BDI Multi-agent Systems,'' in *CLIMA IV – Computational Logic in Multi-Agent Systems*, FL, USA: Fort Lauderdale.

Harsanyi, J.C., and Selten, R. (1972), 'A Generalised Nash Solution for Two-Person Bargaining Games with Incomplete Information,' *Management Science*, 1880-106.

Hendler, J., and McGuinness, D. (2000), 'The DARPA Agent Markup Language,' *IEEE Intelligent Systems*, 15(5), 34–43.

Johnson, M.W., McBurney, P., and Parsons, S. (2005), 'A mathematical model of dialogue,' *Electronic Notes in Theoretical Computer Science*, 144, 181–198.

Jordan, J.S. (1992), 'The exponential convergence of Bayesian learning in normal form games,' *Games and Economic Behaviour*, 4, 4202–4217.

Kalai, E., and Lehrer, E. (1993), 'Rational learning leads to Nash equilibrium,' *Econometrica*, 61(5), 1019–1045.

Kolodner, J. (1993), *Case-Based Reasoning*, San Mateo, CA: Morgan Kaufmann.

Kraus, S., and Subrahmanian, V.S. (1995), 'Multiagent reasoning with probability, time, and beliefs,' *International Journal of Intelligent Systems*, 10(5), 459–499.

Kuznetsov, S.O. (1999), 'Learning of Simple Conceptual Graphs from Positive and Negative Examples,' in *Proceedings of Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD'99*, eds. J. Zytkow and J. Rauch, Lecture Notes in Artificial Intelligence, 1704, 384 – 392.

Kuznetsov, S.O., and Samokhin, M.V. (2005), 'Learning Closed Sets of Labeled Graphs for Chemical Applications,' *Inductive Logic Programming*, 3625, 190–208.

Labrou, Y., Finin, T., and Peng, Y. (1999), 'Agent Communication Languages: The Current Landscape,' *IEEE Intelligent Systems*, 14(2), 45–52.

Laza, R., and Corchado, J.M. (2002), ''CBR-BDI Agents in Planning,'' *Symposium on Informatics and Telecommunications (SIT'02)*, Spain: Sevilla, September 25–27, pp. 181–192.

Mill, J.S. (1843), *A System of Logic, Racionative and Inductive*, London: Longmans, Green and Reader.

Mitchell, T. (1997), *Machine Learning*, Columbus, OH: McGraw-Hill.

Mor, Y., Goldman, C.V., and Rosenschein, J.S. (1995), 'Learn Your Opponent's Strategy (in Polynomial Time),' in *Proceedings of IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*.

Mudgal, C., and Vassileva, J. (2000), 'Bilateral negotiation with incomplete and uncertain information a decision-theoretic approach using a model of the opponent,' in *Cooperative Information Agents*. IV: *The Future of Information Agents in Cyberspace*, eds. M. Klusch and L. Kerschberg, LNAI 1860, Berlin, Germany: Springer.

Muller, H.J., and Dieng, R. (eds.) (2000), *Computational Conflicts Conflict Modelling for Distributed Intelligent Systems*, New York: Springer-Verlag.

Nodine, M.H., and Unruh, A. (2000), "Constructing Robust Conversation Policies in Dynamic Agent Communities," in *Issues in Agent Communication*, eds. F. Dignum, and M. Mark Greaves, Heidelberg, Germany: Springer-Verlag.

Osborne, M.J., and Rubinstein, A. (1994), *A Course in Game Theory*, Cambridge, MA: The MIT Press.

Olivia, C., Chang, C.F., Enguix, C.F., and Ghose, A.K. (1999), 'Case-Based BDI Agents an Effective Approach for Intelligent Search on the World Wide Web. Intelligent Agents in Cyberspace,' in *AAAI Spring Symposium*.

Rosenschein, J., and Zlotkin, G. (1994), *Rules of Encounter*, Cambridge, MA: MIT Press.

Plotkin, G.D. (1970), 'A Note on Inductive Generalisation,' *Machine Intelligence*, 5, 153–163.

Priss, U. (2004), 'A Semiotic-Conceptual Framework for Knowledge Representation,' in *Knowledge Organisation and the Global Information Society*. *Proceedings of the 8th International ISKO Conference*, Ergon Verlag, ed. I. McIlwaine, pp. 91–96.

Rahwan, I., Ramchurn, S.D., Jennings, N.R., McBurney, P., Parsons, S., and Sonenberg, L. (2003), 'Argumentation-based Negotiation,' *Knowledge Engineering Review*, 18(4), 343–375.

Riloff, E. (1996), 'Automatically Generating Extraction Patterns from Untagged Text,' in *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044–1049.

Searle, J. (1969), *Speech Acts. An Essay in the Philosophy of Language*, Cambridge, England: Cambridge University Press.

———. (1979), *Expression and Meaning Studies in the Theory of Speech Acts*, Cambridge, England: Cambridge University Press.

Shanahan, M. (1997), *Solving the Frame Problem*, Cambridge, MA: MIT Press.

Singh, M.P. (2000), "A Social Semantics for Agent Communication Languages," in *Issues in Agent Communication*, eds. F. Dignum, and M. Mark Greaves, Heidelberg, Germany: Springer-Verlag.

Singh, P. and Barry, B. (2003), 'Collecting Commonsense Experiences,' in *Proceedings of the Second International Conference on Knowledge Capture (K-CAP 2003)*. Florida, USA.

Sowa, J. (1984), *Conceptual Graphs, Conceptual Structures Information Processing in Mind and Machine,* Reading, MA: Addison-Wesley.

Stone, P., and Veloso, M. (2000), 'Multiagent Systems. A Survey from a Machine Learning Perspective,' *Autonomous Robotics*, 8(3), 345–383.

Turoff, M., Hiltz, S.R., Bieber, M., Fjermestad, J., and Rana, A. (1999), 'Collaborative Discourse Structures in Computer Mediated Group Communications,' *Journal of Computer-Mediated Communication*, 4(4).

Weiss, G., and Sen, S. (1996), 'Adaptation and Learning in Multiagent Systems,' in Lecture Notes in Artificial Intelligence, Vol. 1042, Berlin/Heidelberg/New York: Springer-Verlag.

Wooldridge, M. (2000), *Reasoning about Rational Agents*, Cambridge, MA: The MIT Press.

Yevtushenko, S.A. (2005). *http//www.sf.net/projects/conexp*

Zeng, D. and Sycara, K. (1997), 'Benefits of Learning in Negotiation,' in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, Menlo Park, CA AAAI Press, pp. 36–42.

Appendix: Computing the similarity between scenarios

Unit 1. The generalisation of two vectors of attributes of mental actions.

```
mergeAttrib ([1, -1,x,1,1], [1, x,x,1,-1], K).
mergeAttrib (Attr1s, Attr2s, [A1, A2, A3, A4, A5]):- Attr1s=[A11, A12, A13,
  A14,A15], Attr2s = [A21,A22,A23,A24,A25],
  ifthenelse (A11=A21, A1=A11, A1=x), ifthenelse(A12=A22,
  A2=A12, A2=x),
  ifthenelse (A13=A23, A3=A13, A3=x), ifthenelse(A14=A24,
  A4=A14, A4=x),
  ifthenelse (A15=A25, A5=A15, A5=x).
```

Unit 2. The predicate which generalises two mental actions. Mental actions for opponent and proponent agents mentactt and mentacts are treated differently. Predicate `sameAgent` distinguishes between above predicates.

```
mergeMentalActions(M1f,M2f,Mf):-M1f=..[M1, ObjAttr1, Lnk1],
M2f=..[M2, ObjAttr2,Lnk2], (Lnk1_Lnk2=..[Lnk1, Lnk2];
       (Lnk1_Lnk2=..[Lnk1, Lnk2]; (Lnk1_Lnk2=(Lnk1, Lnk2))),
  (% make mentacT, mentacS
    (M1\=mentactt, M1\=mentacts,
M1=M2, Mf=..[M1, ObjAttr1, (Lnk1)]);% could be (Lnk1, Lnk2)

  (M1=mentacts,M2=mentacts,mergeAttrib(ObjAttr1,ObjAttr2,Attr),
Mf=..[mentacts, Attr, Lnk1_Lnk2]);

  (M1=mentactt, M2=mentactt, mergeAttrib(ObjAttr1, ObjAttr2, Attr),
Mf=..[mentactt, Attr, Lnk1_Lnk2]);

(M1=mentacts; M1=mentactt), M2\=mentactt, M2\=mentacts,
           sameAgent(M1, M2, _), attr(M2,Attr2),
           mergeAttrib(ObjAttr1, Attr2, Attr),
           Mf=..[M1, Attr, (Lnk1_Lnk2)]);

  (M1\=mentactt, M1\=mentacts,(M2=mentacts; M2=mentactt),
           sameAgent(M1, M2, _), attr(M1,Attr1),
           mergeAttrib(Attr1, ObjAttr2, Attr),
           Mf=..[M2, Attr, (Lnk1_Lnk2)]);

  (M1\=mentacts, M1\=mentactt, M2\=mentacts, M2\=mentactt,
           M1\=M2, sameAgent(M1, M2, Last),
attr(M1,Attr1), attr(M2,Attr2), mergeAttrib(Attr1, Attr2, Attr),
       ifthenelse(Last=115, Mf=..[mentacts, Attr, (Lnk1_Lnk2)],
           Mf=..[mentacts, Attr, (Lnk1_Lnk2)],
           Mf=..[mentactt, Attr, (Lnk1_Lnk2)]))).

sameAgent(M1, M2, Last):- atom_string(M1, M1str),
list_text(M1ss, M1str), atom_string(M2, M2str), list_text(M2ss, M2str),
last(M1ss, Last), last(M2ss, Last).

MoreGenMentActions(A, GA):- mergeMentalActions(A, GA, Am),
   A=..[MM, XXXXX, _], Am=..[MM, XXXXX, _].
```

Unit 2. Predicate Aggreg which extracts steps from a sequence of mental actions.

```
aggreg([mentact([1, -1, x, 1, 1],dA7), bringtatts(o26,dA2),
    explains(o26,dA3), disagrees(o26,dA3),
    agreet(o26,dA3), acceptt(o26,dA7),explains(o26,dA10),
    remindS(o26,dA11), bringtatts(o26,dA7),
    acceptt(o26,dA7), understandt(o26,dA7)], K).

aggreg(MentForm, MFA):-
    member(T1, MentForm), T1=..Ts1, [! last(Ts1, ConnectVar),
    findall(AggTerm, (member(AggTerm, MentForm), AggTerm\=T1,
                AggTerm=..Ts2, last(Ts2, ConnectVar)), AggTerms),
    AggTerms\=[],
    append([T1], AggTerms, MFAd), remove_duplicates(MFAd, MFA) !],
    member(ActionSfull, MFA), [! member(ActionTfull, MFA) !],
    ActionSfull=..[A1|_], ActionTfull=..[A2|_], not
      sameAgent(A1, A2, _).

aggreg(MentForm, MFA):-
    member(T1, MentForm), T1=..Ts1, last(Ts1, ConnectVar),
    findall(AggTerm, (member(AggTerm, MentForm), AggTerm\=T1,
    AggTerm=..Ts2, last(Ts2, ConnectVar)), AggTerms),
    AggTerms=[], MFA=MentForm.
```

Unit 3

```
insertAf(O, OAf):- findall(Af, (aggreg(O, OAggrs),
                member(M, OAggrs), list_next(OAggrs, M, MNext),
                M=..[Mp|_], MNext=..[Mnp|_], sameAgent(Mp, Mnp, _),
                                    Af=af(M, MNext)),
                              Afsd), remove_duplicates(Afsd, Afs),
     append(O, Afs, OAf).
```

Unit 4

```
commSubScenAf(O1, O2, Comm):-
    ifthenelse(member(af(_,_), O1), O1m=O1, insertAf(O1, O1m)),
    ifthenelse(member(af(_,_), O2), O2m=O2, insertAf(O2, O2m)),
    findall(Fm, (member(T1, O1m), member(T2, O2m), T2\=T1,
                  (mergeMentalActions(T1, T2, Fm);
```

Unit 5

```
                          (T1=..[af,Maf1,Maf1a], T2=..[af,Maf2,Maf2a],
                           mergeMentalActions(Maf1, Maf2, Mafm),
                           mergeMentalActions(Maf1a, Maf2a, Mafma),
                           Fm=..[af, Mafm, Mafma])
                        )), Fmsd), remove_duplicates(Fmsd, Fms),
```

Unit 6

```
findall(Fg, (member(Fg, Fms), not (member(F, Fms), F\=Fg,
                    % writeln((Fg=F)), get(CCC),
                ((not Fg=..[af|_], not F=..[af|_],
                  moreGenMentActions(Fg, F));
```

Unit 7, 8

```
                (Fg=..[af,Maf1,Maf1a], F=..[af,Maf2,Maf2a],
                 (moreGenMentActions(Maf1, Maf2),
                  moreGenMentActions(Maf1a, Maf2a)))
                )
  )    ), Comm).
```