Analysis of mental attributes for the conflict resolution in multiagent systems

Boris Galitsky

DIMACS, Rutgers University, New Brunswick, NJ 08903
galitsky@dimacs.rutgers.edu

Abstract

This study addresses the problem of constructing an adequate strategy for conflict resolution in the situations with high uncertainty and inconsistent goals. We develop the formal representation for such concepts as informing, deceiving, explaining, offending, forgiving, pretending, reputation etc. to model the various forms of multiagent conflict behavior. These concepts are defined in the basis of mental states triple *intention-knowledge-belief* with an arbitrary predicate for action.

We use the multiagent simulation tool with metalanguage support (MS) with the embedded metapredicates of mental state and action to provide the automatic agent with the ability to model the human agents with various degrees of reflection (Galitsky, 1998a). This tool raises the inference flexibility and expressiveness means to represent the formal scenarios of multiagent interaction and conflict resolution methodology.

MS capabilities of conflict resolution are demonstrated in the domain of the advertisement scheduler for the broadcasting industry. Automatic agent models the mental states and behavior of the group of human agents during the process of building the schedule with inconsistent and informal constraints and uncertain interaction between these agents. Reasoning about action and change, belief, and temporal reasoning; automatic, incremental, interactive and multiagent modes of the automatic agent demonstrate its capabilities for conflict detection and resolution.

1. Introduction

The conflict resolution problem is one of the important problems among the models of the multiagent systems. Such issues as inconsistent information from various agents, reasoning over attacking conflicts, distributed assessment situations, conflict resolution in a dialog, conflicts for learning, belief systems for conflicts, conflict avoidance and reduction, resource allocation problem and others, are actively addressed for last few years (Robert et. al., 1997, Kraus & Zlotkin, 1995, Zlotkin & Rosenschein 1996, Das et. al. 1997, Ephrati & Rosenschein, 1996, Kraus 1995, Klein 1993). The agents, analyzed in these approaches, are the human beings, computers and knowledge bases.

This study is focused on the conflict resolution in the situation, where the deep analysis of agents' mental states is essential. It is insufficient for adequate modeling to have the belief analysis system, inconsistency elimination system, a truth maintenance system, a model for distributed agents, or a specific dialog model, taken separately. This study involves the domain, where the conflict resolution system must cover the wide spectrum of mental attributes such as various degrees of intention, knowledge, belief, reflection. Besides, the system should be capable of representing such concepts as convincing, deceiving, pretending, explaining a reason, an object and a feature, offending, forgiving, reputation, etc. (Galitsky, 1998a).

As a basis for conflict formal definition, (Cholvy, 1998) proposes a logical definition of the notion of conflicts. There is a conflict when P holds, where P is one of the following formulas (depending on the considered case):

$\varphi \& \neg\varphi$;
believe(A, φ) & *believe*$(B, \neg\varphi)$;
believe(A, φ) & *believe*$(A,$ *believe*$(B, \neg\varphi))$;
obligation_modality(A, φ) & *prohibition_modality*(A, φ);
want(A, φ) & *want*$(B, \neg\varphi)$;
action(A, φ) & *prohibition_modality*(A, φ);
obligation_modality(A, φ) & $\neg$ *action*(A, φ).

φ can include space and time constraints.

In this study, we analyze more complex expressions of various forms of belief and express various modalities involving the concepts of knowledge and intention, extending the BDI model (See, for example, Cohen & Levesque, 1987). We are interested in a conflict case, irreducible to the constraint satisfaction settings, which does not require considerations of agents' mental attributes (Galitsky 1998b). Therefore, we relate the first case above to the traditional CSP. In addition, we mention that the traditional game-theoretic notion of (distributed, common) knowledge (Samet 1987) is insufficient to handle the peculiarities of the agents' mental states under conflict resolution.

This is rather computational, than analytical study. Instead of presenting a complete multiagent axiomatic system and deducing its specific behavior as a set of theorems, we introduce the family of definitions for the multiagent mental concepts, required for the conflict resolution. These generic definitions are reusable for an arbitrary domain, being added to the domain-specific component non-mental state and action. Therefore, we suggest not the stand-alone axiomatic, but the family of mental concepts which can be added to the existing multiagent (axiomatic) system targeting the specific

behavior.

The concept definitions are presented via the MS extension of first-order language to highlight the specific of MS PROLOG implementation. These definitions could be constructed in a different way as long as the certain syntactic analysis features are realized within the predicates (MS-extension specifics). We use the traditional notions of the logical programming (Variables are capitalized, quantifiers are skipped, default conjunction and temporal sequences are represented by term enumeration (symbol & is equivalent to symbol ,).

The paper is organized in the following way. Section 1 introduces the multiagent system and necessary concepts, develops the principles of conflict resolution and presents the example of multiagent conflict scenario. Section 2 presents the scheduling domain, algorithms and autonomy control for the incremental, interactive and multiagent modes.

1.1 Description of the multiagent system

The representation of the scenario agents takes into account the modern approaches to reasoning about intention (for example, Cohen &Levesque,1987), action, state, knowledge and believe and includes the following:

- Agent performs the *actions* (scheduling / unscheduling).
- Agent' actions lead to the *states (scheduling with certain constraint violations).*
- Agent' *intentions* to achieve a *state* or perform an *action*
- Agent' *knowledge and believe* about what agent's scheme include for itself and other agents.

want(Agent, do(Agent, Action)).	agent wants to perform an action
want(Agent, do(DAgent, Action)).	agent wants another agent to perform an action
want(Agent, know(Agent,What)):- (believe(Agent, know(KAgent, What)), ask(Agent, KAgent, What)).	agent wants (himself) to know
believe(Agent, want(WAgent, know(WAgent, What))) :- prefer(Agent, tell(Agent,WAgent, What), OtherAction).	agent believe that other agent wants to know
believe(Agent, want(WAgent, know(KAgent, What))):- believe(Agent, inform(WAgent,KAgent, What)). believe(Agent, want(WAgent, know(KAgent, What))) :- not know(KAgent, want(WAgent, know(KAgent, What))), inform(Agent,KAgent, ask(KAgent, WAgent, What)).	agent believes that someone else wants the third person to know
believe(Agent, want(WAgent, know(KAgent, want(Agent, What))) :- believe(Agent, inform(WAgent,KAgent, want(Agent, What))).	agent believes that someone else wants the third person to know what this agent wants

Fig.1. The examples of the mental attribute formulas. Various formulas are built in the basis of *want-know-believe* and *ask/inform in addition.* The action/ state predicates have the inmost occurrence: *do(Agent, Action)* or *What.* All well-formed formulas, constructed from the mental metapredicates, are interpretable by the system. We refer the reader to www.dimacs.rutgers.edu/~galitsky/MS/ma/wkb.html for more definitions of mental concepts in the basis of *want – know - believe.*

Each agent has intention to achieve certain value of his own state parameter or some value of another agent's state parameter. Besides, each agent has intentions to know certain parameters of the other agents' states and their intentions. The agents have knowledge and belief concerning the states, intentions and knowledge of the other agents and themselves. Knowledge can be once acquired, and belief can be acquired and changed.

We extend the first order language with metapredicates *want, know, believe* such that the formulas *want(Agent, φ), know(Agent, φ), believe(Agent, φ)* are well-formed if and only if the formula φ is well-formed.

In the metapredicates of mental states (*want, know, believe)* and mental action *(inform(ActiveAgent, PassiveAgent, φ))* the formula variableφ ranges over all valid formulas. (See (Galitsky 1998) for the technique of metalanguage reasoning). The formalism of Metalanguage Support (MS) delivers the expressive means to the agent's reasoning language, implementing the capability of autonomy adjustment. It occurs as a result of the agent's

decision concerning the mental states of the other agents.

We clarify the difference between the first-order language extension towards higher orders and towards metalanguage support. If we have a predicate *p(X, q(Y))* , this is the second–order formula if the quantification of q is possible, and p(X, q(Y)) can be satisfied by unification with the term *p(X, q(Y)) = p(x, q(y)), (x,y* – constants, *X,Y* – variables). If the satisfaction process for p requires the syntactic analysis of q(Y) (in other words, the fact that p holds depends on the syntactic properties of the term q(Y)), then p is a metapredicate, and the notion that its second argument ranges over formulas is important. In the examples below, the semantic of mental concepts (metapredicates) is built as checking of certain syntactic properties of the argument's formula instance.

The modern approaches to reasoning about intention, knowledge and belief stay within the bounds of traditional axiomatic method with completeness and soundness properties. Our experience of applied reasoning, involving mental states and actions shows that these approaches are insufficiently rich to present, in particular, the set of mental states, accessible from a given one. The logical property of soundness, inherent to the traditional calculus, prevents one from obtaining all the formulas (theorems), which hold for a particular mental state. Therefore we have to explicitly verify the validity of all well-formed mental formulas at each inference step. The reasoning in the majority of examples in this paper has to break the soundness property to be capable of explaining some multiagent phenomenology, inconsistent from the viewpoint of traditional axiomatic method. The set of intuitively valid formulas for a mental state frequently has multiple inconsistencies and cannot be handles by the sound knowledge representation formalism (compare with Fagin et. al.1995, Konolige,1986).

1.2 Basic mental actions towards conflict resolution: to inform and to deceive

If there is an agent who has some knowledge and believes that if this knowledge could be shared with the other agent or a few agents, the following will happen. These agents reevaluate their intentions and/or ways of their satisfaction such that the conflict is eliminated. We express the meaning of informing via the basis of *want-know-believe*:

```
inform(Who, Whom, What ) :-
   want(Who, know(Whom, What)),
   believe(Who, not know( Whom, What)),
   believe(Who, want(Whom, know( Whom, What))).
```

If there is a conflict and there is an agent who believes that there is no truthful knowledge that could affect the decisions of the other agents, this agent can initiate a deception towards conflict elimination.

```
deceive(Who, Whom, Fake, DeceptionGoal):-
     want(Who, DeceptionGoal),
     want(Whom, not DeceptionGoal),
     know(Who, not Fake),
          % we process the clause, which
          % links Fake and DeceptionGoal
     clause(dig(Whom, Place), Body),
     clause_list(Body, Bodys),
          % to analyze a clause components, we need to
          % convert it to a list
     member(know(Whom, Fake), Bodys),
          % the part below verifies that the cheat was
          % successful for Who, because Whom accepts
          % Fake and performs WhoWill
     clause(reaction, R), assert(R),
     know(Whom, Fake), DeceptionGoal.
```

The reader can compare this definition with its partial case in the example of scenario below (Section 1.5 , where the *DeceptionGoal* is particular action *dig(Investig, Field).*

1.3 Conflict development: to offend, to forgive, to reconcile and to explain

We start with the definition of unintentional offend. Ignoring modalities and tenses, we state, that unintentional offend is based on the lack of knowledge that the offending action *do(Who, Action)* is unwanted.

```
offend( Who, Whom, Action ) :- want(Who, Action),
  not want(Whom, Action),
  not know(Who, not want(Whom, Action)),
  do(Who, Action).
```

We remind the reader, that the default temporal relation between the terms is the order these terms occur in a clause.

To be forgiven, the offender has to demonstrate by some way that the offense was actually unintentional. It is necessary for the offender *Who* to inform *Whom* that *Who* would not *do* that *Action* if *Who* knew *Whom* did not like (*want*) it.

```
forgive( Whom, Who, Action) :-
  offend( Who, Whom, Action ),
  inform(WhoElse, Whom,
          not know(Who, not want(Whom, Action)) ),
  believe(Whom, (know(Who, not want(Whom, Action))→
                          not do(Who, Action) )).
```

If *Who* is unable to convince *Whom* (to make him *believe*) that the offend was unintentional, the other agent *Counselor* is required to explain the actual situation to *Whom.*

```
reconcile( Counselor, Who, Whom, Action ) :-
  offend( Who, Whom, Action ),
  not forgive( Whom, Who, Action),
  explain(Counselor, Whom,
          not know(Who, not want(Whom, Action)) ),
believe(Whom, (know(Who, not want(Whom, Action))→
                          not do(Who, Action) )).
```

While explaining, a *Counselor* helps *Whom* to build the

deductive link between particular facts and general knowledge, Whom possesses in accordance to the *Counselor's belief. The Counselor* knows this deductive link himself, *believes* that this link is unavailable for *Whom* and also *believes* this link will be established after *Whom* is *informed* with *PremiseFact*

explain(*Counselor, Whom, Fact*) :-
know(Counselor, PremiseFact→Fact),
believe(Counselor, believe(Whom, not know(Whom,Fact)),
believe(Counselor,
(inform(Counselor, Whom, PremiseFact) →
believe(Whom,Fact))).

This definition gives *Whom* the explanation **why** the *Fact* holds.

One can compare our semantic for *explain* with the other option, the commonsense definition of the concept **to explain by example**, what feature could a particular object possess". *Counselor explains Whom* what *Feature* could possibly *Object* possess, using the *FeatureList* as the set of examples. The following has to hold to give *Who* an ability to explain *Whom* the *Feature* of the *Object: Who knows* that *Feature,* and *Whom* does not *know,* what could be a *Feature* of the *Object.* Then the *Counselor informs Whom* about the example *FeatureList.*

explain_by_example(Counselor, Whom,
Feature(Object, FeatureList)) :-
know(Counselor, Feature), not
know(Whom, Feature(Object, FeatureList)),
know(Who, Object),
inform(Counselor, Whom, FeatureList).

It seems rather hard to fit two following concepts in the single definition: one explains some **facts** about objects and one explains the concept of having **a feature** for a given object.

We remind the reader, that all the above concepts are indeed defined in the *want-know-believe* basis.

1.4 Conflict resolution: the concepts of reputation and pretending

To show that our approach can handle rather sophisticated forms of behavior, we present the analysis how the agents reputation can be affected while negotiation. Let us assume that an agent is asked to reschedule some other appointment to be able to attend the primary one. This agent might think that if he easily agrees to reschedule the other appointment, the other agents would think he is not an "important" or "busy" person, and his reputation would drop. In such a case, pretending could come into play to avoid the influence on the agents' reputation.

More complex patterns of behavior arise when the agents perform transition to pretending. This type of behavior allows more degrees of freedom to the agents' decision-making. Agents' pretending eliminates the necessity to consider the possible influence of an agent's action on its reputation.

To define the *reputation, we* use the following intuition. First, the reputation is a common belief, a capability to predict a specific subset of the mental states of an agent. Second, each reputation estimator agent believes that if his belief concerning particular person is his reputation, than the other reputation estimator agents should have the same belief. In other words, distinguishing beliefs of the different agents is not a reputation. Furthermore, the same belief of multiple agents is not a reputation if these agents do not possess this believe about themselves.

For an *Agent*, who is subject to behavior estimate, for any estimator agent *EstAgent*, for any *Agent's* mental state S (the set of mental formulas) there is predicted mental state, S_p such that the following holds. If *EstAgent* believes that $(S \rightarrow S_p)$, then he also believes that any other agent, who claims his belief of *Agent's* next mental state, does the same prediction.

The mental state constraint *reputation(Agent, S, Sp)* is the reputation, if its application to a current mental state of an agent gives the prediction of his next mental state in accordance to the following.

$\forall$ *Agent* $\forall$ *EstAgent*
believe(EstAgent, reputation(Agent, S, Sp)),
believe(EstAgent,
($\forall$ *OtherEstAgent reputation(OtherEstAgent, S, Sp))).*

Having the assumption that each agent action is caused by his mental state, we do not need to take physical states into account to predict the agent behavior unless the modality over an agent's physical capability comes into play.

To introduce the concept of pretending, we first use the operator notation for knowledge and pretending. The presentation of the axioms for pretending, expressed in terms of modal operators, will be followed by the definitions in terms of metapredicates. This is another illustration that metapredicates deliver sufficiently expressive language to express such concepts as pretending. We denote by P_iF the fact that agent i pretends that the fact F holds; K_iF denotes the knowledge of fact F by the agent i. We do not use special symbols here to express the observation that the agent i **pretends for** another agent j; we express this fact indirectly.

1) General definition: an agent i pretends to the agent j that the fact F holds if he knows that j will understand the pretending: a) i knows that j knows that i pretends, b) i knows that F does not hold and that j knows that F does not hold, c) i assumes this pretend will be accepted.

$K_iK_j\,P_i\,F \;\&\; K_i\,not\,F \;\&\; K_i\,K_j\,not\,F \rightarrow P_i\,F$.

2) The pretend addressee either accepts the pretend (pretends that he knows the fact) or reject it (not know it).

$P_i\,F \rightarrow P_j\,K_j\,F \vee not\,K_j\,F$.

3) If an agent i pretends that F_1 holds for agent j and pretend that F_2 holds for agent m, and j can inform m about some fact G (in particular, it could be that $G=M1$), than i has to keep pretending that the conjunction of F_1 and F holds

$K_j P_i F_1$ & $K_m P_i F_2$ & $(K_j G \rightarrow K_m G) \rightarrow F=F_1$ & F_2 & $P_i F$.

4) If an agent i pretends that F_1 holds and agent j pretends that F_2 holds, and this pretending has been accepted by both of them, than both of them aware that neither F_1 nor F_2 holds.

$P_i F_1$ & $P_j F_2$ & $P_j K_j F_1$ & $P_i K_i F_2 \rightarrow K_i$ not $(F_1$ & $F_2)$ & K_j not $(F_1$ & $F_2)$.

5) An agent can pretend only about his own knowledge $P_i K_j F \rightarrow i = j$.

We proceed to the metapredicate clauses for pretending.

1') *pretend(Who, Fact) → inform(Who, Whom, Fact), know(Who, know(Whom, pretend(Who, Fact))), know(Who, not F), know(Who, know(Whom, not Fact)).*

2') *pretend(Who, Fact) → inform(Who, Whom, Fact), (pretend(Whom, know(Whom, Fact); know(Whom, Fact)).*

3') *inform(Agent, Agent1, F_1), pretend(Agent, F_1), inform(Agent, Agent2, F_2), pretend(Agent, F_2), (inform(Agent1, Agent2, F_2); inform(Agent2, Agent1, F1)) → pretend(Agent, F_1 &F_2).*

6) The definition of pretending, preserving consistency: If *Who* has started to pretend that *SmthBefore* had hold and he knows, that *SmthBefore* implies *Smth*, *Who* has to continue pretending that *Smth* holds.

```
pretend(Who, Whom, Smth):-
        pretend(Who, Whom, SmthBefore),
      know(Who, (know(Whom, SmthBefore→ Smth)))).
```

To conclude the Section, we remind, that **if the agents pretend, their reputation is not affected.**

1.5 The multiagent scenario with conflict resolution

We present the scenario with the conflict of three agent, concerning the digging up a field. It is self-contained: all concepts, necessary for conflict resolution, are enumerated. We refer the reader to (Galitsky, 1998a) for more details concerning the settings for a formal scenario and MS reasoning for linking its components. There are *KU* (general knowledge), *KS* (specific knowledge), *H* (history of multiagent interactions) and *R* (final reaction of the chosen agent).

Once during a World War I a soldier receives a letter from his wife saying that it's a time to dig up a potato field, but nobody can help.

The husband writes the letter back " Do not dig up the field. There is my gun hidden there"

In a week, the wife responds: "The whole field is dug up by the investigator. I started to plant potatoes".

The problem domain includes the knowledge that letters were subjected to review by police and that the weapon ownership was prohibited.

This is the definition, how an investigator *investig* can obtain knowledge of *Matter* reading the *Letter* from *Smb* to *SmbTo*:

inform(Smb, SmbTo, Letter). This *Letter* mentions the *Object*, which is the subject of investigator's search; the investigator wants (himself or somebody else) to find this Object: *want(investig, find(Who, Object))*. The investigator extracts from the *Letter* all information *Matter* about this *Object*. Syntactically, *Matter* consists from all the terms from *Letter* which contain *Object*. This definition means that the investigator gets all information about his *Object* of interest (including the other agents) from any *Letter* and agents *Smb* and *SmbTo*.

```
(KU.1) know(investig, Matter):- agent(Smb),
agent(SmbTo),
        Smb\= investig,SmbTo\=investig,
      inform(Smb, SmbTo, Letter),
      want(investig, find(Who, Object)),
   %The part of the clause below performs the search of
   %terms Matter in Letter, containing the argument Object
    expand(Letter, Letters), var_const(Letters, Lettersc),
    member(Object, Lettersc),
    member(Matter, Lettersc), Matter \= Object,
    Matter=..TermArgs, var_const(TermArgs, TermArgsc),
          % there is canonical formula conversion to
          % analyze the occurrence of a specific term
    member(Object, TermArgsc).
```

An agent, who wants to find an *Object* and knows that this *Object* is located in a *Place*, digs this *Place*. The first clause, which expresses the digging condition, is the following:

```
(KU.2) dig(Investig, Place) :-
    know(Investig, locate(Object, Place)),
    want(Investig, find(Who, Object)).
```

Below we conventionally use the verb tenses in the definition comments, not expressing the temporal relationships explicitly.

An agent *Smb* responds to another agent *Addressee* to his request *inform(Addressee, Smb, Matter)* that nobody can dig a *Place*, if *Smb* failed his plan concerning *Whom* to dig this *Place*:

```
(KU.3) respond(Smb, Addressee, not(agent(Who),dig(Who,
Place))):-
        agent(Smb), not plan(Smb, dig(Whom, Place)),
        inform(Addressee, Smb, Matter),
          %below is check that Matter was about dig
         expand(Matter, MatterE),
         member(dig(Agent, Place), MatterE).
```

This definition is required for direct dissipation.

We present the definition for the situation, when an agent is asking another agent to act conversely to his initial will. After an agent *Who* received from *Whom* the *Letter* with the intention of *Whom*, *Who* asks *Whom* to perform an *Action = not Intention*, because of the *Reason*. The Reason is something that the competing agent *Investig* does not

know, but *Whom* has to know to logically link *Reason* and *Action*. The latter procedure resumes the clause.

(KU.4) ask(Who, Whom, (Reason, Action)) :-
inform(Whom, Who, Letter),
clause_list(Letter, Letters),
member((want(Whom, Intention)), Letters),
(Action= (not Intention)),
want(Who, not know(Investig, Reason)),
% find linking clause
clause(hDissip, H), assert(H),
inform(Investig, Who, Info),
clause(know(Invest, Reason), RBody),
clause_list(RBody, RBodys),
member(Info, RBodys), member(Intention, RBodys),
Intention \= Info.

We proceed to the definition of an agent's planning of performing an action (here - dig) by another agent *Who* or himself. The easiest way is when *Who* (has already) performed digging (or ready to do it):

(KU.5) plan(Smb, dig(Who, Place)) :-
agent(Who),dig(Who, Place).

Here we ignore the verbs' tenses to simplify the scenario consideration, eliminating the temporal reasoning.

Another possibility is when the other agent *Who* wants to dig, and *Smb* himself knows that he will not (cannot) dig:

plan(Smb, dig(Who, Place)) :-
want(Who, dig(Who, Place)), Who \= Smb,
know(Smb, not dig(Smb, Place)).

The third option is sometimes possible. If neither himself nor the other agents done this action and do not want (cannot) perform it, *Smb* can introduce some false information to make the other agent *Who* do that action (deceive this agent).

(KS.1) plan(Smb, dig(Who, Place)) :- agent(Who),
Who\=Smb, want(Who, not dig(Who, Place)),
not want(Smb, not (dig(Who, Place)))
deceive(Smb, Who, Fake, dig(Who, Place)).

We introduce the concept of deceiving, when an agent *Who*, achieves the action *WhoWill* to be performed by *Whom* by means of informing *Whom* that *Fake* holds. *Who* should want *WhoWill*, besides, it should not be true that *Who* want *Whom* not to perform *WhoWill* (PROLOG peculiarity of the negation as failure) and *Whom* should not want to do *WhoWill*. Besides, *Who* should know that *not Fake*.

(KS.2) deceive(Who, Whom, Fake, WhoWill):- (WhoWill =
dig(Whom, Place)),
want(Who, dig(Whom, Place)),
not want(Who, not dig(Whom, Place)),
want(Whom, not (dig(Whom, Place))),
know(Who, not Fake),
%we process the clause, which links Fake and WhoWill
clause(dig(Whom, Place), Body),
clause_list(Body, Bodys),
member(know(Whom, Fake), Bodys),
% the part below verifies that the deception was
% successful for Who, because Whom accepts
% Fake and performs WhoWill
clause(reaction, R), assert(R),
know(Whom, Fake),
WhoWill.

We note that only the investigator's digging condition actually needs specification what action (dig) to perform. To obey the generality of knowledge representation, the other definitions should contain the quantifier "for all Action " over the second order variable for Action. We write *dig* instead of *Action* in our definitions to simplify the syntax, though higher-order PROLOG extension is available for this example.

The *H* component enumerates the agents:

agent(husband). agent(investig). agent(wife).

The investigator wants to *find* a *gun*. He does not want to dig a *Place*.

(H.1) want(investig, find(investig, gun)).
(H.2) want(investig, not dig(investig, Place)).

The *wife* writes to husband that she wants somebody to dig her field and nobody (can) do it.

(H.3) inform(wife, husband, (want(wife, dig(Smb, field)),
not dig(Smb, field))).

The *husband* wants somebody to *dig* this field, he knows he cannot do it himself and he does not want his *wife* to do it:

(H.4) want(husband, dig(Smb, field)).
(H.5) know(husband, not (dig(husband, field))).
(H.6) want(husband, not (dig(wife, field))).

The *husband* knows that there is no *gun* in the *field*

(H.7) know(husband, not locate(gun, field)).

The scenario reaction is the husband's decision how to dig the field:

?-plan(husband, dig(Who, field)).

He chooses the following:

(R) inform(husband, wife,(locate(gun, field),
not dig(wife, field))),

to inform the wife that the gun is located in the field and that the wife should not dig it.

The usual husband's respond would be the impossibility to have anybody to dig the field

(R_{dissip}) respond(husband, wife, not (agent(Who),
dig(Who, field))).

To transform this scenario into a trivial one, we deduce what would be naturally implied by the fact that the *husband* asks his *wife* not to *dig* the *field*. The *husband* does not want his *wife* (or anybody else) to dig the *field* because he does not want the investigator to know the *gun* location.

(H_{dissip}.4) want(husband, not know(investig, locate(gun, field))).

We conclude, that the *investig* should have informed the *husband* about his interest to the *gun*.

$(H_{dissip}.8)$ *inform(investig, husband, want(investig, find(investig, gun)))*.

This scenario is available as a PROLOG demo, where the main calling predicate is the husband's planning (See http://dimacs.rutgers.edu/~galitsky/MS/dig).

2 Application: the scheduling in the conditions of conflict between automatic and human agents

The following conflict resolution problem is suggested as a benchmark (http://www.cert.fr/fr/dcsd/PUB/AAAI99/scenario).

An initiator suggests a set of possible time intervals for a meeting to a set of invitees. They in turn have to check their calendars for an interval that fits best and they have to inform the initiator about their result. In most cases, especially if more than two invitees are involved and the set of alternatives is small, it is not possible to generate a mutually accepted appointment date. Hence, the initiator has to resolve the conflict by applying heuristics. The initiator has to convince some of the invitees to reschedule their calendar dates to free a certain interval.

We will present the practical application of the appointment scheduling problem, when the assignment of broadcast commercials will replace invitees arrival times, human scheduling agents will replace the invitees themselves and the automatic agent will replace the initiator. This domain involves the "customer profile" of a particular broadcast advertisement and its priority in respect to other advertisements and in respect to the broadcast program breaks they fill.

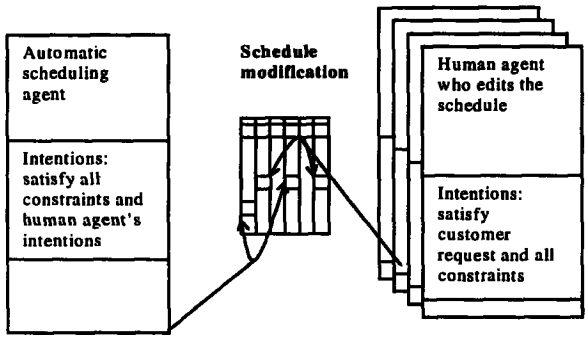

Fig.2. The scheme of interaction of the automatic agent with the human agents. The temporal constraints of the various nature are set by the customers (human agents) in addition to the standard ones. Conflicts between the agents arise while modifying the schedule (each day is a column, an ad can be moved from one brake of one day to the other break of the same or other day).

2.1 The scheduler for broadcasting industry

In this Section we present the background of our scheduling domain. From the reasoning prospective, the scheduling algorithm itself, taken separately from the multiagent representation, comprises the predicates for **actions** and **states**. We present the detailed description of our domain of broadcast advertisement scheduling to highlight the peculiarities of the multiagent interaction and conflict resolution.

Assigning the commercials (ads) to the time intervals of a broadcasting program (breaks) is one of the most important and complicated operations in the creation of TV or radio program schedule. Variety of constraints and optimization criteria together with the "traffic director"(agent) - scheduler interactions are the reasons for the special requirements to the deductive means of the system. The scheduler operates in real time, when after assignment to a break, an advertisement (ad) can be aired, bumped, canceled, replaced in various combinations. It occurs while a "traffic director" and assistant personnel negotiate with the advertiser and manipulate with the schedule. This negotiation procedure distinguishes from that of presented in the majority of studies, considering negotiation and resource distribution towards the common goals (see, for example, Ephrati and Rosenschein 1996).

We pose the scheduling problem as the construction of the mapping from the finite set C (*ads*) into the finite set B (*breaks*), the ordered set of time intervals where ads are substituted into).

C is induced by the *contract* set N. Each contract N contains the set of constraints P_N, common for its ads. They include the ad duration, time interval within a day, valid days of week, separation between the ads of the same contract and of the same type, running within a particular broadcast program, ad frequency, set on a daily or weekly basis, and ad type (weather, political, etc). *Constraint formula* is a conjunction of predicates, expressing these constraints.

The break properties $B^1,..., B^m$ are the break duration, break start time, belonging to a broadcast program, and combination in a consecutive series of breaks with preferred duration. The break properties are the *arguments* of a constraint formula for an ad, which could possibly be scheduled into this break. For example, to express that the car dealer ad c is scheduled into the 60 second weather break b, we write

car_dealer(c, weather(b)), length(b,60).

Hence, *to map an ad into a break* means to satisfy the constraint formula for this ad by the substitution of the arguments (properties) of this break. The constraint formula for an ad of the contract N, scheduled into the break b, is the first-order formula

$\varphi_N = p_c^{N1}(B^s,..., B^t) \& ... \& p_c^{Nn}(B^u,..., B^v)$,

where B^s,B^t,B^u,B^v are the properties of the break b.

p_c^{N1} ,...,p_c^{Nn} are the predicate constants of the ad c, and

$B^x...B^t,...,B^u...B^v$ are their variables (properties of b).

To express the constraints for given ad c from the contract N and another ad c' from the contract N', we introduce the metapredicate $p^{N_1}(\varphi_{N'},..., B^s)$. For example, to express the fact that the given supermarket ad should be at least 20 minutes (1200 seconds) after any banking ad, scheduled in the news break, we write

after(*supermarket(c), banking(c', news(b)), 1200*).

after is indeed a metapredicate, because it analyzes the syntax structure of the second argument (which ranges over formulas) to be satisfied. If the second argument was the constraint formula for an unscheduled break, ***after*** would behave differently (it would mean the certain separation from a break, not from an ad).

Finally, the general case for the constraint formula for an ad, including the limitations for the neighboring ads c',... is the following:

$\varphi_c = p_c^{N_1}(\varphi_{c'},..., B^t) \& ... \& p_c^{Nn}(B^u,..., B^v)$,

where $p_c^{N_1}$,...,p_c^{Nn} are the metapredicates.

The *priority* function is introduced to express the competition of two ads, c_1 and c_2 , for one break b. The *absolute priority* is assigned by each contract to all its ads. The relative priority component of an ad c depends on the break b it has been scheduled into and, possibly, on properties of the neighboring ads $p_{c'}^{Nn}$. The impact of the ad constraints and break properties is specific for each broadcasting station, so the scheduling algorithm must be able to adopt a particular way of the computation of relative priority. We have the following function for priority:

priority(c,b)= absolute_priority(c)+
relative_priority(c,b, C_n).

Priority parameter introduces the following limitation. For each ad c and break b it has been scheduled into, there is no *bumped* ad c_{bump} with the higher priority which could be substituted instead of c into b. After the scheduling occurs, each ad is either scheduled (assigned to a break) or *bumped* (not currently assigned to a break).

The *scheduling criterion* is the maximal revenue for all the ads running for the fixed time unit (day, week, and month).

To obtain the *scheduling rule* from the constraint formula, we divide the formula satisfaction into two steps. The first (search) step serves as a criteria to find a pretender for a break, and the second (match) actually verifies the rest of constraints, expressed by this formula. Formally, we assign the status for each variable of a constraint formula from the set {*search, match*}, where the *search* status variables are instantiated while choosing the break on the first step, and the *match* status variables are instantiated on the second step. This construction becomes trivial if the constraints could be expressed in the first-order language (the constraints link a single ad with a single break).

Dividing the instantiation into two steps, we provide the expressive means of the schedule rule language, transitioning from the declarative constraint formula to multiple procedural scheduling rules. For any computationally tested set of constraints in the MS language, instantiation states {*search, match*}are sufficient for derivation of the rule delivering the optimal scheduling.

The result of a scheduling rule application for a time unit (hour, day or week) is the division of the set of all ads into scheduled and bumped. A particular rule is capable of successful scheduling of the certain subset of the set of all ads. Some other rule is required to schedule the currently bumped spot. The sequence of scheduling rules is called the *scheduling algorithm*; the algorithm is *valid* if an addition of a new rule to the end of the sequence does not increase the revenue by the amount, higher than a fixed *revenue threshold.*

We call the scheduling algorithm *ideal*, if it delivers the optimum of the scheduling criteria and satisfies the priority limitation.

To define the concept of scheduling algorithm, we first fix the language of scheduling rules, which includes language-object for ad-break limitations and metalanguage for ad-ad limitations. Then we freeze the ad constraints, break properties and the resume the choice of relative priority and actual revenue functions. The first rule of the *scheduling algorithm* covers as much contracts (ads) as possible for the highest priority. The second (third, etc.) rule schedules the lower priority ads and the bumped higher priority ones. Each consecutive rule handles the unsatisfied (bumped) set of ads of the previous ones (schedules the bumped ads).

We do not present the details of construction of the optimal algorithm, it could be implemented as an exhaustive search over the priority diapason for each k. Heuristics of the reduction of this search depend on the particular constraints and priority computation; they are not worth mentioning. See (Zweben, M 1994) for the iterative repair technique for CSP, applicable to our scheduling domain, if the intervention of human agents is excluded.

2.2 The modes of multiagent interaction while scheduling

Any scheduled ad can be subject to the manual reassignment by a human agent. His or her motivations for this manipulation is usually unknown for the automatic agent, it can reconstruct the corresponding mental state of the human agent if it occurs on the regular basis. One human agent can assume, for example, that another human agent has the preferred breaks to schedule the ads of the important advertiser. Hence, the break history, which includes the sequence of its states and breaks, it has been assigned to, and corresponding agents, performing these operations, serve as a basis to reconstruct the mental states of agents.

The reconstruction procedure contains two passes

through the totality of all ads. The first pass is based on the attempt to establish the basic mental states and making required assumptions. The second pass tries to reveal the more complex behavior patterns, confirming the maximal number of these assumptions. This operation is performed through the whole historical sequence of all ads; note that different ads can be involved in the same assumptions.

Before starting functioning in the particular environment, the system generates the rules and their order to derive the scheduling algorithm, given the initial set of contracts. Then, given the new portion of contracts, the system creates the schedule for as many time units as required.

The ad scheduling system performs real-time evaluation of the algorithms. On the unit-by-unit basis, the older contracts expire and the newer are added to the system. Having the current scheduling algorithm, the system reorders the rules to adjust to the specifics of new contracts. The order of rules in the scheduling algorithm is computationally verified to be sufficient in the sense of approaching the revenue threshold.

Receiving a new portion of ads, it is necessary to unschedule some ads to redistribute them together with the new ones. It is impossible to unschedule all ads because of the possible loosing of implicit scheduling intentions (for example, after manual editing) and because of the performance drop in case of multiple overall rescheduling. So it is necessary to unschedule 5-10% of all ads; the system must be able to construct unscheduling conditions on the fly.

This is the conflict between the necessity to schedule new ads and human agents' intention to change the location of scheduled ads in as low degree as possible. To resolve this conflict, the automatic agent tries to reconstruct the mental state of the human agent, while assigning a particular ad. The automatic agent then unschedules all ads with the reconstructed mental states.

Under interactive scheduling, results of the automatic scheduling with the incremental feeding by new contracts are subject to the manual editing by a human agent. In other words, the automated scheduling scenario is intervened by the agent manipulations in accordance to the rules, unknown for the system. The system is thus designed to compensate the agent action on one hand and to develop the predicting capabilities for acceleration of the interactive scheduling on the other hand. It is performed by means of the reconstruction of mental attributes, based on the history of the ad states.

Interactive scheduling is considered in the scenario framework (Galitsky 1998a). An *action* of scheduling *agents* is an application of a scheduling algorithm. Having the results, the system is capable of building the explanation metapredicate to restore the agent's mental attributes *S*, if manual editing has a limited complexity and permanent motivations. Interactive scheduling assumes the common intentions of the system and human agent, and both of them know it.

The most complicated scheduling scenarios arise when multiple agents participate in the scheduling with their specific motivations and rules. The human agents are represented to have intentions concerning the scheduling rules and their knowledge and other agents' knowledge, knowledge about the scheduling constraints, their intentions, their own knowledge and each others. Besides the automatic scheduling, the system has to perform the modeling of the human agent behavior to compensate the inconsistency of scheduling results.

For example, an agents *bm* (Broadcast program manager) wants to assign the higher priority ads to a weather breaks, and he knows, that the other agent *td* (Traffic director) wants more political ads on Friday afternoons. *td* wants the third agent *ad* (assistant traffic director) to move the car dealer ads to the morning drive breaks, and he knows, that *bm* assumes there are too many ads to have enough available (unscheduled) breaks on Monday morning. Our approximation of this scenario includes the following:

want(bm, schedule(bm, (higher_priority(c, weather(b)))).
know(bm, want(td, schedule(td, (political(c,
friday(b),afternoon(b))))).
want(td, schedule(ad, (not (car_dealer(c, morning)),
car_dealer(c, evening)))).
know(td, know(bm, not morning(b))).

The reader can think of these representations as the results of mental attributes reconstruction

The system obtains as much of this information as it is possible judging on the agents' actions. Finally, the task for the system is to schedule the car dealer ads on Monday such that the schedule is least affected by the agents' editing.

Results and discussion

Advanced modes of the automatic scheduling agent, presented above, were created in MS PROLOG and finally written in C++ as the optional utilities for the main application. The scheduler prototype was implemented in Marketron, Inc. and tested at various broadcast stations to compare the performance and constraint satisfaction accuracy with the broadcasting scheduling systems, currently available on US market. The implementation of advanced multiagent reasoning allowed the significant increase of the overall scheduling quality after the system training in the particular personnel environment.

We showed that the defined mental concepts match the general intuition and technical commonsense reasoning traditions on one hand and allow adequate application on the other. Computational experiments show that the number of such concept definition, functioning simultaneously (within a single context), for any practical application lies below a hundred. Since each concept potentially realizes multiple meanings, and each meaning

combination requires a specific search, operation with more than a hundred of concepts would cause the flexibility drop. The latter is hardly desirable for the automatic multiagent system, where the conflict resolution results are frequently more important than the system adaptability to function multiple domains simultaneously.

Besides, the number of mental concepts, participating in the reasoning under the multiagent constraint resolution, must be below the critical value such that the following situation does not occur. If two distinguishing concepts are *attenuated* to the same meaning (see Galitsky 1998a for details), the model of multiagent behavior can become invalid (though the reasoning might still preserve consistency).

Three levels of conflict complexity were presented in the study:

1) If a conflict does not involve inconsistent mental attributes, it can be resolved in terms of the traditional constraint satisfaction methodology.
2) To resolve a conflict with the agents' intentions, knowledge, and beliefs, one agent is able inform another agent, convince or deceive him, or present some explanation of a conflict reason. It occurs in accordance to the clauses, described in the Section 1.
3) A conflict, where the agents demonstrate the behavioral patterns of the higher complexity, involving mutual reasoning about their mental states with a deep reflection or prediction of many steps in advance, needs specific considerations. The partial case was addressed, where the agents were concerned with their reputations, and displayed the pretending type of behavior.

The generic definitions for these concepts are helpful for the design of intelligent software, where the modeling of user's mental attribute is important (compare, for example, with (Decker et. al., 1997, Zlotkin & Rosenschein 1996).

References

1. Robert, S., Amant, R., Cohen, P.R., 1997. Evaluation of a Semi-Autonomous Assistant for Exploratory Data Analysis, in *International Conference on Autonomous Agents'97.*
2. Kraus, S., Zlotkin, G. 1995. Multiagent negotiation under time constraints AI 75(2): 297-345.
3. Galitsky, B.A. 1998a. The formal scenario and metalanguage support means to reason with it, *DIMACS Tech. Report #98-27*, Rutgers University.
4. Galitsky, B.A. 1998b. The concept of formal scenario in multiagent constraint programming. *Proceedings of the DIMACS Workshop on constraint programming and large-scale discrete optimization*, Rutgers University.
5. Zlotkin, G. and Rosenschein, J. S. 1996. Mechanism design for automated negotiation, and its application to task oriented domains. *AI 86*, 195-244.
6. Cohen, P.R. and Levesque, H.J. 1987, Intention = Choice + Commitment, in: *Proceedings AAAI-87*, Seattle WA 410-415.
7. Das, S.K., Fox, J., Elsdon, D., and Hammond, P., *1997.* Decision making and plan management by autonomous agents: theory, implementation and applications, in: *International Conference on Autonomous Agents'97.*
8. Ephrati,E., and G. and Rosenschein, J. S. 1996. Deriving consensus in multiagent systems *AI 87* 21-74.
9. Kraus, S. 1995. Negotiation and Cooperation in multi-agent environment, IJCAI-1995.
10. Decker,K., Pannu, A., Sycara,K. and Williamson, M.,1997. Designing Behaviors for Information Agents , in *International Conference on Autonomous Agents'97.*
11. Cholvy, L., 1998. Summary of *Conflict is inevitable, tolerable, and constitutive* discussion of ECAI Workshop "Conflicts among agents: avoid or use them", www.cert.fr/fr/dcsd/DCSDPUB/ECAI98/synthese/node6
12. Klein, M.1993. Supporting Conflict management in cooperative design teams Journal on Group Decision and Negotiation. Special Issue on the 1992 Distributed AI Workshop. Volume 2, pp. 259-278, 1993.
13. Zweben, M 1994, Scheduling and rescheduling with iterative repair, *Intelligent scheduling* , Zweben & Fox, eds., *Morgan Kaufmann Publ.*
14. Fagin,R., Halpern,J.Y., Moses,Y., Vardi,M.Y. 1995 Reasoning about knowledge MIT Press, Cambridge,MA, London, England.
15. Konolige, K.1986. A deduction model of belief. *Morgan Kaufmann Publ.*
16. Samet, D. 1987. Ignoring ignorance and agreeing to disagree. MEDS discussion paper, NW Univ.