# Negotiation As Conversations About Acceptable Behavior

Mihai Barbuceanu
Enterprise Integration Laboratory
University of Toronto
4 Taddle Creek Road, Rosebrugh Building
Toronto, Ontario, Canada, M5S 3G9
mihai@ie.utoronto.ca

**Abstract**

Negotiation is an interactive goal discovery process aimed at eliminating goal conflicts that impede joint or individual work. In this paper we report on ongoing work addressing the integration of two technologies for attacking negotiation. The first is conversation based agent interaction, which provides the means to specify and execute long running structured interactions where agents exchange communicative action based messages. Using this, agents communicate their requests, proposals, counter-proposals, etc. about other agents' behavior. The second is a 'negotiation solver', a formal reasoning engine based on a formal behavior description language (in terms of which requested behavior is described) and several constraint propagation methods allowing the discovery of behaviors that do not violate the constraints of the involved parties, or violate them in a manner that can be characterized and understood (e.g. minimally). These enables parties to better understand the current state of the negotiation process and how to proceed further. The formality of the methods employed offers guarantees (like minimality of violation) which will increase the level of trust of the parties using this method. We illustrate the approach with examples from the telecommunications industry.

## 1 Introduction

Current models of collective behavior like e.g. joint intentions [5] start from the premise that agents have already determined they have common goals which they wish to pursue together. Such a determination once made, the joint intentions model for instance stipulates how agents should behave towards each other while working for the common goal - for example, an agent should inform the others if it has discovered the common goal is unachievable or has already been achieved. But how do agents discover if they have common goals in the first place? Or, if agents' individual goals are in conflict impeding agents ability to achieve them, how do agents reformulate their goals in order to avoid the conflicts? A seller may start with the goal of selling item `I` for price `X`, while a buyer's initial goal may be to have `I` paying `Y`. Unless `X=Y` these are inconsistent. A customer `A` may set up a phone call to a customer support agent `B`, but if `B` can not take the call it will be impossible for `A` to obtain the desired service. In the end, the seller and buyer may agree on a price `Z` (normally different from both `X` and `Y`), while `A's` call may be forwarded to `C` who is available and may provide the same service as `B`. In both situations, in spite of initially conflicting goals, agents have subsequently discovered and adopted common or non-conflicting goals, which enabled them to pursue cooperative action. In both situations, these goals might have been discovered through negotiation.

Negotiation is an interactive goal discovery process employed by autonomous agents to create common goals for cooperative action or to eliminate conflicts among individual goals that would otherwise obstruct their work in the community. We see two main facets of negotiation. The first is a *process* facet, that deals with the structure of the interactive communication among parties. Parties request, propose, accept reject, etc. goals and behaviors until some agreement is reached or negotiation fails. The second facet has to do with agents *reasoning about goals and behaviors*. What are the consequences of adopting or not this goal? What useful goal can be adopted that does not violate somebody else's requests? Is there anything acceptable in this proposal, etc.?

In this work, we discuss an architecture for negotiation that integrates two technologies covering both aspects. The first is our conversational interaction technology [1], enabling agents to carry out multiple structured exchanges in terms of communicative actions. The second is our representation and reasoning about obliged and forbidden behavior [2] which we have recently extended to deal with a wider class of behaviors and with algorithms that discover behaviors that guarantee certain properties like *minimal violation* of specified obligations and interdictions or various forms of consistency with these.

## 2 The Negotiation Model

From the 'negotiation process' point of view, we assume that agents interact by carrying out conversations, that is long running structured interactions where agents exchange messages expressed in terms of communicative actions like *request, propose, accept reject*, etc. Several communicative action languages for agent communication exist, including KQML [6] and the FIPA ACL [7]. We have built a coordination language [1] that provides conversational constructs - conversation plans, conversation rules, etc. - that can be used in conjunction with *any* communicative action based communication language. Conversations allow negotiating parties to order and structure the exchange of requests, proposals, counter-proposals, rejections, accepts, etc. according to existing plans that express known interaction conventions that may lead to the discovery of joint or non-conflictiong goals and common courses of action.

From the 'reasoning about behavior' point of view, we address expressing the content of requests, proposals, etc. and especially reasoning about this content. Our assumption here is that a useful class of negotiations are naturally expressed in terms of constraints on agents' behaviors, where a behavior generally consists of complex actions expressed in a formal action language. For example, when agent `A` tries to set up a phone call to agent `B`, it may request that it be *not forwarded* to certain numbers if `B` will not take the call directly. Or a customer agent may require an order to be *delivered at a certain date and time*. Or a customer may try to *return* merchandise to the store and *refuse to exchange* it with anything similar. Our architecture provides a behavior description language allowing the specification of complex actions as choice, parallel and sequential compositions. To this, we add definitions of obliged and forbidden behavior as behavior that induces a cost if not performed (obligation) or a cost if performed (interdiction). To reason about obliged and forbidden behavior, we introduce cost propagation semantics that, given initial obligations and interdictions, propagates new costed obligations and interdictions to other actions in an action network. This allows agents to choose behavior based either on minimizing the total violation cost to be paid (assuming in general not all obligations and intedictions can be satisfied) or by adopting other forms of cost consistent behavior.

In general, assume `B` receives a message from `A` containing a set of obligations and interdictions constraining its behavior. Figure 1 shows what happens inside `B` when such a request arrives. First, the constraints in the request are translated from the external interlingua to their internal counterpart, verified for consistency and locally represented as requested behavior. Second, the consequences of the requested constraints are determined, especially in relation with `B`'s own (local) obligations and interdictions and with other obligations and interdictions it may be committed to. This process, called deontic propagation for reasons that will become clear later on, determines which of the requested constraints can be satisfied and which can not. In this process, if `B` encounters conflicting obligations (interdictions), it decides which to violate based on minimizing the total violation cost associated with the violated constraints or on other cost consistency conditions. If none of the constraints imposed by `A` is violated, it means that `B` can consistently integrate `A`'s request into its own behavior and normally `B` will continue with scheduling the requested actions and then with executing them. If however some of `A`'s constraints have to be violated, (and as shown in figure 1 this may as well be detected later on during execution as a consequence of new constraints dynamically generated by executed actions) `B` can not satisfy the request and will reveal the violated constraints to `A`, perhaps with some explanations for the reasons of failure. `A` may now revise its request in various ways. It may drop constraints, it may add constraints, or it may raise or lower costs (within the limits of authority conferred by its role) to make `B` violate other constraints. The revised set of constrains is sent back to `B`,which will repeat the same cycle. The process may end with both agents agreeing on a set of constraints that `A` still wants and `B` can satisfy - a common goal - or may terminate before any agreement is reached. The structure of the interaction is

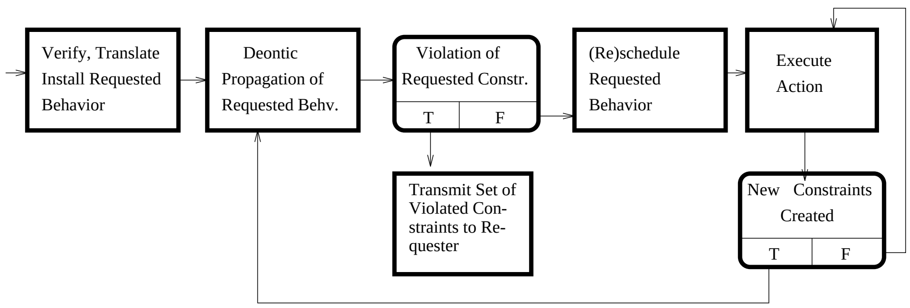


Figure 1: Negotiation Architecture

enforced by having agents execute appropriate conversations described in our conversation formalism.

# 3 Agent Technology in Support of Negotiation

In this section we review our approach in more technical detail, expanding on the issues that have been mentioned previously.

*Behavior Description Language.* Agents act and this means we need a language to describe, reason about and execute agent behavior. Our language contains two types of actions, *composed* and *atomic*. Composed actions consist of other (sub)actions, while atomic actions do not. Both types of actions have `[earliest-start, latest-end]` execution time intervals within which they must be executed (discrete time assumed). Atomic actions have specified finite durations. We allow three kinds of compositions. *Sequential* compositions, $a = seq(a_1, a_2, ...a_n)$ denote that all component actions $a_i$ must be executed in the given order, inside the time interval of $a$ (their superaction), without temporal overlapping. *Parallel* compositions, $a = par(a_1, a_2, ...a_m)$ denote that all component actions must execute within the time interval of $a$, but with temporal overlapping allowed. Finally, *choice* compositions, $a = choice(a_1, a_2, ...a_p)$ denote execution of only a non-empty subset of subactions within the time window of the superaction, also with overlapping allowed.

Executionwise, choices have 'or' semantics in that a choice gis 'on' - meaning will be executed and writen $On(g)$ - iff at least one component is on and 'off' iff all components are off. Sequences and parallels have 'and' semantics (on iff all components on, off otherwise), but the components of a sequence are extended to include besides subgoals the binary orderings $(a_i, a_{i+1})$ meaning $a_i$ has to be executed *before* $a_{i+1}$. This requires a new rules for inferring the on-off status of sequences and orderings, for example if $On(a_i, a_{i+1})$ then $Off(a_{i+1}, a_i)$ and the corresponding propagations to/from the sequences containing these orderings.

An action $a$ *forbidden* with cost $c$, writen $F(a, c)$, is defined as one whose execution results in a state of violation that induces a cost $c$ for the executing agent. Similarly, an *obliged* action, $O(a, c)$, is one whose non-execution induces a cost and a *permitted* action is one which is not forbidden. In on-off terms, if an action is on, then all interdiction costs are paid, while if it's off all obligation costs will be paid.

We have also defined an axiomatization allowing the propagation of obligation and interdiction costs across action networks. Without getting into the formal details of these (to be presented elsewhere), the axiomatization allows us to perform various cost inferences, some of which are informally as follows:

(1) If $On(g)$ and $g$ is an 'and' type composition (parallel or sequence) and $O(g, c)$ then $O(g_i, c)$ for each subgoal $g_i$ of $g$.

(2) If $Off(g)$ and $g$ is an 'or' type action (choice) and $F(g, c)$ then $F(g_i, c)$ for each subgoal $g_i$ of $g$.

(3) If $On(g)$, $O(g, c)$, and $g$ is a choice all of whose subgoals except one are off, then the single on subgoal $g_j$ is $O(g_j, c)$.

(4) If $Off(g)$, $F(g, c)$, and $g$ is a parallel or sequence all of whose subgoals except one are off, then the single on subgoal $g_j$ is $F(g_j, c)$.

As shown next, cost propagation semantics allows us to define an acceptability criterion for costed behaviors.

*Reasoning About Behavior.* Let $N = \{g_1, ...g_n\}$ be an action (goal) network. An on-off labeling of the network is an association of either 'on' or 'off' label to each action. The labeling defines an executable behavior (or is execution consistent) iff the labeling of each composed node is consistent with its execution semantics (as defined above). Let also $\{O(g_j)...\}$ and $\{F(g_k)...\}$ be a list of obligations, respectively interdictions on some goals in $N$. If N is executable (consistently on-off labeled), the total violation cost of $N$, $VC(N)$, is computed as the sum of obligation costs for all 'off' actions plus the sum of interdiction costs for all the 'on' actions. If $VC(N) = 0$ then all obligations and interdictions have been satisfied. Otherwise, some have been violated. Our main method of reasoning about behavior, called *deontic propagation* starts with an action network $N$ and an initial list of obligations and interdictions, and determines an executable behavior (consistent on-off labeling) that either optimizes or satisfies a given criterion. Here are some criteria.

(1)Minimization of the total violation cost $VC$ (defined as above). This finds the behavior that maximally satisfies all the stated obligations and interdictions. When the obligations and intedictions come from other agents, this can be construed as the maximally 'social' behavior of the agent, that violates others' requests as little as possible. If we maximize $VC$, then we get the maximally 'anti-social' behavior of the agent, one that violates as much as possible others' requirements.

(2)Finding a cost consistent behavior. A behavior is cost-consistent if for every composed action $g$, if $On(g)$ then the total propagated obligation cost on the goal is greater or equal than the total propagated interdiction cost, and if $Off(g)$, the total propagated interdiction cost is greater or equal than the total propagated obligation cost. Cost propagation follows the rules mentioned previously. This criterion defines acceptable behaviors, where the choice of what to do or not is consistent with the costs associated with doing or not doing actions.

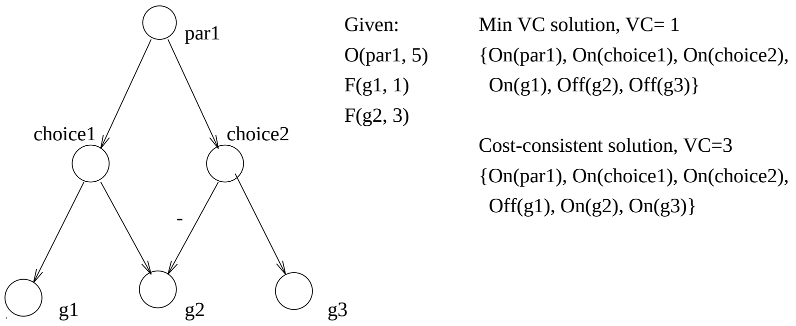


Figure 2: Deontic propagation

Figure 2 illustrates the difference between these solutions. For the given network (`choice1` and `choice2` are choices, `par1` is a parallel, the rest are atomic, and `g2` occurs negated in `choice2`) the minimum $VC$ is 1 (paying only the cost of $On(g1)$), and the solution where we pay the cost of $On(g3)$ although not minimal, is still cost-consistent because by propagation we have $O(choice2, 5)$, $F(choice2, 3)$ and $5 > 3$.

In the general framework described, this reasoning is intractable (the 'or' and 'and' type execution semantics plus the existence of negation allow a reduction from satisfiability). For the general case we use a *branch-and-bound* backtracking search method that can:

1. Minimize or maximize the total violation cost ($VC$) or total obligation or interdiction costs on specified subsets of goals. E.g. 'Find the labeling that minimizes $VC$ for the set of goals $G \subset N$'.

2. Satisfy user defined cost or other conditions. E.g. 'Find the labeling that has $VC < 5$ for goals $G \subset N$.

3. Any of the above, but with pre-imposed partial on-off labelings. E.g. 'Find min $VC$ for $N$ when $On(g_1)$ and $Off(g_2)$'.

With restrictions on the behavior description language (like elimination of negated subgoals or tree structured networks) we also have tractable methods, like the arc-consistency type method described in [2].

*Action Minimization, Scheduling and Execution.* Deontic propagation finds executable behaviors that optimize or satisfy given criteria. But for execution the agent also needs to know the final order of actions that satisfies all sequencing and resource constraints as well as the time windows in which each action has to be executed. These are achieved using a finite capacity, constraint based scheduler. Before scheduling, the agent also minimizes the number of actions to be executed, by determining a minimal set of actions that, when executed, will achieve all 'on' actions in the behavior. This is based on classifying all goals in a lattice for a subsumption relation (similar to classification as used in description logics). An action $a$ subsumes an action $b$, writen $subsumes(a, b)$, iff executing

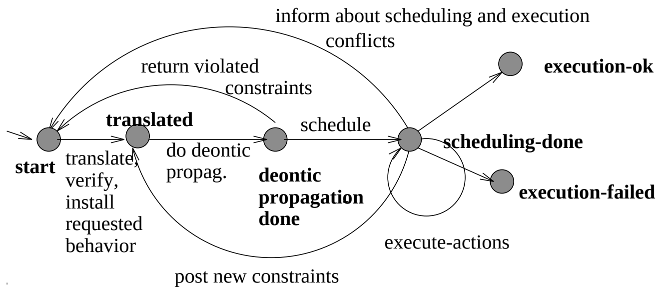


Figure 3: Conversation plan doing deontic propagation, scheduling and action execution

$b$ also achieves $a$. Finally, action execution is carried out by an executive that makes sure the real time is within the time window for each action and monitors the success or failure of each action.

*Conversation Plans.* To support flexible integration and combination of the above services (deontic propagation, minimization, scheduling, execution, etc.) in the negotiation architecture we use *conversation plans*. These are described in detail in [1]. Briefly, a conversation plan is a description of both how an agent *acts locally* and *interacts* with other agents by means of communicative actions. It consists of states (with distinguished initial and final states) and rule governed transitions together with a control mechanism and a local data base that maintains the state of the conversation. The execution state of a conversation plan is maintained in *actual conversations*. To decide which conversations to instantiate and to update the agent's data base when events take place, we provide *situation rules*. The top level control loop of an agent activates all applicable situation rules (suggesting conversations to initiate) and then executes new or existing conversations as appropriate.

The conversation plan in figure 3 shows a possible incarnation of the generic negotiation scheme in figure 1. It shows how an agent receives a set of obligations and interdictions that another agent requests it to satisfy, translates and verifies them, performs deontic propagation (going to state `deontic-propagation-done`) and, if no requested constraints are violated, plans and schedules the required actions (going to state `scheduling-done`). Then it executes the planned/scheduled actions in rule `execute-actions`, during which if new constraints are posted propagations will be redone. If no execution failure occurs, the plan ends in state `execution-ok`, otherwise it ends in `execution-failed`. If during deontic propagation the agent determines that it can not satisfy some of the requested constraints, or if actions can not be scheduled or planned, the violated constraints may be sent back to the sender for revision.

# 4 Negotiating Service Provisioning in Telecommunications

Consider the feature interaction problem, a general service creation problem in the telecommunications industry [3], on which we are working with industrial partners.. The creation and administration of connections can use various levels of functionality, or *features*, that provide different services to subscribers or the telephone administration.

Here are a few examples for the features that are usually available (modern telecommunication services may have many hundreds of such features):

(1) *Incoming Call Screening*: the calee will refuse all calls from callers in an incoming call screening list.

(2) *Call Forward*: the calee will forward the incoming call to another number.

(3) *Retry*: if the calee is busy, either the the caller or the callee will try to establish the connection later.

(4) *Outgoing Call Screening*: the caller does not allow to be connected to some specified directory numbers.

The feature interaction problem arises from the fact that often combinations of features interact in undesired ways, affecting the intended functionality of the provided services. For example, *Incoming Call Screening* and *Retry* may conflict if *Retry* is done without checking that the number belongs to the incoming call screening list - we shouldn't call back numbers that are not accepted in the first place. Similarly, *Call Forward* and *Outgoing Call Screening* may conflict if a caller is forwarded to a number that it does not wish to be connected to.

Assume now `A` and `B` are agents responsible for establishing voice connections amongst their users and consider figure 4 which shows the goal structure used by agent `B` when receiving a call from `A`.

*Scenario 1: Dealing with the above features.* `A` has *Outgoing Call Screening* for numbers #1 and #2, and `B` has *Incoming Call Screening* with `A` in its incoming call screening list (meaning `B` does not want to talk to `A` directly). The set of constraints that `A` sends to `B` is

```
{(obliged take-my-call :from A :cost 5)
 (forbidden ch1 :from A :cost 9)
 (define choice ch1 forward-to-Moldova forward-to-Hong-Kong)}.
```

When receiving this message, `B` first translates, verifies and installs the requested behavior. `take-my-call` is translated to `ReceiveCall`, `forward-to-Moldova` and `Forward-to-Hong-Kong` are translated to `To#1` and `To#2` respectively and a new goal, `ch1` is installed among `B`'s goals (in interrupted lines in figure 4). As `A` is on `B`'s screening list, `B` will also post it's own interdictions for `Accept`and for `Retry`. Propagation finds a behavior with $VC = 0$, namely one where `A` is forwarded to `To#1`, because the other two possible forwarding numbers are forbidden by `A` and both `Accept` and `Retry` are forbidden by `B`. This behavior is immediately executable as it does not violate any constraint.

Alternatives to this scenario include:

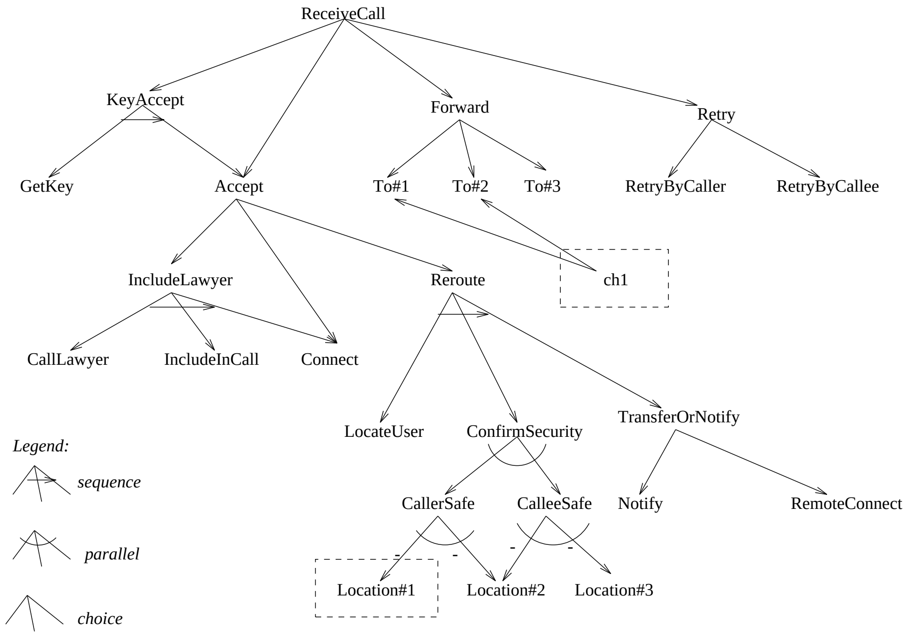


Figure 4: Deontic propagation applied to feature interaction.

1. A has `Outgoing Call Screening` to all of #1, #2 and #3. In this case there is no zero violation cost solution, but if A has different violation costs for the three numbers, B can forward to the smallest cost one. This may be done automatically by B or with A's approval, in which case the negotiation cycle is extended to include this step.

2. A does not want B to know that it does not want to be forwarded to #1 and #2. In this case, A does not send the corresponding interdiction, but requires B to confirm any forwarding. B replies by informing A about all three possible numbers to forward to and A either calls one of these directly (so that B doesn't know which it is) or asks to be forwarded to a specific one (so that it is not clear which number was avoided).

*Scenario 2: The Key Protection feature.* Assume now that B has a *Key Protection* feature that requires that certain callers input a key before being connected to B. If A is among these callers, B will place an obligation for `KeyAccept`. Assuming also that A is not one of the callers whose call is accepted only in the presence of a lawyer and that B is not away, interdictions will also be placed on `IncludeLawyer` and `Reroute`. Again, this has a zero cost behavior consisting of A entering the key and, if successful, being connected to B. A possible variant is that A is among those callers whose call is only taken in the presence of B's lawyer. Detecting this forces B to place an obligation on `IncludeLawyer`. The zero cost behavior in this case has B try first call his lawyer and, if successful, take `A`'s call. We note in this discussion that forbidding or obliging various goals depends on checking a variety of conditions about the agents' state, including agents' own databases that contain screeed out callers, callers required to enter valid keys, or even callers for whom a lawyer's presence is required. We provide several ways in which these verifications can take place, including situation rules, conversation rules or specific rules attached on goals that determine if the current situation warrants posting obligations or interdictions on the goal.

*Scenario 3: Security concerns.* Another class of features deal with the situation in which the called person is away, and calls need to be rerouted to his current location. One concern in this situation is that both callers and calees may not wish to be connected to/from locations considered risky from a security point of view. In this case, callers are allowed to specify constraints on the locations to which they accept or not to be connected. Similarly, callees agents may not connect specified callers to the callee if the callee's location is not safe enough. In figure 4 this situation is dealt with by introducing a new alternative to the `Accept` goal, which does `Reroute` when possible. Both caller and callee specify their unsafe locations as negated subgoals of a parallel composition. This ensures that none of the specified locations can be accepted. The caller will send its unsafe locations in the message with an associated interdiction cost. These locations will be added dynamically to the calee's action network (as shown in the figure). In case there are no zero cost solutions (e.g. the user is at an unsafe location) and all violations come from security goals, the min-violation solution will also represent the least risky solution. The caller may or may not wish the take this risk, and this will be decided during negotiation.

# 5 Final Remarks

Starting from the view of negotiation as an interactive goal discovery process aimed at eliminating goal conflicts that impede joint or individual work, we have reported on our work to integrate two technologies that support negotiation.

The first is our previous conversational technology, which provides the means to specify and execute long running structured interactions where agents exchange communicative action based messages. This supports the process dimension of negotiation. The second supports the reasoning dimension of negotiation and comes in the form of a 'negotiation solver', a formal reasoning engine based on a formal behavior description language and several constraint propagation methods allowing the discovery of behaviors that do not violate the constraints of the involved parties, or violate them in a manner that can be characterized (e.g. minimally). These enables parties to better understand the current state of the negotiation process and the available options. The formality of the methods employed by the 'negotiation solver' offers guarantees (like minimality of violation) which will increase the level of trust of the parties using this process. As expected, the two dimensions are intertwined. Interactive communication provides new goals and behaviors to reason about, while the results of reasoning guide the nature of future communication.

The approach is also application independent. We have illustrated it in the telecommunications industry, but we are equally looking at other domains like supply chain management and other forms of business negotiation.

## 6 Acknowledgments

This research is supported, in part, by the Manufacturing Research Corporation of Ontario, Mitel Corp., CITO, Natural Science and Engineering Research Council, Digital Equipment Corp., Micro Electronics and Computer Research Corp., Spar Aerospace, Carnegie Group and Quintus Corp.

## References

[1] Barbuceanu, M. and Fox, M. S. 1997. Integrating Communicative Action, Conversations and Decision Theory to Coordinate Agents. *Proceedings of Automomous Agents'97*, 47-58, Marina Del Rey, February 1997.

*Gives details about converstion plans and their use in agent interaction. We are using conversation plans here for controlling an agent's behavior in the negotiation process.*

[2] Barbuceanu, M. 1998. Agents That Work Together in Harmony by Knowing And Fulfilling Their Obligations. *Proceedings of AAAI'98*, Madison, Wisconsin, 1998.

*Gives details about the behavior description language, an earlier arc-consistency type constraint propagation method to reason about obligations and the agent architecture based on these.*

[3] Cameron, E.J., N.D. Griffeth, Y.J. Lin, M.E. Nilson, W.K. Schnure, and H. Velthuijsen. A Feature Interaction Benchmark for for IN and Beyond. In L.G. Bouma and H. Velthuijsen, editors, *Feature Interactions in Telecommunication Systems*, 1-23, Amsterdam, May 1996. IOS Press.

*Classical paper defining and classifying feature interactions in telecommunications. Used here for constructing our feature interaction example. Note that 'feature interaction' is also a general software maintenance and extension problem. Vendors of switching equipment currently offer centralized, hard-coded solutions, in the form of multi-million lines proprietary software which is very hard to maintain and extend. An agent solution based on declarative representations and formal reasonig would improve this situation radically.*

[4] Castelfranchi, C. 1995. Commitments: From Individual Intentions to Groups and Organizations. *Proceedings of ICMAS-95*, AAAI Press, 41-48.

*Makes a case for the need to integrate obligations in any model of coordinated behavior.*

[5] Levesque, H, Cohen, P.R. and Nunes, J. On Acting Together. *Proceedings of AAAI'90*, Menlo Park, CA.

*Classical paper introducing the joint intentions model of collective behavior. We have argued elsewhere that this does not scale up to complex organizations where agents have multiple, contradictory goals. We have also proposed obligations and our reasoning model with obligations as a way to explain coordinated behavior with contradictory or different rather than common goals.*

[6] Finin, T. et al. 1992. Specification of the KQML Agent Communication Language. The DARPA Knowledge Sharing Initiative, External Interfaces Working Group.

*Older specification of KQML. We believe that simply listing the 'communicative actions' in the agent communication language is not enough. One has to describe the structure of interactions and exchanges as well, and this is what conversation plans provide. We do not believe that general 'grammars' of speech-act interactions, as recently proposed by some KQML designers, will prove very useful because most interaction patterns are more specific to the domain or application and because the agent's actions are tightly coupled with the messages it exchanges which requires a way to describe them together, which is what our coordination language provides.*

[7] http://drogo.cselt.stet.it/fipa/

*FIPA has a KQML-like language, but with a better designed set of speech acts. Any of KQML, FIPA or similar languages can be used in our coordination system.*

[8] Griffeth, N. and Velthuijsen, H. A Negotiation Protocol for a Telecommunications Service Provisioning Environment. Proposal for FIPA, 1998.

*Describes an agent negotiated solution to feature interaction whose basic intuitions are very much similar to our work. However, it does not advance the kind of infrastructure that we have, and certainly lacks in the formal aspects of our reasoning model.*

[9] Meyer, J. J. Ch. 1988. A Different Approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic. *Notre Dame J. of Formal Logic* 29(1) 109-136.

*Paper reformulating deontic logic in a dynamic logic framework. Initially we tried to direcly use their semantics to reason about obligations and interdictions. But Meyer does not have violation costs, the treatment of sequences is in fact more complex and to account for the temporal semantics of action execution, especially if we introduce finite durations and finite capacity resources, we need some form of action scheduling. Thus, from the compact dynamic deontic logic of Meyer we arrived at the current solution where we separate deontic reasoning in the propagation engine and then account* *for temporal concerns by means of scheduling. This decomposition is more powerful than Meyer's initial framework.*

[10] Shoham, Y. 1993. Agent-Oriented Programming. *Artificial Intelligence* 60, 51-92.

*A first version of programming with speech-acts, obligations, and others. Our work may be seen as a later and more complete version of AOP.*

[11] Traum, D.R. and Allen, J.F. 1994. Discourse Obligations in Dialogue Processing. *Proceedings of the 32$^{th}$ Annual Meeting of the ACL*, Las Cruces, NM, 1-8.

*Argues for the insufficiency of shared goals and intentions to explain coordinated behavior, and for the need to introduce obligations in the picture.*