Resolving Conflicts in Agent Guidance

Karen L. Myers David N. Morley
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com morley@ai.sri.com

Abstract

For agent technology to be accepted in real-world applications, humans must be able to customize and control agent operations. One approach for providing such controllability is to enable a human supervisor to define *guidance* for agents in the form of policies that establish boundaries on agent behavior. This paper considers the problem of conflicting guidance for agents, making contributions in two areas: (a) outlining a space of conflict types, and (b) defining resolution methods that provide robust agent operation in the face of conflicts. These resolution methods combine a guidance-based preference relation over plan choices with an ability to extend the set of options considered by an agent when conflicts arise. The paper also describes a PRS-based guidance conflict-handling capability applied within a multiagent intelligence-gathering domain.

Introduction

Many potential applications for agent technology require that humans and agents work together in order to accomplish tasks effectively. This requirement is especially important for domains where task complexity precludes formalization of agent behaviors for all possible eventualities. In such domains, the availability of mechanisms by which a human supervisor can provide direction will enable agents to be informed by the experience, breadth of knowledge, and superior reasoning capabilities that a human expert can bring to the problem-solving process.

In previous work, we defined a framework for *agent guidance* that supports dynamic directability of agents by a human supervisor (Myers & Morley 2001; 2002). Guidance imposes boundaries on agent behavior, thus enabling a human to customize and direct agent operations to suit his or her individual requirements. The guidance framework focuses on two types of agent directability, namely *adjustable agent autonomy* and *strategy preferences*. Guidance for adjustable autonomy enables a supervisor to vary the degree to which agents can make decisions without human intervention. Guidance for strategy preferences constitutes recommendations on how agents should accomplish assigned tasks. For example, the directive "Use helicopters for survey tasks in sectors on the west coast" imposes restrictions on how resources can be used to perform a certain class of task.

User guidance provides a powerful mechanism for runtime customization of agent behavior. However, it also introduces the potential for problems in the event that the guidance recommends inconsistent responses. Such conflicts cannot arise with adjustable autonomy guidance, but are a significant issue for strategy preference guidance. Robustness of operations requires mechanisms for detecting these conflicts and responding in a manner that does not jeopardize agent stability.

This paper addresses two main issues related to the topic of conflicting strategy preference guidance for agents. First, it identifies different types of conflict that can arise. Second, it defines automated techniques for resolving the conflicts. Our approach combines the selective dropping of problematic pieces of guidance with a proactive capability to eliminate the source of conflicts by modifying current agent activities. We do not consider interactive techniques for conflict resolution in this paper, although they would certainly play an important role in a comprehensive conflict resolution system.

The conflict resolution techniques defined in this paper have been implemented within the Taskable Reactive Agent Communities (TRAC) framework (Myers & Morley 2001; 2002), which provides a domain-independent guidance capability for PRS agents (Georgeff & Ingrand 1989). The techniques have been applied within the context of a simulated disaster relief task force in which a human supervisor must manage a team of agents engaged in a variety of information-gathering and emergency response tasks. Within this testbed, referred to as TIGER (TRAC Intelligence Gathering and Emergency Response), agents control simulated trucks and helicopters in the performance of their assigned tasks.

We begin with a description of the agent model that underlies our work on agent guidance, followed by a definition of strategy preference guidance. Next, we outline different types of conflict that can arise with agent guidance and our conflict resolution methods. Finally, we describe a realization of the conflict resolution mechanisms within the TIGER framework and discuss related work.

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Agent Model

We adopt a typical Belief-Desire-Intention (BDI) model of agency in the style of (Rao & Georgeff 1995). BDI agents are so-called due to the three components of their "mental state": *beliefs* that the agent has about the state of the world, *desires* to be achieved, and *intentions* corresponding to actions that the agent has adopted to achieve its desires.

Each agent has a library of *plans* that defines the range of activities that an agent can perform to respond to events or to achieve assigned tasks; our plan model is based on (Wilkins & Myers 1995). Plans are parameterized templates of activities that may require variable instantiations to apply to a particular situation. In a standard BDI framework, there are two types of plan: *fact-invoked* plans for responding to changes in the beliefs of the agent, and *goal-invoked* plans for decomposing tasks into constituent subgoals and actions.

Each plan has a *cue* that specifies a stimulus that activates the plan, either a new goal (for a goal-invoked plan) or a change in the agent's beliefs (for a fact-invoked plan). A set of *preconditions* associated with plans defines gating constraints that must be satisfied for a plan to be applied. A plan is said to be *relevant* to a world change (e.g., new goal or belief change event) if the plan cue matches the stimulus, and *applicable* if, additionally, the plan preconditions are satisfied relative to the agent's current beliefs. The *body* of a plan specifies how to respond to the stimulus, in terms of actions to perform and subgoals to achieve.

An agent's plan library will generally contain a range of plans describing alternative responses to posted goals or events. Sets of these plans may be *operationally equivalent* (i.e., they share the same cue and preconditions) but differ in the approach that they embody. Some form of metacontrol policy can be defined to select among such alternatives, such as strategy preference guidance.

A BDI interpreter runs a continuous *sense-decide-act* loop to respond to changes in its operating environment. At the start of each cycle, the interpreter collects all new goals and events (i.e., changes in its beliefs about the world). Next, it determines whether there are any plans in its library that are applicable to these changes. From this set, it selects some subset for execution and creates intentions for them. Finally, some bounded number of steps for each current intention are performed.

The guidance framework assumes that agents are capable of fully autonomous operation. More concretely, an agent's plan library covers the range of activities required to perform its assigned tasks. This assumption means that agents do not depend on the human supervisor to provide knowledge to complete tasks. Within this setting, guidance provides customization of agent behavior to suit the preferences of the human supervisor. In many applications, such guidance will enable superior performance, given that few plan libraries will reflect the experience, breadth of knowledge, and reasoning capabilities that a human supervisor can bring to the decision-making process.

Strategy Preference Guidance

Strategy preference guidance expresses recommendations on how an agent should accomplish tasks. These preferences could designate classes of plans to employ or restrictions on plans that should not be employed, as well as constraints on how plan variables can be instantiated. For example, the directive "Try contacting Nongovernmental Organizations for information before sending vehicles to towns on the west coast" expresses a preference for selecting among operationally equivalent plans. The directive "Only use helicopters for survey tasks in sectors that are expected to be inaccessible by truck for more than 1 week" restricts the choice of resource type for instantiating certain plan variables.

Representation of Guidance

Our language for representing agent guidance builds on three main concepts: the underlying *agent domain theory*, a *domain metatheory*, and the connectives of first-order logic. Using these elements, we develop the main concepts underlying our model of agent guidance. These consist of an *activity specification* for describing abstract classes of action, a *desire specification* for describing abstract classes of goal, and an *agent context* for describing situations in which guidance applies.

Domain Metatheory A standard domain theory for an agent consists of four types of basic element: *individuals* corresponding to real or abstract objects in the domain, *relations* that describe characteristics of the world, *goals* that an agent may adopt, and *plans* that describe available means for achieving goals.

The *domain metatheory* provides an abstracted characterization of elements of the domain theory that highlights key semantic differences. As discussed in (Myers 2000a), a metatheory can yield a rich vocabulary for describing activity, thus providing a powerful basis for supporting user communication. The main concepts within our metatheory for agent guidance are *features* and *roles* (similar in spirit to those of (Myers 1996)) defined for agent plans and goals.

Consider first plans. A *plan feature* designates an attribute of interest for a plan that distinguishes it from other plans that could be applied to the same task. For example, among plans for route determination, there may be one that is *Optimal* but *Slow* with a second that is *Heuristic* but *Fast*; each of these attributes could be modeled as a feature. Although the two plans are operationally equivalent (i.e., same cue and preconditions), their intrinsic characteristics differ significantly. Features provide the means to distinguish between such operationally equivalent alternatives.

A *plan role* describes a capacity in which a domain object is used within a plan; it maps to an individual variable within a plan. For instance, a route determination plan may contain variables *location.1* and *location.2*, with the former corresponding to the *Start* and the latter the *Destination*.

In analogous fashion, roles and features can also be defined for goals. For example, a goal of informing another party of task progress may have a *Communication* feature and *Recipient* role associated with it. These metatheoretic

constructs can be used to specify the class of goals that involve communicating with the commander.

Activity and Desire Specification An *activity specification* characterizes an abstract class of plan instances for an agent. Our domain metatheory provides the basis for defining an activity specification, in terms of a set of required and prohibited features on a plan, as well as constraints on the way in which plan roles are filled.

Definition 1 (Activity Specification) *An* activity specification $\alpha = \langle F^+, F^-, R, \phi \rangle$ *consists of*

- *a set of* required features F^+
- *a set of* prohibited features F^-
- *a set of* roles $R = [R_1, \ldots, R_k]$
- *a* role-constraint *formula* $\phi[R_1, \ldots, R_k]$

For example, the following activity specification describes the class of plan instances with the feature *Survey* but not *Heuristic*, where the variables that fill the roles *Start* and *Destination* are instantiated to values in the same sector.

<{*Survey*}, {*Heuristic*}, {*Start, Destination*},
{*(= (sector Start) (sector Destination))*}>

A *desire specification* constitutes the goal-oriented analogue of an activity specification, consisting of a collection of required features, prohibited features, roles, and role constraints for goals.

Agent Context Just as individual plans employ preconditions to limit their applicability, guidance requires a similar mechanism for defining scope. To this end, we introduce the notion of an *agent context*. While plan preconditions are generally limited to beliefs about the world state, our model of agent context focuses on the full operational state of an agent, characterized in terms of its beliefs, desires, and intentions. Beliefs are specified in terms of constraints on the current world state. Desires are specified as desire specifications describing goals that the agent has adopted. Intentions are specified as activity specifications describing plans currently in execution by the agent.

Our model of agency assumes a hierarchical collection of plans and goals; furthermore, agents are capable of multitasking (i.e., addressing multiple goals in parallel). Within a given phase of the BDI executor cycle, an agent's goals can be scoped in three ways:

- *Current goal:* the goal for which the BDI interpreter is selecting a plan to execute
- *Local goals:* the current goal or any of its ancestors
- *Global goals:* any goal of the agent

By distinguishing these different scopes for goals, guidance can be localized to more specific situations. Plans being executed can be scoped in a similar fashion.

Definition 2 (Agent Context) *An* agent context *is defined by a tuple* $\kappa = \langle \Phi, \Delta, A \rangle$*, where*

- Φ *is a set of well-formed formulae.*
- $\Delta = \Delta^C \cup \Delta^L \cup \Delta^G$ *is a set of current, local, and global desire specifications, respectively.*
- $A = A^L \cup A^G$ *is a set of local and global activity specifications, respectively.*[1]

Strategy Preference Strategy preference guidance consists of two components: an *agent context* and a *response activity specification*. The activity specification designates the class of recommended plan instances to be applied (i.e., choice of plan and variable instantiations for designated roles) when the agent enters a state that matches the designated agent context.

Definition 3 (Strategy Preference) *A* strategy preference *rule is defined by a pair* $\langle \kappa, \alpha \rangle$ *where* κ *is an agent context and* α *is an activity specification.*

Example

To illustrate strategy preference guidance, we consider an example from a simplified description of the TIGER domain.

Within TIGER, the demand for intelligence gathering and other services generally exceeds the capabilities of the available agents. As a result, task management constitutes one of the key functions of TIGER agents. We assume that tasks have associated properties such as type (e.g., survey, evacuation, medical emergency) and priority.

A TIGER agent controls a single vehicle (either a truck or a helicopter) and can work on at most one task at any time. When an agent receives a task in the form of a goal (*task t*), it must decide whether to start on the task immediately, to postpone the task until other tasks are completed, or to drop the task, leaving it for other agents to perform. We use the following predicates to represent the task execution state of the agent:

- (*available*) – the vehicle that the agent is controlling is available for use
- (*doing t*) – the agent is doing task *t*
- (*pending t*) – the agent will do task *t* after it has completed other tasks
- (*ignored t*) – the agent has determined not to do task *t*

Using a simple notation for describing plans, Figure 1 defines three plans for responding to the goal (*task t*). In the bodies of these plans, we represent the action of asserting a fact ϕ into the agent's belief set by $+\phi$ and retracting ϕ by $-\phi$. We use the notation $\{a_1, \ldots a_k\}$ to represent performing k actions in parallel within the body of a plan. The notation $a_1; a_2$ denotes action sequencing.

The first plan encodes a response to (*task t*) when not engaged in another task; it involves asserting (*doing t*), retracting (*available*), and then performing *t*. The second plan records the task as pending in the case where the agent is busy. The third plan ignores the task. Additional plans (not shown) provide the capabilities to start a pending task when the active task completes, to perform a task, etc.

Features for these plans reflect their inherent semantic differences: *Adopt* indicates that the plan results in the agent

[1] Because the motivation for guidance is to influence the choice of plan for the current goal, the agent context excludes an activity specification for the current plan.

Name: *Immediate-Response*
Cue: $(task\ t)$
Preconditions: $(available)$
Body: $\{-(available), +(doing\ t)\}; (do\ t)$
Features: *Adopt* Roles: $NewTask = t$

Name: *Delay-Response*
Cue: $(task\ t)$
Preconditions: $\neg(available)$
Body: $+(pending\ t)$
Features: $Adopt, Delay$ Roles: $NewTask = t$

Name: *Ignore-Response*
Cue: $(task\ t)$
Preconditions: *true*
Body: $+(ignored\ t)$
Features: *Ignore* Roles: $NewTask = t$

Figure 1: TIGER Task Management Plans

deciding to perform the goal task, although maybe not immediately; *Delay* indicates that the plan results in execution of t being delayed; *Ignore* indicates that the plan results in the agent deciding not to perform t (the complement of *Adopt*). The role *NewTask* is used to reference the new task under consideration. For goal $(task\ t)$ we associate the feature *TaskResponse* and role $CurrentTask = t$.

With these features and roles, we can define strategy preference rules to provide guidance for responding to a new task. For example, the guidance "Adopt medical emergency tasks that involve more than 5 people" could be represented by the following strategy preference rule:

Agent Context:
  Current Desire Specification:
    Features+: *TaskResponse*
    Roles: *CurrentTask*
    Constraint:
      (and (= (task-type CurrentTask) medical)
        (> (task-number-affected CurrentTask) 5))
Response Activity Specification:
  Features+: *Adopt*

The agent context states that the guidance is applicable when the current goal has the feature *TaskResponse* and the role *CurrentTask*, so that the task that instantiates *CurrentTask* has type *medical* and more than five people affected. The response activity specification designates any plan with the feature *Adopt*. Given the plans defined above, that would mean either *Immediate-Response* or *Delay-Response*.

Guidance Semantic Model

Space limitations preclude a full description of the semantics for guidance satisfaction defined in (Myers & Morley 2002). We present a brief summary here.

The semantic model for guidance satisfaction interprets strategy preference rules as a filter on the plan instances that an agent can execute. A strategy preference rule is deemed to be *relevant* to a given BDI executor cycle iff the agent context of the rule matches the current execution state. For a relevant rule to be satisfied, the BDI interpreter must not select a plan instance that violates the rule's activity specification. One interesting consequence of this model is that a strategy preference rule that is not *relevant* to the current decision cycle is trivially satisfied.

When a BDI agent needs to expand a goal, it determines the *applicable plans* for the goal and selects one of these to add to its intentions. Under the guidance satisfaction model, the set of plans from which the agent selects is restricted to those that satisfy the strategy preference rules: the agent identifies the relevant strategy preference rules and filters out plan instances that do not match the suggested response. The BDI interpreter selects one of the remaining plans to execute for the current goal.

Types of Guidance Conflict

Guidance can lead to two types of conflict: *plan selection* and *situated guidance*.

Plan Selection Conflict

A plan selection conflict occurs when multiple pieces of guidance make incompatible recommendations for responding to a goal within a given cycle of the BDI executor. Conflicts of this type can arise in different forms. Here, we distinguish between *direct* and *indirect* conflicts.

A *direct conflict* arises when guidance yields contradictory plan selection recommendations. At the plan level, such conflicts can arise through explicitly contradictory directives (e.g., guidance that reduces to the constraints *Execute plan P* and *Don't execute plan P*), or implicitly because of in-place control policies (e.g., guidance that reduces to the constraints *Execute plan P* and *Execute plan Q* in the context of a control policy that allows only one response to any posted goal). Conflicts can also arise at the level of variable bindings (e.g., *Instantiate role R to A* and *Instantiate role R to B*, where $A \neq B$).

An *indirect conflict* among guidance occurs when there is no direct conflict, yet the plans recommended by the guidance cannot complete successfully because of interplan interactions. Such a situation could arise due to future contention for resources, deadlock/livelock, or race conditions (among others). The problem of indirect conflict arises for any multithreaded system, not just systems in which guidance has been used to select activities.

Direct conflicts are easy to detect, as they lead to incompatible recommendations for responding to a posted goal. In contrast, it is generally difficult to detect *a priori* the plan interference problems that underlie indirect conflicts. Because such interference problems remain an open research area in the agents community, we focus exclusively on direct conflicts in the remainder of this paper.

Situated Guidance Conflicts

The semantic model from (Myers & Morley 2002) interprets guidance as a filter on the set of otherwise applicable

plan instances for a particular goal or event. For example, consider a situation in which all TIGER vehicles are in use for various tasks, and the human supervisor has asserted the guidance Adopt medical emergency tasks that involve more than 5 people from the previous section. Suppose an emergency event arises. The relevant task adoption plans (namely *Immediate-Response* and *Delay-Response*) each require the availability of a vehicle. Because all vehicles are in use, no immediate response plans could be adopted for the event. Had there been a vehicle available, however, an emergency response of some form would have been adopted. Furthermore, according to the filtering semantic model, the declared guidance will eliminate from consideration the only applicable response, namely *Ignore-Response*, because it does not satisfy the guidance recommendations.

In this case, there is a clear expectation on the part of the human supervisor for the system to adopt a task in response to the emergency. Supporting this reaction requires a generalization of the filter-based semantic model described above.

More generally, this type of conflict arises in situations where a plan p is relevant for a current goal g but some precondition C of p does not hold, making p inapplicable. The unsatisfied condition may be blocked by a contradictory belief of the agent, or some already executing activity. This type of situation can arise independent of guidance. Our interest in such situations relates to cases where guidance would recommend the execution of p but the violation of C eliminates its consideration. In other words, the prior activity or state condition conflicts with the intent of applying the guidance. For this reason, we call this phenomenon a *situated guidance conflict*, as it depends on the consideration of guidance within a particular execution state of an agent.

In certain situations, there may be no recourse to address the violated conditions (e.g., consider a requirement for favorable weather). However, others may be *resolved* by undertaking appropriate actions in the domain. Proactive response of this type lies at the heart of our techniques for resolving for this class of guidance conflict.

Conflict Resolution

The original semantic model for guidance interpretation has a *passive* flavor in that it simply filters otherwise applicable plan instances that violate current guidance. This passive semantic model has the virtue of simplicity, but it eliminates the applicability of guidance in many interesting situations.

One problematic situation arises when guidance makes contradictory recommendations (e.g., *"Execute plan P"* and *"Don't execute plan P"*) as described above for the case of plan selection conflicts. Because the passive semantic model eliminates plans that violate current guidance, such a situation would lead to the selection of no plan. As noted above, the filter-based semantics also leads to trivial satisfaction of guidance in cases where a more proactive interpretation would be preferred.

Meaningful resolution of guidance conflicts requires a richer semantic model for guidance satisfaction. Our approach builds on the definition of satisfaction of an individual piece of guidance from (Myers & Morley 2002). However, we adopt a preference-based approach that seeks to maximize guidance satisfaction relative to stated priorities. Furthermore, instead of reducing the set of plans that the agent considers (by filtering plans that violate guidance), we expand the set to include options that would otherwise be discarded as inapplicable in the current execution state.

Preference Semantics for Plan Selection

Our approach to resolving plan selection conflicts involves identifying a plan instance that 'best satisfies' current guidance, through the definition of a partial order over plan instances. We assume that strategy preference rules include a *weight* reflecting the relative strength for that preference.

Different criteria could be used for combining and comparing the weights associated with the strategy preference rules to produce the partial order. We adopt an approach that rewards guidance satisfaction while punishing guidance violation, as characterized by the following *guidance ranking function* for plan instances.

Definition 4 (Guidance Ranking of a Plan) *Let p be a plan instance for a goal g and Q be the current set of guidance, where $Q_p^+ \subseteq Q$ is the set of guidance satisfied by p and $Q_p^- \subseteq Q$ is the set violated by p. The* guidance ranking *of p is defined as follows in terms of the priority $GPriority(q)$ associated with each guidance rule q:*

$$GRanking(p) = \sum_{q \in Q_p^+} GPriority(q) - \sum_{q \in Q_p^-} GPriority(q)$$

Candidate Plan Expansion

The ranking of applicable plan instances is insufficient to resolve situated guidance conflicts. Our approach involves expanding the set of plan instances considered for application to a given goal. This expanded set builds on the agent's library of predefined plans, extending plans that guidance might recommend to compensate for violated applicability conditions. The expansion process requires the satisfaction of certain prerequisites related to *resolvability* and *cost-benefit analysis*.

Resolvability The unsatisfied applicability conditions of the guidance-recommended plan must be *resolvable*. In particular, there must be identified methods (called *resolution plans*) that can be invoked to achieve the unsatisfied conditions.

Cost-Benefit Analysis The benefits in following the prescribed guidance must outweigh the costs associated with executing the resolution plans.

We consider these two requirements in turn, and then describe the process for expanding the set of plan instances to be considered for a given goal.

Resolvability Resolution plans could be defined for a wide range of conditions. Resource availability constitutes one important class; in this paper, we focus on conditions related to serially sharable resources (i.e., resources that can be used sequentially but not simultaneously).

In this context, resolution plans must free the resource employed by the current activity to enable its use by the

guidance-recommended plan. Within the context of serially sharable resources, *cancellation* of the prior activity constitutes one obvious solution. In addition, prior activities could be modified to eliminate the dependence on the conflicted resource, say by *substituting* a different resource or by *delaying* the prior activity. We call a current activity that is impacted by a resolution plan a *conflict task*.

In addition to establishing resolvability conditions, resolution plans must also leave an agent in a coherent state. This requirement means that a resolution plan must consider the in-progress effects of any current activities that are to be modified or canceled to ensure that appropriate 'clean-up' processes are invoked. For example, consider an emergency response task in which a truck has picked up a set of supplies to deliver to a designated location. Early termination of such an activity might involve, at a minimum, delivering those supplies to a local depot (for delivery by some other agent). Definition of *recovery mechanisms* of this type is not restricted to agents under human supervision via guidance, but rather constitutes a problem for any agents that need to dynamically modify their activities at runtime.

Checkpoint schemes constitute one standard technique for ensuring state coherence when activities may need to be terminated prematurely. With checkpoint schemes, coherent states are saved periodically to enable rollback to a consistent state in case of unrecoverable failures. Because agents operating in dynamic environments will generally perform activities that change the world in irrevocable ways, such schemes are not viable. Instead, agents require *forward* recovery methods that can take actions to transition from an unable situation to some known, safe state.

Formulation of forward recovery methods presents several problems. In general, the specific actions to take could depend both on the state of execution of the in-progress plans and on the status of other executing activities and world state properties. In the worst case, a unique recovery method would be required for each such situation. For this reason, forward recovery mechanisms for agents are generally implemented in an *ad hoc*, domain-specific manner.

The resolution plans for task management within TIGER were defined by hand. They consist of plans for *delaying* and *terminating* prior tasks. Because the number of subtasks involved with the survey and emergency response tasks is relatively small (i.e., fewer than five), the number of possible combinations of state to consider is correspondingly low. Furthermore, tasks are undertaken independently of each other, so cross-task interactions were not an issue.

Cost-Benefit Analysis The value in modifying existing activities to enable activation of new guidance-recommended activities depends on a range of factors. We consider a model grounded in the following two concepts:

- *Resolvability Cost:* the cost of executing any required resolution plans
- *Activity Priorities:* the priorities of the recommended activity and any *conflict activities*

Different approaches can be considered for combining the above factors to determine the appropriate response to a situated guidance conflict. We describe three general approaches here.

One approach involves defining a multidimensional objective function that determines when the guidance-recommended activity should be undertaken. Such a function would cover every possible situated guidance conflict, thus enabling a fully automated conflict resolution method. Multidimensional evaluation functions are notoriously difficult to define, as they must relate values that are not directly comparable. The need to consider all combinations of values complicates matters further.

In contrast, a mixed-initiative approach could be adopted in which a human supervisor determines the appropriate course of action based on the factors cited above. This approach avoids the cost of defining *a priori* comparison functions and provides maximum flexibility at runtime. However, situational conflicts may arise frequently in some domains, thus burdening the user with a high level of decision making involvement.

A third approach involves the definition of policies (similar to guidance) for determining the conditions under which modification of earlier activities should be undertaken. Such policies would involve constraints on the various factors described above. For example, "Only modify ongoing activities when the priority of the new task exceeds that of the original, and the recovery cost is less than 0.5". For situations where the policies are sufficient to identify a response, conflicts will be resolved automatically. In other cases, the human supervisor would be engaged to make the appropriate decision. As such, this third approach combines the benefits of predefining certain responses with the flexibility of runtime decision-making by the human supervisor for situations that cannot be readily characterized ahead of time.

Expanded Set of Candidate Plans Our approach to expanding the set of candidate plans for a goal g involves the dynamic creation of a set of *proactive plans*. Each proactive plan is a variant of a *potentially applicable plan* – a relevant plan for g whose applicability is blocked by one or more unsatisfied but resolvable preconditions. The body of the proactive plan incorporates activities from appropriate resolution plans to achieve the blocked preconditions, as well as the actions from the potentially applicable plan. In this way, proactive plans extend the set of actions that can be taken by an agent for a given goal. The following definitions capture these notions more precisely.

Definition 5 (Resolvable Condition) *A condition ϕ is* resolvable *in a given BDI executor state iff there is some instance p^r of a resolution plan such that $Cue(p^r) = \phi$ and for every $\phi' \in Pre(p^r)$, $Believed(\phi')$ holds in the BDI state.*

Definition 6 (Potentially Applicable Plan Instance) *A* potentially applicable plan instance *for a goal g is a relevant plan instance for g for which not all preconditions are satisfied but all unsatisfied preconditions are resolvable.*

Definition 7 (Proactive Plan) *Let p be a potentially applicable plan instance with $Cue(p) = g$. Let $Pre(p) = \Phi^F \cup \Phi^T$ where $\Phi^F = \{\phi_1^F, \ldots, \phi_m^F\}$ are unsatisfied and $\Phi^T = \{\phi_1^T \ldots \phi_n^T\}$ are satisfied. Let $p_1^r, \ldots p_m^r$ be resolution plans such*

Name: *Resolve-by-Delay*
Cue: $(available)$
Preconditions: $(doing\ t')$
Body:
$(pause\ t'); \{+(available), -(doing\ t'), +(pending\ t')\}$
Features: *DelayCurrent* Roles: $CurrentTask = t'$

Name: *Resolve-by-Termination*
Cue: $(available)$
Preconditions: $(doing\ t')$
Body: $(kill\ t'); \{+(available), -(doing\ t'), +(ignored\ t')\}$
Features: *DropCurrent* Roles: $CurrentTask = t'$

Figure 2: TIGER Resolution Plans

that $Cue(p_i^r) = \phi_i^F$ *and for every* $\phi \in Pre(p_i^r)$, $Believed(\phi)$ *holds in the current BDI state. The plan* p' *defined below is a proactive plan for* g.

- $Cue(p') = g$
- $Pre(p') = \Phi^T \cup \bigcup_{1 \le i \le m} Pre(p_i^r)$
- $Body(p') = \{Body(p_1^r), \ldots, Body(p_m^r)\}; Body(p)$
- $Features(p') = Features(p) \cup \bigcup_{1 \le i \le m} Features(p_i^r)$
- $Roles(p') = Roles(p) \cup \bigcup_{1 \le i \le m} Roles(p_i^r)$.

Definition 8 (Proactive Plan Set) *The* proactive plan set *for goal* g, *denoted by* $Proactive(g)$, *consists of the proactive plans that can be constructed from the potentially applicable plan instances for* g *and the available resolution plans.*

Note that the applicability of a proactive plan within the current BDI executor loop is guaranteed, as its preconditions are drawn from a set of known satisfied conditions.

As noted above, cost-benefit considerations should be taken into account when deciding how to augment the original set of applicable plan instances for a current goal with proactive plans. Thus, in general, only a subset of the set $Proactive(g)$ of proactive plans for g would be included.

Given the expanded set of candidate plans for application to the current goal g, selection among them can be performed in accord with the preference semantics outlined earlier, through application of the guidance ranking function of Definition 4.

TIGER Conflict Resolution

TIGER supports situated conflict resolution for task management activities. Within this context, resource contention arises because each vehicle is limited to use on at most one task at a time. TIGER includes resolution plans (see Figure 2) that achieve $(available)$ (i.e., the availability of the vehicle) by delaying and terminating prior tasks, thereby enabling a new task to be executed immediately. The resolution plan *Resolve-by-Delay* delays the current task t' via the goal $(pause\ t')$. The resolution plan *Resolve-by-Termination* terminates the current task via the goal $(kill\ t')$. The goals $(pause\ t')$ and $(kill\ t')$ also perform appropriate clean-up actions. When an agent is given a new task but $(available)$ is unsatisfied, it can synthesize two proactive plans (shown in Figure 3) from the plan *Immediate-Response* and the resolution plans of Figure 2.

Name: *Proactive-Immediate-Response-Delay*
Cue: $(task\ t)$
Preconditions: $(doing\ t')$
Body: $(pause\ t'); \{+(available), -(doing\ t'), +(pending\ t')\}$;
$\{-(available), +(doing\ t)\}; (do\ t)$
Features: $Adopt, DelayCurrent$ Roles: $NewTask = t, CurrentTask = t'$

Name: *Proactive-Immediate-Response-Termination*
Cue: $(task\ t)$
Preconditions: $(doing\ t')$
Body: $(kill\ t'); \{+(available), -(doing\ t'), +(ignored\ t')\}$;
$\{-(available), +(doing\ t)\}$
Features: $Adopt, DropCurrent$ Roles: $NewTask = t, CurrentTask = t'$

Figure 3: Sample Proactive Plans

For the small number of plans in this example, resolution rules might seem unnecessary – one can achieve the same effect by extending the agent's plan library to include the proactive plans. However, with a greater number of plans, the ability to factor out the recovery mechanisms eliminates duplication and reduces the potential for error. Alternatively, one could add explicit subgoals of achieving $(available)$ (in this case) into the base plans. Because the choice of method for achieving $(available)$ would occur after the choice of the base plan, this approach would preclude the use of guidance to select among options.

The cost-benefit analysis used by TIGER for determining which proactive plans to consider for a given goal consists of the following three conditions. First, resolvability costs are computed as a simple heuristic related to expected time remaining for task completion; a threshold is defined so that nearly complete tasks are not interrupted. Second, the priority of the new task must be at least as high as that of the resolvable task. Third, there must be at least one current strategy preference rule whose agent context is satisfied and whose activity specification matches the proactive plan. In the future, we intend to replace this fixed criteria with a policy-based approach similar to that described earlier.

Related Work

Most work on conflict within the agents community has focused on conflicts *among* agents (e.g., the papers in (Tessier, Chaudron, & Muller 2000)) rather than on guidance for controlling an agent. Typically, explicit interagent protocols are used to negotiate solutions to detected conflicts.

Conflicting advice has been considered previously in the context of advising an automated planning system (Myers 2000b). Detection and resolution techniques in that work have a markedly different style from the approach outlined in this paper, being grounded in heuristic search control methods for plan generation.

The rule-based reasoning community has considered a similar problem dealing with resolving rule conflicts. In

(Chomicki, Lobo, & Naqvi 2000), rule conflicts are resolved by ignoring triggering events that cause conflicts. The work of (Jagadish, Mendelzon, & Mumick 1996) uses a declarative set of metarules to constrain how a set of rules should be executed. Their metacontrol language is grounded in the specifics of rules and rule firing (e.g., ordering rules, disabling rules, requiring simultaneous triggering of rules), which differs substantially from our more expressive and user-focused guidance language.

The work in (Lupu & Sloman 1999) presents a simple language for defining policies to manage a distributed set of objects. Static conflict checking is performed when policies are defined (rather than at runtime, as in this paper). Their conflict resolution methods include straightforward concepts such as explicit priorities and preferring negative to positive rules; in addition, they incorporate more advanced notions of specificity over rules and distance metrics to generate overall precedence relationships.

In (Dignum *et al.* 2000), preferences over BDI agent behaviors are expressed through a multimodal deontic logic for social norms and obligations. Metalevel norms and obligations are used to resolve conflicts that arise.

The dialog community has focused on techniques for detecting and resolving conflicts that arise in the performance of collaborative tasks (e.g., (Qu & Beale 1999)). Techniques of this type could be used as the basis for interactive resolution of the conflict types explored in this paper.

Conclusions

Many applications that could benefit from agent technology impose the requirement that humans retain control over agent operations. The guidance framework of (Myers & Morley 2001; 2002) enables a human supervisor to assert strategy preference rules to influence agent behavior, thus providing a powerful mechanism for directing agent operations. However, it also introduces the possibility for problems in the event that conflicting guidance is declared.

This paper identified two classes of guidance conflict, namely *plan selection* and *situated guidance*, and showed how such conflicts transcend the purely filter-based semantics introduced originally for agent guidance. To address such conflicts, we defined a generalized semantic model for guidance satisfaction that incorporates the complementary notions of guidance-derived plan preference relations and plan option expansion. Within this model, the set of plan options available to an agent is extended to include plans that would not normally be considered for execution. These options can be synthesized dynamically by combining guidance-relevant plans whose applicability is blocked with resolution plans that can achieve the blocked applicability conditions. Accompanying conflict resolution methods for this model were presented that ensure robustness of agent operations in the face of conflicting guidance.

Acknowledgments This work was supported by DARPA under the supervision of Air Force Research Laboratory contract F30602-98-C-0160.

References

Chomicki, J.; Lobo, J.; and Naqvi, S. 2000. A logic programming approach to conflict resolution in policy management. In Cohn, A. G.; Giunchiglia, F.; and Selman, B., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '00)*. Morgan Kaufmann Publishers.

Dignum, F.; Morley, D.; Sonenberg, E. A.; and Cavedon, L. 2000. Towards socially sophisticated BDI agents. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000)*.

Georgeff, M. P., and Ingrand, F. F. 1989. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*.

Jagadish, H. V.; Mendelzon, A. O.; and Mumick, I. S. 1996. Managing conflicts between rules. In *Proceedings of the ACM Symposium on Principles of Database Systems*.

Lupu, E., and Sloman, M. 1999. Conflicts in policy-based distributed systems. *IEEE Transactions on Software Engineering, Special Issue on Inconsistency Management* 25(6).

Myers, K. L., and Morley, D. N. 2001. Human directability of agents. In *Proceedings of the First International Conference on Knowledge Capture*.

Myers, K. L., and Morley, D. N. 2002. Policy-based agent directability. In Hexmoor, H.; Falcone, R.; and Castelfranchi, C., eds., *Agent Autonomy*. Kluwer Academic Publishers.

Myers, K. L. 1996. Strategic advice for hierarchical planners. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*. Morgan Kaufmann Publishers.

Myers, K. L. 2000a. Domain metatheories: Enabling user-centric planning. In *Proceedings of the AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems*.

Myers, K. L. 2000b. Planning with conflicting advice. In *Proceedings of the Fifth International Conference on AI Planning Systems*.

Qu, Y., and Beale, S. 1999. A constraint-based model for cooperative response generation in information dialogues. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*.

Rao, A. S., and Georgeff, M. P. 1995. BDI agents: From theory to practice. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95)*.

Tessier, C.; Chaudron, L.; and Muller, H.-J., eds. 2000. *Conflicting Agents*. Kluwer Academic Press.

Wilkins, D. E., and Myers, K. L. 1995. A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation* 5(6).