Metacognition for Detecting and Resolving Conflicts in Operational Policies

Darsana P. Josyula, Bette Donahue, Matt McCaslin and **Michelle Snowden**
Bowie State University
Bowie, MD 20715, USA
darsana@cs.umd.edu

Michael L. Anderson
Franklin & Marshall College
Lancaster, PA 17604, USA
michael.anderson@fandm.edu

Tim Oates and **Matthew D. Schmill**
University of Maryland Baltimore County
Baltimore, MD 21250, USA
oates@cs.umbc.edu

Donald R. Perlis
University of Maryland
College Park, MD 20742, USA
perlis@cs.umd.edu

Abstract

Informational conflicts in operational policies cause agents to run into situations where responding based on the rules in one policy violates the same or another policy. Static checking of these conflicts is infeasible and impractical in a dynamic environment. This paper discusses a practical approach to handling policy conflicts in real-time domains within the context of a hierarchical military command and control simulated system that consists of a central command, squad leaders and squad members.

All the entities in the domain function according to preset communication and action protocols in order to perform successful missions. Each entity in the domain is equipped with an instance of a metacognitive component to provide on-board/on-time analysis of actions and recommendations during the operation of the system. The metacognitive component is the Metacognitive Loop (MCL) which is a general purpose anomaly processor designed to function as a cross-domain plug-in system. It continuously monitors expectations and *notices* when they are violated, *assesses* the cause of the violation and *guides* the host system to an appropriate response.

MCL makes use of three ontologies—indications, failures and responses—to perform the notice, assess and guide phases when a conflict occurs. Conflicts in the set of rules (within a policy or between policies) manifest as expectation violations in the real world. These expectation violations trigger nodes in the indication ontology which, in turn, activate associated nodes in the failure ontology. The responding failure nodes then activate the appropriate nodes in the response ontology. Depending on which response node gets activated, the actual response may vary from ignoring the conflict to prioritizing, modifying or deleting one or more conflicting rules.

1 Introduction

Effectively dealing with global threats such as terrorism requires cooperation and collaboration across multiple government agencies having different operational policies. Operational policies include the pre-set communication and action protocols established for effective cooperation and collaboration between agents. These policies are typically represented declaratively using deontic concepts such as permissions, obligations, and prohibitions (Kagal 2004).

Informational clashes can arise within one policy or across multiple ones; when policies from multiple sources are combined (fused or integrated), the chances of such inconsistencies grow rapidly. Conflicts in operational policies can cause autonomous agents to run into situations where an action by an agent violates one policy or the other. Therefore, there is tremendous interest in statically checking these policies for inconsistencies.

A standard approach for consistency checking of policies is to use formal automated reasoning over their declarative representations. If there is a contradiction, it will eventually be revealed. This approach, however, is problematic for at least three reasons. First, it is computationally intractable for the majority of expressive access control languages, such as the popular XACML (Kolovski 2008), and even for some that are extremely simple (Dinollt, Benzinger, and Yatabe 1994). Second, agencies may be unwilling to publish their policies for automated reasoning and other forms of scrutiny. Third, beyond detecting inconsistencies, resolving these conflicts may require additional knowledge that is not available in the inconsistent policies; or inferential capability that is not available in the formal logics that check them for (in)consistency. This paper illustrates how a metacognitive component that can detect and deal with anomalies can provide an agent the ability to handle operational policy violations in real-time domains.

Consider this example: a request for signals intelligence (SIGINT) data on the location of enemy troops is denied because the communication channel used to convey the data is secure and the requestor lacks proper clearance. A human might respond in any number of appropriate ways, such as modifying the query to omit the components of the data denied by the security policy, bringing the matter to the attention of superiors with the power to override the policy, or consulting a different source. An automated system, on the other hand, might simply keep trying forever, or take some other inappropriate action unless it had been programmed with specific instructions for this situation.

The issue then is how to design a system that can respond effectively in situations it was not explicitly designed for

and regarding which it does not have explicit knowledge. Our hypothesis is that this ability can largely be captured by a general-purpose anomaly-processor which, when coupled with an existing host system, improves the system's robustness.

The host system for this experiment is a hierarchical autonomous mission command and control simulated system that consists of unmanned aircrafts and ground robots that can take the roles of squadron members or squadron leaders and a central command. All the entities in the domain function according to pre-set communication and action policies. An instance of the metacognitive component operates on each entity to provide on-board/on-time analysis of anomalies during the operation of the system. Anomalies that an entity faces may be specific to that entity. The metacognitive component we are using is the Metacognitive Loop (MCL) designed to function as a cross-domain plug-in anomaly processor. The following sections examine the MCL architecture, the host system and how MCL helps deal with different kinds of operational policy conflicts within the host system.

2 MCL

The underlying conceptual apparatus of MCL (Schmill et al. 2007; Anderson et al. 2008) is to notice anomalies, assess their importance and cause, and guide a response into place. The general MCL architecture has three sets of ontologies corresponding to the Note-Assess-Guide loop: an *indications* ontology for anomaly types to note, a *failure* ontology for use in assessment, and a *response* ontology for selecting repair types to guide, as in Figure 1.

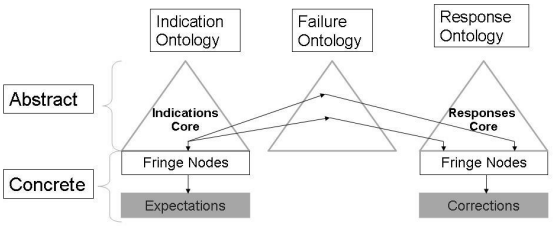

Figure 1: An overview of the MCL ontologies

The core nodes of each ontology are implemented as Bayesian networks. These core nodes represent abstract and domain-general concepts concerning anomalies and how to respond to them. These nodes are linked within each ontology to express relationships between the concepts they represent. They are also linked between ontologies, allowing MCL to employ a number of Bayesian algorithms for reasoning over ontologies.

At the bottom of the indication and response ontologies are the "fringe" nodes. The fringe nodes below the indications core represent concrete, specific information about the anomaly and those below the responses core represent specific correction information.

MCL is linked to the host through two interfaces as shown in Figure 1. At the input interface, expectations are directly linked to the indications ontology through indication fringe nodes. At the output interface, the response ontology's fringe nodes are linked to a set of possible corrections that the host could employ. When an actual perturbation occurs in the host, MCL will detect the expectation violation through the input *fringe* nodes. It will then attempt to map it into the MCL core so that it may reason about it abstractly. MCL's reasoning process then produces an output which is articulated through the output fringe nodes in the form of an action that the host is able to carry out.

2.1 Indications

The core indication nodes represent general, abstract classes of sensory events and expectation types that help MCL disambiguate anomalies when they occur. Fringe nodes are the first to receive an expectation violation. As such, they are defined much more concretely than core nodes: they zero in on specific properties of expectations. For example, the fringe nodes of the indications ontology will encode information such as what type of sensor is being monitored (internal state, time, or reward). The core nodes synthesize this information provided by the fringe nodes and translate it into abstract, specific instances of an indicator such as "deadline missed".

Expectation nodes represent concrete predictions of how the host system's sensors and state should behave over a period of time and under foreseeable circumstances. Expectations are generated dynamically based on what the host system is doing; the expectations for an aircraft autonomous vehicle would be different depending on whether it was landing or taking off, for example. Expectation nodes may be specified by the system designer or learned by MCL, and are linked dynamically into indication fringe nodes when they are created.

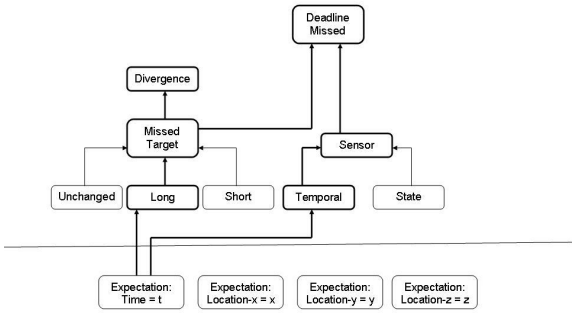

Figure 2: A fragment of the MCL indication ontology

The metacognitive reasoning process is initiated when an expectation is violated. MCL monitors all active expectations as it waits in the Note phase. Once a violation occurs,

the corresponding expectation node is activated. Reasoning with the MCL ontologies can be likened to a spreading activation algorithm (Gaines et al. 2002); through this process, activation spreads into the MCL core along abstraction links.

When a violation occurs, the expectation node that has been directly violated is set to true, and the fringe nodes have their values dynamically updated. Belief (in the Bayes network sense) is then propagated along abstraction links within the indication core. Finally, fringe-to-core links combine the beliefs of individual fringe nodes into specific indicated events.

Figure 2 shows an expectation violation that happens when a squad member takes too long to reach a location. The fringe nodes express the way in which the expectation was violated (the time taken was longer than expected) and the sensor type involved in the violation (temporal sensor). Once all violated expectations have been noted, and belief propagation is finished, the Note phase of MCL is complete.

2.2 Failures

Once the Note phase has been completed, MCL moves to the Assess stage, in which indications are used to hypothesize a cause of the expectation violation. The failure ontology serves as the basis for processing at the Assess stage.

The failure ontology helps deal with the potentially ambiguous nature of indications. In many cases, a single indication might suggest several potential failures. Similarly, a single failure might only be suspected when a subset of indications are present. The mapping between indications to failures, then, might be one-to-many or many-to-one. This rich connectivity is lost without all three ontologies.

Nodes in the failure ontology are initially activated based on activation in the indication ontology, since indication nodes are linked to failure nodes. The interontological links between indication ontology and failure ontology express which classes of failures are plausible given the active indication events.

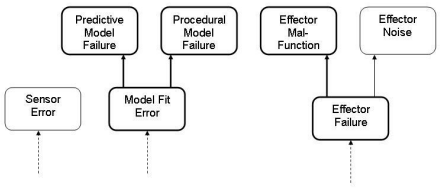

Figure 3: A fragment of the MCL failure ontology

Figure 3 shows a fragment of the MCL failure ontology. Dashed arrows indicate links from the indications ontology leading to the “sensor failure”, “effector failure” and “model error” nodes.

A time exceeded indication can be associated with either of these types of failure. The remaining links in Figure 3 are intraontological and express specification. For example, an effector may fail in two ways: it may be malfunctioning or there could be noise causing it to fail. Either of these is a refinement of the “effector failure” node. As such, “effector malfunction” and “effector noise” are connected to “effector failure” with specification links in the ontology to express this relationship. As in the Note phase, reasoning follows Bayesian belief propagation along specification links to more specific nodes.

2.3 Responses

Outgoing interontological links from active failure nodes will allow MCL to move into the Guide phase. In this phase, potential responses to hypothesized failures are activated, evaluated, and implemented in order of their expected utility. Figure 4 shows a fragment of the MCL response ontology. Pictured are both core and fringe responses and host-level responses. Host-level responses are concrete actions that can be implemented by the host system. Host system designers specify the concrete ways in which MCL can effect changes, such as, in this case, by repeating the action.

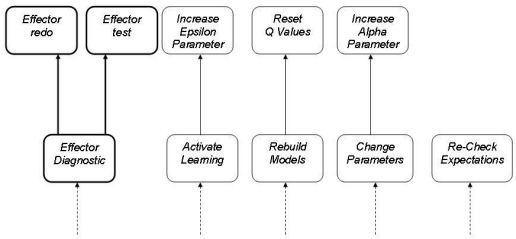

Figure 4: A fragment of the MCL response ontology

In the portion of the response ontology pictured, links from the failure ontology are pictured as dashed arrows. These links cause initial activation in the “effector diagnostic” node. Like the failure ontology, internal links in the response ontology are primarily specialization links; they allow MCL to move from general response classes to more specific ones, eventually arriving at concrete responses. In our example, concrete nodes correspond to repeating actions or testing actions.

2.4 Closing the loop

Once MCL has arrived at a concrete response in the Guide phase, the host system can implement the response. All the activated failures and responses are only considered candidates, and hence MCL must verify that a response is working before it considers an expectation violation addressed.

After a response has been chosen, the state of the three ontologies is stored while the necessary action is taken. MCL *re-enters* the Note phase, waiting to receive feedback from the effected repair. If no new expectation violations are received, then the changes effected during the repair are left in

place, and the violation is considered addressed. If the violation persists, or a new one occurs, then MCL deactivates the invalidated candidate response, and revisits its options for recovery.

3 Test Domain - Afghan-World

Afghan-World is the virtual environment created to test the ability of MCL to detect, analyze and respond to operational policy violations. The physical environment is divided into friendly and enemy territories with agents assigned to protect specific regions. Agents are governed by different operational policies and may not leave their assigned region or enter enemy territory without specific mission instructions from central command. The agents in the Afghan-World include a central command that creates squad leaders and their squads that are made up of unmanned aerial vehicles and land robots assigned for a given geographical area.

Central command will continue to monitor all actions taken during a mission watching the performance of unit members and looking for any anomalies in mission objectives, policy enforcement and overall effectiveness. It can transmit further instructions or recommendations to the squad leader when anomalies occur or further clarification when policy conflicts require it.

The role of the squad leader is to share common mission goals and assignments with squad members, assign specific assignments to each member and then to accompany the squad on the mission to monitor its activities. During the mission the squad leader will communicate with central command as needed and with other squad leaders to coordinate total coverage of the area and to share information other squad leaders may have about enemy activity that could affect their area. Each squad leader has total control over its squad for the mission, requesting data reports and coordinating the activities in the area assigned to the squad.

The unmanned vehicles and robots communicate with their respective squad leader and other squad members to either provide a response to a command message or a request for information. They continuously update their squad leaders with mission parameters. Mission parameters include the current location and speed of the vehicles and robots, current mission status ($aborted$, $delayed$, $doing$ or $done$), current operational policy enforcement and features of enemy targets ($time$, $location$ and $speed$).

The application utilizes a client-server architecture where the squad members are the clients and multiple instances of squad leaders operate on different servers where communication occurs over TCP/IP sockets. The squad leaders and the command center also follow a client-server model. The squad leaders communicate with each other following a peer-to-peer model. Squad members also communicate with other members of their squad following a peer-to-peer model.

4 Operational policies in Afghan-World

The opertational policies for this domain are listed below. These policies are translated and represented by the host system as expectations that need to be monitored by its MCL.

4.1 Geographical Accessibility Policy

This security policy specifies the access rights of the aerial vehicles and land robots. The specific rules of this policy are as follows:

- The unmanned vehicles and robots cannot enter restricted access zones to accomplish their missions because of danger to military and civilian assets.
- The unmanned vehicles and robots cannot travel outside their assigned geographical area to accomplish a mission.

4.2 Secure Communication Policy

This security policy specifies the rules for information sharing between the different entities in the domain. The specific rules of this policy are as follows:

- Squad members can only communicate directly with their squad leader or other members of the same squad.
- Squad members must respond to/act on all squad leader requests for actions or information within a pre-specified number of seconds "t".
- The squad leaders can communicate with each other if the central command has permitted such communication.

4.3 Assets Protection Policy

This policy specifies the action protocol for the land robots and aerial vehicles. The goal of this policy is to minimize the damage to different entities of the system. The rules that come under this policy are as follows:

- The unmanned vehicles and robots must act to protect themselves against enemy and collateral actions at all time.
- The unmanned vehicles and robots cannot take actions to accomplish their mission if said actions will result in civilian casualties.

5 Metacognitive Monitoring and Control in Afghan-World

Each entity in the domain —squad member, squad leader and command center—is coupled with an MCL instance by specifying the MCL fringe nodes associated with that entity. Thus, the fringe nodes are designed with indications and responses tailored to the host system even though the generalized core MCL on each entity is the same. The MCL component of each entity monitors the expectations for that entity, notes any expectation violations, assesses the cause of each violation and guides a response strategy into place. The expectation violations can be the result of conflicts within a single security policy or between two or more policies.

The following scenarios illustrate how metacognitive monitoring and control using MCL allow expectation violations to be identified and evaluated so that a corrective action recommendation can be returned to the entity involved.

5.1 Secure Communication Violation

Suppose an unmanned vehicle does not receive or fully understand a communication from its squad leader and so does not send the required response, sends bad information or does not comply with the squad leader's request within t seconds. This will cause an expectation violation within the Secure Communications Policy of the squad leader. The MCL component of the squad leader notes that (i) the unmanned vehicle never responds to the request, (ii) it responds but does so beyond the time requirement or (iii) it is observed performing an action other than requested in the message. The MCL component notes these anomalies as violations of the expectations that it keeps track of. An example of an expectation is, if the squad leader tells the unmanned vehicle to abort mission, the unmanned vehicle must then turn around so its distance to the base becomes smaller than its distance when it received the message.

When the squad leader's MCL notes the anomaly, it has the option to choose from different responses. These responses include (i) perform a systems check on its own transceiver and recalibrate it, (ii) change the radio frequency or communication array (from short-range to long-range) to attempt to improve reception, (iii) ignore the problem as inconsequential, (iv) resend the message with an incrementing $messagesent$ variable and a response error indicator, (v) request an unmanned vehicle sighting report from another member of the squad to determine if the unresponsive vehicle was damaged or destroyed, (vi) request the vehicle send a mission status report to identify corrupted data, (vii) send a mission update message and wait for the proper reply or action, (viii) send the unmanned vehicle or the entire squad an abort mission message if the success of the mission is endangered by faulty communications, or (ix) re-evaluate the time requirement set for responses.

5.2 Geographical Accessibility and Secure Communication Conflict

Suppose a squad member is given a mission by its squad leader to move to location X even though that location is within one of its restricted zones. In this scenario, any action by the vehicle will violate either the Secure Communication Policy or the Geographical Accessibility Policy. The Secure Communication Policy requires unmanned vehicles to respond/act on all squad leader requests for actions or information within t seconds and the Geographical Accessibility Policy prohibits vehicles from entering restricted areas. This is an example of an inter-policy violation since the two rules are from two different security policies.

The expectations associated with each policy ensures that any policy violation manifests as an expectation violation that can then be noted by MCL. When MCL notes this anomaly, the system can respond in various ways based on the situation. For instance, if the security policy rules have associated weights, then the response could be to abide by the rule with the higher weightage. Other possible responses that the unmanned vehicle can enact include (i) request a new version of the restricted zones list, (ii) request a restricted access zone override, (iii) request a mission update to ensure that it has not received corrupted data, (iv) perform a mission abort if the problem is sufficiently serious, (v) ignore the violation and finish the mission, or (vi) request the squad leader for a security policy update that outlines how to handle the situation.

5.3 Assets Protection and Geographical Accessibility Conflict

Suppose the unmanned vehicle is under attack close to the border of its assigned geographical territory or a restricted zone and must take actions for self-preservation that involve crossing into one of these areas which is prohibited. This scenario could lead to violation of either the Geographical Accessibility Policy or the Assets Protection Policy. Geographical Accessibility Policy requires that the vehicle stay within a geographical area; but the Assets Protection Policy may require the agent to move out of its assigned geographical area. The MCL component of the unmanned vehicle can notice the expectation violations from the observations that the vehicle receives and can guide a corrective action into place.

The response may be to (i) ignore the violation when it arises, (ii) prioritize costs and benefits of each action and choose an action based on this cost-benefit analysis, or (iii) request specific overrides from the squad leader to modify or delete an existing policy or create a new one to handle this situation.

5.4 Assets Protection Violation

Suppose the unmanned vehicle or robot is under attack and is required to take an action to protect itself as per the Assets Protection Policy. If the vehicle or robot is also currently located in a densely populated civilian area any response could result in endangering the civilian population which is also prohibited by the same policy. Since both conflicting rules are from the same policy, this is an example of an intra-policy conflict.

The MCL component of the unmanned vehicle can note this anomaly and respond in different ways to deal with the situation including (i) ignore the violation, (ii) choose to give priority to either civilian assets or self-preservation, (iii) request a specific override (civilian assets or self-preservation) command from the squad leader, or (iv) request a new or updated security rule from the squad leader to handle this situation.

5.5 Secure Communication Violation for Squad Leaders due to Infiltration

Suppose the enemy infiltrates the network and sends a squad leader erroneous observations using the signature of one of its unmanned vehicles or robots. Examples of erroneous observations include speed and location of different squad members or targets. In this scenario, the squad leader may be unaware of the problem and will continue to process the data as though it came from its squad member. However, if the data sent by the intruder conflicts with information sent by other members of the squad, the MCL component of the squad leader can notice the discrepancy between this data

and the information that it received from other sources. Although finding contradictions can be a very difficult problem to handle generally, in this case the situation is sufficiently circumscribed to make it feasible. Given a limited number of objects (vehicles, squads, etc.) that can be assigned a limited number of properties (speed, location, etc.) then every observation is essentially predicating a property of an object. It is straightforward to check that all properties assigned to a given object at a given time are in agreement across the various sources of the observation.

The MCL's corrective response may be to (i) request the unmanned vehicle to send current surveillance data, (ii) request its squad to change frequencies and/or message encryption methods, (iii) distrust one or more sources, or (iv) request a mission abort if the infiltration is deemed a danger to the success of the mission.

5.6 Secure Communication Violation for Squad Members due to Infiltration

Consider another scenario involving enemy infiltration when the enemy sends a command to a squad member pretending to be the squad leader. This violates the Secure Communication Policy requirement that unmanned vehicles only listen to directions from their squad leaders. The unmanned vehicle is unaware that it is breaking the Secure Communication Policy by accepting requests and directions from a prohibited source. However, the MCL component in the squad leader can help detect this anomaly when it notices that the unmanned vehicle has strayed from mission parameters [1]. MCL could notice this anomaly if the observations (like speed and location) regarding the compromised vehicle do not match the expectations that the squad leader has for those mission parameters. These observations could either come from the compromised vehicle (if the communication link to the squad leader is working properly) or from other squad members in the area.

The MCL component could analyze the violation by requesting a current mission status from the unmanned vehicle. If there are no discrepancies, the squad leader can handle the error as a communications problem as outlined in Section 5.1. If an infiltration is detected, the squad leader can (i) change frequencies and/or the encryption method and send a mission update to the unmanned vehicle, (ii) request that the mission be aborted if the infiltration is deemed a danger to the mission or (iii) ignore the violation.

The MCL component of the unmanned vehicle can also identify this anomaly if its expectations about its observations of its squad mates are violated. In this case, the unmanned vehicle can (i) request a mission update from the squad leader, (ii) confirm the timestamp and order number it received with the other squad members, (iii) request a change in radio frequency or encryption if it identifies a network infiltration, or (iv) distrust one or more sources.

[1] See Section 3

5.7 Assets Protection and Secure Communication Conflict

Suppose an aerial vehicle is inside enemy territory and comes under attack from enemy units. In accordance with the Assets Protection Policy the unmanned vehicle takes action to ensure its survival by initiating a stealth mode that terminates all transmission that may give away its location to enemy interceptors. While the vehicle is operating in this mode its squad leader sends it a request for information. At this point the vehicle's two options are to violate the Assets Protection Policy by ending stealth mode to respond to a squad leader's request or violate the Secure Communications Policy by remaining in stealth mode to comply with Assets Protection Policy requirements.

The MCL on board the aerial vehicle will immediately note that both options constitute security policy violations and will have to determine which violation will ensure the greatest chance that the mission will be a success. If the mission hinged on the vehicle's well being, say it was taking pictures of an enemy position, the MCL would be able to determine that the Secure Communications Policy would need to be violated rather than the Assets Protection Policy. However, if the pictures were already transmitted, MCL could violate the Assets Protection Policy in order to transmit other mission critical data and ensure the success of the mission.

The MCL on board the squad leader can note a violation in the Secure Communications Policy response time and could respond by (i) resending the request immediately, (ii) ignoring the violation and resending the request after a certain time period, or (iii) tasking another squad member to get a visual on the violating vehicle to determine if the violation was due to physical damage.

5.8 Illustration

Figures 5 and 6 illustrate how MCL notices an anomaly and corrects the anomaly when one of the squad members deviates from its assigned mission path. Figure 5 shows one squad member moving away from its correct path. The MCL component of the squad leader notices the anomaly because of a failed deadline expectation violation. The MCL component then assesses the failure and issues a corrective action of resending the mission goal to the squad member. As a result, the squad member returns to its assigned path. Figure 6 shows this behavior.

6 Related Work

A collaborative negotiation system is used to resolve conflicts in agent teams in (Jung and Tambe 2000). The system detects conflicts by either explicitly evaluating proposals sent by teammates or by evaluating "role constraints" that specify the maintenance goals for successful role performance. Conflicts are resolved by adopting conflict resolution as a team goal such that team members propose various arguments for resolving the conflict, evaluate the arguments and accept one of the suggested proposals. The maintenance goals that are associated with role constraints are similar in

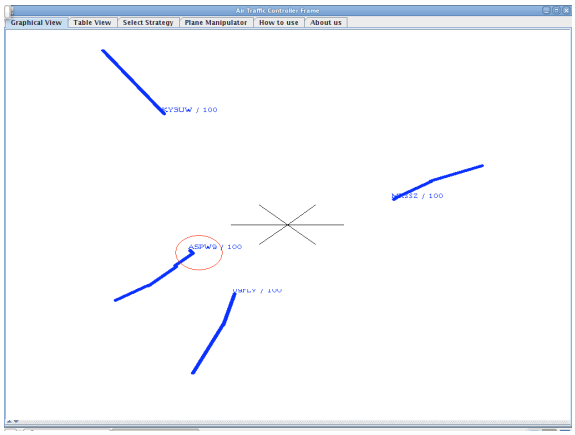

Figure 5: When one squad member deviates from its prescribed mission path.

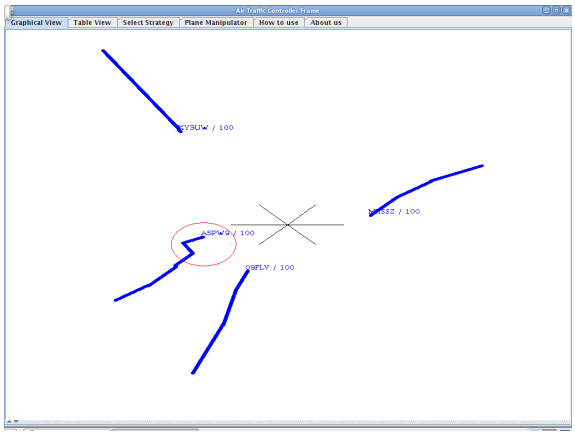

Figure 6: After MCL issues the corrective action to return the squad member to follow its goal.

essence to the expectations in our approach. While this approach uses a separate mechanism for conflict detection and resolution, our approach uses a uniform method to detect all anomalies including policy conflicts—by representing expectations and detecting expectation violations when they occur.

Abductive reasoning techniques are used to detect policy conflicts and refine policies within an Event Calculus (Kowalski and Sergot 1986) based formalism in (Bandara, Lupu, and Russo 2003). In this work, conflicts in the policy specifications are detected by adopting each conflict type as a goal and explicitly querying the system specification for event sequences that would result in that conflict type. This approach is not practical for dynamic agents operating under partially specified policies. Our focus is not in detecting all possible conflicts in policy specifications but those that may occur during the operation of a dynamic agent.

An Execution Monitoring class of security policies that contain enforcement mechanisms capable of terminating objects that violate one of the policies being enforced is presented in (Schneider 2000). This approach is suitable when termination of the violating object is an option, but that is not always a cost effective or feasible way to manage resources in our domain.

The inter-vehicle communication system in (Leinmüller et al. 2006) recognizes nodes that are cheating about their position using a number of independent sensors that give an estimation of another node's trustworthiness with respect to position claims. The trustworthiness of nodes proven to have given false location information is reduced so that such nodes are isolated. Isolating a security breach may not be a viable option for domains with hierarchical control structures.

A conflict resolution architecture for multi-agent systems in the domain of an Air Traffic Management System that favors noncooperative de-centralized conflict management methods over centralized cooperative methods is presented in (Tomlin, Pappas, and Sastry 1998). De-centralization helps manage the computation complexity and communication limitations. This work is complementary to ours in providing host-specific response strategies to deal with anomalies noted by the MCL components of the different entities - local squad members, central command and the squad leaders.

7 Conclusion

Static checking to detect policy violations is impractical and infeasible in dynamic environments. The paper describes a metacognitive monitoring and control technique to deal with intra-policy and inter-policy conflicts in dynamic domains. The technique is based on maintaining expectations, keeping track of expectation violations, mapping expectation violations to appropriate types of failures and mapping failures to appropriate responses to deal with the violations. The paper illustrates the application of the technique in a simulated command and control environment.

8 Acknowledgements

This research has been supported in part by grants from NSF, AFOSR and NASA.

References

Anderson, M. L.; Fults, S.; Josyula, D. P.; Oates, T.; Perlis, D.; Schmill, M. D.; Wilson, S.; and Wright, D. 2008. A Self-Help Guide for Autonomous Systems. *AI Magazine*.

Bandara, A. K.; Lupu, E. C.; and Russo, A. 2003. Using Event Calculus to Formalise Policy Specification and Analysis. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 26. Washington, DC, USA: IEEE Computer Society.

Dinollt, G. W.; Benzinger, L.; and Yatabe, M. 1994. Combining components and policies. In *Proceedings of the Computer Security Foundations Workshop*, 22–33.

Gaines, D. M.; Wilkes, D. M.; Kusumalnukool, K.; Thongchai, S.; Kawamura, K.; and White, J. H. 2002.

SAN-RL: Combining spreading activation networks and reinforcement learning to learn configurable behaviors. *Mobile Robots XVI* 4573(1):80–91.

Jung, H., and Tambe, M. 2000. Conflicts in agent teams. In Catherine Tessier, L. C., and Müller, H.-J., eds., *Conflicting Agents Conflict Management in Multi-agent Systems*. Kluwer Academic Publishers. 153–167.

Kagal, L. 2004. *A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments*. Ph.D. Dissertation, University of Maryland Baltimore County, Baltimore MD 21250.

Kolovski, V. 2008. *Logic-Based Framework for Web Access Control Policies*. Ph.D. Dissertation, University of Maryland, College Park.

Kowalski, R. A., and Sergot, M. J. 1986. A logic-based calculus of events. *New Generation Computing* 4(1):67–95.

Leinmüller, T.; Maihöfer, C.; Schoch, E.; and Kargl, F. 2006. Improved security in geographic ad hoc routing through autonomous position verification. In *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, 57–66. New York, NY, USA: ACM.

Schmill, M.; Josyula, D.; Anderson, M. L.; Wilson, S.; Oates, T.; Perlis, D.; and Fults, S. 2007. Ontologies for reasoning about failures in AI systems. In *Proceedings from the Workshop on Metareasoning in Agent Based Systems at the Sixth International Joint Conference on Autonomous Agents and Multiagent Sytems*.

Schneider, F. B. 2000. Enforceable Security Policies. *ACM Transactions on Information and System Security* 3(1):30–50.

Tomlin, C.; Pappas, G. J.; and Sastry, S. 1998. Conflict Resolution for Air Traffic Management: A Case Study in Multi-Agent Hybrid Systems. *IEEE Transactions on Automatic Control* 43(4).