Identifying Terrorist Activity with AI Plan Recognition Technology

Peter A. Jarvis, Teresa F. Lunt[*], and Karen L. Myers

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025, USA.
Peter.Jarvis@sri.com Karen.Myers@sri.com

Palo Alto Research Center[*]
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
tlunt@parc.com

Abstract

We describe the application of plan recognition techniques to support human intelligence analysts in processing national security alert sets by automatically identifying the hostile intent behind them. Identifying the intent enables us to both prioritize and explain the alert sets for succinct user presentation. Our empirical evaluation demonstrates that the approach can handle alert sets of as many as 20 elements and can readily distinguish between false and true alarms. We discuss the important opportunities, for future work, that will increase the cardinality of the alert sets supported by the system to the level demanded by a deployable application. In particular, we outline opportunities to bring the analysts into the process and the opportunities for heuristic improvements to the plan recognition algorithm.

Keywords: Plan Recognition, Intelligence Analysis.

Introduction

Events in the United States during 2001 tragically demonstrated the nation's vulnerability to acts of terrorism. U.S. security agencies had information available at that time that could have been used to thwart the World Trade Center attack. However, it was not utilized as it was residing within the ocean of other intelligence leads then under consideration [U.S. Senate Report 2002].

Significant research has focused on the problem of uncovering the critical pieces of intelligence information that can be used to thwart an attack from a large body of intelligence leads [DISCEX 2003]. The data mining approaches that have been explored for this purpose can sift through vast quantities of information, but suffer from a high false alarm rate and do not help analysts link together separate facts and events [Ning & Dingbang 2003].

We have developed a proof-of-concept prototype for a tool to automate the analysis currently undertaken by humans by exploiting plan recognition techniques from the artificial intelligence community. Our thesis is that we can significantly improve the quality of the information passed to human analysts if we can automatically discover a significant causal coherence between disparate activities. Our analysis can also aid in the explanations of hypotheses by presenting them in the context of the evidence.

We structure this paper as follows. We first present our Computer Aided Plan Recognition (CAPRe) architecture. We then describe the modeling framework that we use to represent terrorist behavior before detailing the plan recognition algorithm we have developed to match observations with the model. Our experimental section evaluates the performance of the system on alert sets with range of signal-to-noise properties. We close by reflecting on what we have learned and define avenues that must be explored by future work before the capability can be deployed in an operational setting.

Computer-Aided Plan Recognition (CAPRe) Architecture

Our approach, illustrated in Figure 1, involves specifying *a priori* attack templates that are compared against observables to infer whether a particular attack matching one or more templates might be under way. The templates also support the anticipation of future steps so that data collection can be directed, and early interventions to interfere with the plan are possible.

Attack Templates

CAPRe's attack template library contains a description of attack activities structured hierarchically with a specification of the conditions under which they can be combined. The library forms a description of the action physics for a particular application domain that can be used to construct plan instances tailored to specific target requirements, not a library of known attack plan cases.

We draw on the rich hierarchical action representation developed and refined in the automated planning community during the past 30 years (Fikes and Nilsson 1971, Tate 1977). These representations have found application in areas as diverse as spacecraft control (Muscettola et al. 1997) and oil spill response planning (Bienkowski et al. 1994). Figure 2 presents a sampling of the templates in our library for terrorist attacks on a national infrastructure. A template contains information organized into the following slots:

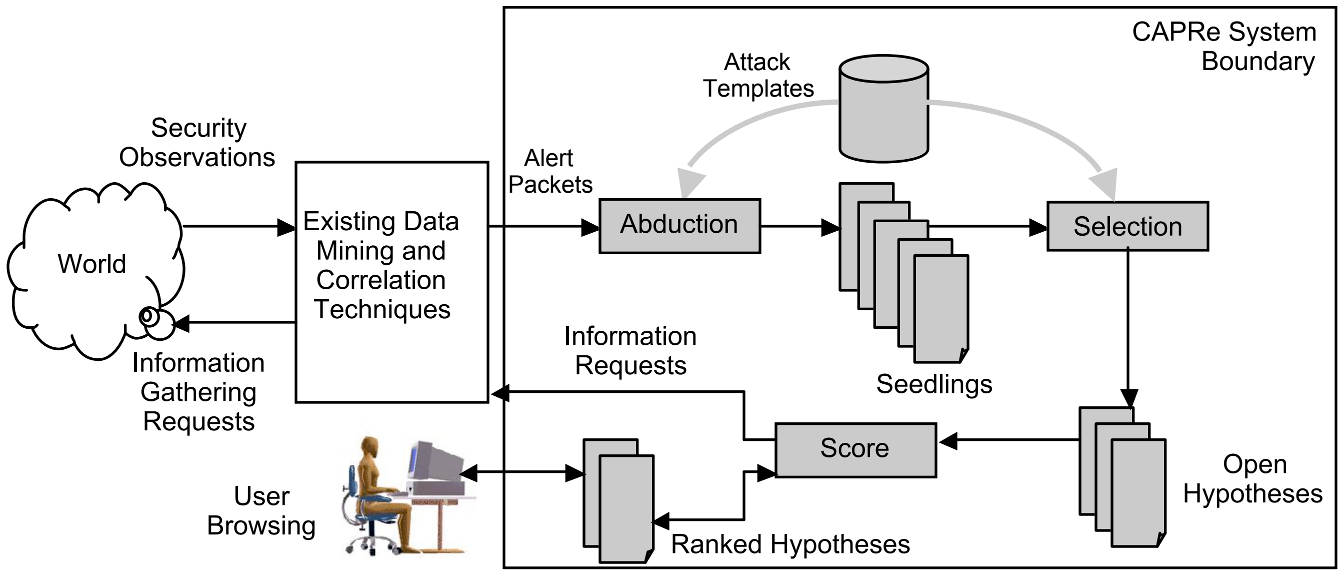


Figure 1: CAPRe Architecture

:template Physical_Attack
:purpose destroy(?group ?target)
:tasks
1. reconnaissance(?group ?target)
2. prepare_attack(?group ?target)
3. attack(?group ?target);

:orderings 1 > 3, 2 > 3;
:end_template

:template Reconnaissance_of_Target
:vars group ?group, target ?target;
:purpose reconnaissance(?group ?target)
:tasks
1. recon_security(?group ?target)
2. recon_structure(?group ?target);

:end_template

:template Research_Structure
:**purpose** recon_structure(?group ?target)
:tasks
1. obtain_structural_plans (?target ?group)
2. take_job (?group ?target)
3. structural_engineering_training (?group);

:conditions member(?person ?group);
:effects retrieved_blue_prints(?person ?group) :at 1.
hr_records(?group ?target) :at 2.
enrollment(?group ?engineering_school) :at 3.
:end_template

Figure 2: Example Templates

- **:vars** - variables that provide typed parameter descriptions for a template
- **:purpose** - the overall purpose of the template
- **:tasks** - set of labeled lower-level tasks to be performed to achieve the purpose
- **:orderings** - temporal constraints on the execution of actions defined in terms of task labels
- **:preconditions** constraints on the execution of a task (e.g., valid driver's license required to rent a car)
- **:effects** - changes to the world that result from the execution of tasks within a template

We define three properties on each task and effect within a template[1]. Each property can take the value *high*, *medium*, or *low*. While more complicated schemes are possible, we decided that this simple scheme would be the most accessible to our user community.

- **Frequency** of the occurrence of a task or effect in normal behavior. For example, car rentals are assigned a *high* frequency, while missing person reports are assigned a *low* frequency.
- **Accuracy** of normal observations of a task or effect. For example, missing person reports are *highly* accurate, while a witness's recollection of a suspicious car's license tag is generally *low* accuracy.
- **Gathering Cost** records the cost of making an observation. Accessing an online database is considered a *low* cost, while an observation that demands a door-to-door search by law enforcement officials is a *high*-cost operation.

Frequency and accuracy properties are exploited during the plan recognition process to score hypotheses or to filter observation lists. Cost is used during the information-gathering planning phase to determine the cost benefit of a particular information-gathering action.

[1]We have omitted these properties from Figure 2 because of space constraints.

$$\begin{tikzcd}[column sep=large]
\text{Retrieve\_blueprints (John\_Doe, Springfield\_dam)} \arrow[r, "\text{Research Structure}"] & \text{ReconStructure (?group, Springfield\_dam)} \arrow[r, "\text{Reconnaissance of target}"] & \text{Reconnasissance (?group, Springfield\_dam)} \arrow[r, "\text{Physical Attack}"] & \text{Attack (?group, Springfield\_dam)}
\end{tikzcd}$$

Figure 3: Example Seedling

Plan Recognition Process

We first provide an overview of our plan recognition process before describing each element in detail. The broad approach is adapted from our earlier work on abductive plan sketch completion (Myers 1997).

Process Overview

Informally, the plan recognition task is to take a set of observed actions and world state changes and a collection of (attack) templates and produce a set of plans that offer potential explanations for the observations within the template set.

Consider the observation *retrieved_blueprints (John_Doe, Springfield_dam)* in the context of the templates shown in Figure 2. We can work backward from this observation through the *research structure* template to conclude that the group of which Mr. Doe is a member is attempting to obtain control information about the Springfield dam. Working back another two steps, we can explain the control information attempt as part of the broader reconnaissance component of a physical attack on this dam. Figure 1 labels this reasoning as the abductive phase of the plan recognition activity that results in a set of *seedling* explanations for each observation.

The *selection* phase of the plan recognition activity takes the set of seedlings generated for all observations and seeks subsets that can be combined consistently with respect to the templates. For example, if we had observed another individual with links to the same organization as Mr. Doe enrolling in an engineering school, then we could combine these two observations to form an *open hypothesis.* If Mr. Doe and this new individual were associated with two different organizations, then these seedlings would not be able to be combined as they violate the *member* precondition in the *Research_Structure* template.

We now describe these two phases in more detail.

Seedling Generation

We use an abductive inference procedure to identify the *seedling hypotheses* that provide candidate explanations for a set of observed activities A. Each seedling hypothesis tops out in an element of the goal space, *G,* for a domain model. While *G* could be defined explicitly, we use the set of template purposes that do not appear as tasks in templates. *Destroy(?group, ?target)* is one example member of *G* that is shown in Figure 2.

Definition 1 (Seedling) The seedling for a task α is the set of labeled linear graphs:

$$A \xrightarrow{O_n,\sigma_n} G_n \xrightarrow{O_{n-1},\sigma_{n-1}} G_{n-1} \xrightarrow{O_{n-2},\sigma_{n-2}} \dots \xrightarrow{O_1,\sigma_1} G_1$$

where $G_1 \in G$, and O_j is a template with purpose G_j and a subtask T such that σ_j is a most general unifier of T and $(G_{j+1})^{\beta}$, for $\beta = \cup_{n \geq i \geq j} \sigma_j$ and $A = G_{n+1}$.

Consider the following observation set:

{retrieved_blue_prints(doe_corp, springfield_dam), enrollment(john_smive, springfield_eng_school), retrieved_blue_prints(doe_corp, sand_dam }.

Figure 3 shows the seedlings that will be generated for these observations given the template set shown in Figure 2. Technically, this figure should show variable bindings, but we have simplified the presentation to the propositional case.

Implementing a seedling generation procedure is simple given the above definition. The only concern is the runtime of the procedure. We define the *abstraction factor* for a task *T* to be the number of template schema subtasks that unify with *T.* We define α to be the maximum abstraction factor for all the tasks in a domain definition. Let l_A be the difference in abstraction level between observations *A* and the most abstract goal in the goal space *G.* The sum

$$\sum_{a \in \text{observation}} \alpha^{la}$$

(where ε =) is a loose upperbound on the construction time for the seedling explanations for a set of observations.

We show empirically the time spent in the seedling generation phase in our experiment section. While a procedure with exponential bounds is cause for concern, we have found in practice that typical domain definitions contain 8 to 10 abstraction levels, and the abstraction factor rarely exceeds 6 (Myers 1997).

We provide two mechanisms that allow the user to influence the seedling generation process. The *frequency filter* parameter allows the user to specify the maximum occurrence frequency of the observations that should be considered. This allows the user to filter out high- or medium-frequency events. The user can also specify classes of events and effects to ignore.

Seedling Selection

The selection phase seeks to combine the seedlings generated in the first phase to form a set of open hypotheses with each member offering an explanation of the intent behind a cluster of seedlings.

Figure 3, all three seedlings would be clustered into a single seedling group, *SG*, as they share the common *destroy* top-level predicate. We now iterate through the powerset of each seedling group (ignoring those of cardinality < 2 and generating the set incrementally) to identify the seedlings that can be combined. The powerset of *SG* that we consider consists of the following sets:

1. {blue_prints_springfield, engineering_training_smive}
2. {blue_prints_springfield, blue_prints_sand}
3. {blue_prints_sand, engineering_training_smive}
4. {blue_prints_springfield, engineering_training_smive, blue_prints_sand}

Corresponding seedling steps can be combined to form an open hypothesis if the following conditions are satisfied:

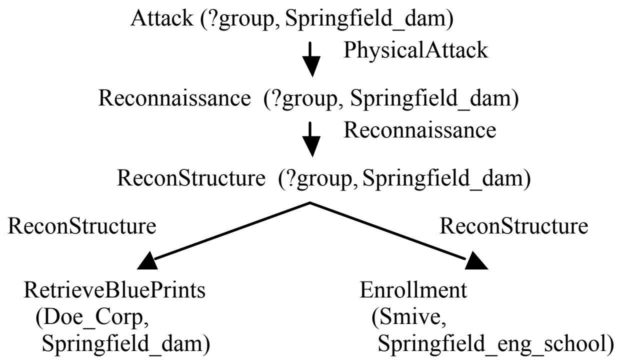
- The purpose statements unify.
- The steps use a common template.
- All constraints in the template are satisfied.
- The bindings entailed by the purpose and task statements across the steps are consistent.

The combination process starts with the top-level step of each seedling in the set under consideration. Consider the members of Set 1. The top-level steps can be combined as the combination conditions are satisfied. Now consider the members of Set 2. The top steps of these two seedlings cannot be combined, as the binding for the *target* variable is inconsistent across the seedlings. Return to Set 1; the combination process continues by considering the next steps in each seedling under consideration. A valid open hypothesis is produced if all steps in all seedlings could be combined.

Figure 4 shows the open hypothesis produced by combining the members of Set 1. While the group behind the attack is not named, the template constraints state that Doe Corp and the individual known as Smive are associated with the group behind the attack.

The concern with the composition procedure, as with the seedling construction phase, is the computational complexity of the procedure. The size of a power set of *n* elements is 2^n.

We carefully structure our search and exploit search pruning to maximize the activity cluster size that we can consider. Our primary strategy is to search through the powerset of seedlings in ascending cardinality order. This strategy has several advantages. First, we can terminate our search as soon as we find a cardinality level with no open hypotheses. Second, we can use a no-good learning strategy. Once we have found a set of seedlings that cannot combine, we can prune all other sets that contain that set as a subset. We show the benefit of no-good learning in the experiments section.



Attack (?group, Springfield_dam)
PhysicalAttack
Reconnaissance (?group, Springfield_dam)
Reconnaissance
ReconStructure (?group, Springfield_dam)
ReconStructure
ReconStructure
RetrieveBluePrints (Doe_Corp, Springfield_dam)
Enrollment (Smive, Springfield_eng_school)

Figure 4: Example Open Hypothesis

Implementation

We have implemented a prototype of our CAPRe architecture in Java. Figure 5 shows the user interface to this system. The top pane displays the current open hypotheses with observations bolded. The left and center bottom panels allow the user to control the seedling generation and compositions process and the scoring function used to sort the open hypotheses. We currently support a simple scoring scheme that rates open hypotheses according to the number of observations that support them. The bottom right panel displays the constraint on the currently selected hypotheses.

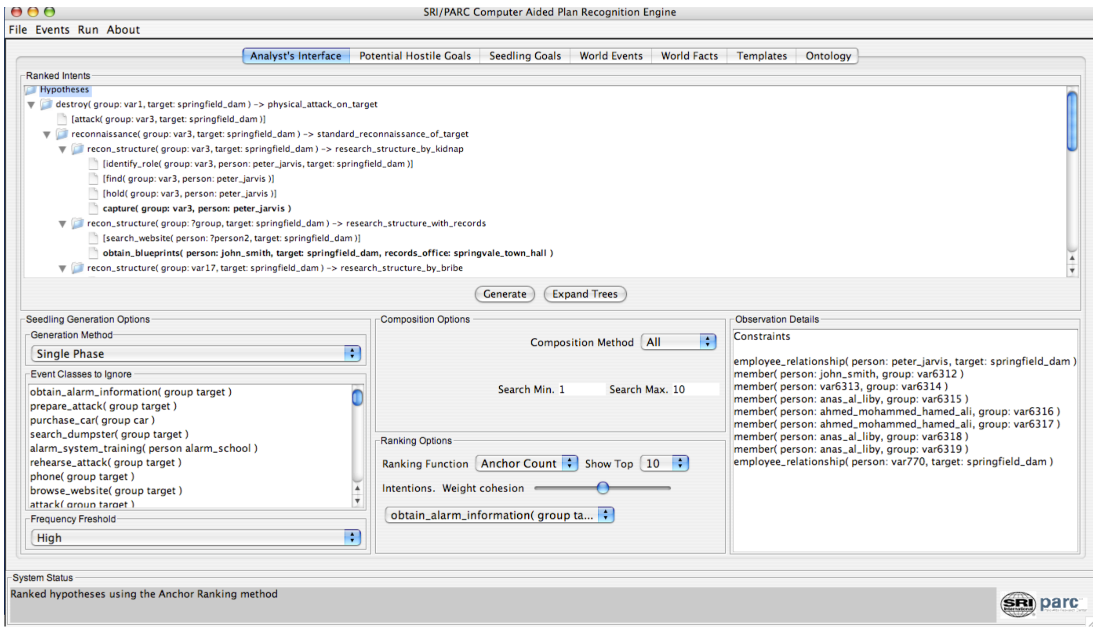
Our current implementation includes only the plan recognition portion of the architecture. Implementing and evaluating the generation of information-gathering plans is left for further work.

Experiments

Our empirical evaluation of the CAPRe implementation examined the performance of the system on a range of alert sets. We focused on variations in the following properties of alert sets:

- **Number of alerts** is the total number of alerts in a set.
- **Signal-to-noise ratio** is the number of alerts in a set that are part of a malicious plan (the target plan) divided by the total number of false alerts in the set.
- **Noise coherence** refers to the maximum number of false alerts in a set that can be combined consistently to form a coherent attack plan.

Table 1 presents the results of our empirical examination of CAPRe. The input alert sets were crafted by hand to include evidence for a target hypothesis together with noise with the appropriate signal-to-noise and coherence properties. We recorded both the runtime of the system and the position of the target hypothesis within the sorted list of hypotheses identified for each set.



```
Ranked Intents
Hypotheses
▼ destroy( group: var1, target: springfield_dam ) -> physical_attack_on_target
    [attack( group: var3, target: springfield_dam )]
  ▼ reconnaissance( group: var3, target: springfield_dam ) -> standard_reconnaissance_of_target
    ▼ recon_structure( group: var3, target: springfield_dam ) -> research_structure_by_kidnap
        [identify_role( group: var3, person: peter_jarvis, target: springfield_dam )]
        [find( group: var3, person: peter_jarvis )]
        [hold( group: var3, person: peter_jarvis )]
        capture( group: var3, person: peter_jarvis )
    ▼ recon_structure( group: ?group, target: springfield_dam ) -> research_structure_with_records
        [search_website( person: ?person2, target: springfield_dam )]
        obtain_blueprints( person: john_smith, target: springfield_dam, records_office: springvale_town_hall )
    ▼ recon_structure( group: var17, target: springfield_dam ) -> research_structure_by_bribe

Seedling Generation Options
  Generation Method: Single Phase
  Event Classes to Ignore
    obtain_alarm_information( group target )
    prepare_attack( group target )
    purchase_car( group car )
    search_dumpster( group target )
    alarm_system_training( person alarm_school )
    rehearse_attack( group target )
    phone( group target )
    browse_website( group target )
    attack( group target )
  Frequency Freshold: High

Composition Options
  Composition Method: All
  Search Min. 1    Search Max. 10

Ranking Options
  Ranking Function: Anchor Count    Show Top: 10
  Intentions.  Weight cohesion
  obtain_alarm_information( group ta...

Observation Details
Constraints

employee_relationship( person: peter_jarvis, target: springfield_dam )
member( person: john_smith, group: var6312 )
member( person: var6313, group: var6314 )
member( person: anas_al_liby, group: var6315 )
member( person: ahmed_mohammed_hamed_ali, group: var6316 )
member( person: ahmed_mohammed_hamed_ali, group: var6317 )
member( person: anas_al_liby, group: var6318 )
member( person: anas_al_liby, group: var6319 )
employee_relationship( person: var770, target: springfield_dam )

System Status
Ranked hypotheses using the Anchor Ranking method
```

Figure 5: CAPRe User Interface

Maximum Noise Coherence		1			4			8		
No. Events	Signal/ Noise	1/3	1/1	3/1	1/3	1/1	3/1	1/3	1/1	3/1
8	**Time**	0:00:00.1	0:00:00.1	0:00:00.1	0:00:00.1	0:00:00.1	0:00:00.1	0:00:00.1	0:00:00.1	0:00:00.1
	Rank	**1/7**	**1/5**	**1/3**	3/3	**1/2**	**1/2**	2/2	*Joint 1st*	**1/2**
16	**Time**	0:00:00.7	0:01:00.4	0:01:26.0	*0:00:01.7*	0:01:02	0:01:25.9	0:03:06.1	0:01:00.1	0:00:23.8
	Rank	**1/13**	**1/9**	**1/4**	*Joint 1st*	**1/3**	**1/2**	3/3	*Joint 1st*	**1/2**
20	**Time**	0:00:09.7	2:27:02.4	0:47:34.0	0:00:06.3	0:47:28.2	0:48:19.4	0:28:01.0	0:48:20.5	0:54:59.4
	Rank	**1/16**	**1/12**	**1/8**	5/5	**1/4**	**1/3**	3/3	**1/2**	**1/2**
25	**Time**	0:19:49.3	*	*	*	*	*	*	*	*
	Rank	**1/19**	*	*	*	*	*	*	*	*

Table 1: Experimental Results

Apple PowerMac G5 1.8 GHz, 500 MB RAM. * denotes no result after 12 hours.

We sorted the set of hypotheses by the number of seedlings combined to generate hypotheses. The more seedlings composed to form a hypothesis the higher its support, and therefore the higher its score.

Examining Table 1 reveals that the time CAPRe takes to identify intent increases exponentially with the number of events in an alert set. This is the behavior that we predicted from the appreciation that our seedling composition step must consider the powerset of the seedlings explanation generated for an alert set. CAPRe is currently limited to alert sets of about 20 actions on state-of-the-art hardware.

The noise coherence and signal-to-noise ratio properties of alert sets affected both the runtime and accuracy of CAPRe. Consider first the group of three results with a noise coherence of 1. In this case, noise cannot mislead the recognition process as each mistaken hypothesis can be supported by only one alert. In this situation, CAPRe consistently ranks the target hypothesis first. When we move to alerts with a noise coherence of 4, CAPRe ranks incorrect hypotheses higher when the signal-to-noise ratio favors the noise. The target hypothesis is ranked first again when the signal-to-noise ratio is 1 or favors the signal. When the noise coherence is adjusted to 8, the results show that it becomes increasingly difficult for CAPRe to correctly identify the target hypothesis. This is understandable, as the noise has become coherent and is dwarfed in cases where it outnumbers or equals the target activities.

We draw two conclusions from the experiments. First, the runtime performance degrades exponentially with alert set size, and for practical purposes 20 alerts is the limit. Second, the accuracy of the system falls off as the cohesion of the noise exceeds that of the actual attack activity.

Summary and Further Work

We have introduced the CAPRe architecture for automating the deep analysis of security alert clusters in order to reduce the load on human security analysts. Our proof-of-concept demonstration illustrates that the technology is capable of recognizing the intent behind events in an alert set and of presenting that intent succinctly to a human user. Our empirical investigation concluded that the technology could process alert clusters of as many as 20 actions and demands that noise (false alerts) be less casually coherent than the components of the attacks.

Further work is needed to move CAPRe to a position where it is ready for operational deployment. We recommend that further work focus on the following three areas:

- **Real Alert Sets:** CAPRe has benefited from a close development relationship with the intelligence community. However, it is essential that future work have access to actual or at least analyst-generated alert sets. Research can then focus on addressing the issues raised by actual rather than projected signal-to-noise ratios and coherence factors.
- **Mixed-Initiative Paradigm:** the number of seedlings generated for an alert set is the critical factor in determining the runtime of the composition phase. Intelligence analysts often have deep insights into the attack activity in progress and events that are supporting false conclusions. Future work should explore a mixed initiative approach where seedlings are presented in a digestible form to the analysts for filtering.
- **Heuristic Development:** We propose to explore two approaches to improving the algorithmic performance of CAPRe. First, we will explore the information-gathering planning concept defined in the architecture to enable us to perform plan recognition on only low-frequency actions in an alert cluster before examining the cluster for alerts that support the set of hypotheses generated. This approach would have the key benefit of reducing the number of seedlings generated for an alert set. Second, we will explore the inclusion of probabilities of observing template tasks given a template purpose in a way similar to that used by Goldman et al. (1999). We will use this information to rank seedlings according to the probability that the observation supports the goal of each. A simple cutoff strategy can then be used to prune unlikely seedlings and again reduce the number of seedlings passed into the computationally expensive combination phase.

References

Bienkowski, M., des Jardins, M., and Desimone, R., 1994. SOCAP: System for Operations Crisis Action Planning. In Proceedings of the ARPA/Rome Lab 1994 Knowledge-Based Planning and Scheduling Initiative Workshop, February.

DISCEX 2003, Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03) 0-7695-1897-4/03, IEEE.

Goldman, R., Geib, C., and Miller, C., 1999. A New Model of Plan Recognition. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 245-254.

Fikes, R., and Nilsson, N., 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence*, Vol. 5, No. 2, North-Holland publishing Co., Amsterdam, Netherlands.

Myers, K., 1997. Abductive Completion of Plan Sketches. In Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), AAAI Press, Menlo Park, CA.

Muscettola, N., Smith, B., Fry, C., Chien, S., Rajan, K., Rabideau, G., and Yan, D., 1997. In Proceedings of the IEEE Aerospace Conference, Aspen, CO.

Ning, P., and Dingbang, X., 2003. Adapting Query Optimization Techniques for Efficient Intrusion Alert Correlation. In proceedings of the 17th IFIP WG 11.3 Working Conference on Data and Application Security.

Tate. A., 1977. Generating Project Networks. In Proceedings of the International Joint Conferences on Artificial Intelligence, pp. 888-893.

U.S. Senate Select Committee on Intelligence and U.S. House Permanent Select Committee on Intelligence, 2002. Joint Inquiry Into Intelligence Community Activities Before and After the Terrorist Attacks of September 11, 2001. *S. Rept. No. 107-351*, December. http://www.gpoaccess.gov/serialset/creports/911.html. Accessed October 8, 2003.

Acknowledgments

The authors thank Tom Garvey and other members of SRI's AI Center who have commented on this work. The DARPA Cyber Panel program through subcontract MP201427 under contract N66001-03-C-8004 supported the work.