CADRE: A System for Abductive Reasoning over Very Large Datasets

Daniel F. Bostwick, Daniel B. Hunter, and Nicholas J. Pioch

BAE Systems Advanced Information Technologies
6 New England Executive Park
Burlington, MA 01803
{daniel.bostwick, daniel.hunter, nicholas.pioch}@baesystems.com

Abstract

CADRE is a system for the detection of complex events in relational data. It implements a form of abductive reasoning that combines data-driven and pattern-driven inferencing to efficiently search for matches in massive amounts of data. It has been applied to a number of pattern detection problems, most notably to the problem of threat detection in massive amounts of data. This paper describes the details of CADRE processing and compares CADRE with other systems for abductive inference. We show that CADRE has unique features that make it especially suitable for the problem of pattern detection in very large relational databases.

CADRE: Continuous Analysis and Discovery from Relational Evidence

With the increasing threat of global terrorism, and an ever-growing sea of computerized intelligence data, manual analysis techniques cannot provide enough coverage to reliably detect threatening activity. To date, most systems used by intelligence and law enforcement agencies have been limited to *link analysis* tools such as Analyst's Notebook® (I2 Inc. 2004), which enable rapid exploration of small- to medium-sized relational data sets. These tools depict data as a graph, in which nodes are usually people, places, or events, and links are binary relations holding between them. Analysts detect threats by visually inspecting the data.

While manual link analysis methods can be usefully applied to small data sets, they break down when data sets become large or densely connected, when relevant clues are so widely scattered that they cannot be conveniently localized on a single visual display, or when inference is required to understand the significance of disparate pieces of evidence in combination. These challenges become all the more difficult when the task is to detect a terrorist threat before an attack occurs, as opposed to investigating after the fact. This suggests a need for new relational information extraction methods to help automate link analysis, including our system, CADRE: Continuous Analysis and Discovery from Relational Evidence.

CADRE implements a form of abductive inference to produce the best explanation of a set of observed facts. Much of the previous work on abductive inference, described in the penultimate section of this paper, is not suitable to the sort of problem domain that CADRE was designed for — the detection and prediction of threat events from massive amounts of data with low observability of attributes and links and where the hypothesis space is vast.

Threat detection involves correlating a relatively small number of clues scattered among very large datasets that contain mainly noise and clutter, where noise consists of random events that may happen to fit a subpattern of the threat pattern while clutter consists of events conforming to a nonthreat pattern having some similarity to the threat pattern Note that for this type of problem, we are not given a set of observations to explain, but rather must search for relevant observations, melding information extraction with abductive inference. We regard the key challenge for this sort of problem to lie in the efficient generation of hypotheses, rather than in the evaluation of hypotheses already generated.

CADRE deals with the hypothesis generation problem for large hypothesis spaces by carefully tailored combination of bottom-up and top-down processing, which focuses the search for hypotheses in such a way that the number of hypotheses generated is manageable. In addition, we have implemented a very compact encoding of large numbers of hypotheses that avoids the need to store full hypotheses explicitly. To deal with the incompleteness and sparseness of relevant data, we employ a hierarchical, constraint-based representation of patterns that allows incremental filling-in of a hypothesis at different levels and that permits constraint-based inferences about missing data.

The following sections describe CADRE's approach to handling the problem of hypothesis generation and evaluation for large, noisy datasets. We begin by describing the types of representations used by CADRE for hypotheses and data. Next we explain the details of

CADRE's hypothesis generation and evaluation process. We conclude with a comparison of CADRE with previous approaches to abductive inference.

CADRE Representations

The inputs to CADRE are:

- A set of patterns
- Relational evidence
- A procedure for evaluating hypotheses

Given these inputs, CADRE produces new high-value hypotheses, where each hypothesis incorporates a set of elements from the relational evidence which match the pattern.

CADRE evidence, patterns, and hypotheses are represented in the language of the AIT Knowledge Server (AKS). The AKS is a computationally-efficient, constraint-based knowledge server that provides a semantics richer than conventional frame system attribute/value relations [Minsky 1975]. The AKS language represents concepts as a class hierarchy with multiple inheritance, slots with type and cardinality constraints, and default inheritance of slot values. In addition, it incorporates certain constructions from description logics, such as enumerations and class unions [Hunter and Bostwick 2005, Everett et al 2003].

An AKS *context* is a set of classes that share the same implicit set of assumptions. Contexts provide a mechanism for partitioning a knowledge base into self-consistent chunks. Contexts enable reasoning within a subset of the knowledge base, and they allow different parts of the knowledge base to be inconsistent with each other, facilitating "what if" reasoning. In the AKS, contexts are arranged in a single (tree-structured) inheritance hierarchy such that anything that is true in a context is also true in its subcontexts. This hierarchical structure of AKS contexts is vital to CADRE's ability to efficiently represent many alternative hypotheses.

Pattern Representation

CADRE patterns are represented as a set of AKS classes with constraints among them. For example, a pattern for a Mafia-style contract killing might look like this:

```
Class MurderForHire : event
   mafiaGroup : criminalOrganization
   contractor : person
   hitman : person
   target : person
   downPaymentToHitman : payment
   finalPaymentToHitman : payment
   confirmation : phoneCall
   killing : murder

   hitman subset mafiaGroup.members
   contractor subset mafiaGroup.members
   killing.victim = target
   killing.perpetrator = hitman
   downPaymentToHitman.payor = contractor
   downPaymentToHitman.payee = hitman
   confirmation.caller = hitman
   confirmation.recipient = contractor
   killing.timeOfEvent =downPaymentToHitman.timeOfEvent
                        + [0,14] days
   finalPaymentToHitman.timeOfEvent = killing.timeOfEvent +
                                      [1,5] days
   confirmation.timeOFEvent = killing.timeOfEvent + [0,6] hr
End
```

In this example, the pattern MurderForHire has a number of attributes (called *slots* in a frame system). Each of these slots has a name, type and cardinality. For example the slot for the person ordering the murder is named *contractor* and is of type *person*. The default cardinality is that there is exactly 1 contractor. The type of a slot can be an AKS class, like *person*¸ or a primitive type like *number* or *string*.

In addition to slots, the *MurderForHire* pattern has constraints between its slots that limit the fillers of the slots. The constraint killing.victim = victim specifies that the filler of the *victim* slot of the filler of the *killing* slot must be equal to (that is, be the same instance of *person*) as the filler of the *target* slot. The AKS language also supports interval constraints between slots. The pattern specifies that the killing, for example, must occur 0 to 14 days after the down payment.

One important feature of CADRE patterns is that they are hierarchical. That is, a pattern can be composed of numerous subpatterns. CADRE can first hypothesize instances of the subpatterns then use those hypotheses to build up matches to the top level pattern. For example, the *MurderForHire* pattern could be a subpattern of a pattern describing an attempt by a Mafia group to take over a particular industry. Individual *MurderForHire*s could be related by constraints on geographic location, time, industrial ties of the victims, and the specific Mafia group involved.

Evidence Representation

CADRE represents evidence as AKS instances. For example, a particular payment might be represented like this:

```
Instance Payment001 : payment
   payor = mr_x
   payee = mr_y
   amount = 500 dollars
   timeOfEvent = 4/2/2004 4:31 pm
End
```

In this example, *mr_x* and *mr_y* are instances of *person*.

Over the lifetime of CADRE, we have interfaced with various external evidence sources: text files, relational databases, XML. In all cases, though, the evidence is represented internally to CADRE as AKS instances.

Hypothesis Representation

A single hypothesis is represented in CADRE as a set of AKS instances that satisfy the constraints of the pattern. The instances that comprise a hypothesis may be either instances from the evidence or hypotheses for subpatterns.

In the domains to which we have applied CADRE, there are typically many consistent ways of matching data to a pattern. Managing the large number of hypotheses that result was a central concern for the design of CADRE, as one of the problems historically associated with abductive reasoning has been an explosion in the number of hypotheses generated as the number of rules used for abduction increases. To address this issue, CADRE employs a representation analogous to the track-tree representation in multi-hypothesis tracking, a widely used technique in data fusion applications [Blackman and Popoli 1999]. CADRE applies two main concepts from multi-hypothesis tracking to relational link discovery: 1) an algorithm for assembling a global hypothesis on-the-fly from the best candidate local hypotheses, and 2) a packed tree representation of local hypotheses in which child node in the tree represents an elaboration of additional links in the hypothesis. Each node is a separate context corresponding to a partially filled-out hypothesis. The root context corresponds to the raw evidence data. The tree is extended by making queries within each current leaf context, based on the assumptions of that context, and creating new child contexts corresponding to the different data matches obtained. The hypothesis tree structure compactly represents hypotheses via *context inheritance;* each context inherits the information present in the parent context and so only needs to store a single match to a query. In Figure 1, hypothesis_1 is extended in two ways to form hypothesis_1a and hypothesis_1b. Because of context inheritance hypothesis_1b only needs to store the association between the *confirmation* slot and *call001*.

Since global hypotheses can be constructed by simply selecting one local hypothesis per context tree, it follows that a set of m context trees implicitly represents h^m distinct global hypotheses, if h is the average number of local hypotheses in each context tree. Since the total number of local hypotheses is hm, the overall compression factor achieved by our factored hypothesis representation is given by:

$$\frac{h^{m-1}}{m}.$$

For a collection of 25 datasets processed by CADRE, we calculated an average compression ratio of 4.4×10^{56}.

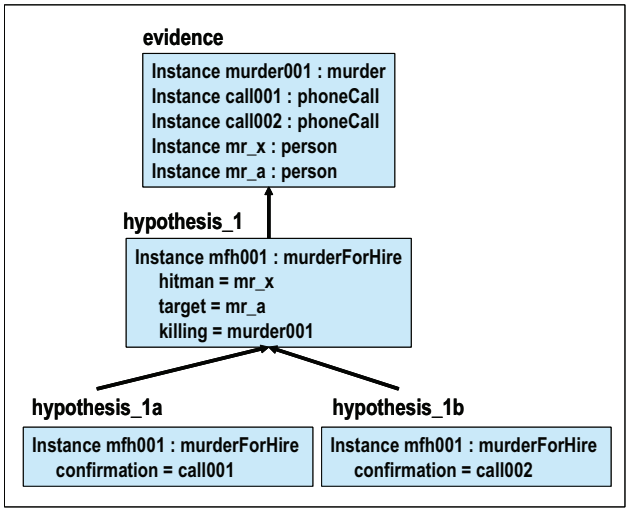

Figure 1. Hypotheses are stored in a context hierarchy

CADRE Operation

CADRE's operation may be divided into three distinct procedures:

- Triggering new hypotheses
- Refining hypotheses
- Global hypothesis generation

Triggering is the procedure of determining when one or more pieces of evidence fit together in such a way as to warrant forming a new hypothesis. Refining a hypothesis involves incrementally making abductive inferences to add pieces of evidence to a hypothesis. Global hypothesis generation takes many local hypotheses, some of which may be alternative explanations for the same event, and creates a set of high value hypotheses that is consistent.

Triggering New Hypotheses

The first step in CADRE's operation is called *triggering*. During triggering, CADRE issues queries, called *triggering rules*, to the evidence to find matches to small subpatterns that can serve as seeds for growing more complete matches to the pattern. Each match to a triggering rule results in a new hypothesis in its own AKS context.

Good triggering rules have the following features:

1. They involve events or combinations of events that are not common in the evidence database
2. They involve events or combinations of events that are strongly associated with the pattern of interest
3. The subpattern matched by the triggering rules is strongly connected, via AKS constraints, to the rest of the pattern

The first feature is important for reasons of efficiency and feasibility. For example, although it is part of the *MurderForhire* pattern that one person phones another, it

is not feasible to attempt to find *MurderForHire*s by *first* sifting through *all* phone calls; there are likely to simply be too many phone calls in the evidence database that are not part of a *MurderForHire*. Looking at phone calls may only be feasible *after* a set of individuals has been identified through other means. The second feature provides assurance that we are not likely to miss instances of the target pattern by focusing on features that, although perhaps sometimes found to occur in pattern instances, are frequently missing. Finally, a good starting place for hypothesis generation should have the property that it allows us to "unravel" the rest of the pattern in the data by following links from the subevents and individuals detected in triggering to other subevents or individuals that may be involved in the whole pattern.

In the past we have used handcrafted triggering rules informed by a detailed study of the problem domain. Recently, however, we made some progress toward automating triggering rule generation. Given a pattern where one slot is designated as the *anchor*, CADRE can use the constraints on that slot to automatically generate a triggering rule in much the same way it generates abductive queries during hypothesis refinement (see the following section). While this approach still requires a human to choose an appropriate anchor, it is much less burdensome than manually authoring a rule.

To further automate the trigger rule generation process we have implemented a scheme which utilizes inductive logic programming to generate a trigger rule that maximizes some metric. The procedure searches through the space of possible triggering rules by combining query clauses from rules generated from the constraints on any slot in the pattern. The search begins with generating some seed rules. Seed rules are generated from the unary constraints (typically type constraints) on slots in the pattern. As the search continues, the candidate rules are extended by adding n-ary constraints between slots already in the rule and other slots in the pattern.

Each candidate rule is given a score based on some metric. For our initial experiments we used the time to detect a pattern as the metric. For each candidate rule we searched for elements in the evidence that satisfy the conditions of the rule, we call this a *trigger match*. Then we use CADRE's hypothesis refinement mechanism, described below, to attempt to grow the trigger match into a match for the full pattern. The mean time it takes to find a match to the pattern for a given candidate rule leads to a score for the rule. The faster the time, the higher we score the rule.

Our initial results show that we can generate a rule that maximizes our metric for non-hierarchical patterns. While these results are encouraging, there is opportunity for further work in this area. Our approach to learning triggering rules does not require ground truth data, making use instead of constraints in patterns that have already been created. Similarly, the metric used – mean time to match the pattern – does not require ground truth Although this approach has the advantage that it can be applied to any evidence set, when ground truth data is available, other machine learning techniques could be applied to learn triggering rules and alternative metrics such as precision and recall used to score them.

Hypothesis Refinement

Following triggering, CADRE extends the triggered hypotheses in a top-down process we call *hypothesis refinement*. Hypothesis refinement takes as input the set of triggered hypotheses and emulates the manual link analysis process by incrementally expanding each partial match with new linked evidence that satisfies constraints involving known slots.

We can view the evidence and pattern as graphs. Triggering thus produces a match of a subgraph of the pattern to a subgraph of the evidence. In this formulation, hypothesis refinement can be explained as the process of incrementally extending this match based on the pattern description. For a given hypothesis, CADRE generates queries for an unfilled slot, S, by examining constraints that relate the unfilled slot to filled slots. Each query results on a (possibly empty) set of fillers for S. For each filler, V, CADRE extends the hypothesis in a child context where V is assigned to S.

For example, given the following hypothesis and evidence instances and class constraints on the *MurderForHire* class, CADRE will query for fillers of the *confirmation*.

```
Instance mfh001: MurderForHire
    hitman = mr_x
    target = mr_a
    killing = murder001
End

Instance murder001 : murder
    victim = mr_a
    perpetrator = mr_x
    timeOfEvent = 4/1/2004 10:13am
End
```

CADRE will query for phone calls slot that satisfy the constraint that mr_x is the caller, and the call occurs between 10:13am and 4:13pm on 4/1/2004. That these are the correct conditions to place on the phone call can be seen by examining the constraints on the *MurderForHire* class and the slot fillers on *mfh001* and *murder001*.

The constraint:
confirmation.timeOFEvent=killing.timeOfEvent+[0,6] hr
and the fact that
murder001.timeOfEvent = 4.1.2004 10:13am
leads to the condition that the confirmation occurs between 10:13am and 4:13pm on 4/1/2004.

The constraint:
confirmation.caller = hitman
and the fact that
mfh001.hitman = mr_x
leads to the condition that the caller is *mr_x*.

For each phone call matching to the query, CADRE will abductively infer that the call is a confirmation of the killing, as in Figure 1. Furthermore, by straightforward reasoning about other constraints on the pattern, CADRE can infer the identity of the contractor (because of the constraint that the recipient of the call must be equal to the contractor).

Hypothesis refinement halts when every hypothesis either has no unfilled slots, or every abductive query results in no additional matches. At this point CADRE has generated a set of *local hypotheses* that have been maximally associated with available evidence.

Globally Consistent Hypotheses

While each local hypothesis is self consistent, that is, satisfies the constraints on the pattern, there may be inconsistencies between local hypotheses. The hypotheses 1a and 1b in Figure 1 cannot both be true since they specify different phone calls as the filler of the *confirmation* slot. At this point CADRE can generate a *global hypothesis,* which is a set of consistent high value local hypotheses. From an operational point of view a user of the CADRE system may wish to see alternative high value explanations for the same event. In such a situation CADRE could simply present a ranked list of hypotheses, or allow the user to explore alternate explanations for the same events.

In order to compute a consistent global hypothesis or even to rank local hypotheses, CADRE requires a method to assign a score to hypotheses. We have explored both general-purpose and domain-specific scoring methods. For one application we developed an evaluator that computes the conditional probability that the hypothesis is an instance of a given pattern. This probabilistic evaluator utilizes deep knowledge of the target pattern as well as clutter patterns. Clutter patterns are patterns which closely resemble the target pattern. It computes P(Pattern|Hypothesis) for each clutter pattern in addition to the target pattern. For other applications we have utilized a simpler evaluation metric where we count the number of slots in the pattern that are filled in a hypothesis.

CADRE uses a greedy algorithm to generate global hypotheses. First the highest scoring local hypothesis, L1, is added to the global hypothesis, H. Then all local hypotheses that are inconsistent with L1 are eliminated. The highest scoring remaining local hypothesis, L2, is then added to H and local hypotheses that are inconsistent with L2 are eliminated. The process continues until every local hypothesis is either in H or has been determined to be inconsistent with a member of H.

Comparison with Previous Research

CADRE implements a form of abductive inference to produce a plausible explanation, or set of explanations, of a set of observed facts. Abductive inference has been a topic of discussion among logicians and philosophers for more than a century (e.g. [Peirce, 1903]), but implementations of abductive reasoning within the Artificial Intelligence community are of more recent origin. In the 1980's, much of the research on abductive inference was focused on a particular type of problem, that of diagnostic problem solving ([Reiter, 1987], [de Kleer and Williams, 1987]). In diagnostic problem solving, we are given a set S of observed *symptoms*, and set D of *disorders*, and some kind of *causal model* M relating disorders to symptoms (e.g. a set of causal relations between disorders and symptoms). The task is to find a minimally sufficient set of disorders that account for the symptoms. (Formally: find a subset E of D such that E together with M imply S and such that no proper subset of E together with M implies S.) Examples of such problems include medical diagnosis and diagnosis of faults in electronic circuits.

[Peng and Reggia, 1990] develops this line of reasoning into a formal theory of diagnostic problem solving called *parsimonious covering theory*. There are some points of contact between Peng and Reggia's algorithm for diagnostic problem solving and CADRE's processing. There is a notion of seed hypotheses and a notion of incremental refinement of hypotheses. In addition, Peng and Reggia introduce the idea of first generating complete hypotheses through a deterministic process and then choosing a best hypothesis by evaluating the probability of competing hypotheses using a probabilistic model.

Peng and Reggia's abductive inference algorithm is designed to search efficiently through a space of hypotheses represented as subsets of a relatively small enumerated set of "disorders." It was not designed to handle complex relational data involving very large numbers of entities and events. It is therefore not directly applicable to the sort of problem CADRE was designed to handle, although the conceptual framework for abduction developed by Peng and Reggia is useful in illuminating what CADRE is doing.

Other work in the 1980's on abductive inference in a quite different domain has a much closer affinity to CADRE's approach. [Charniak, 1988] presents a theory of "abductive unification" for the domain of motivation analysis. Motivation analysis is a means of understanding the actions of a character in a story by explaining those actions in terms of the character's motivation. Charniak's representational framework is very close to that of CADRE

– a frame-based (script), hierarchical representation of actions with constraints among actors and roles. Moreover, Charniak's approach is based upon a kind of consistency checking very similar to what CADRE uses in refining hypotheses: in determining whether a slot can be filled with a particular data item, Charniak tests the consistency of assuming that the slot value is identical to the data item by seeing if the data item violates any of the constraints involving that slot. CADRE's constraint checking is similar but more general in that it can deal with multi-valued slots, not just functional slots, and can express a wider range of constraints than Charniak's formalism can.

We have been strongly influenced by Charniak's approach and have extended his ideas in CADRE. By introducing triggering rules, we address, at least partially, the question of where initial hypotheses come from. Charniak's consistency checking algorithm forms the basis for our hypothesis refinement step, in which the initial hypotheses are extended with new evidence or pruned if they are found to be inconsistent with the evidence.

In the 90's Bayesian networks [Pearl, 1988] emerged as a popular form of probabilistic inference. Bayesian networks can perform abductive inference by computing the most probable assignment to hidden variables conditional on assignments to some subset of the observable variables. (A special form of this computation is routinely used in Hidden Markov Models, which can be regarded as Bayesian networks with a specific structure.) As with the work on diagnostic problem solving, however, the limited representational power of Bayesian networks renders them unsuitable as solutions to the problem of abductive inference over very large relational datasets (although they may be used in evaluating hypotheses generated by other means). There are two representational problems with standard Bayesian networks: (1) they have essentially the representational power of propositional logic and cannot easily capture the relational information contained in data about complex events with subevent structure and actor roles; and (2) despite the compact factored representation of probability distributions provided by Bayesian networks, the entire prior hypothesis space for even a moderately large relational database is so large that any Bayesian network attempting to capture it would be computationally intractable. (For example, some nodes would have tens or hundreds of thousands of states.)

There has been work in recent years on extending the expressive power of Bayesian networks (e.g. [Koller and Pfeffer, 1998]). Relational Probabilistic Models, for example, are a way of imposing a Bayesian causal model on a relational database schema (or a frame-based pattern representation). This addresses problem (1) above but does not obviate problem (2). The CADRE approach is to use deterministic processing to generate a relatively small set of candidate hypotheses, which may then be evaluated by Bayesian networks (or other kinds of probabilistic models such as HMMs) of manageable size.

We conclude this comparison by mentioning two systems whose representation of patterns is very similar to CADRE's, but whose inference procedures differ in interesting ways.

The CAPRe system [Jarvis et. al., 2005] is similar to CADRE in more than name. It makes use of a hierarchical representation of terrorist attack plans modeled on representations used within the AI planning community. In essence, their representation decomposes complex, high-level actions into subtasks; a subtask may itself be complex and if so a pattern (template in the terminology of [Jarvis e. al., 2005]) exists which decomposes it into actions further down in the hierarchy. Action decomposition bottoms out in observable actions. It should be clear that this representation can be mapped directly into CADRE's hierarchical class representation.

CAPRe's inference procedure, however, is quite different from that of CADRE. For each observable action that matches a slot in a bottom level pattern, CAPRe constructs a hypothesis that embeds the action and its performers in increasingly higher level patterns, up to the top level attack hypothesis. Since many different primitive actions may in fact be part of a single attack activity, the hypotheses generated by single actions must be merged to collect together consistent actions into a single hypothesis.

With 20 observable actions, processing time for CAPRe ranged from 6 minutes to 55 minutes, depending upon the characteristics of the data set. (Unfortunately, as the signal to noise ratio – the ratio of actions that are part of an attack plan to those that are not – increases, processing time increases.) Indeed, the authors admit that "CAPRe is currently limited to alert sets [sets of observable actions] of about 20 actions on state-of-the-art hardware." ([Jarvis et. al., 2005,, p. 79].) In contrast, CADRE has processed datasets containing 100,000 events and 10,000 individuals in 10 minutes. The difference lies in CADRE's combination of bottom-up and top-down processing. The bottom-up process of triggering provides CADRE with plausible starting points for hypotheses; refinement is a top-down process that assimilates additional evidence in a highly directed manner.

Finally, another system that has interesting similarities and differences with CADRE is SRI's LAW [Wolverton et. al., 2003]. LAW represents patterns and data as graphs. This is an inessential difference since it is straightforward to transform a frame-based representation into a graph representation and vice-versa. Hierarchical patterns can be represented in LAW and both temporal relations and role relations can be expressed as annotated edges in the graph. LAW's processing differs from that of CADRE in two major respects: (1) LAW makes use of a graph-edit distance metric to evaluate the quality of a match; this

allows LAW to accept as a "close" match data patterns that do not strictly conform to a pattern; by contrast, CADRE, while capable of accepting partial matches in the sense of hypotheses with missing elements, will reject a hypothesis if its elements do not strictly conform to the dictates of the hypothesis pattern. (2) LAW makes use of the A* search algorithm in refining a pattern, whereas CADRE deterministically iterates through unfilled slots in search of matching data.

On the other hand, there appears to be no explicit context mechanism in LAW nor does there appear to be any attempt to form a single global hypothesis – LAW returns all valid matches found.

The comparison between LAW and CADRE is perhaps the most intriguing of those mentioned since they appear to have similar core functionality with different but complementary add-ons to this functionality. Techniques from LAW such as fuzzy matching might usefully be applied within CADRE and similarly, techniques from CADRE, such as the context mechanism and global hypothesis generation, might be grafted onto LAW.

Conclusions

CADRE has significant advantages over other systems for abductive inference when the hypothesis space is so large that exhaustive enumeration of hypotheses is intractable. CADRE constructs hypotheses incrementally, starting from data known to be highly relevant, then using constraint-based reasoning to guide the search for additional pattern matches. This approach allows CADRE to be usefully applied to realistic problems involving massive amounts of data in which instances of the target pattern are sparse and incomplete.

Although we have focused in this paper on the domain of terrorist threat detection, we wish to emphasize that CADRE is a domain independent inference tool. We have also applied CADRE to the detection of complex events in video data. Video processing systems were run on videos of meetings to perform speech recognition, person detection, and motion detection. Utilizing those video processing outputs and patterns of interesting behaviors, CADRE discovered instances of those behaviors in the videos. CADRE performed very well in this domain achieving a mean 96.5% precision and 84.5% recall while detecting instances of four different patterns of complex events.

Acknowledgements

This work was sponsored by the U.S. Government, in part, by the Air Force Research Laboratory, under Contract F30602-01-C-0195.

References

Minsky, M. 1975. A framework for representing knowledge. In P. Winston ed., *The Psychology of Computer Vision*. New York: McGraw-Hill, pp. 211-280.

Blackman, S. and Popoli, R. 1999 *Design and Analysis of Modern Tracking Systems*, Artech House, Norwood, MA.

Everett, J., Bostwick D, and Jones, E. 2003 Rapid knowledge base design via extension of mid-level knowledge components. In *International Conference on Integration of Knowledge-Intensive Multi-Agent Systems (KIMAS '03) (Cambridge, MA, Sep. 30-Oct. 4, 2003)*. IEEE Press, Cambridge, MA, 535-541.

Hunter, D and Bostwick D. 2005. Default Reasoning with Contexts. In *Context and Ontologies: Theory, Practice and Applications*, Technical Report WS-05-01, AAAI Press, Menlo Park, CA, 112-115.

Peirce, C.S. 1903. Abduction and induction. In J. Buchler (Ed.), *Philosophical Writings of Peirce*, pp. 150-156. New York: Dover (1955).

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57-95.

de Kleer, J. and Williams, B. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32:97-130.

Peng, Y. and Reggia, J.A. 1990. *Abductive Inference Models for Diagnostic Problem-Solving*. New York: Springer Verlag.

Pioch, N., Hunter, D., White, J., Kao, A., Bostwick, D. and Jones, E. 2004. Multi-hypothesis abductive reasoning for link discovery. *Proceedings of LinkKDD-2004*. Seattle, WA. 22-25 August, 2004.

Charniak, E. 1988. Motivation analysis, abductive unification, and nonmonotonic equality. *Artificial Intelligence*, 34:275-295.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann.

Koller, D. and Pfeffer, A. 1998. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, WI: AAAI Press.

Jarvis, P.A., Lunt, T.F., and Myers, K.L. 2005. Identifying terrorist activity with AI plan-recognition technology. *AI Magazine* 26(3):73-81.

Wolverton, M., Berry, P., Harrison, I., Lowrance, J., Morley, D., Rodriquez, A., Ruspini, E., and Thomere, J. 2003. LAW: A workbench for approximate pattern matching in relational data. In *The Fifteenth Innovative Applications of Artificial Intelligence Conference (IAAAI-03)*.