AFRL-RI-RS-TR-2009-91
Final Technical Report
April 2009

NATIONAL CENTER FOR MULTISOURCE INFORMATION FUSION

Calspan-UB Research Center, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2009-91 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/ /s/

JOHN J. SALERNO
Work Unit Manager

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
APR 09	Final	Jun 06 – Dec 08

4. TITLE AND SUBTITLE
NATIONAL CENTER FOR MULTISOURCE INFORMATION FUSION

5a. CONTRACT NUMBER
FA8750-06-C-0184

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63789F

6. AUTHOR(S)
Jared Holsopple, Shanchieh Jay Yang, Mike Kuhl, David Hall, Rakesh Nagi, Stu Shapiro, Moises Sudit, Brian Panulla, Mike Kandefer, Patrice Seyed, Zhang Ying, Adhijit Gosavi, Kevin Costantini, Greg Tauer

5d. PROJECT NUMBER
NIFR

5e. TASK NUMBER
CM

5f. WORK UNIT NUMBER
FS

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Calspan-UB Research Center, Inc.
4455 Genesee Street
Buffalo NY 14225-1928

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/RIEA
525 Brooks Rd.
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2009-91

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88-ABW-2009-0144

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The National Center for Multisource Information Fusion (N-CMIF) research was a joint collaboration between CUBRC, University at Buffalo (UB), Rochester Institute of Technology (RIT) and Penn State University (PSU) to address information fusion research gaps present in situation, threat, and impact assessment (JDL levels 2 and 3) as well as sensor management (JDL level 4) and visualization. While much of the research conducted under N-CMIF emphasized computer security, the research also aimed to address the problems in a manner applicable to other domains. Major accomplishments in N-CMIF include (1) addressing current gaps in information fusion and computer security; (2) the enhancement of the Cyber Attack Simulator (a tool to automatically generate cyber attack scenarios for a given computer network); (3) Future Situation and Impact Awareness (FuSIA) (a level 2/3 framework implemented in Java that enhances situation awareness by identifying the current and future impact of a situation as well as providing run-time performance metrics to evaluate the quality of the assessments); (4) two different sensor management techniques for computer networks; (5) semantic and contextual reasoning of cyber attacks; and (6) the visualization of cyber attacks.

15. SUBJECT TERMS
Information Fusion, Cyber Situation Awareness, JDL, Modeling and Simulation, Visualization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			John J. Salerno
U	U	U	UU	155	19b. TELEPHONE NUMBER (*Include area code*) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Table of Contents

List of Figures

List of Tables

1 Objective

Establishment of the National Center for Multi-Source Information Fusion (N-CMIF) Research provides our nation's defense, intelligence and homeland security communities a single point of access to world-class researchers and scientists who have distinguished themselves in this discipline. The center has focused its efforts in solving the growing problems of exploiting massive quantities of diverse, and often low-confidence, information relevant to tactical and strategic decision-making. Additionally, the Center provides a core focal point for the performance of basic and applied research that defines, designs, and facilitates the development and use of techniques for combining and understanding diverse data. Such understanding is the cornerstone of improved performance, reliability and trustworthiness of analysis and decision-making support for defense, national intelligence and other missions directly serving the security and interests of the United States.

Work performed at the Center on this contract includes:

- Task 4.1.1 - Development of Level 2 (Situation Assessment) and Level 3 (Threat Assessment) Information Fusion Technologies that will provide crucial decision-support information for command personnel, intelligence analysts and other operational personnel. Focus is being placed on the development of algorithms and techniques that are scalable and applicable to multiple domains. Practical application of this work is initially being demonstrated utilizing data obtained from computer network intrusion detection sensors and build on the existing Information Fusion Engine for Real-Time Decision-Making (INFERD)/Event Correlation for Cyber Attack Recognition System (ECCARS) architecture.
- Task 4.1.2 – This task is focused on the development of a comprehensive high level fusion framework that includes the addition of Levels 2, 3 and 4 type tools to the ECCARS architecture. In particular use the optimization skills of the N-CMIF partners to create models that aid the process refinement for Situational Awareness and Impact Assessment.
- Task 4.1.3 - In conjunction with AFRL/IF scientists, N-CMIF staff are collaborating on defining (and where possible) and implementing the transition of research-based fusion capabilities developed in Tasks 4.1.1 and 4.1.2 into operational environments.
- Task 4.1.4 - Reexamination of the overall role of the human in Information Fusion processes, to include methods for knowledge extraction and development of man-machine Interfaces to include the visualization of fused information in a command or intelligence analyst environment.
- Task 4.1.5 – Research leading to the development of Information Fusion tools that operate in and support Network Centric Interface Architectures.

- Task 4.1.6 - Fusion process architectural analyses to better understand and define optimal ways in which fusion process Levels interface and interact.
- Task 4.1.7 – Research and develop new methods for exploiting multi-source data sets in very complex environments.

The following is a list of major accomplishments in N-CMIF:

- Detailed research of level 2 and level 3 fusion ideas
- Detailed research and understanding of using information fusion to aid in cyber security decision-making
- FuSIA: Future Situation and Impact Awareness (Level 2/3 Framework)
- Virtual Terrain (Security-based computer network representation)
- Mathematical Techniques for Level 4 Fusion
- Enhancement of Semantic Network Processing System (SNePS) and integration with cyber security
- Cyber Security Visualization
- Cyber attack generation and simulation
- Deterministic and stochastic model elicitation for Nation building

These major accomplishments are described in detail below in their appropriate sections.

2 Information Fusion Research and Development (4.1.1)

The Joint Director's Laboratory (JDL) fusion model discusses a general guideline in which to design data fusion systems[1]. However, information fusion research has primarily focused on levels 0 and 1 fusion, which deal with the detection and correlation of sensor readings. As a result, decision-makers are presented with a large amount of data that they must manually interpret. In many domains, especially in cyber security, the data is overwhelming for even a large group of decision-makers and analysts. It is therefore of interest to further research levels 2 and 3 since they are aimed at automatically analyzing the data to present a situational view of what is going on. In addition, level 4 seeks to refine the processes used in levels 0-3 in an attempt to make the processes more efficient and/or accurate.

In the past few years, cyber attacks have become an increasing concern to not only the government, but also a large organization and even the average home computer user. Current cyber security research has primarily focused on levels 0 and 1, with very little research into any of the other fusion levels. As a result, one task of N-CMIF was to conduct extensive information fusion research in JDL levels 2, 3, and 4 with an application to cyber attacks. The research conducted involved a detailed look at what aspects comprise each level of data fusion and what factors need to be taken into account in order to implement an intelligent data fusion system.

Figure 1 illustrates the overall architectural vision as set forth in year 1 of N-CMIF. Throughout the course of the project, the overall vision did not change, but the components of the orange box changed slightly. The TANDI and Virtual Terrain Assisted Cyber Attack Impact Assessment (VTAC) elements (described in this document) were merged together to form a level 2/3 architecture called Future Situation and Impact Awareness (FuSIA).

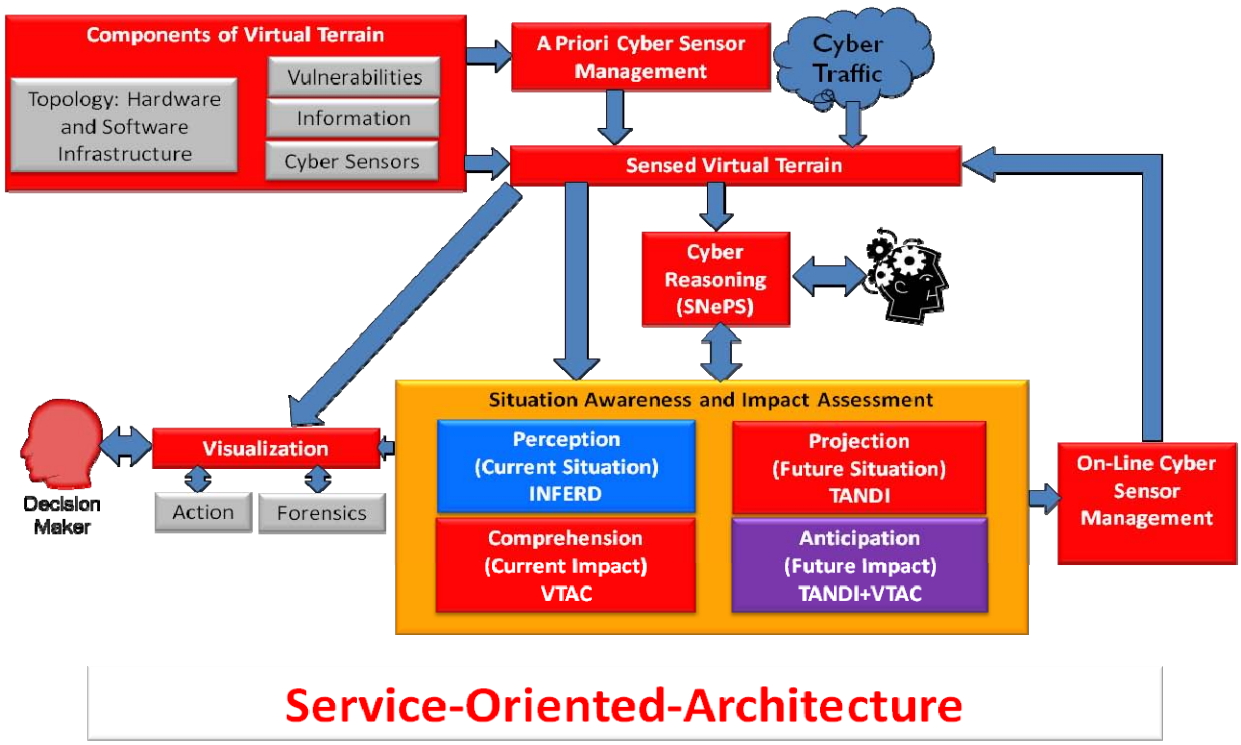


Figure 1: Overall Architectural Vision

2.1 Background and Existing Cyber Security Capabilities

Prior to N-CMIF, the CUBRC/Rochester Institute of Technology (RIT) team developed an array of prototype information fusion tools for cyber security. One of the goals of the N-CMIF project was to refine these tools as various related aspects in information fusion are researched. The following is a list of tools that were refined or overhauled during N-CMIF:

- Information Fusion Engine for Real-time Decision-making: INFERD
- Threat Assessment of Network and Data Information: TANDI
- Cyber Attack Simulator

A high-level overview of each of these tools will be provided in this section, but we encourage the reader to review the referenced publications for more explicit detail into the methodology and implementation of each tool.

Information Fusion Engine for Real-time Decision-making (INFERD) [2] is a tool developed by CUBRC and Alion Technologies under the ECCARS contract. INFERD is a JDL level 1 fusion tool that correlates related observations. While INFERD was implemented to be a general tool, its initial application was to address the problem of alert correlation in the computer security domain. INFERD uses a graphical model called a *guidance template* to drive the correlation process. While INFERD has been applied to other domains, the original use of INFERD was for the detection of multi-stage cyber attacks. The guidance template INFERD used grouped

observables into different groups and used constraints on IP addresses to correlate alerts together into an attack track. The primary observables used were alerts from intrusion detection sensors (IDS). INFERD also provided for memory management and ambiguity issues. More details on INFERD can be found in [2].

Built by RIT and CUBRC under the supervision of AFRL, Threat Assessment of Network Data and Information (TANDI) was a level 3 tool built to project cyber attacks one step into the future [5]. TANDI used a combination rule to fuse "how" (Guidance Template) and "where" (Logical Topology) the hacker can attack to determine "what" (Information Hierarchy) has been or will be attacked. Many of the concepts set forth by TANDI were refined into the various aspects of situation estimation and projection in the FuSIA level 2/3 architecture developed under N-CMIF. More information on TANDI can be found in [5] and the refinements to the TANDI concepts will be discussed in the appropriate sections.

For many reasons, companies and organizations are very reluctant to release data regarding past cyber attacks. As a result, developers of computer security tools need to primarily rely on domain knowledge with a small amount of data to properly analyze and report potentially malicious attacks. While Skaion Corporation developed a set of test data for AFRL to test alert correlation engines (such as INFERD), the number of scenarios is relatively small and is focused on only one type of network. This prompted CUBRC and RIT to develop a cyber attack simulation tool, originally developed using Arena, to create cyber attack scenarios. This simulator tool generates data in the same format as the original dataset provided by Skaion so that existing testing tools could be reused. In addition, the simulator provided flexibility to change the network configuration to see how different attacks could be executed on different networks. Like the Skaion data, this simulator focused on generating IDS alerts as the primary output instead of actual network packets. This approach presented two distinct advantages over generating network packets. First, it generates a manageable amount of data and it secondly reduces the complexity of the simulator since less data needs to be generated. In addition, the situation and impact assessment approaches already were developed by analyzing IDS alerts and system log entries, so network packet data was not necessary. Section 2.9 discusses the refinements made to the cyber attack simulator to not only improve performance, but to also implement requirements set forth by the impact and threat assessment methodologies under N-CMIF.

2.2 General Level 2/3 Considerations

A given application domain greatly influences the design of a higher level data fusion system. Before developing the data structures and algorithms, one needs to understand the requirements and existing technologies of the domain to successfully implement a higher level data fusion system. This section presents a list of questions that should be answered to identify

the requirements of a given application domain. Each of these questions is then answered with respect to the cyber domain.

2.2.1 What events are of interest (EOI) to the analyst?

An event of interest (EOI) is simply an event that affects an analyst's decision-making process. An event is an occurrence of one or more activities on one or more objects. An EOI can be detected, undetected, or inferred. Ideally, an EOI would be detectable with a relatively high accuracy, but the current sensors available may force an EOI to be inferred from other data or to simply not be detectable. An EOI that cannot be detected or inferred will reduce the overall accuracy of any high level fusion assessment that requires it. While such an EOI may spark the development of new sensing technologies, it may still need to be incorporated into the model of the domain and processed accordingly.

Currently, cyber attacks are detected by the use of intrusion detection sensors (IDSs). IDSs detect potentially malicious activities usually based on a pattern of computer network traffic. While these detected activities are still used today as a primary means of decision-making to the analysts, when considering the ultimate goal of defending against cyber attacks, these may not be the events of interest to the analysts.

When the analysts look at these events, they are concerned not with the actual activity, but rather with the *effect(s)* of the activity. For example, when an analyst sees a NETBIOS attack on an unpatched Windows host, he isn't concerned with the fact that it was a NETBIOS attack, he is concerned with whether that attack potentially compromised the host or its data. If one breaks down the example, the analyst is using domain knowledge in conjunction with the observables to determine whether the event is interesting or not. If the domain knowledge could be modeled in such a way that an algorithm could determine the high level effect, the analyst would be provided with a very fast indicator of the effect of the activity and react accordingly. Therefore, the researchers took the approach of defining the EOIs of cyber attacks to be an effect, or impact, of an activity. Example EOIs would be "the discovery of the Mail Server" or "the compromise of an employee workstation".

However, analysts would mainly be interested in high impact EOIs as indicators for events that need to be addressed. High impact EOIs would be events that result in the compromise of a host, service, mission, or data in the monitored computer network. Low impact EOIs would be reconnaissance attacks. Details for determining the impact of an EOI is discussed in Section 2.5.8 .

2.2.2 Can these events be detected or inferred?

The effects of detected activities currently cannot be detected in the cyber domain. However, the effects can be inferred based on knowledge of the computer network and its vulnerabilities.

To make these inferences possible, the researchers developed the Virtual Terrain, which is discussed in Section 2.4.3.1.

2.2.3 How frequently do the events of interest occur?

The EOIs occur very frequently in the cyber attack domain. Many companies and organizations are reporting thousands or millions of IDS alerts per day. However, high impact EOIs occur with a much lower frequency than low impact EOIs. The frequency at which high impact EOIs occur, though, can vary based on a number of parameters such as the size, exposure, and vulnerability of the network.

2.2.4 How quickly does the analyst need to react to various events of interest?

This question poses a rather heavy restriction on the processing requirements of the data fusion system. In the cyber attack domain, a high impact EOI needs to be addressed and mitigated as quickly as possible. Cyber attacks can progress at a very high rate, so every second that a security breach is not addressed can potentially cause severe consequences. While some attacks do occur over a long period of time, it is still of interest to address the breach as quickly as possible to avoid any future attacks.

2.2.5 How quickly do the events of interest need to be presented to the analyst?

Given the answer to the previous question, the events of interest should be presented within seconds to the analyst.

2.3 Prototype Impact Assessment Software/Framework: Future Situation and Impact Awareness (FuSIA)

The research conducted in this section is relevant to task 4.1.1.2 and 4.1.1.5.

One of the major deliverables in N-CMIF was the development of a software framework and architecture capable of level 2 and level 3 fusion. While some level 2 and level 3 concepts were already developed in INFERD and TANDI, there were still many gaps that needed to be filled to complete the framework. So before discussing the details of each level of fusion, the high-level view of the framework will be illustrated to see how everything ties together. Subsequent sections will then identify the details associated with each component – corresponding to both cyber attacks as well as the general considerations.

The developed framework is known as FuSIA: Future Situation and Impact Awareness. The architecture of FuSIA is illustrated in Figure 2. FuSIA is comprised of three main components that deal with estimating and assessing both the current and potential future situations. The first component corresponds to level 2 fusion and is known as Estimation. This component processes correlated observations (such as attack tracks output by INFERD) alongside a model of the environment to create an estimate of the situation. A situation is defined as a set of

activities and object states in the given environment. This situation estimate corresponds to an estimate of the *current* situation.

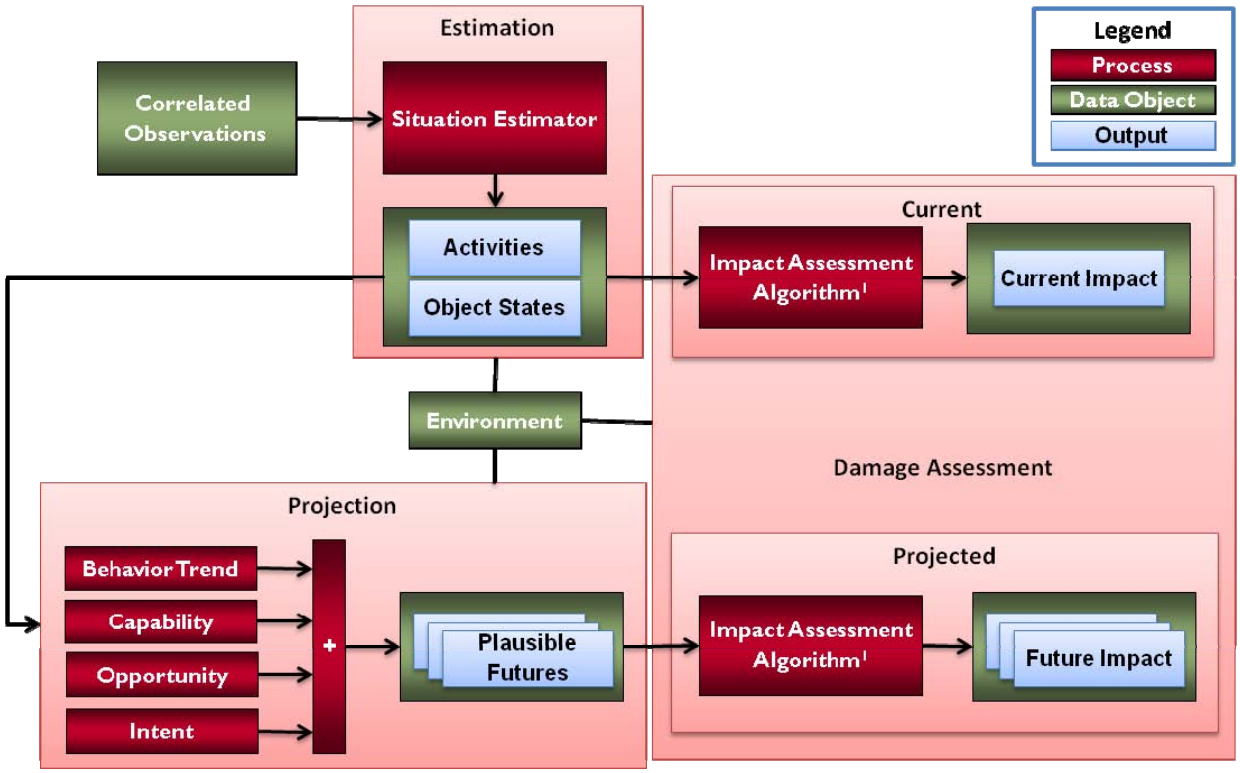


Figure 2: FuSIA Architecture

The second component is Projection and corresponds to level 3. This component generates possible future situations known as *plausible futures* by fusing multiple projection algorithms together. The final component is Damage Assessment that determines the current and projected impacts Projection and Estimation outputs. The final output of the system is an estimate of the current situation, impacts of the current situation, and impacts of all plausible futures.

Each of these components and their relevance to both general fusion and cyber attacks is detailed in subsequent sections. The following list can be used as a quick reference for the sections each component is detailed in:

- Environment – Section 2.4.3.
- Situation Estimator- Section 0.
- Activities- Section 2.4.5
- Object States- Section 2.4.4 and 2.4.6.
- Projection- Section 2.5.1.
- Behavior Trend- Section 2.5.5.

- Capability- Section 2.5.3.
- Opportunity- Section 2.5.4.
- Intent- Section 2.5.6.
- + (Fusion of projection algorithms) – Section 2.5.1.
- Plausible Futures- Section 2.5.1.
- Damage Assessment- Section 2.5.8.
- Impact Assessment Algorithm- Section 2.5.8.1 and 2.5.8.2.
- Current/Future Impact- Section 2.5.8.3.

2.3.1 Implementation Details

This sub-section describes high-level details of the FuSIA implementation. It was written to be very modular and independent of application domain. Since one goal of N-CMIF was to build upon existing cyber attack tools, the specific implementation of FuSIA is provided for that domain.

FuSIA was implemented using Java 6 and was built to be domain independent. However, as will be illustrated by this document, extensive domain and environment knowledge is required for proper situation and impact assessment. Therefore, many of the classes and interfaces provided by the FuSIA architecture need to be implemented for a given application. FuSIA was built to be modular in that compatible modules can simply be written and easily integrated into the architecture. In order to ensure compatibility each module developed for FuSIA must contain the following information:

1. Observable Type – the data type of observables the module is able to process.
2. Correlated Observation Type – the data type of correlated observations used.
3. Object Type – the data type of objects that the module provides assessments for.
4. Environment Type – the data type of the environment the module processes.

In order for two modules to be compatible, they must agree on each of the above four data types. This ensures no potentially erroneous results or runtime errors due to data inconsistencies. Further implementation details and documented code can be found on the N-CMIF deliverables CD.

2.4 Situation Modeling and Estimation (Level 2)

The research conducted in this section is relevant to tasks 4.1.1.1, 4.1.1.4, and 4.1.1.6.

Given the downfalls of TANDI discussed in Section 2.1, it was evident that more details regarding what is needed for level 2 fusion needed to be analyzed. The first task was to define the knowledge and information that comprises a situation. The following lists the elements identified that comprise a situation:

- Environment – the environment being protected. This includes not only tangible entities and relationships, but also intangible entities and relationships such as vulnerabilities and social networks.

- Objects of Interest – assets that an analyst is interested in protecting these could be tangible or intangible assets and are part of the environment.

- Activities – the activities can be executed by either the red or blue team and have some effect on the environment.

- Object states – the states objects in the environment. The states defined are specific to the domain and could vary between different types of objects.

The object states are a function of the environment, objects of interest, and activities. The details of these four elements are described in later sub-sections.

2.4.1 Situation Model Content Requirements

The next task was to define various requirements for a situation model that can feasibly be used to represent a situation in a given application domain. These requirements impose guidelines on the content present in the model.

1. The situation model must be understandable or convertible to a model easily understood by the analyst.

2. The situation model must be able to be processed or convertible to a model used by one or more (semi) automated algorithms.

3. The situation model must be detailed enough to provide the analyst or automated algorithm with sufficient detail.

4. The situation model cannot be so detailed that it makes the model too complex to populate, understand, or process.

Requirements 1 and 2 are tradeoffs with each other. An automated process could be a heuristic or based on a well-known mathematical model such as fuzzy logic or Markov models. The models used by a mathematical model might not be easily understood (in the context of the application domain) by an analyst. Likewise, a model easily understood by an analyst may not be directly applicable to a mathematical model. Therefore, the situation needs to take into account the type of processing that may be needed to populate or process over the situation. A key point is that there may not be a *single* way to model a situation, but there should be well-defined conversions between the situation representations so that inputs and outputs contain the same information.

Requirements 3 and 4 are also tradeoffs with each other and illustrate the necessity of carefully considering the granularity of detail in the situation model. A situation model will always be an estimate of the actual situation. A situation model with more details will generally be a better estimate than a model with fewer details. However, a more detailed situation model could be more difficult (or impossible) to process, populate or understand. This means that the application domain needs to be carefully analyzed to determine the information that is most relevant to the accuracy of the model. The optimal level of detail that addresses the tradeoff of feasibility and accuracy varies greatly between application domains and the expected use of the situation model for level 2 and level 3 fusion.

2.4.2 How Many Situations?

"How many situations?" may not seem like an obvious question because it may seem that that there is only a single situation that needs to be modeled. However, information fusion is only able to develop an *estimated situation* from observations and knowledge of the *true situation*. It is therefore likely that an information fusion system could determine there to be multiple estimates of the situation. To ensure clarity, the term "situation" will imply "estimated situation" unless otherwise explicitly stated.

The cyber attack domain is unique in that multiple hackers can attack independently of each other without knowing that anyone else is attacking, so it is therefore necessary to be able to model multiple situations with respect to each different attacker. This is accomplished by using a single attack track in INFERD to correspond to a single situation. The details of how a cyber situation is modeled are outlined in subsequent sections.

2.4.3 Environment

The environment contains the entities and relationships relevant for defining and processing the situation model. It is perhaps the most critical and complex element in higher level fusion since it contains the information necessary to infer what has, is, or will happen to the environment over time.

One could consider an environment to be synonymous with an ontology. However, use of the term ontology may imply that the environment should be modeled using an ontological data structure such as OWL [24]. While OWL may work well in some applications, it may be too slow to process or too complex to model in some domains. In general, the data structure used to model the environment must:

1. Be designed with consideration to the requirements of the application domain. For example, a domain such as cyber that requires fast reaction times would require a model that can be processed using various techniques in real-time.

2. Contain data and information necessary to properly model the environment.

3. Be convertible to other data structures if data structure cannot be used directly.

4. Be understandable by domain experts so that they have the ability to change the details of the model if needed.

2.4.3.1 Environment for Cyber Attacks: Virtual Terrain

While ontologies offer advanced inferencing and modeling capabilities, they can be too computationally expensive for very fast processing. However, due to the domain requirements outlined in Section 2.2, a model developed to model a computer network must be able to be used in real-time. We therefore created a graph-based model of a computer network. The advantages of using a graph-based model are as follows:

1) Graphs have well-known traversal algorithms with well-defined computational complexity.
2) Computer network architectures are generally drawn up using a graph structure, so it closely mimics an actual computer network and allows it to be understandable by the analyst.
3) Common computer network algorithms used for routing, switching, and filtering traffic can be copied provided that the communication links are modeled as arcs.

The virtual terrain is a graphical representation of a computer network containing information relevant for a security analysis of a computer network. The virtual terrain was built to mimic the structure of an actual computer network. Since the components and behavior of computer network components are very well defined and predictable, the same networking algorithms can be executed on the virtual terrain to elicit information that aids in the decision-making process of a security analyst.

Figure 3 shows an example network, while Figure 4 shows an illustration of the equivalent virtual terrain. This is a fictional, but realistic network representing how the various elements can be modeled by the virtual terrain. The virtual terrain is defined by an Extensible Markup Language (XML) [42] schema from which XML files for specific virtual terrains can be generated. XML will provide the flexibility to change the virtual terrain definition as new network technologies are introduced or more information is added. The virtual terrain data structure and parsing is currently implemented using Java 2.0.

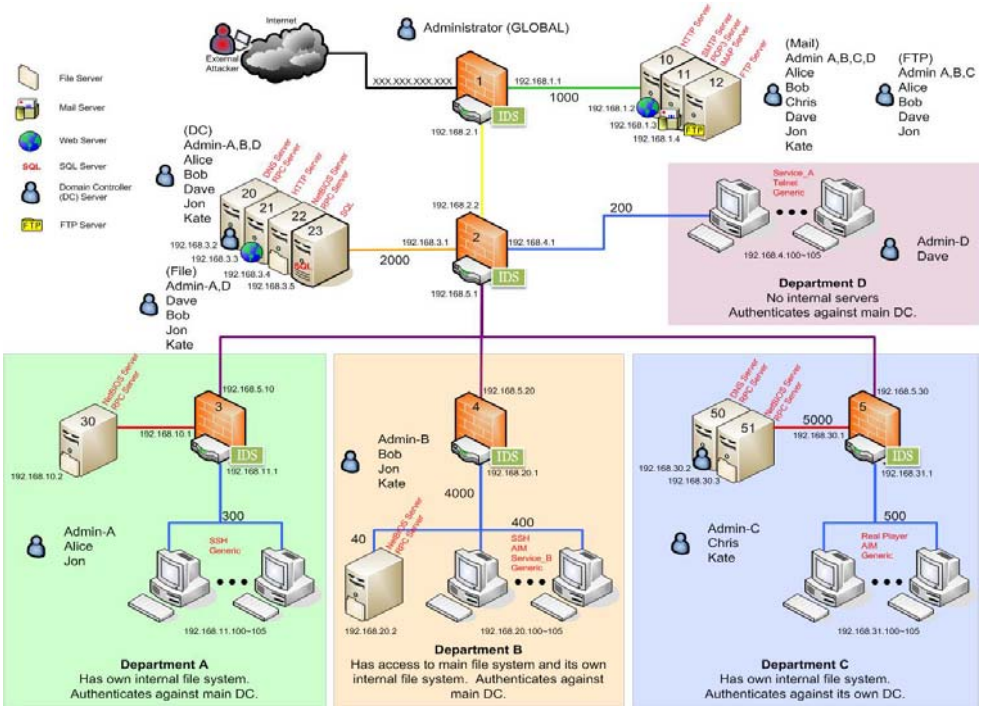


Figure 3: Illustration of an example computer network.

Due to the complexity of computer networks and the amount of detail the virtual terrain can model, it would be unrealistic to assume that the virtual terrain can be manually populated by any single person or group of people. However, there is currently no single tool available that is able to query the computer network to completely populate the virtual terrain. There are, on the other hand, network discovery tools available such as NESSUS [15] that can be utilized to aid in the population of the virtual terrain. As network security research progresses and the need for more intelligent network discovery tools increases, the authors envision the eventual ability to extract the contextual information contained in the virtual terrain automatically.

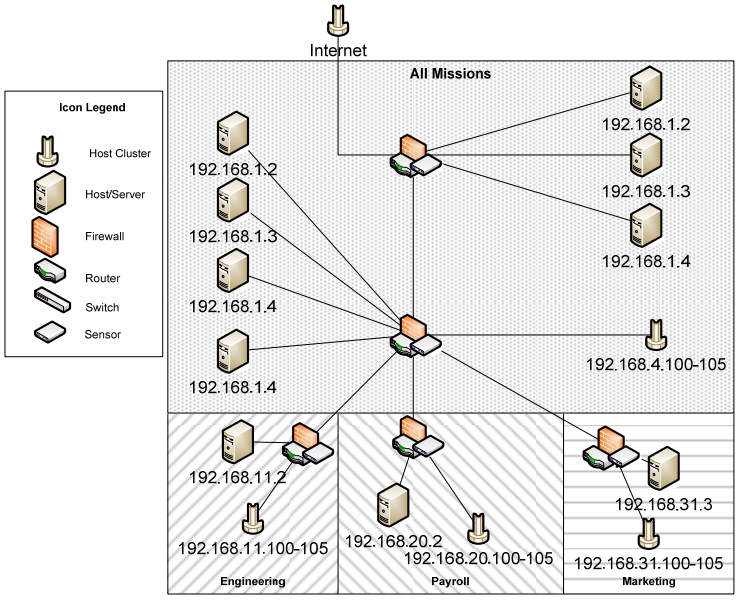


Figure 4: Diagram of virtual terrain for the computer network illustrated in Figure 3. Exposures, vulnerabilities, firewall rules and users are not shown.

The remainder of this section will discuss the various elements in the virtual terrain and their relationships to each other. Each sub-section will also provide a brief discussion on how the respective elements and relationships are populated into the virtual terrain.

2.4.3.1.1 Missions

A mission is a task or process that is carried out by a computer network. Example missions on a corporate network would be payroll, human resources, engineering, etc. A military network might contain missions pertaining to surveillance or intelligence. Regardless of what the mission actually is, one or more entities on a computer network are associated with each mission with varying criticalities. Each criticality is defined between 0 and 1. A criticality of 0 indicates that the given entity is not related to the mission, whereas a criticality of 1 indicates that the entity is absolutely essential to the correct operation of the mission. Criticalities defined between 0 and 1 indicate the importance of that entity to the given mission. The virtual terrain in Figure 4 contains three missions. The boxes surrounding the entities illustrate which missions each entity is associated with.

To illustrate the idea of criticalities, suppose a mission is executed by two subnets connected by a single router such that redundancy exists in each subnet. The redundancy implies that a single host is not critical to the execution of the mission. However, the router is highly critical to the mission since it is a single point of failure. Such a mission would be modeled by assigning a low criticality to each subnet, but a high criticality to the router.

Since computer networks are often designed to perform different tasks, the missions and associated criticalities must be manually entered by the analysts.

2.4.3.1.2 Hosts, Server, Host Clusters, and Subnets

Hosts and servers are represented by nodes and each is identified using the host name(s) or IP address(es) associated with it. A complex computer network may contain thousands of hosts, many of which are configured very similarly (such as computers in the same subnet). Therefore, the virtual terrain supports a "host cluster", which is just an aggregate node of multiple computers. This aggregate node greatly reduces the complexity of the virtual terrain graph since every host in the host cluster is assumed to have the same physical connectivity. Note that the workstations in each department are represented by a single node in the virtual terrain depicted in Figure 4. While each of these nodes only represents six hosts, the complexity of the virtual terrain would not change even each department had many more workstations (assuming each is configured similarly).

Internet IP addresses are modeled using a host cluster object. This host cluster object represents all IP addresses not present in the virtual terrain.

Hosts are stored using lookup tables so that they can efficiently be referenced when information needs to be elicited from them. In addition to the IP address(es) and host name(s), each host contains a list of services as well as one or more mission references.

Hosts can be added to the virtual terrain using Nessus and other network discovery tools. Wireless hosts can also be discovered by network discovery tools or can manually be entered. The virtual terrain can always be updated when new hosts are added or removed from the network.

2.4.3.1.3 Services, Privileges, and Exposures

Every host or server exists to perform a specific task. These tasks are accomplished by various services running on the computer. Each service has the potential to be exploited and possibly lead to the compromise of the host. When a service is compromised, the hacker usually gains privileges equivalent to the privilege of the service. Therefore, each service is defined with a given privilege. Operating systems are modeled as a service. There are also some actions that can occur on all hosts regardless of operating system or running services, so these are captured by a "General" service. Services can also be defined to be part of specific missions.

A list of exposures (and vulnerabilities) corresponding to IDS alerts is contained by each service. These exposures can be filtered further if the specific version of the service is known. Some exposures (such as "ICMP PING NMAP") are simply reconnaissance actions and cannot actually compromise the service or host. Therefore, each exposure can be defined with its own

privilege. This service/exposure mapping in the virtual terrain allows actual IDS alerts to be overlaid onto the virtual terrain in real-time to provide an estimate as to whether the attack was successful and to what extent the hacker has compromised something on the network. An example mapping of the services and exposures is illustrated in Figure 5. The "..." nodes simply represent other exposures present in the actual virtual terrain but that are not listed in the diagram. With the exception of "ICMP PING NMAP", each of the listed exposures correspond to an exploit of the service, so they are defined with a "System" privilege. The detection of one of these alerts on the mail server would indicate that the service was likely compromised by that attack. Since those exploits correspond to a system-level privilege, it can also be assumed that the server was compromised.

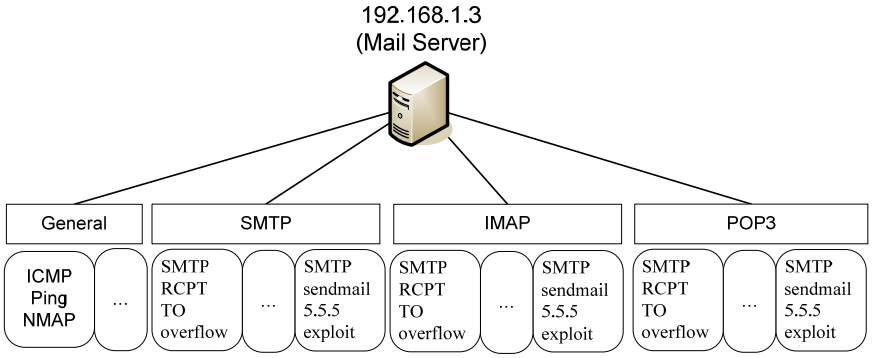


Figure 5: Example service/exposure mapping for the mail server

There are currently no tools available that can automatically map all possible exposures to a service. Nessus [15] scans do automatically provide some of these mappings, so these are populated directly into the virtual terrain. However, it does not have the ability to identify every exposure on the host. Other discovery tools, such as NMAP, also provide the ability to identify services running on the host, but do not provide any information as to which exposures are present on the host. To overcome this problem, a database containing mappings between services, default privileges, service versions, and exposures was developed. This database was populated manually using subject matter experts in addition to using information from known vulnerability databases such as CVE [9] and Nessus.

2.4.3.1.4 Routers, Switches, Sensors and Firewalls

Switches and routers determine the direction in which network traffic flows, while access lists and firewalls can be used to configure what type of traffic is permitted or restricted on the communication medium. Sensors monitor network traffic for suspicious activity and some can also be configured to also restrict network traffic. Each of these entities is modeled by a node in the graph. The firewalls and access lists are modeled by a table that defines whether the specific ports, IP addresses, or protocols are permitted. This table is also present on all hosts and servers that contain a host-based firewall. Table 1 shows an excerpt of the firewall rules

between the hosts stored at the main firewall. The listed protocols are the protocols permitted on the given link. An 'X' indicates that traffic is not permitted between the hosts.

Due to the configuration of networks, routers, switches, sensors and firewalls may be transparent to network discovery tools. Therefore, the locations of these elements are likely to be populated manually. However, the configurations of each of these elements can be automatically parsed to populate the firewall rules in the virtual terrain.

Table 1: Excerpt of firewall rules from one of the firewalls defined in the virtual terrain

FIREWALL TRAFFIC	Internet	192.168.1.1	192.168.2.1	192.168.2.2	192.168.3.1	192.168.4.1	192.168.5.1	192.168.5.10	192.168.5.20	192.168.5.30
Internet	X	OPEN	OPEN	X	X	X	X	X	X	X
192.168.1.1	SMTP Webmail Web FTP	X	POP3 IMAP Web	X	X	X	X	X	X	X
192.168.2.1	Web	POP3 IMAP Web	X	POP3 IMAP WEB	X	X	X	X	X	X

2.4.3.1.5 Physical and Wireless Links

An arc between two nodes in the virtual terrain indicates a physical connection between the two entities represented by the nodes. The virtual terrain treats corded and wireless communication mediums as "physical" links. The arcs imply that data can be sent across the medium, but it does not necessarily mean that the data will actually be received by the destination entity (due to firewall rules or access lists). Note that the virtual connectivity (i.e. whether two hosts can communicate) is defined based on the routing and firewall rules. The algorithm on whether traffic can actually flow between two links is detailed in Section 2.4.6.1.

In the virtual terrain, access points for wireless clients are modeled using a switch and the IP ranges defined for connecting to the access point should be manually defined. While Nessus can be used to determine unrestricted connectivity between computers in the same subnet or network segment, firewalls could potentially prevent the identification of physical connections. Therefore, the physical connectivity between switches, routers, and different network segments must be manually entered.

2.4.3.1.6 Users

Users are modeled into the virtual terrain since they can also be a vulnerability to the network. Weak passwords or disgruntled employees can lead to the inappropriate use of user privileges on the computer network. While individual users are not typically identified in IDS alerts, logs may contain usernames, so this information could potentially be utilized in threat and impact assessment..

Users are modeled in the virtual terrain and relationships are defined between users and the hosts they are able to log into. Users are also divided into user groups that contain the privilege levels each user has. The privilege level is important since a hacker can potentially cause more damage with an administrator account as opposed to a typical user.

2.4.3.2 Virtual Terrain Comparison

Now that the virtual terrain and its uses have been detailed, it is logical to compare the virtual terrain to other existing computer security models. Two models, attack graphs and vulnerability trees, have been proposed for use in computer security research. These two models and the virtual terrain suffer from the fact that incomplete information in defining the model can lead to an inaccurate analysis. Due to the detailed require for each model, it may actually be very difficult to completely populate these models using today's tools given the complexity of large computer networks. However, network discovery tools are constantly becoming more intelligent and provide the users with more information, so it is not unrealistic to assume that such tools could be utilized in the future to capture the information needed to completely populate each model. This comparison will assume that all models are completely and accurately populated.

2.4.3.2.1 Virtual Terrain vs. Attack Graphs

Attack graphs are used to determine if designated goal states can be reached by attackers attempting to penetrate computer networks from initial starting states [8]. Philips and Swiler [6] proposed the use of attack graphs for vulnerability assessment, while Jajodia [7] used attack graphs to correlate IDS alerts into individual attacks and provide a threat assessment for the network. A comprehensive review of attack graphs was conducted by Lippman and Ingols [8] in 2005.

While specific implementations vary, attack graphs are generally acyclic graphs that represent possible steps in a given attack. Most implementations only model a single goal per graph, but there are some, such as Ning's [10], that can model multiple goals in a single graph. Nodes in the graph represent the individual steps in the attack, while the arcs represent the possible transitions between the steps.

Lippman and Ingols [8] discussed problems associated with all attack graph implementations. One problem is the actual generation of the attack graph models. Many approaches require manual entry of the attack details. Since these graphs are acyclic, attack graphs cannot model bi-directional communication between hosts in a single graph, so multiple graphs must be created to model this communication. Likewise, multiple graphs must be created for multiple goals if the attack graph only supports a single goal. In addition, experts must be used to generate the model since the step transitions require knowledge of exactly how attack steps could occur. However, Jajodia[6] proposed the use of Nessus to populate the attack graphs and Phillips proposed to use attack templates to aid in the generation of these graphs. While some elements of the virtual terrain need to be populated manually, only knowledge of the network is required to populate the model since it relies on external algorithms to elicit information for an analysis of attacks. Also, because the virtual terrain does not have an acyclic restriction, all transitions can be modeled using a single virtual terrain. Another downfall of attack graphs is that they assume static communication between two nodes. Due to firewalls and routers, this may not be a valid assumption since different types of traffic might be permitted or denied between two hosts. The virtual terrain models the firewall rules and router access lists explicitly, so it contains the information necessary to determine if traffic was filtered or not.

Despite these issues with attack graphs, they can be a very powerful analysis tool when properly populated. Phillips and Swiler [6] used the attack graph as a Bayesian network to identify the most likely sequence of actions that could take place. Jajodia [7] and Ning [10] were able to use them to hypothesize about missing or unobserved actions. These analyses are possible and NP-complete due to the acyclic nature of the attack graph. However, such analyses may be difficult in the virtual terrain since it does not have an acyclic restriction on the graph.

2.4.3.2.2 Virtual Terrain vs. Vulnerability Trees

A vulnerability tree is a special attack graph that is modeled as a tree. The root of this tree is the final goal of the attack, the leaves represent logical starting points, and the other nodes represent various steps of the attack. A traversal of the tree from a leaf node to the root would represent one possible sequence of attacks. Each node can also contain a complexity that corresponds to how difficult the attack is to execute. Vidalis and Jones [17] proposed the use of vulnerability trees for vulnerability assessment, while Schneier [12] focused his vulnerability tree research on computer security.

While vulnerability trees were initially developed for vulnerability assessment, the vulnerability tree seems to lend itself very well to TIA. If the tree was extended to include the observables associated with each step (i.e. the IDS alerts), incoming alerts could be mapped directly to the nodes in the vulnerability tree.

Like attack graphs and despite the simple structure vulnerability trees utilize to model potential attacks, they suffer from scalability issues. Since one vulnerability tree contains the possible sequences to accomplish one single goal, multiple vulnerability trees need to be defined for multiple goals. Such an approach could make vulnerability tree definition very difficult for a large network, which would likely require the definition of many large and complex vulnerability trees. The virtual terrain, on the other hand, is able to implicitly model all known attacks and goals into one model. Scalability of the virtual terrain is restricted to the scalability of the algorithms used to analyze the virtual terrain. Like the attack graph, communication between hosts is assumed to be absolute and do not take into account firewall rules, which may actually inhibit certain steps from ever taking place. In the same light, a new type of attack or network reconfiguration could potentially cause the structure of the vulnerability tree to drastically change. The virtual terrain, though, models firewall rules and changes to the network or attack types just require small changes to the virtual terrain model since attack steps are not modeled explicitly by the virtual terrain.

Vulnerability trees give the analyst immediate feedback regarding the hacker's potential goals as well as the number of steps he is away from accomplishing the goal. However, the virtual terrain does not explicitly define goals, so the goals need to be logically implied by the analyst or by some pre-defined algorithm. By looking at the parent node of an alert mapped to the vulnerability tree, the analyst can easily identify a next plausible action by the hacker and act accordingly. Such an analysis can be performed on the virtual terrain by using the algorithm described in Section 2.5.5. That analysis is limited to the assumption that the hacker will not execute new attacks. Finally, vulnerability trees allow the analyst to quickly identify any steps in the attack that may have been skipped or unobserved.

Attack graphs and vulnerability trees provide simple algorithms in which to analyze potential attacks. However, both models suffer from scalability problems since they can become difficult to model for large networks. On the other hand, the virtual terrain is able to model large networks, but requires more complex analysis algorithms. A possible extension of this comparison is to determine if attack graphs or vulnerability trees can be automatically and dynamically generated from the virtual terrain. Dynamic creations of these data structures could potentially reduce the scalability and data population problems of attack graphs and vulnerability trees.

2.4.4 Objects of Interest

The objects of interest (OOI's) are the objects in the environment that are relevant to the estimation of threat and impact. OOI's can be either tangible or intangible and are defined as an element in the environment. As part of a situation, an OOI can be affected by various activities (see Section 2.4.5) and have a certain state (see Section 2.4.6).

2.4.4.1 Objects of Interest for Cyber Attacks

When an analyst is trying to determine the consequences of an attack, he is looking at the impact to missions, hosts, and services on the computer network. Therefore, the missions, hosts and services (all present in the virtual terrain) are considered to be the OOI's for a computer network.

2.4.5 Activities

Activities are any event that is executed within the environment. Activities can be executed by both blue and red teams. Activities can be related to each other both causally and temporally. Causal relationships are a cause/effect relationship, whereas temporal relationships are only determined by the time in which the activities occur. It should be noted that activities are not limited to the detectable activities. It may be possible to infer that activities occurred. Likewise, it may be possible that a given activity was not detected or known to happen. The notion of detectable and undetectable activities can greatly influence the design of the data fusion system so that it could, if possible, be robust to undetectable activities.

2.4.5.1 Activities for Cyber Attacks

Activities in the cyber attack domain are typically detected by IDSs and are reported in real-time to the analyst. Each alert generated by an IDS alert contains several fields of information that can be used to infer the impact of the event:

- Signature
- Source IP/Port
- Destination IP/Port
- Timestamp
- Protocol

Since many of the IDSs are signature-based, it is possible that a relevant activity could go undetected. This influences the design of a cyber security data fusion system because it is possible that there is missing information. The IDS sensor primarily used in this research was Snort [13].

2.4.6 Object States

Object states correspond to the current state(s), or the current condition, of an OOI. While object states must be defined specific to the domain and also the object itself, the needs of the analyst as well as any reasoning algorithms should drive what the states are. Examples of states are operable, inoperable, damaged, normal, etc. In order to permit an OOI to be in multiple states at once, we decided that there can be multiple classes of states. Within a class, object states are mutually exclusive of each other, which implies that an OOI can only be in one

state at a given time within a class of states. In addition, each class contains an "Unknown" state that implies that the state cannot be determined given the observable data.

2.4.6.1 Defining and Determining Object States for Cyber Attacks

For cyber attacks, we defined a single class of states that correspond to states of interest to the analyst. The following lists the different states we defined for cyber attacks:

- Normal – there are no indicators or implications made that indicate anything other than normal operations
- Attacked – the OOI was targeted by a malicious activity, but the activity was unsuccessful or had no effect on the object.
- Discovered – a reconnaissance-type attack was targeted at the ooi, but no sensitive information was revealed to the attacker.
- Partially Compromised – the ooi was successfully targeted by a malicious activity and partial control of the OOI is given to the attacker or some sensitive information may have been revealed to the attacker.
- Compromised – the OOI is in complete control of the attacker or a very large amount of sensitive information was revealed to the attacker.

Now that the states were defined, we needed to come up with a way to automatically determine each of the states within a given certainty. While the steps of this algorithm were originally developed using domain expertise, the high-level steps in the algorithm were validated after reading a whitepaper released by ArcSight, which is one of the COTS tools available for cyber security [25]. According to ArcSight, analysts traditionally look at network traffic dumps, firewall logs, and the actual computer, to determine the success of an attack. While automated techniques could be developed to perform these tasks, the processing occurs over a potentially large amount of data. Recall, however, that information needs to be processed very fast in the cyber domain – to where even a few seconds may be too slow to process a *single* attack since many networks receive thousands or millions of alerts per day. The virtual terrain, however, could be used to quickly estimate this data since the routing, firewall, and vulnerabilities are already captured.

Recall that for cyber attacks, an activity contains an alert signature, source and destination IPs, protocols, and source and destination ports. These are the main fields used to determine the state of a given object. It should be noted that the automatic algorithm listed below solely uses the activity and virtual terrain to determine the routing path, firewall filtering, and vulnerability information relevant to the activity.

The steps of the algorithm are as follows:

1. Determine if the attack actually reached the target
    a. Using the virtual terrain, determine the path between the source and destination IPs. IP addresses that are not explicitly defined in the virtual terrain are assumed to be from the Internet. Most routers use Open-Shortest Path First (OSPF) algorithms to determine the path of the network traffic, so this algorithm is mimicked using the virtual terrain to get the best estimate the packet traveled. It should be noted that OSPF may not always be used, so depending on the network setup, it may make sense to implement a different algorithm to determine the path. This step is equivalent to an analyst looking at dumps of network traffic to determine where the packets were routed.
    b. If a path cannot be found between the source and destination, it is assumed that the packet was dropped and the activity was unsuccessful. The state of the OOI is set to "Attacked". This would be the equivalent of a router dropping a packet.
    c. If a path is found, each node is analyzed for any firewall rules that may have filtered the traffic. Using the virtual terrain, this process is very similar to the way firewalls actually work. It simply steps through the list of firewall rules and compares the IP, protocol, and port information with the firewall rules. If firewall rules are not present on the virtual terrain (whether they are simply not included or not known), then the algorithm assumes that no filtering occurred and assumed that the given firewall did not filter the traffic. It is therefore paramount to keep the firewall rules updated on the virtual terrain to ensure the most accurate assessment.
    d. If any firewall rules are violated in the defined path, then it is assumed that the network traffic was filtered out and the OOI state is set to "Attacked".
    e. If none of the firewall rules were violated, then the attack is assumed to have reached the target.
2. If the attack reached the target, determine if the target is vulnerable to the attack
    a. Using the virtual terrain, try to match the alert signature with any vulnerability on the target. The current implementation performs a case-insensitive string comparison between the name of the exposure and the signature of the alert. Again, it is paramount to ensure that this mapping process provides an accurate representation of the vulnerabilities. The current implementation makes the assumption that if a vulnerability/exposure is not present on the host, that the host is not vulnerable to the attack. It is very possible to override this functionality and create a different mapping functionality.

b. If the signature is mapped to a vulnerability on the virtual terrain, then it is assumed the target is vulnerable to the attack. Otherwise, the OOI is not vulnerable to the attack and assigned a state of "Attacked".

3. If the target is vulnerable to the attack, determine the effects of the attack.
    a. Since the alert itself may not include the anticipated effects of the attack, the virtual terrain is used to determine the effect of the attack. Recall that each exposure is assigned a given privilege level corresponding to the privilege level obtained by the hacker should that vulnerability be executed on the host. This privilege level is used to determine the state of the OOI.
    b. If the privilege level is "None", then the exposure corresponds to a reconnaissance attack and the state of the OOI is set to "Discovered".
    c. If the privilege level is "User", then the exposure grants user-level access to the hacker on the host, but does not give the hacker complete control of the system. Therefore, the state of the OOI is set to "Partially Compromised".
    d. Finally, if the privilege level is "System", then the exposure grants system-level access to the hacker, which means that the hacker can have complete control of the system. This sets the OOI state to "Compromised".

By default, all OOI's begin with a state of "Normal" and the state is only changed if they are affected by a given activity.

The above steps can be executed by every alert processed. For a given attack track (grouping of alerts), the final state of an object is the worst state assigned to an object after all alerts in that attack track were processed. It should also be noted that the developed code also provides for an uncertainty value to be assigned to the state. However, uncertainty is not calculated with the current implementation.

2.5 Level 3 (Threat Assessment)

The research conducted in this section is relevant to tasks 4.1.1.1, 4.1.1.4, and 4.1.1.6. Discussions on scalability and applicability to other domains are included for relevance to tasks 4.1.1.1.1 and 4.1.1.1.2.

Level 3 (Threat Assessment) is comprised of two separate tasks. The first task is projection. This task attempts to generate plausible future situations for an analyst to be aware of. The second task is impact assessment, which determines the negative on different OOI's in the environment based on the known and implied observables. This impact assessment is performed not only for the current estimated situation, but also for the possible future situations. The interpretation of results from these tasks is discussed in Section 2.5.8.3.

2.5.1 Projection

In addition to knowing what the current situation, it is beneficial to try to project the current situation into the future. A key point is that the term "projection" is not to be confused with "prediction". Prediction identifies guess as to what *will* happen in the future. Projection merely identifies what *can* happen in the future. This distinction is important because the exact future is not always known due to random chance or missing knowledge. Projection allows the analyst to look at possible future situations, heretofore referred to as *plausible futures*, and be prepared for multiple outcomes.

The notion of using plausible futures introduces another tradeoff in developing a methodology in which to project situations into the future. Identifying a large number of plausible futures could overload the analyst with too much information. Likewise, not identifying a plausible future could lead to erroneous decision-making. At the same time a plausible future may be unlikely to occur, but it could also present a large threat to the environment (see Section 2.5.8 for more information regarding threat and impact assessment and a discussion of this problem). To further increase complexity, one could infinitely project a plausible future to create plausible futures of plausible futures. For simplicity, we decided to only look at plausible futures with respect to the next event.

It is therefore of interest to determine how to model a plausible future. A plausible future is simply a projection of a situation into the future. So a plausible future only needs to be defined by the differences between the plausible future and the current situation. We model a plausible future by assigning a plausibility score and uncertainty to the object(s) of interest that changed state. Section 2.5.2 discusses in more detail what these scores represent.

Now that it is known how to model a plausible future, it is now necessary to actually determine a plausible future. There are possible ways to address this problem:

1) Create a single comprehensive algorithm that tries to project the situation using domain knowledge.
2) Create multiple algorithms that do their own projection and fuse the results together.

The first option is ideal, but may be unrealistic to implement. There is perhaps too much complexity or too much missing information for a comprehensive algorithm to be implemented with a reasonable degree of accuracy. The second approach is the focus of the remainder of this section.

By using the logical topology and guidance template, TANDI sought to fuse "where" and "how" to determine "what" was compromised [5]. This separation helped to reduce the complexity of the analysis. However, the weighted average method to fuse the assessments together

provided contradictory results in some instances. This contradiction motivated a rework of how projection results were fused together.

Each projection algorithm calculates or assigns plausibility scores to each entity of interest over the environment. Because multiple plausibility scores are assigned to a given object, there must be a method in which to fuse plausibility scores together. The methods for combining plausibility scores evolved as our understanding of impact assessment and situation projection increase.

There is quite a bit of literature outlining that capability, opportunity and intent are required for a threat to be present [26]. However, combining these three aspects using a single algorithm may be too complex to implement or even understand. For this project, we decided to develop and implement a "Capability" and an "Opportunity" algorithm to demonstrate how the fusion of multiple projection algorithms would occur. "Intent" was not considered because we determined it to be too difficult, if not impossible, to determine the intent since the actual source of the attack is not very reliable or well-known. For the sake of this level 3 architecture, we assume that the intent of an attack will always be malicious, which considers the worst case scenario.

There are several advantages to using multiple projection algorithms versus a single algorithm. First, multiple algorithms reduces the complexity of a single algorithm since various aspects can be ignored. Also, since the projection of situations into the future is likely to be an unknown process, there is likely no single known way to perform this projection. Therefore, each algorithm can be interpreted as a single expert's assessment. The fusion of the algorithms will help to increase the plausibility score for certain plausible futures and perhaps decrease the plausibility score for other plausible futures that may contain conflicting information. The method of combining multiple expert opinions is very similar to the Dempster-Shafer combination process, which is outlined in the next section.

The remainder of the section will outline the various Dempster-Shafer combination rules tested. It will also outline the implementations of the capability and opportunity projection algorithms used.

2.5.1.1 Use of Dempster-Shafer Theory for Projection Algorithm Fusion

The details of Dempster-Shafer Theory (DST) can be found in [27]. However, while there is common terminology for DST, the notation is often varied between different publications. To ensure clarity, the following notation will be used for DST:

Frame of discernment: θ

Power set of frame of discernment: 2^{θ}

Mass function: $\sum_{A\subseteq\theta} m(A) = 1$

Belief function: $Bel(H) = \frac{1}{1-m(\emptyset)} \sum_{X\subseteq H} m(X)$

Basic DST combination rule: $m_1 \otimes m_2(A) = \sum_{X,Y\subseteq\theta, X\cap Y=A} m_1(X)m_2(Y)$

2.5.1.2 Plausibility Score Fusion Method #1: Using just plausibilities

The first theory used to combine plausibility scores was the classical Dempster-Shafer combination rule using only plausibility scores. A plausibility score, p, was defined as a member of {unknown, [0,1]} to represent how plausible something was to happen. This allowed us to define the Dempster-Shafer frame of discernment in the following manner:

$$\theta_p = \{P, N\} \tag{1}$$

Where P represents the plausibility the event will happen, and N represents the plausibility that the event will *not* happen. Since these are mutually exclusive, it does not make sense to define the power set 2^θ.

Mass functions are undefined for p = unknown, but were defined as follows otherwise:

$$m(A) = \begin{cases} p & for\ A = \{P\} \\ 1-p & for\ A = \{N\} \\ 0 & otherwise \end{cases} \tag{2}$$

The fused plausibility is simply then calculated by applying the basic combination rule and calculating Bel(P).

A limitation of this approach is that it cannot weight algorithms against each other. In other words, it does not have the ability weight a more reliable assessment higher than a less-reliable assessment. Also, as will be illustrated at the end of the section, since an unknown threat score does not have a defined mass function, it is ignored by the Dempster-Shafer combination rule. This ignorance can lead to a possible inconsistency of results.

2.5.1.3 Plausibility Score Fusion Method #2: Adding in Reliabilities

To overcome the weighting limitation of the first theory, a reliability, $r \in [0,1]$, was calculated or assigned to each plausibility score. This provided us with ability to weight more reliable assessments higher than others. The reliability of an assessment can be affected by different factors:

1. Missing or ambiguous sensor readings
2. Uncertainty in sensor readings
3. Missing, incorrect, or out-dated environment information

4. Uncertainty in environment
5. Simplifying assumptions for complex models
6. Poor knowledge of how to model projection algorithms (i.e. lack of known probability distributions)

2.5.1.4 Plausibility Score Fusion Method #2-A: Discounted Combination Rule (DCR)

This method uses the discounted combination rule proposed by Shafer in 1976. This method discounts the beliefs of each mass function and then combines the *beliefs* by averaging them. For a given mass function m_i, the discounted belief is represented as:

$$Bel_i^{\alpha_i}(A) = (1 - \alpha_i)Bel_1(A) \tag{3}$$

where:

$$\alpha_i = 1 - r_i \tag{4}$$

The combination rule is different from the classical combination rule in that it combines beliefs instead of mass functions. The combination rule is simply just the average of the discounted belief functions. It can therefore be used in the following manner to determine the fused plausibility:

$$\overline{Bel}(P) = \frac{1}{n}\left(Bel_1^{\alpha_1}(P) + \cdots + Bel_n^{\alpha_n}(P)\right) \tag{5}$$

A limitation of this approach, however, is that there is no method to calculate the reliability of the fused beliefs.

2.5.1.5 Plausibility Score Fusion Method #2-B: Single Mass Method with Reliability (SMMR)

This was the researcher's first attempt to model reliability into the Dempster-shafer fusion such that the plausibility and reliability could be extracted from the fused result. The first step is to extend the frame of discernment to include reliability (originally proposed by Haenni):

$$\theta_r = \{R, U\} \tag{6}$$

$$\theta_e = \theta_p \times \theta_r = \{P, N\} \times \{R, U\} = \{PR, PU, NR, NU\} \tag{7}$$

Where R represents "reliable" and U represents "unreliable". Each plausibility score is then modeled by the following mass function:

$$m_i(A) = \begin{cases} p_i r_i & for\ A = \{PR\} \\ p_i(1 - r_1) & for\ A = \{PU\} \\ (1 - p_i)r_i & for\ A = \{NR\} \\ (1 - p_i)(1 - r_i) & for\ A = \{NU\} \end{cases} \quad (8)$$

The fused mass function can then be calculated using the basic combination rule:

$$m_{fused} = m_1 \otimes m_2(A) \quad (9)$$

Using m_{fused}, the plausibility can be calculated using Bel({PR,NU}) and the reliability can be calculated using Bel({PR,NR}).

2.5.1.6 Plausibility Score Fusion Method #2-C: Two Mass Method with Reliability (TMMR)

Like the above approach, it involves including the reliability into the frame of discernment and using the basic combination rule to calculate the results. However, each assessment is modeled by two mass functions (one to model the plausibility, and the other to model the reliability:

$$m_i^1(A) = \begin{cases} p_i & for\ A = \{PR, NU\} \\ 1 - p_i & for\ A = \{PU, NR\} \\ 0 & otherwise \end{cases} \quad (10)$$

$$m_i^2(A) = \begin{cases} r_i & for\ A = \{PR, NR\} \\ 1 - r_i & for\ A = \{PU, \{NU\} \\ 0 & otherwise \end{cases} \quad (11)$$

This changes the combination rule to the following:

$$m_{fused} = m_1 \otimes m_2(A) = m_1^1 \otimes m_1^2 \otimes m_2^1 \otimes m_2^2(A) \quad (12)$$

Using m_{fused}, the plausibility can be calculated using Bel({PR,NU}) and the reliability can be calculated using Bel({PR,NR}).

2.5.1.7 Plausibility Score Fusion Method #2-D: Marginalized Two Mass Method with Reliability (MTMMR)

This is an extension to the method proposed by Haenni [28]. For each plausibility score, two mass functions are created identically to the TMMR method. These two mass functions are then fused using the basic combination rule to create a single mass function:

$$m_1(A) = m_1^1 \otimes m_1^2 \quad (13)$$

This mass function, however, is defined with respect to θ_e. Mass functions can be "marginalized" in order to map a given mass function to a different frame of discernment. This mass function can then be marginalized to remove the "irrelevant" part to the analysis. If the mass function is marginalized to θ_p, then the plausibility can be determined. If the mass function is marginalized to θ_r then the reliability can be determined. We use the $\downarrow \theta_j$ symbol to indicate that the mass function was marginalized to the frame of discernment θ_j. Therefore, the marginalized mass function $m_i^{\downarrow\theta_p}(A)$ "removes" the reliability and puts the mass function in reference to θ_p. Therefore, the plausibility can be calculated using Bel(P). Likewise, for the marginalized mass function $m_i^{\downarrow\theta_r}(A)$ the reliability can be calculated using Bel(R).

2.5.2 Modeling Plausibility and Reliability

2.5.2.1 Plausibility

We originally defined plausibility on [0,1]. A very low score indicated that there was a low plausibility of the event happening. A very high score indicated a very high plausibility of the event happening. Before integrating reliability into the assessments, an unknown plausibility was simply ignored by the Dempster-Shafer fusion. However, once reliability was incorporated, an "unknown" score could be interpreted as a "score with a very low reliability". The term "very low" is used because a reliability of zero would cause the basic combination rule to always return a 0. However, this method caused inaccurate results because there was no clear notion of what number should be for an unknown plausibility.

To overcome this problem, we decided to keep the plausibility defined on [0,1]. However, we changed the interpretation of the values. A plausibility score defined on (0,0.5) indicates the plausibility that the event will *not* happen. A value closer to 0 indicates a high plausibility that the event will not happen. A plausibility score defined on (0.5,1) indicates the plausibility that the event will happen. A value closer to 1 indicates a high plausibility that the event will happen. A plausibility score of 0.5 indicates an indifference to whether the event will happen or not and is used to represent an "unknown" plausibility score.

2.5.2.2 Reliability

Reliability is defined on [0,1] where 0 indicates a low reliability and 1 indicates a high reliability.

2.5.3 Projection Using Capability

The basic notion of the capability algorithm was to analyze the demonstrated capabilities of the hacker to determine which OOIs are vulnerable. There were two versions of the capability algorithm that were developed.

The first implementation was a basic rules-based assignment algorithm. Utilizing *only* the activities in a given situation, this method suffered from the fact that it required a long list of

activities corresponding to a single situation in order to assess the capability of the hacker. Such a limitation was not very helpful in cyber security since ideally the number of activities would be minimal in a situation. This essentially meant that the capability algorithm would assign very few (if any) plausibility scores for an attack. Also, the capability algorithm would be worthless for scenarios that exhibited no repetition. This required a complete re-work of the algorithm in order to provide a better projection.

The second capability algorithm performs a statistical correlation based on conditional probability within a track on which Version is being 'tampered' with (changing states). A history is stored that records per track which versions have been 'tampered'.

The score assigned to a version V on a track with V_1, V_2, V_3,… that have been tampered is evaluated as the conditional probability of $P(V|V_1 \cup V_2 \cup V_3, \dots)$.

The reliability assigned to a version V on a track is calculated as follows:

$$r = (1 - 2 \cdot e) \cdot L \tag{14}$$

$$e = Z\sqrt{\frac{pq}{n}} \tag{15}$$

L = Confidence Level (95%)
Z = Normal Z-Score for a Confidence Level L
p = proportion
q = 1-p
n = sample size

Updating history

```
if V_A is a newly tampered version on track T then
   T_history = track T version history
   add to T_history the Unordered Pair (V_A, State of V_A)
end if
```

Calculating Version Score

Let T be the tracks and let $T_{current}$ be the current track being calculated for, let V be versions, let S be a state change of a version

Note: sets cannot have duplicate elements

```
for A = all alerts in T_current
   V_A = target Version of A
   S_A = state change of V_A due to alert A
```

```
    add to V_set pair (V_A, S_A)
end for
for (V_A, S_A) = all elements of V_set
    T = all tracks that have (V_A, S_A)
    Add to T_set all elements of T
end for
for V = all Versions in the Virtual Terrain
    if V has been tampered in T_current then
        score of V = 1.0
        reliability of V = 1.0
    else
        supporting_tracks = 0
        for T = all elements of T_set
            if V exists in a pair in T then
                supporting_tracks ++
            end if
        end for
        score of V = supporting_tracks / size(T_set)
        reliability of V = (1 – confidence_interval) * confidence_level
    end if
end for
```

2.5.3.1 *Benefit of Statistical Capability Algorithm*

Using 10 scenarios generated by the cyber attack simulator, the original (old) and final (new) capability algorithm were tested against each other. Figure 6 shows a comparison of the two capability algorithms. The "% Projected" method captures the percentage of objects that had a plausibility score assigned to it before it was acted upon by a future activity. It illustrates that the old capability algorithm only provided a projection, on average, less than 10% of the time. The new capability algorithm provided a substantial increase in the number of projected objects.

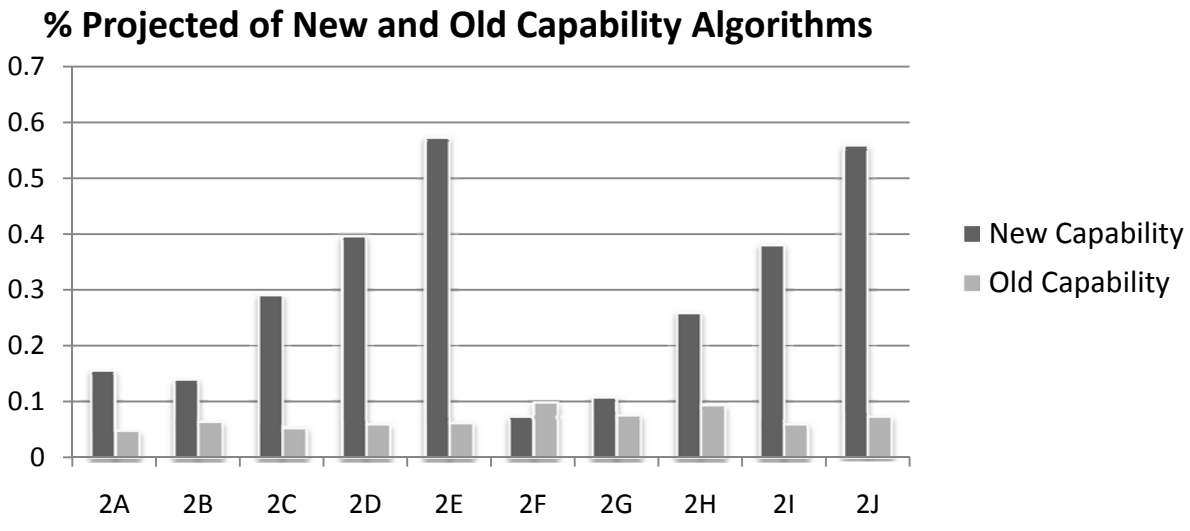


Figure 6: Comparison of the Two Capability Algorithms

It should be noted that since a dynamic statistical approach is now used, the capability algorithm must first "ramp up" in order to become more accurate at projections.

2.5.4 Projection Using Opportunity

For each iteration (new compromised host on a track), the Opportunity Algorithm consolidates hosts that have been compromised into a single node and performs a breadth first search starting at the newly compromised host. The breadth first search goes through all connected network components (routers and hosts) to aggregates the firewall rules of routes in the network to a single set of exposed rules per host. Each set of exposed rules defines the set of open path from a compromised host to a non compromised host. The result of the algorithm is a set of hosts that are reachable by their defined exposed rules from the set of compromised hosts.

For a given track, let H be hosts, Let C be the consolidated compromised hosts, let R be routers, let F be firewall rules, let N be Network Components

```
If H_A is a newly compromised host then
   N_HA = neighbor network components of H_A
   for all N_HA
      exposed_rules of N_HA = (exposed_rules of N_HA) ∪ (firewall_rules of H_A to N_HA)
      parent of N_HA = H_A
   end for
   push neighbor network components of H_A onto search_list
   for all elements in search_list
      N = removed last element of search_list
      if N has not been visited then
         if N is a Router then
            N_N = neighbors of N
            for all N_N
               exposed_rules of N_H = (exposed_rules of N_N) ∪
                                      ((exposued_rules of N) ∩
                                        (firewall_rules of parent of N to N_N))
               parent of N_N = N
            end for
            push all N_N onto search_list
         else if N is a Host then
            add N to reachable_host_set
         end if
      end if
   end for
end if
```

2.5.4.1 Opportunity Algorithm Optimizations

While the assignment strategy for the opportunity algorithm was largely unchanged, the opportunity algorithm was originally very inefficient in that it employed a depth first search for all target/host pairs. This caused a large number of redundant calculations. The algorithm was later revised to perform a breadth first search from the compromised hosts. Table 2 shows the

speedup in time complexity with respect to the number of compromised hosts (V) and the number of possible targets (E).

The opportunity algorithm was executed using ten test scenarios simulated using the Cyber Attack Simulator. Table 3 illustrates an average of 2.51 speedup over the original opportunity algorithm, with only scenario 2A exhibiting a speedup below 2 (1.87). It should be noted that the optimizations to the opportunity algorithm only affected the processing time and not the plausibility scores or reliabilities.

Table 2: Complexity of Opportunity Algorithm

	New Opportunity Algorithm	**Original Opportunity Algorithm**
Route Search	Breadth First Search	Depth First Search for all (target, host) pairs
Time Complexity	O(|V|+|E|)	$O(|V|^2 + |E|^2)$
Memory Complexity	O(|V|+|E|)	O(|V|+|E|)

Table 3: Speedup of Opportunity Algorithm after Optimizations

Scenario	2A	2B	2C	2D	2E	2F	2G	2H	2I	2J	Average
Speedup	1.87	2.01	2.29	2.63	3.53	2.05	2.06	2.36	2.82	3.48	2.51

2.5.5 Projection Using Behavior

The Behavior algorithm utilizes Variable Length Markov Models (VLMM) to capture the sequential behaviors in each attack track. For each ongoing attack track, the suffix of attack events will be used to compare to the VLMM and predict by combining the various order Markov models.

Each attack sequence consists of one or more events (e.g., cyber alerts). Multiple attributes may be used to describe each event. In the cyber domain, these attributes include attack description, attack category, target IP, network protocol, etc. Each of these attributes can be used to form a sequence of symbols from the sequence of events, i.e., attack track. The sequences of symbols based on different attributes are the basis to form a suffix tree, and the statistics stored in the suffix tree then is used to build the VLMM for behavior prediction.

Let $s = \{x_1, x_2, ..., x_m\}$ be an unfolding sequence of attack actions. Our objective is to predict the next action (x_{m+1}) given s and given a data-set containing representative attack tracks. Let $P_n(x)$ denote the probability of the next symbol in s being x according to an n_{th} order Markov model. Note that n can be at most m, which is the length of the already observed context. For a given set of representative attack sequences, the entire context of length m may not match any suffix

in the tree; that is, this newly observed sequence s is not part of the data-set. Let H be a set containing all suffixes of representative attack tracks (in the case of this research, H is contained in a suffix tree), and *f* be the size of the longest suffix of s such that $s_f = \{x_{m-f+1}, ..., x_m\}$ exists in H. A VLMM determines P(x) by combining or blending $P_i(x)$, ∀ i = {−1, 0, 1, ..., f} according to the equations below. The intuition is that past observations (captured by the symbols following suffixes of s that exist in H) are representative of the future.

$$P(X) = P(X_{m+1} = x) = \sum_{j=-1}^{f-1} \sum w_j \times P_j(x)$$

$$w_j = \left(1 - e_j\right) \times \prod_{k=j+1}^{f} e_k \tag{16}$$

$$e_j = \frac{1}{C_j + 1}$$

Suffix tree creation Psuedocode:

```
learn sequence( string sequence )
for offset from 1 to sequence.length do
  current node = root
  current node.incident edge frequency += 1
  for i from offset to sequence.length do
    child = current node.get child( sequence[ i ] )
    if ( child != null )
      current node = child
      current node.incident edge frequency += 1
    else
      child = createBranch( current node,
      sequence, i,
      sequence.length - i )
      break
    end if
  end for
end for
create branch( suffix tree node parent,
string sequence,
int offset, int length )
descendent[ 0 ] = parent
for i from 1 and j from offset, i < length + 1 do
  new node = sequence[ i ]
  new node.incident edge frequency = 1
  new node.parent = descendent[ i -1 ]
  descendent[ i ] = new node
end for
return descendent[ length ]
```

Prediction pseudocode:

```
predict( string context, character c )
accumulated escape = 1
for i from 1 to context.length do
  sub context = context[ i, context.length ]
  append context = sub context & c
```

```
  nom = # matches for append context in suffix tree
  denom = # matches for sub context in suffix tree
  probability = nom / denom
  escape prob = compute escape(context)
  prediction = prediction + probability × (1 - escape prob ) × accumulated escape
  accumulated escape = accumulated escape × escape prob
end for
escape prob = compute escape(context)
nom = number of matches for c in suffix tree
denom = total number of observed characters
probability = nom / denom
prediction = prediction + probability × (1 - escape prob )× accumulated escape
accumulated escape = accumulated escape × escape prob
probability = 1 / alphabet size
prediction = prediction + probability × accumulated escape
return prediction

compute escape( string context )
nom = number of characters following context in suffix tree
denom = number of matches for context in suffix tree
escape prob = 1 / ( 1 + denom )
return escape prob
```

2.5.6 Projection Using Intent

While it would be ideal to project activities based on intent, it is very difficult (or even impossible) to accurately do without knowledge of the attacker. Since cyber attacks are typically executed by an unknown person (perhaps disguising himself as another unknown person), it is generally impossible to know the intent of the hacker. We decided to not use intent as a projection algorithm.

2.5.7 Modeling Plausible Futures

The output of the Dempster-Shafer combination algorithm is a list of plausibility scores and reliabilities for each OOI in the environment. Recall that for cyber security, these OOI's are missions, hosts, and services. However, some combination of hosts and services help to perform a mission. However, a mission is often comprised of many non-cyber elements. Therefore, projecting a plausible future with respect to a mission may not make sense since the current cyber architecture is unable to capture the complete view of the mission. Also, projecting a plausible future with a specific service may not make sense because the compromise of services running on the same machine may have the exact same effect (and thus be identical plausible futures). Therefore, plausible futures are generated based on the plausibility scores assigned to a given host.

The process by which plausible futures are generated is very similar. Since a plausible future is just a projection of a given situation, the plausible future is the given situation with one or more additional OOI's compromised. This allows impact assessment (see next section) to use the exact same algorithm regardless of whether it is the current situation or a plausible future. For every host that has a plausibility score assigned to it, a plausible future is created. It is also possible to assign a plausibility score threshold to limit the number of plausible futures

generated, but this could potentially rule out a low plausibility but high impact event – which may not be desirable to the analyst.

2.5.8 Impact/Damage Assessment

Impact/damage assessment is responsible for determining the impact of a set of activities to OOIs on the network. We model impact by assigning an impact score between 0 and 1 inclusive to a given OOI.

2.5.8.1 Virtual Terrain Assisted Impact Assessment for Cyber Attacks (VTAC)

Let H, S, U, and N be the set of hosts, services, users, and subnets (including the entire network), respectively. The set of hosts include individual hosts, routers, servers, host clusters and server farms. A single entity is defined in the virtual terrain (VT) for a set of hosts or servers if they are configured identically and with the same access privileges. The set S contains the types of services offered in the entire network. An enterprise network typically offers the same type of services, e.g., FTP or SMTP, in different subnets. The different service instances are denoted as $r \in R$. If there is exactly one instance for each service type, then $R \equiv S$. For any pair of sets X and Y, the notation $X(y)$ denotes a subset of X deduced with $y \in Y$. For example, $R(h)$ means the set of service instances offered in the host node h, $S(r)$ deduces the service type of the service instance r, $N(h)$ returns the subnet h resides, and $U(h)$ determines the set of users who have access to h. The associations between different elements in different sets are defined in the VT with directional arcs.

Each host node $h \in H$ contains a set of service instances $R(h)$, and each service instance $r \in R(h)$ contains a set of exposures $E(r)$. The exposures are the vulnerabilities of the corresponding service and shall match to IDS alerts in the VT. The mapping is done by referring to Nessus [15] scans, Common Vulnerabilities and Exposures (CVE) dictionary [9], and IDSs. For work reported in this paper, we consider only a subset of Snort alerts [13]. In addition, each host node may contain an allowed or denied list, which will be referred to as the firewall list for convenience. This firewall list may contain IPs, port numbers, and/or protocols, and will be particularly comprehensive if the host node is a router. The physical connectivity of host nodes are defined in the VT, and the firewall rules are checked against the attributes in IDS alerts to determine the logical connectivity.

Based on the above definitions, the following subsections define the impact scores for the hosts (I_H), the services (I_S), the users (I_U), and the subnets or the entire network (I_N). In the absence of an agreed upon definition of 'impact,' all impact scores are defined with respect to the total loss of the entity being assessed, and are normalized to a value between 0 and 1. A weight or criticality of each sub-component with respect to an entity, therefore, is defined.

2.5.8.1.1 Impact Scores

Host Impact: The potential damage done to a host with respect to its services and their importance to the host.

The impact to a host $h \in H$ depends on the services running and the asserted exposures, $E^{*}(r,t)$, of each service r due to the attack actions observed up to time t. For each service instance $r \in R(h)$ on the host h, the asserted exposure scores, α_e, $e \in E_{-}(r,t)$, can be derived based on CVSS [9] and normalized in [0, 1]. The maximum of the asserted exposure scores is defined to be α_r, representing the maximum damage imposed to the service instance r.

$$\alpha_r = \max_{e^{*}(r,t)} \alpha_e \,;[0,1] \tag{17}$$

The impact score for a host is the average of α_e scores weighted by the respective service criticality ($c(r,h)$). Note that $c(r,h)$ is defined to represent the importance of the service instance r to its host h.

$$I_H(h) = \frac{\sum_{r \in R(h)} c(r,h) \cdot \alpha_r}{\sum_{r \in R(h)} c(r,h)} \tag{18}$$

Service Impact: How potentially compromised a particular service is across the entire network.

To calculate the impact score for a service type $s \in S$, every instance of a given service running in different hosts on the network is analyzed. For each instance $r : S(r) = s$, the same rule is used as above to get α_r. Similar to the definition of I_H, I_S is a weighted average of α_r but with weights $c(r,s)$, which represents the criticality of the service instance r to the overall service s.

$$I_S(s) = \frac{\sum_{r \in R(s)} c(r,s) \cdot \alpha_r}{\sum_{r \in R(s)} c(r,s)} \tag{19}$$

User Impact: The potentially effect to each user's use of the hosts under attack.

There are different interpretations in assessing cyber attacks to users. One interpretation is to assess whether a user privilege on different machines has been compromised. This, however, requires host-based IDS to report privilege information on attack actions. Accurately reporting this information is not as easy as it seems and this work does not utilize such information. Alternatively, user impact can be thought as the level of confidence a user can still use the machines he or she has accounts on. In realizing this notion, I∪(u) is defined as a weighted average of $I_H(h)$ for $h \in H(u)$ with weights $c(h,u)$.

$$I_U(u) = \frac{\sum_{h \in H(u)} (c(h,u) \cdot I_h(h))}{\sum_{h \in H(u)} c(h,u)} \quad (20)$$

Network/Mission Impact: The potentially damage on a subnet or the entire network or mission.

A mission is comprised of a subset of elements in the computer network. For simplicity, when referring to the Network/Mission Impact, the term "network" will refer to either the entire network or a mission. The network impact score will allow an analyst to monitor the health of a subset of or the network. Intuitively, the calculation may be an aggregation of I_H, I_S, or I_U within the subset of or the entire network. Consider the target $n \in N$ and a set of weights with respect to this target. The impact score $I_N(n)$ can be defined as follows for $X = H, S, U$.

$$I_N(n) = \frac{\sum_{x \in X(n)} (c(x,n) \cdot I_X(x))}{\sum_{x \in X(n)} c(x,n)} \quad (21)$$

All of the impact scores can be determined for one or many attack tracks. This allows for a ranking of attack tracks in terms of their impact to different parts of the network. In fact, network analysts may focus on the most affected part of the network (e.g., subnet, services, or users) and trace the multistage attacks that cause the damage.

2.5.8.1.2 Reference Impact Scores

The four impact scores defined thus far allow network analysts to examine and focus on the current damage done to the network. This may be sufficient to perform a rudimentary analysis, however it seems as though these scores are lacking a reference point to the potential damage that can actually be done. Since the current impact scores largely rely on exposure damage scores and different criticalities, depending on the network configuration, impact scores may be incapable of reaching the maximum score of 1.0. To put the impact scores into reference with the current and potential situation, we propose two functions for each of the components being analyzed. These new reference scores will give an analyst a better overall view of the current situation. The two reference scores are defined as follows:

- **Impact score for X with Maximum Exposure for all Hosts (IX-MEH)** - Represents the highest possible impact score that a component could have, given that for all of the services related to it, regardless whether it is asserted or not, their exposure with the highest damage score is asserted.
- **Impact score for X with Maximum Exposure for asserted Services (IX-MES)** - Represents the highest possible impact score that a component could have, if for all of

the asserted services related to it, their respective exposure with the highest damage score is asserted.

The IX-MEH score can put the impact scores into perspective by showing the analyst the maximum damage that can be done to the X component. The calculation of IX-MEH for host, service, user, and network is similar as described in the previous sections, except for equation (22). Instead, equation (23) is used to calculate α, analyzing all existing exposures, not only the asserted ones.

$$\alpha_r = \max_{e^*(r,t)} \alpha_e \; ; [0,1] \tag{23}$$

Where this time, the iteration is done over $E^*(r)$, all exposures associated with the service instance r. The score resulting from the second algorithm, IX-MES, shows the analyst the maximum potential impact score given the currently asserted services. This results from an attacker fully exploiting the services of which they have already attacked, only on the hosts that they have already attacked.

2.5.8.1.3 Logical and Illogical Attacks

In determining the impact scores, VTAC does more than blindly process the incoming correlated alerts. Since alerts are grouped with others that potentially belong to the same sequence of a multistage attack, VTAC can determine whether a step is 'illogical' based on the prior steps in the same sequence and the attributes defined in the VT, such as the firewall rules. An attack step is illogical if the corresponding alert contains conflicting attributes to the VT definition. There are a few possible reasons for illogical steps: (1) the alert is a false positive, (2) the VT definition is wrong or not updated, (3) there are coordinated attacks or wrongly correlated alerts, and (4) it is a new attack and wrongly categorized to a known alert message. Determining the root cause for the problem is out of the scope of this papered report. VTAC will merely identify the illogical steps and leave it to the analysts to determine whether to process these steps in determining the impact scores. The algorithm used to determine illogical steps is the same as the algorithm described in Section 2.4.6.1. Any activity that does not change the state of any element in the VT is considered to be illogical.

2.5.8.2 State-Based Impact Assessment Algorithm

VTAC was developed with the intent of assessing the impact of the current situation. As a result, it was largely developed with the need to process the observables to determine the impact. However, to reduce complexity in the projection, the capability and opportunity projection algorithms only assign plausibility scores at a host and service level. Since they do not attempt to project any observables, VTAC would be insufficient in calculating the impact of a plausible future. Therefore, a state-based impact assessment algorithm was developed to

overcome this problem. The state-based impact assessment algorithm is a simple calculation that assigns an impact based on the severity of the object state. Each object state is assigned a value, *v*, that corresponds to the relative threat of that state with respect to the other states. The values are assigned as follows: Normal (0), Attacked (1), Discovered (2), Partially Compromised (3), and Compromised (4). For each OOI, the final impact score, *i*, is calculated by normalizing the v values:

$$i = v / 4 \tag{24}$$

2.5.8.3 Ranking Current Situation(s) and Plausible Futures

After plausibility scores and impact scores have been calculated, it may be useful to rank the attack tracks, or situations, against each other. This will allow the analyst to quickly determine which situations to rank. However, given the parameters such as impact and plausibility, it is not obvious how the situations should be ranked. This section goes through an example to illustrate that this process does not involve the combination of these parameters, but rather an analysis of multiple parameters and rankings.

Table 4 shows example FuSIA outputs. There are two situations and each situation had three plausible futures generated (denoted by X.Y where X is the situation identifier and Y is the plausible future identifier). In this example, there are two missions of interest to the analyst, so there are two impact scores generated for each situation and plausible future. Plausible futures 1.3 and 2.2 are highlighted because they each have a situation of a very low plausibility, but an extremely high impact. Such instances imply that a very devastating situation is possible, but unlikely to occur. This is one of the key reasons that ranking situations cannot involve the combination of impact and plausibility because even these low plausibility, high impact situations would be of interest to the analyst.

Table 4: Example FuSIA Outputs

Situation	Plausibility	Mission 1 Impact	Mission 2 Impact
1		0.50	0.25
1.1	0.58	0.70	0.30
1.2	0.62	0.52	0.99
1.3	0.01	**0.99**	0.26
2		0.60	0.20
2.1	0.78	0.61	**0.98**
2.2	0.13	0.62	**0.97**
2.3	0.12	0.61	0.45

Table 5 shows each situation ranked against each other using various metrics in each column. The top-ranked situation is shown in bold. The "ratio" scores are just normalizations of the scores with respect to the maximum score in the column to the left. It should be noted that with respect to Mission 1, Situation 2 is ranked higher in all instances except the current impact. With respect to Mission 2, Situation 1 is ranked higher in all instances except the average future impact. The intent of this table is to illustrate that current impact may not always be more important to look at than future impact (and vice versa).

Table 5: Example Impact Rankings

Mission 1 Situation Rankings						
Situation	Current Impact	Current Impact Ratio	Max Future Impact	Max Future Impact Ratio	Avg Future Impact	Avg Future Impact Ratio
1	0.50	0.83	**0.99**	**1.00**	**0.74**	**1.00**
2	**0.60**	**1.00**	0.62	0.63	0.61	0.83
Mission 2 Situation Rankings						
Situation	Current Impact	Current Impact Ratio	Max Future Impact	Max Future Impact Ratio	Avg Future Impact	Avg Future Impact Ratio
1	**0.25**	**1.00**	**0.99**	**1.00**	0.52	0.65
2	0.20	0.80	0.98	0.99	**0.80**	**1.00**

Table 6: Example Plausible Future Rankings

Mission 2 Plausible Future Rankings				
Future	Plaus Score	Impact Score	Max Plausor Impact	Plaus*Impact
1.1	0.58	0.30	0.58	0.17
1.2	0.62	**0.99**	**0.99**	0.61
1.3	0.01	0.26	0.26	0.00
2.1	**0.78**	0.98	0.98	**0.76**
2.2	0.13	0.97	0.97	0.13
2.3	0.12	0.45	0.45	0.05

Table 6 shows example rankings of plausible futures, with the highest ranked plausible future shown in bold in each column. Due to the low plausibility, high impact scores, it is also inconclusive on how to rank plausible futures against each other. Depending on the ranking

strategy, situation 2.2 is either ranked very high, or very low. It is also never the top-ranked situation.

2.6 FuSIA Screenshot

Using scenarios developed in VTAC, the following is a screenshot of FuSIA. This interface was designed for debugging purposes to illustrate the output of FuSIA as a prototype. The upper-left window shows a list of the different attack tracks, or situations, and associated performance statistics. The right window shows the objects of interest, object states with respect to the selected situation, plausibility scores from projection algorithms, and the impact to each object of interest. The bottom window illustrates the activities associated with the selected situation. This contains the fields relevant to the activity such as IP addresses and protocols. A fully-functional demo of FuSIA is available in the N-CMIF deliverables CD. The demo can be executed as a standalone application with only FuSIA running, or it can be executed to accept real-time input from INFERD.

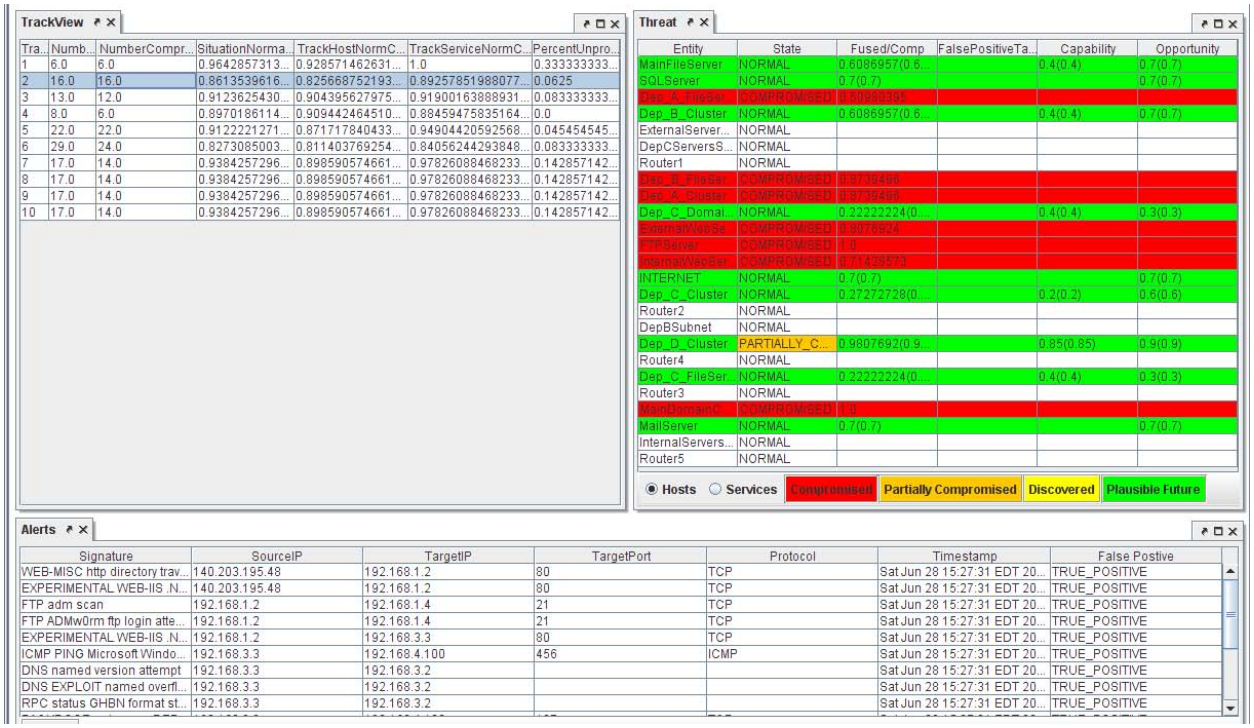


Figure 7: Example FuSIA Output

2.7 Level 4 (Process Refinement)

The research conducted in this section is relevant to tasks 4.1.1.1, 4.1.1.4, and 4.1.1.6.

Sensors for network security can generally be divided into four types: host intrusion detection sensor (HIDS), host intrusion prevention sensor (HIPS), network intrusion detection sensor (NIDS) and network intrusion prevention sensor (NIPS). When they are applied to network

security purposes, they must be placed at an appropriate location in the network and turned on. Information generated by these sensors is also needed to be extracted to a single or multiple fusion nodes. Therefore, three steps are necessary to sensor management for cyber network security:

- Locating sensors in a cyber network;
- Turning on sensors to detect or prevent cyber attacks; and
- Extracting information from working sensors to fusion nodes.

According to the above steps, we need to solve the following three problems:

- Where shall we locate and how do we choose sensors in a cyber network?
- When shall we turn on or turn off a sensor?
- When shall we extract information from a sensor?

To solve these problems in an objective manner, we need to develop a sensor management model for fusion process refinement. We can decompose the process refinement process into two main categories: (1) Off-Line Sensor Management and (2) On-Line Sensor-Management. In this report we document our work on off-line sensor management. We have developed a detailed Mathematical Model which will aid in the location and setting of cyber sensors in a startup network. By solving the model we can conduct an effective sensor management for cyber network security.

Currently, sensor management for cyber network security depends a lot on human judgment and subjective experience. This may result in inaccurate decisions in many cases. Therefore, our research is dedicated to finding an objective way to conduct effective sensor management. To achieve this, we develop a mathematical model based on the daily operation of sensors and networks. By solving the model we can get the solutions to sensor management in an optimal rather than ad hoc manner.

A difficulty for sensor management is that it contains two different types of decisions: strategic decisions and tactical decisions. In daily operations, the location of a sensor cannot be changed once it is placed in the network. Therefore, the decisions for sensor locations are strategic decisions. However, we need to make different decisions during different periods when we consider whether to extract information from a sensor. Therefore, they are tactical decisions for the decisions to extract information from them. Using constraints, we can encompass these two types of decision problems into a single model.

There are three objectives for this research work:

- Developing a mathematical model for off-line sensor management for the cyber network security;

- Conducting effective sensor management for the cyber network security by solving the model.

By achieving the research objectives above, we can gain the following benefits:

- Conducting sensor management in a scientific and objective way based on the optimal solution of the mathematic model;
- Realizing a rapid response to environmental changes in a dynamic environment.

The main technical findings are as follows:

- A mathematical method has been proposed and formulated to determine optimal locations of fusion nodes;
- The sensor management model is totally linearized to facilitate software computation;
- A Java program has been coded to calculate and optimize initial sensor settings;
- A Java program has been coded to test the performance of the mathematical model on cyber networks. This program consists of a random network topology generator, a random parameter generator, and a mathematical model formulator and solver, which is powered by a COTS optimization package CPLEX.

2.7.1 Mathematic Model in Cyber Security

2.7.1.1 Fusion Node Placement

The number and locations of fusion nodes are decided by the solution to the model below:

$$\min \sum_{i\in I}\sum_{j\in J} c_{ij}y_{ij}$$

$$s.t. \quad \sum_{j\in J} y_{ij} = 1\,, \qquad \forall i$$

$$x_j - y_{ij} \geq 0, \qquad \forall i,j$$

$$\sum_{j\in J} f_j x_{jk} \leq B$$

$$x_{jk} = 0 \; or \; 1 \qquad \forall i,j$$

$$y_{ij} = 0 \; or \; 1 \qquad \forall i,j$$

where

$x_{jk} = \begin{cases} 1, \\ 0, \end{cases}$ if fusion node j is placed at location k / otherwise

$y_{ij} = \begin{cases} 1, \\ 0, \end{cases}$ if location i is assigned to fusion node j / otherwise

c_{ij} the cost of extracting information from location i to fusion node j

f_j the financial cost to place fusion node j

B the budget to place fusion nodes

2.7.1.2 *Sensor Management Model*

The topology of a sample cyber network is show as Figure 8.

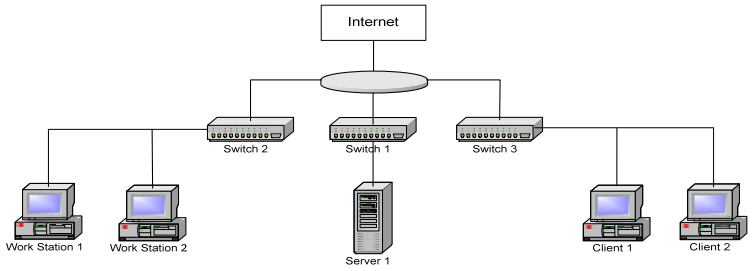


Figure 8: Topology of a Cyber Network

In the model, the decision variables are as follows:

$$x_{sl} = \begin{cases} 1, & \text{if sensor } s \in S^{HD} \cup S^{HP} \cup S^{ND} \cup S^{NP} \text{ is located at } l \in L^H \cup L^N \\ 0, & \text{otherwise} \end{cases}$$

$$y_{slt} = \begin{cases} 1, & \text{if sensor } s \text{ is turned on at location } l \text{ during time period } t \in T \\ 0, & \text{otherwise} \end{cases}$$

$$z_{slt} = \begin{cases} 1, & \text{if extract information from sensor } s \text{ at location } l \text{ during time period } t \\ 0, & \text{otherwise} \end{cases}$$

$$q^E_{mat} = \begin{cases} 1, & \text{if machine } m \in L^H \text{ is protected from attack } a \in A^I \cup A^O \text{ from an external IP during time period } t \\ 0, & \text{otherwise} \end{cases}$$

$$q^I_{mat} = \begin{cases} 1, & \text{if machine } m \in L^H \text{ is protected from attack } a \in A^I \cup A^O \text{ from an internal IP during time period } t \\ 0, & \text{otherwise} \end{cases}$$

The parameters in the model are as follows:

- L^H: set of host machines
- L^N : set of network locations
- S^{HD} : set of HIDSs
- S^{HP} : set of HIPSs
- S^{ND} : set of NIDSs
- S^{NP} : set of NIPSs
- A^M : set of attacks which can lead to an intrusion
- A^N : set of attacks which cannot lead to an intrusion
- T : set of time periods

- K : set of links
- E_m : set of entry points for machine m
- v_m : the importance of machine m
- w_a: the weight of attack a
- c_s: financial cost for sensor s
- g_s: procession capability needed when sensor s is on
- e_s: bandwidth per link needed when extract information from sensor s
- F : total financial budget
- P_l : procession capability limit for location l
- B_j : bandwidth limit for link j
- Q_m : minimum

$$d_{sa}^I = \begin{cases} 1, & \text{if sensor } s \text{ can detect attack } a \text{ from an internal IP} \\ 0, & \text{otherwise} \end{cases}$$
$$d_{sa}^E = \begin{cases} 1, & \text{if sensor } s \text{ can detect attack } a \text{ from an external IP} \\ 0, & \text{otherwise} \end{cases}$$
$$p_{s_1 s_2} = \begin{cases} 1, & \text{if sensor } s_1 \text{ and } s_2 \text{ can be placed at a same location} \\ 0, & \text{otherwise} \end{cases}$$
$$n_{sl} = \begin{cases} 1, & \text{if sensor } s \text{ can be placed at location } l \\ 0, & \text{otherwise} \end{cases}$$
$$r_{ij} = \begin{cases} 1, & \text{if link } j \text{ is used to extract information from a sensor at location } l \\ 0, & \text{otherwise} \end{cases}$$

Based on decision variables and parameters above, the mathematic model is:

$$\max \sum_t \sum_m v_m P_{mt}$$
$$s.t.$$
$$P_{mt} = \frac{\sum_{a \in A} w_a (q_{mat}^E + q_{mat}^I)}{2 \sum_{a \in A} w_a}, \qquad \forall m \in L^H, t \in T \qquad \text{(definition of protection level)}$$
$$P_{mt} \geq Q_m, \qquad \forall m \in L^H, t \in T \qquad \text{(minimum protection)}$$
$$q_{mat}^E \leq \sum_{s \in S^P} d_{sa}^E y_{smt} + \sum_{s \in S^D} d_{sa}^E z_{smt}, \quad \forall m \in L^H, a \in A^N, t \in T$$
$$q_{mat}^I \leq \sum_{s \in S^P} d_{sa}^I y_{smt} + \sum_{s \in S^D} d_{sa}^I z_{smt} \quad \forall m \in L^H, a \in A^N, t \in T$$

$$q_{mat}^{E} \le \sum_{s\in S^{P}} d_{sa}^{E} y_{smt} + \sum_{s\in S^{D}} d_{sa}^{E} z_{smt} + \sum_{s\in s^{P}} d_{sa}^{E} y_{sl_m^i t} + \sum_{s\in S^{D}} d_{sa}^{E} z_{sl_m^i t}, \quad \forall m \in L^H a \in A^M, t \in T, \; l_m^i \text{ is the } i^{th} \text{entry point of } m$$

$$q_{mat}^{I} \le \sum_{s\in S^{P}} d_{sa}^{I} y_{smt} + \sum_{s\in S^{D}} d_{sa}^{I} z_{smt} + \sum_{s\in s^{P}} d_{sa}^{I} y_{sl_m^i t} + \sum_{s\in S^{D}} d_{sa}^{I} z_{sl_m^i t}, \quad \forall m \in L^H a \in A^M, t \in T, \; l_m^i \text{ is the } i^{th} \text{entry point of } m$$

$$x_{s_1 l} + x_{s_2 l} \le p_{s_1 s_2} + 1, \quad \forall s_1, s_2 \in S^{HD} \cup S^{HP} \cup S^{ND} \cup S^{NP}, \; l \in L^H \cup L^N \quad \text{(sensor compatibility)}$$

$$\sum_{l:r_{lj}=1} \sum_{s} e_s z_{slt} \le B_j, \quad \forall j \in K, t \in T \quad \text{(bandwidth limit)}$$

$$\sum_{s\in S^{HD}\cup S^{HP}\cup S^{ND}\cup S^{NP}} g_s y_{slt} \le P_l, \quad \forall l \in L^H \cup L^N, t \in T \quad \text{(procession capability limit)}$$

$$\sum_{s\in S^{HD}\cup S^{HP}\cup S^{ND}\cup S^{NP}} \sum_{l\in L^H \cup L^N} c_s x_{sl} \le F, \quad \text{(finanacial budget limit)}$$

$$x_{sl} \le n_{sl}, \quad \forall s \in S^{HD} \cup S^{HP} \cup S^{ND} \cup S^{NP}, \; l \in L^H \cup L^N \quad \text{(sensor location)}$$

$$z_{slt} \le y_{slt} \le x_{sl} \quad x_{sl}, y_{sl}, z_{slt} = 0 \; or \; 1, \forall s, l, t$$

2.7.1.3 Explanations to the Constraints in the Sensor Management Model

Generally speaking, the cyber attacks can be divided into two types:

- Attacks that will lead to intrusions;
- Attacks that will not lead to intrusions.

If an attack will lead to an intrusion, there are two ways to protect computers from it:

- Protecting the computer by directly locating HIDSs or HIPSs at it
- Protecting the computer by locating NIDSs or NIPSs at its entry points (switches, hubs, etc.)

Therefore, the constraints based on q_{mat} are introduced, and the related constraints are:

$$q_{mat}^{E} \le \sum_{s\in S^{P}} d_{sa}^{E} y_{smt} + \sum_{s\in S^{D}} d_{sa}^{E} z_{smt} + \sum_{s\in s^{P}} d_{sa}^{E} y_{sl_m^i t} + \sum_{s\in S^{D}} d_{sa}^{E} z_{sl_m^i t}, \quad \forall m \in L^H a \in A^M, t \in T, \; l_m^i \text{ is the } i^{th} \text{entry point of } m$$

$$q_{mat}^{I} \le \sum_{s\in S^{P}} d_{sa}^{I} y_{smt} + \sum_{s\in S^{D}} d_{sa}^{I} z_{smt} + \sum_{s\in s^{P}} d_{sa}^{I} y_{sl_m^i t} + \sum_{s\in S^{D}} d_{sa}^{I} z_{sl_m^i t}, \quad \forall m \in L^H a \in A^M, t \in T, \; l_m^i \text{ is the } i^{th} \text{entry point of } m$$

If an attack will lead to a result other than intrusion, such as reconnaissance or privilege escalation, the only way to protect a machine is to place the HIDS or HIPS, which can prevent the attack, on the host machine. Therefore, the constraint is:

$$q^E_{mat} \le \sum_{s \in S^P} d^E_{sa} y_{smt} + \sum_{s \in S^D} d^E_{sa} z_{smt}, \quad \forall m \in L^H, a \in A^N, t \in T$$

$$q^I_{mat} \le \sum_{s \in S^P} d^I_{sa} y_{smt} + \sum_{s \in S^D} d^I_{sa} z_{smt} \quad \forall m \in L^H, a \in A^N, t \in T$$

2.7.2 Model Validation

2.7.2.1 Values of Parameters

The validation of the model has been tested on the example cyber network shown in Figure 8. The values of parameters are shown as follows:

- Locations for sensors
  - $L^H = \{l_1^H, l_2^H, l_3^H, l_4^H, l_5^H\}$
  - $L^N = \{l_1^N, l_2^N, l_3^N, l_4^N\}$
- $F = 100$
- $P_l = (12, 9, 8, 5, 5, 20, 15, 12, 8)$
- $B_j = (20, 18, 16, 13, 10, 9, 8, 6)$
- $Q_m = (1/2, 5/12, 1/4, 1/12, 1/12)$

Table 7: Values of Information on Hosts

Host	Server	Work Station 1	Work Station 2	Client 1	Client 2
Value	10	8	6	3	1

Table 8: Sensors

Sensor Type	S^{HD}		S^{HP}	S^{ND}	S^{NP}
Sensor	s_1^{HD}	s_2^{HD}	s_1^{HP}	s_1^{ND}	s_1^{NP}
c_s	9	8	11	12	20
g_s	6	5	10	8	12
e_s	6	4	5	10	9

Table 9: Attacks

Attack Type	A^M				A^N	
Attack	a_1^M	a_2^M	a_3^M	a_4^M	a_1^N	a_2^N
Weight	1	1	1	1	1	1

2.7.2.2 Computational Results

The remaining financial budget for the example above is 5. The results for the protection levels are illustrated in Table 10. The results for the remaining procession capability are illustrated in Table 11. The results for the remaining bandwidth are shown in Table 12.

Table 10: Protection Levels

Host	l_1^H	l_2^H	l_3^H	l_4^H	l_5^H
PL	5/6	¾	3/4	2/3	5/12

Table 11: Remaining Procession Capability (PC)

location	l_1^H	l_2^H	l_3^H	l_4^H	l_5^H	l_1^N	l_2^N	l_3^N	l_4^N
PC	2	4	3	0	5	20	3	0	0

Table 12: Remaining Bandwidth

link	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
bw	20	10	2	13	6	5	4	6

2.7.2.3 Results Analysis

Some conclusions based on the computational results shown above are as follows:

- When placing sensors, priority is given to more important hosts. This means that traffic going to more important hosts has a larger number of attacks that can be detected by the sensors placed to protect them.
- All types of resources are fully utilized (in the example we cannot add one more sensor to the network)

2.7.3 Computation Time Performance of the Model on Large-scale Cyber Networks

To study the practicality of commercially available mixed-integer solvers to solve the model, we have tested its time performance on large-scale cyber networks, in which there are about one thousand host machines. Here we list five sets of computational results powered by CPLEX as follows:

Table 13: CPLEX Computational Results

	Test 1	Test 2	Test 3	Test 4	Test 5
Number of hosts	900	950	1000	1050	1100
Number of network locations	61	73	76	73	80
Number of links	960	1022	1075	1122	1179
GUB cover cuts applied	6	0	6	6	3
Clique cuts applied	0	192	103	12	0
Cover cuts applied	66	9	48	54	75
Implied bound cuts applied	0	51	48	3	0
Gomory fractional cuts applied	3	57	66	11	6
Iterations	997	505	649	1255	1079
Solution time (Sec)	1.13	0.58	0.98	1.13	0.94

2.8 Data Fusion Visualization/Representation

The research conducted in this section is relevant to tasks 4.1.1.4. This research was conducted under the supervision of David Hall at Penn State University. Key Penn State IST personnel included; David Hall, Ben Hellar, Brian Panulla, Wade Shumaker and Mark Ballora. The project achieved the following: (1) Development of a new framework for linking situation space to decision space (linking situation awareness displays and understanding to decision making for threat reduction or mitigation), (2) implementation of a Macromedia Flex software infrastructure to provide for situation displays and interaction with CUBRC inference software (the INFERD system), (3) creation of new concepts for data and network displays using an advanced 3-D full immersion visualization environment, and (4) development of new concepts and demonstrations to use aural pattern recognition (sonification methods) for analysis and

recognition of types of cyber attacks. Together these results provide a basis for developing effective cyber-monitoring systems to understand the current status of a cyber network, detection and analysis of cyber-attacks, and new tools for understanding how to evaluate the vulnerability of cyber-systems.

2.8.1 Framework for linking Situation Space to Decision Space

A traditional view of the data fusion problem is shown in Figure 9. Data from multiple sensors or sources are ingested and processed via multiple techniques (across multiple levels of the Joint Directors of Laboratories (JDL) data fusion processing model to achieve a situation display or common operating picture. The process shown in Figure 9 is primarily a data-driven approach in which a situation display is updated with overlays involving data, icons representing the result of pattern recognition techniques, and state vectors representing the results of estimation algorithms. In some cases more automated processing is attempted (e.g., via rule-based logic, Bayesian belief nets, neural nets, or agent-based reasoning) to generate meta-data interpretations to annotate the situation display. While Figure 9 shows a single analyst/decision-maker, often multiple users are involved, with each user having different domains of expertise. In these situations a "common operating picture (COP)" is sought.

This situation is readily adaptable to the cyber domain. In this case we seek to determine the situation or state of a cyber network (e.g., level-1 and level-2 fusion), and to identify and characterize potential threats or attacks (level-3 fusion).

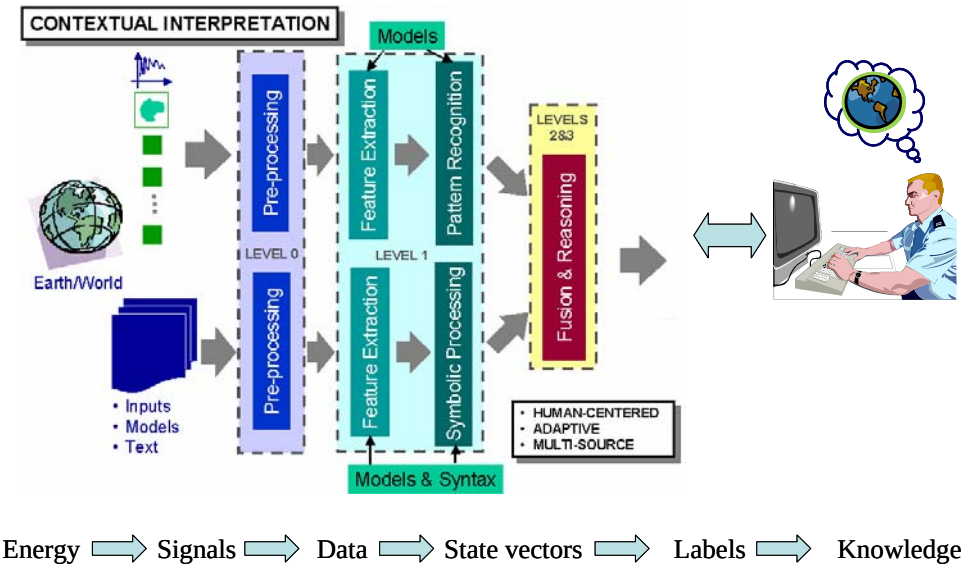


Figure 9: Concept of Information Fusion

In many applications, including tactical military operations, crisis management, monitoring and control of complex systems, the focus of a monitoring system is not merely to develop a dynamic display of the situation, but rather to support a decision-making process. The concept is shown in Figure 10.

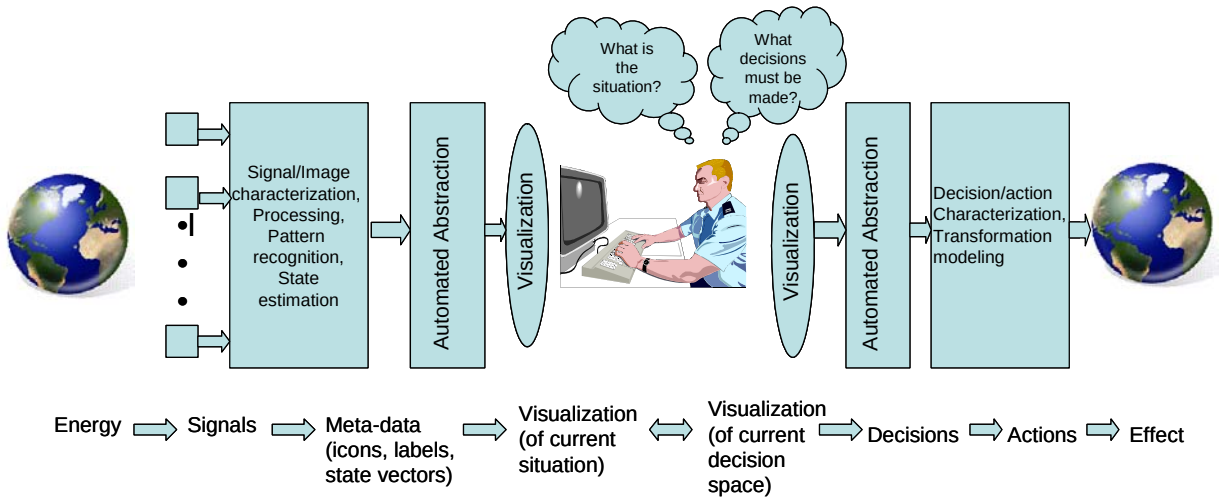


Figure 10: Connection between Situation Space and Decision Space Representation

For these applications, the analyst/decision maker must balance two simultaneous questions; (1) "What is the current situation?" and (2) "What decisions need to be made now?" Indeed, these are inextricably linked. The decisions that must be made should guide what data or elements of the situation need to be paid attention to, and the current situation should guide the analyst in knowing what decisions are necessary at the current time. Hence, the data overload gap may be described, not as simply too much data that obscures or overwhelms the analyst's ability to make sense of the situation, but rather a mismatch or gap between the current understanding of the situation (as represented by the meta data and displays) and the current understanding of the decisions to be made (as represented by meta data and displays such as templates or decision trees). We have developed a general concept of mapping situation space to decision space – a dynamic linking between situation space and decision space. The situation display guides the analyst's attention and understanding of a situation, and a related decision display guides the analyst's attention to understand what decisions need to be made and what information should be gathered and evaluated to understand how the decisions affect the evolving situation and vice-versa.

The conceptual framework shown in Figure 10 is a baseline model of the data fusion process that shows the abstracted transformation from energy to action. An advantage of this model is that it can be tailored and customized for the specific context of its application. Figure 11 shows the transformation model applied to the context of cyber-assurance. The cyber assurance application involves monitoring a computer network system for evidence of attack by virus,

worms, denial of service attacks and other threats. The goal is to obtain a situational awareness of threats and on-going attacks, and guide a network analyst/administrator in how to respond.

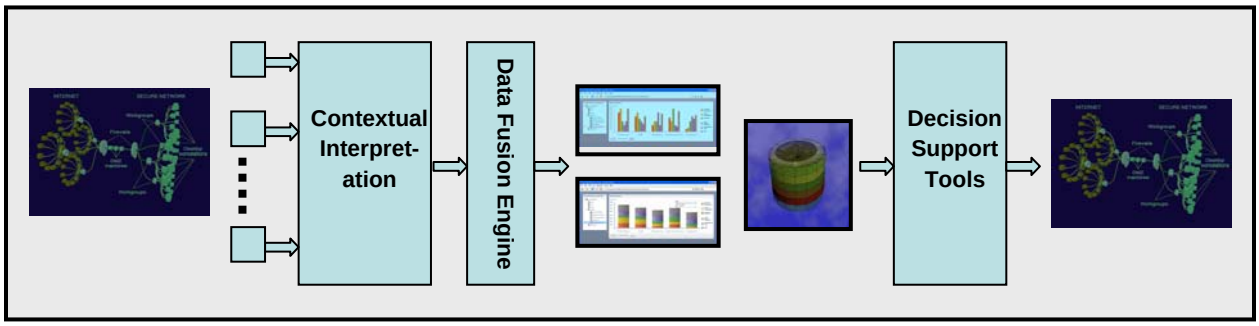


Figure 11: Application of the Transformation Model to Cyber-Assurance

In this example, sensing takes place within a virtual environment as represented by the network diagram (on the left hand side of Figure 11). The sensors are digital, and they report on the digital phenomena of the network. The vast amount of data that is collected through these sensors is processed through a data fusion engine. The data fusion engine passes the sifted metadata to a situation display that is manned by an expert user. This user is now placed in the center of a decision process where he must use decision space display and other decision support tools in order to generate a course of action. The outcome of the user's decision is some action or effect on the surveyed network.

We have developed a general framework and visual toolkit to support development of these concepts, especially for the cyber situational awareness domain. This is described in the next section.

2.8.2 A Software Infrastructure to provide Situation Displays and Decision Support

In order to achieve the goal of the conceptual framework, we developed a framework for visualization of information that has been processed through data fusion. The visualization framework is displayed in figure 4 below. The left-hand side of this figure shows the end results of this framework a decision space user interface (UI) and a situation space UI. In this example, the situation space was developed in Adobe Flex and decision space is a 3-D mockup of a decision aid developed using 3-D visualization technology. Both interfaces are fed XML data over HTTP provided by the visualization data services. The visualization data services use a combination of SOAP/REST, the flash media server to stream relevant multimedia, and flex data services. Together these services feed data in real time to the user interfaces.

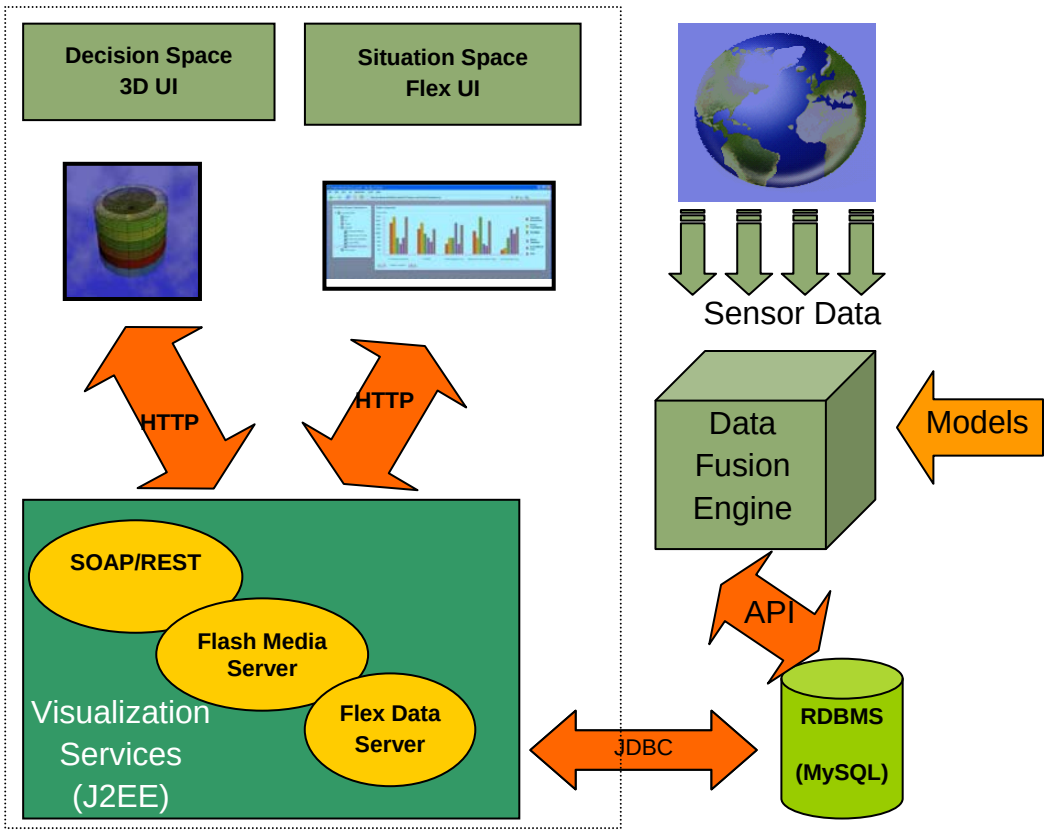


Figure 12: The Visualization Framework

The right-hand side of Figure 12 is a simplification of the data fusion process. Data, as gathered through sensors is processed with a data fusion engine. In this example, the data fusion engine is informed by use inference template models developed by researchers at the Center for Multisource Information Fusion (CMIF) (see http://www.infofusion.buffalo.edu/). The resultant output of the data fusion engine (called Information Fusion Engine for Real-time Decision Making (INFERD)) is stored in a relational database (see Stotz et al [2]). The relational database we use in this example is MySQL. Using a JDBC connection, the relational database is able to provide the visualization data services with all the data it needs.

Together both sides of the visualization framework enable a *real-time visualization* of fused data relationships. This visualization framework provides the foundations for a human in the loop system. Figure 13 shows our user concept of a human in the loop system for data fusion.

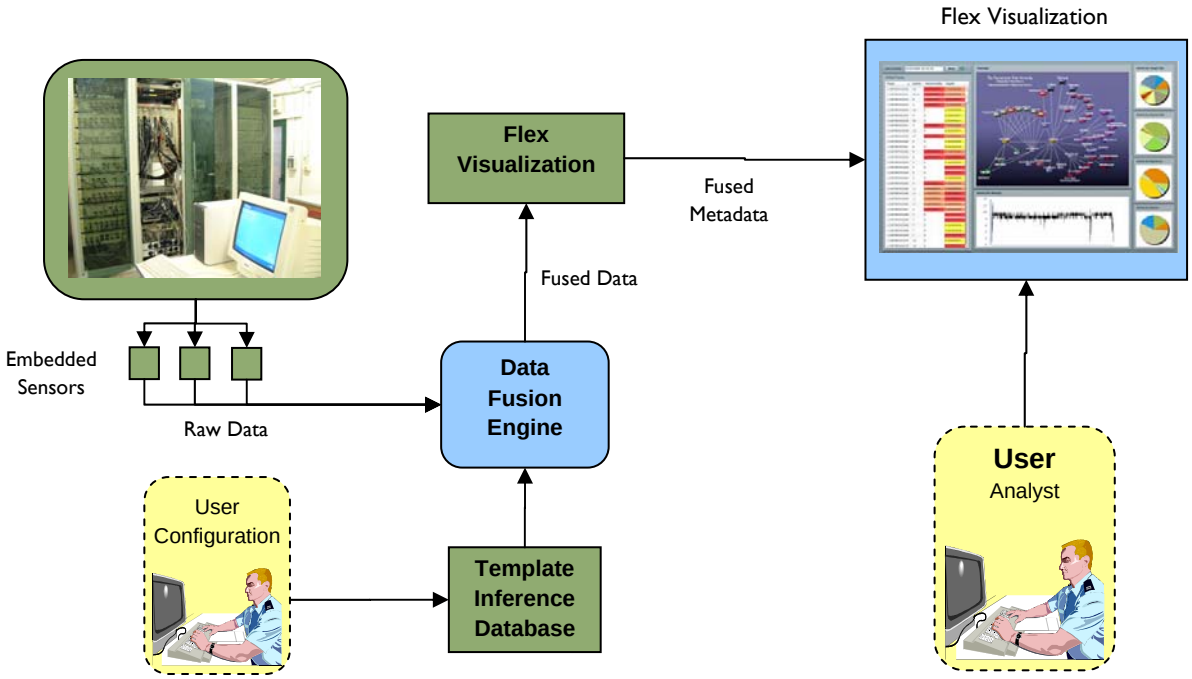


Figure 13: Data Fusion Human-in-the-Loop User Concept

A human has two important roles in this system. The first important way in which the user interacts with a data fusion system is by being a process observer. This is synonymous with the role of a human according to the level 5 process[1] proposed as part of the JDL model. As a process observer, the user configures the template inference database that guides the data fusion engine.

The second role of user in this system is that of an analyst. As an analyst the user has different responsibilities and tasks depending on if their work is embedded in either the situation or decision space.

When a user operates in a situation space, their primary responsibilities are characterized by the tasks of problem detection and diagnosis. The focus of the user should be to first identify that something is happening then follow-up by verifying what is happening. In this effort we developed a prototype of a situation assessment display to aid in cyber security network anomaly detection. A sample display of the prototype is shown in Figure 14. This interface provides the user with real-time visualization of data relationships. By showing the user temporal and topological relationships, and graphical data relationships it enables the user to recognize patterns and perform sensemaking.

[1] We understand that there is not universal acceptance of the level-5 process for the JDL model; if the reader does not recognize the level-5 process, then this can be viewed as a general human-computer-interface (HCI) function.

Initial designs for visualizing network security data with respect to the *virtual terrain* provided by the network topology focused on simplified abstract representations of network hosts and links grouped according to department, services provided, and mission. This non-traditional representation was eventually discarded in favor of a traditional graph-oriented display of the network as shown in Figure 14.

The topology in Figure 14 was drawn from a syndicated XML service provided by other components in the system, and can be updated dynamically to reflect the status of particular network nodes by simply reloading the topology from the central service.

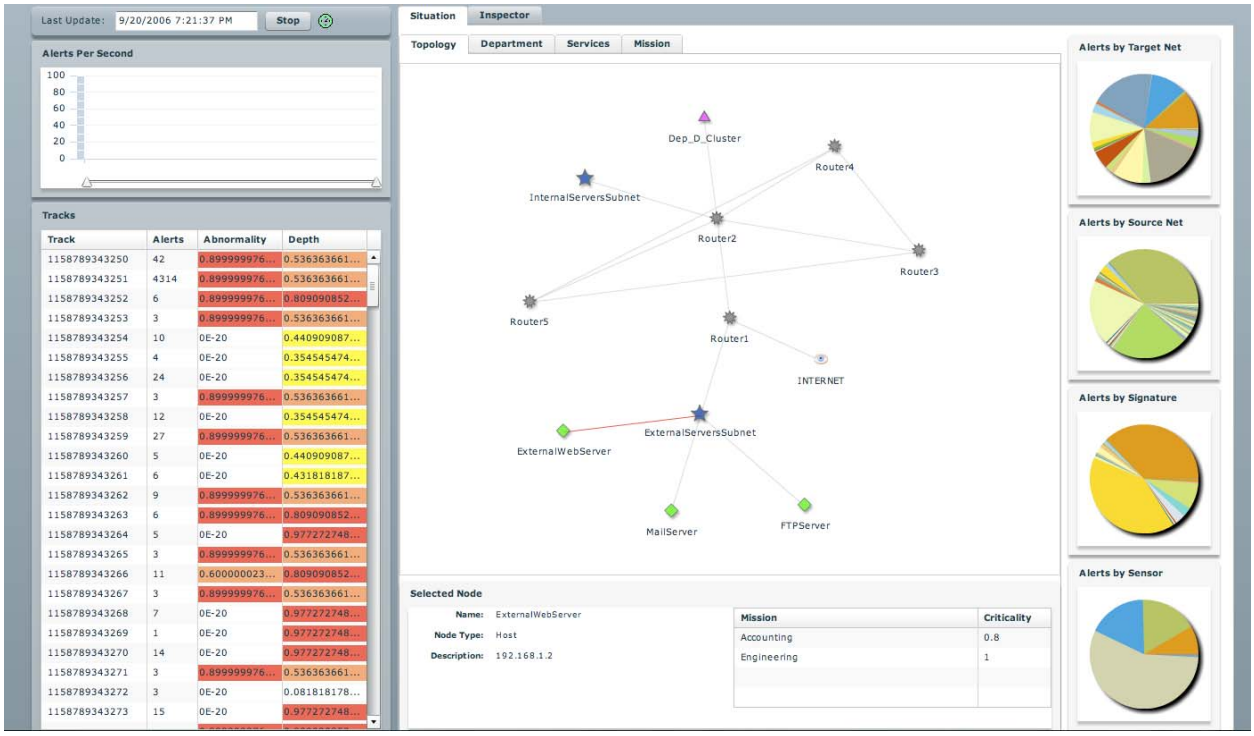


Figure 14: Sample Flex Situation Awareness/Decision Support Display

2.8.3 New Concepts for Data Visualization

In addition to the Flex-based environment for visualization and decision support, we designed and implemented several experimental displays using a 3-D, full immersion display capability at Penn State IST. The displays allow for exploration of several representations of the cyber domain; (1) a geo-physical representation of a cyber network environment (e.g., Figure 15).

(2) a 3-D view of the cyber-network activity (see Figure 8), and finally (3) a general view of network activity partitioned by network function (see Figure 9). While these displays are currently experimental, they provide a promising method for significantly engaging an analyst's visual pattern recognition capability to understand and evolving cyber-network situation and potential threats.

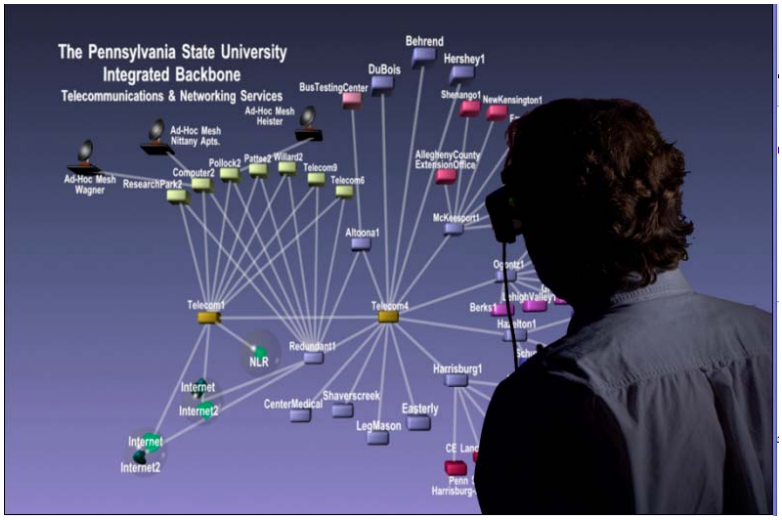


Figure 15: Sample 3-D cyber "physical landscape" display

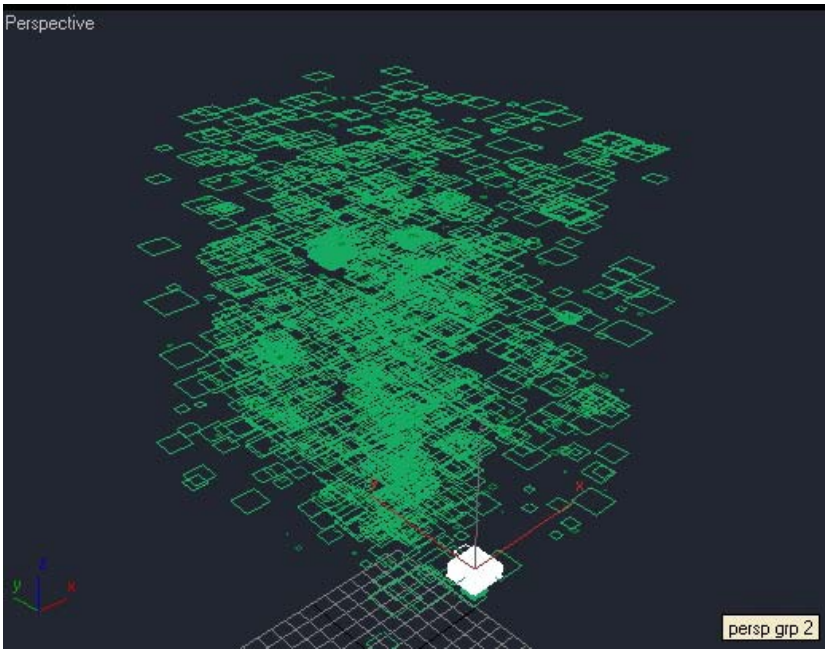


Figure 16: 3-D View of Cyber-space Network Activity (4-D view of target IP space)

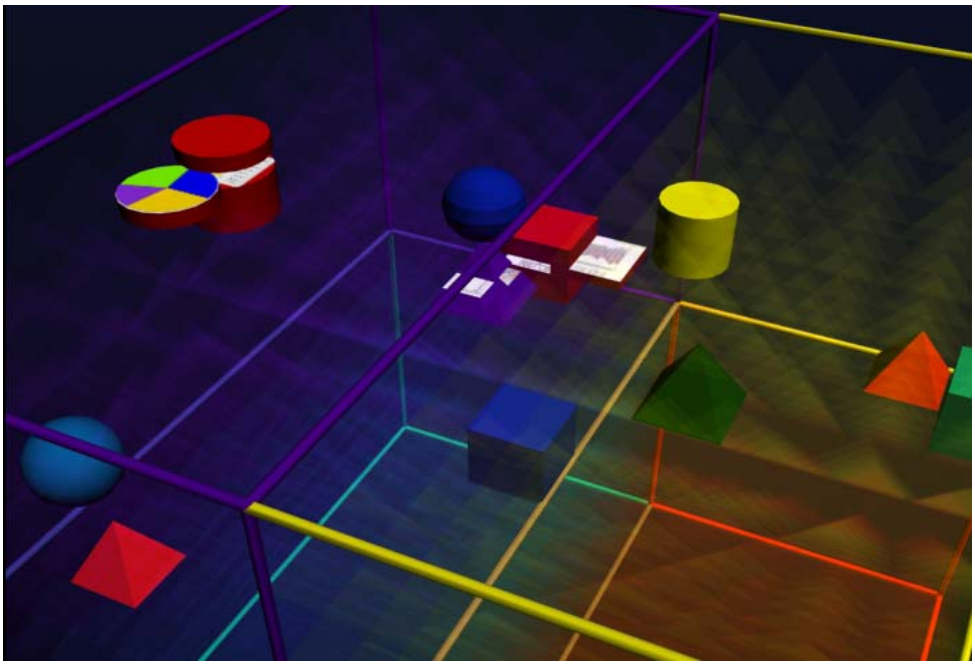


Figure 17: 3-D View of Cyber Network Activity, Partitioned by Network Component Functions

2.8.4 Sonification for Situation Understanding

The final effort under this task involved developing concepts of pattern recognition using sonification – the transformation of data into sound to allow human aural pattern recognition. Initial experiments were conducted to translate information related to network attack messages and tracks into sounds suitable for human ears. While only preliminary the results are promising [22]. We found that certain types of cyber attacks exhibited distinctive patterns that were readily recognizable, even by novice analysts.

2.8.5 Summary

This project provided the opportunity to apply the results of extensive research on traditional data fusion systems (the use of physical sensors to observe and characterize physical targets in the geo-spatial world), to concepts for monitoring the cyber-domain. Our research provided new concepts for situational awareness and decision-making, a new Flex-based software environment to support implementation of visualization and analysis tools, new concepts in 3-D visualization for data analysis and situational awareness, and finally the use of sound to support identification of anomalies and cyber threats.

2.9 Developing Test Data Sets – Cyber Attack Simulator

This section presents a model for simulating the behavior of the intrusion detection system by producing simulated alerts representative of malicious cyber attacks and non-malicious network activity based on the user's specification. With this model, a user can efficiently construct scenarios of various computer networks and cyber attacks and generate the

corresponding alerts. The cyber attack simulator is designed with the primary focus of using its generated alerts as an input to an information fusion engine. Additionally, the simulator works with "virtual terrain" files to read/write real or fabricated network topologies. Figure 18 displays how this simulator and its outputs interact with current information fusion tools.

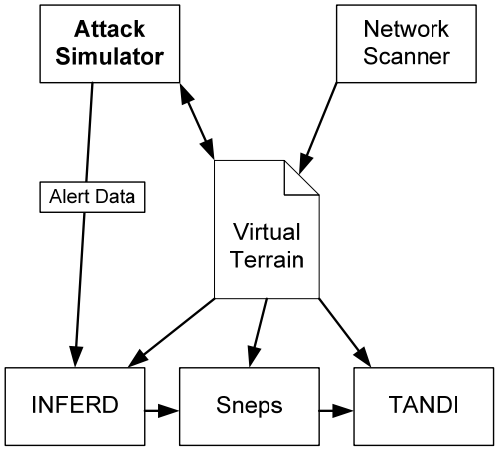


Figure 18: Use of Attack Simulator in Information Fusion Applications

2.9.1 Approach

This work is based on the need for testing situational awareness tools that are being developed to detect and analyze attacks on computer networks. Since conducting cyber attack experiments on computer systems that contain critical data is very undesirable, several alternatives have been used. One alternative consists of setting up a physical computer network absent of any critical data, performing cyber attacks on the network, and collecting data from intrusion detection systems. A second alternative consists of generating synthetic data through the use of simulation.

These two approaches have varying degrees of requirements, capabilities, and limitations. The physical computer network requires the physical machines, networking, and IDS components. Consequently, conducting experiments on various network configurations involving different machines, servers, routing systems, IDS sensors, etc. requires reconfiguration of the network and setting up the network to produce the desired network activity and cyber attacks. The advantage of using the physical network is that the data produced is from a real network as opposed to an abstract representation. This also has some disadvantages in that it is impossible to replicate the experiment exactly, and the data produced is difficult to validate to ensure all desired information is accounted for in the ground truth. Since physical networks are not perfectly reliable, data can be missed, processed incorrectly, etc.

The simulation approach requires knowledge of the desired network and its operation. This information must be captured by the simulation model to represent the behavior of the

network. However, the level of detail included in the model will depend on the goal of the simulation. In this case, the packet level information and computer network traffic details are not needed, so the simulation can be constructed at a higher level to produce alerts caused by cyber attacks and harmless network traffic. Once the framework of the model has been established, various network configurations can be efficiently created and experiments can be conducted with various attack scenarios. Since the simulation experiments are controlled, they can be repeated exactly and all ground truth information is known.

2.9.1.1 Overview of the Simulation Model

A discrete-event simulation model has been developed for generating representative cyber attack and intrusion detection sensor alert data. Although the model is primarily designed to be used in testing cyber situational awareness and analysis tools, other applications such as training of systems analysts may also make effective use of the model. The simulation model was initially implemented in the ARENA simulation software. However, ARENA had several limiting factors that made it difficult to provide the desired level of detail and functionality that was desired of the model. Recent efforts have focused on an object-oriented model written in Java. Figure 19 displays the architecture of the simulation model with respect to its inputs and outputs.

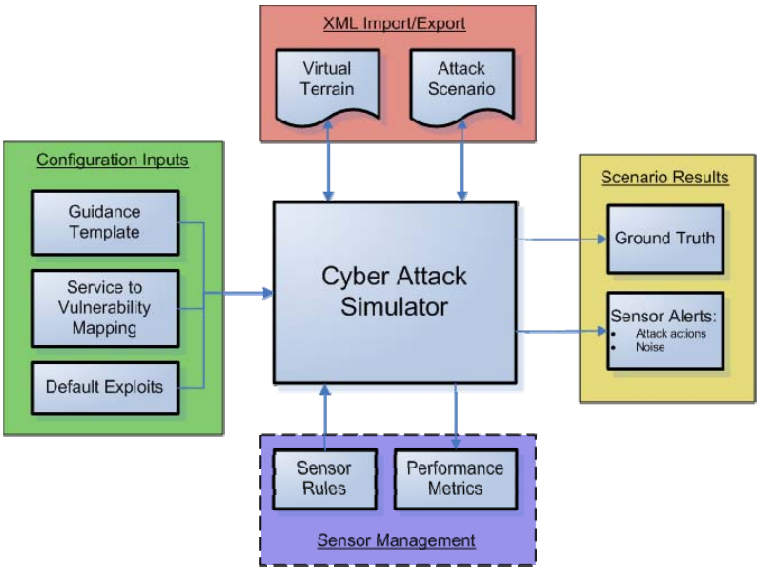


Figure 19: Cyber Attack Simulator Functionality

2.9.1.2 Computer Network Representation

The simulation model provides a user with the ability to construct a representative computer network and setup and execute a series of cyber attacks on certain target machines within that network. IDS sensors that are setup within this network produce appropriate alerts based on the traffic they observe within the network. The alerts produced consist of a combination of the

alerts produced as a result of attack actions and as a result of typical “noise” (non-malicious network traffic that triggers an alert.)

Figure 20 displays an example network interface setup using the Java model. A simple graphical interface is provided to give the user a means to easily create and visualize the network. Simulated computer networks consist of three primary types of devices: machines, connectors, and subnets (machine clusters).

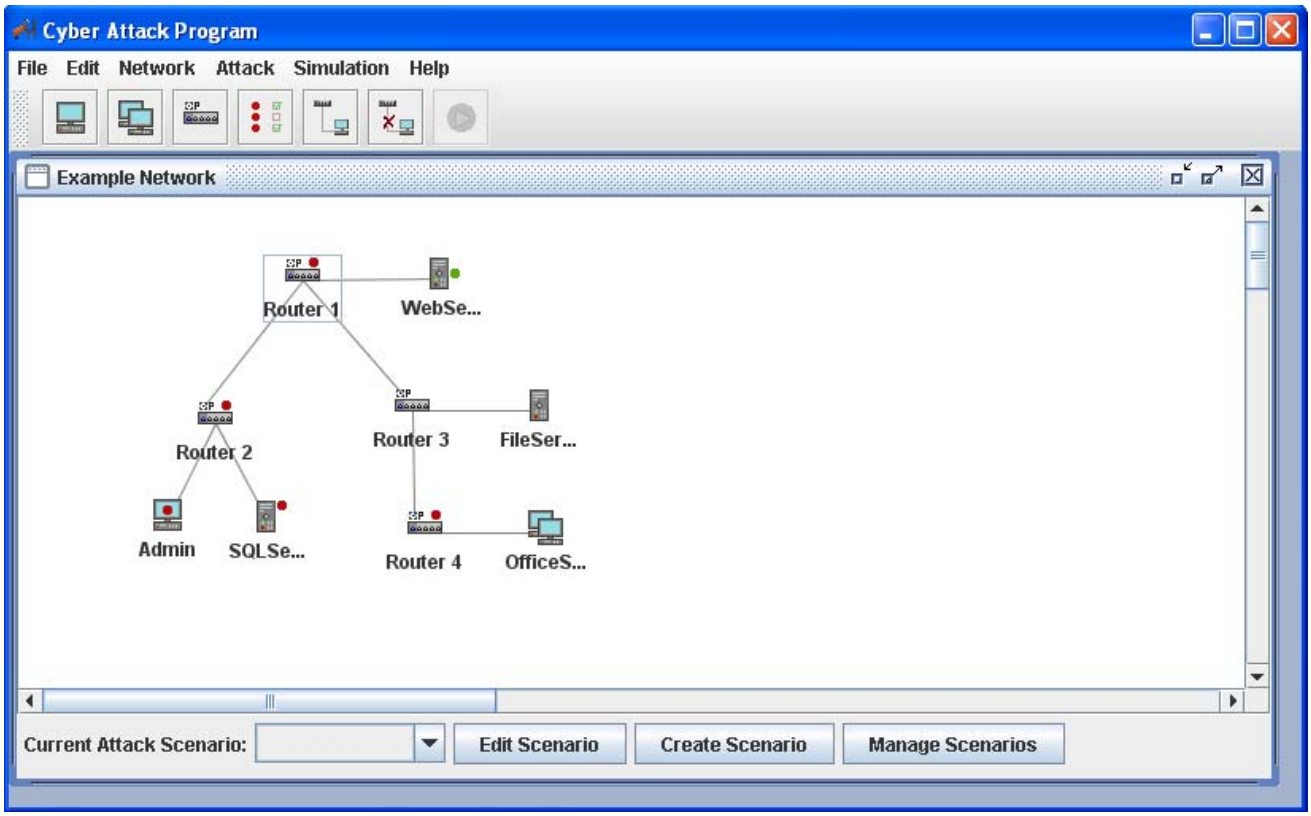


Figure 20: Sample Network Interface in Java Model

A machine can represent a personal computer or a server. Machine characteristics that can be specified include the IP address, the operating system, services running on the machine, and the type of IDS sensor on the machine (if any). For each IDS sensor specified, an associated output file will be generated containing the sequence of alerts produced when the simulation is run.

A connector represents the means by which computers are connected, such as through a switch or a router. The network connectivity plays in important role in establishing the path that an attacker can take through the network. For each path through a connector, a list of allowed or banned ports can be specified. The connector can also contain network IDS sensors, which are

used to monitor any network traffic that travels through the connector and produce alerts corresponding to known potentially harmful actions.

A subnet represents a group of several machines with connectivity to the network that all share a common set of properties (such as the operating system). Machines within a subnet contain the same set of properties that could be specified if the machines were placed into the network individually. The subnet just provides an efficient method of specifying groups of computers (particularly useful when creating large network models).

Connector links are used in the model to connect the modules and represent the connection of machines/subnets to a connector, as well as the connections between connectors themselves.

2.9.1.3 Attack Representation

When a computer network has been created, an attack scenario can be setup and run on the network. An attack scenario consists of a series of specified cyber attacks occurring over a period of time along with a specified quantity of network noise. A user-interface with a series of forms is used to specify the desired scenario. The model structure enables manual or automatic attack generation. In the manual mode, the user can specify all of the details of the attack scenario including the sequence and timing of attack actions as well as the path the attack will take through the computer network. In the automatic mode, the user can specify the goal (ultimate attack action and target computer) of the attack, and the simulation model will generate a random, feasible sequence of attack actions along a path that leads to the goal. Additional parameters that represent the behavior of the attacker can also be specified. These parameters range from 0 to 1 and include the efficiency, stealth, goal focus, and success rate of the attack being modeled. The efficiency refers to how direct the attack is, with 1 representing the most efficient attack path. The stealth parameter refers to how well the attack avoids detection, with 1 being a completely undetectable attack. The goal focus refers to how focused the attack is on achieving the primary goal and avoiding intermediate goals. The probability of success (also considered “skill”) refers to the likelihood of the step succeeding assuming there are no physical (network) impediments.

An attack scenario in the Java model has no limit for the amount of attacks or steps per attack; however, more attacks and steps will result in slower execution. For each type of attack, the user can specify the time between attack steps based on a fixed number or on a random number sampled from an exponential distribution (with a specified mean). The steps/actions available for use in an attack are chosen from a categorized list of 2,004 known exploits in 5 major groups and 23 subgroups. Each exploit has a set of specific properties, which include the exploit signature, the port and protocol used, the machine type and operating system that can be threatened by the exploit, and a vulnerability id. If no specific exploit is selected, one will be chosen at random based on the subgroup. In addition to attacks, the user can specify, the rate

at which non-malicious traffic alerts (noise) is generated, as well as the probability of noise alerts corresponding to each of the action categories.

Once the scenario has been created, the attack information can be saved (along with the network) for future use. The simulation is then run, and the attack scenario is executed. This includes initializing and/or generating the attack steps, simulating the movement of attack and noise traffic through the network, and outputting results. The output of the simulation includes a file listing the actions generated for each attack (known as the "ground truth") and the time the action occurred. In addition, an output file containing IDS alerts is produced for each IDS sensor specified in the modeled network. These files containing IDS alerts are intended to be used to test the situational awareness and analysis tools.

2.9.2 Simulation Methodology

This section discusses in detail the general approaches taken in modeling computer networks, modeling cyber attacks, and simulating cyber attacks and generating corresponding IDS data.

2.9.2.1 Modeling Computer Networks

As described in the previous section, the computer network is modeled using two basic constructs: machines and connectors. The third construct, subnets, represents a group of machines. The modules representing the machines, connectors, and subnets provide a visual representation of the computer network. However, functionally, these modules provide a logical method for the user to enter the data about the computer network including whether the machine can be accessed externally from the Internet. The connecting lines showing the connectivity of the network are used to construct links between the network devices and form the network topology that will be used in the attack generation. The details of the devices (such as the type of IDS) are attributes that can be easily viewed and modified by double-clicking their corresponding representation in the interface to bring up a form to enter or change information.

2.9.2.2 Modeling Cyber Attacks

The simulator allows for both cyber attacks that are initiated by a hacker through the Internet and insider attacks. The progress that a hacker can make in an attack is dependent upon the hacker's capabilities and the vulnerabilities of the network. The methods for modeling and simulating the initiation and progression of cyber attacks through a computer network included in this model are based on Stotz et al [2].

Stotz et al [2] place the sequence of attack actions that a hacker may use into stages that correspond to the hacker's capabilities given the current state of the network. These stages are referred to as Stage 0 through Stage 9. Stage 0 represents reconnaissance activities on the external part of the computer network where the attacker is using exploits to simply gain more

information about the network. Stage 0 – Stage 4 represent hacker actions on external machines, and Stage 5-Stage 9 represent hacker actions on internal machines. Table 14 lists some typical hacker actions that correspond to an attack stage.

Table 14: Typical Hacker Actions in a Cyber Attack

Stage	Typical Action
0	Recon Footprinting (External)
1	Intrusion User (External)
2	Escalation Service (External)
3	Intrusion Root (External)
4	Goal Denial of Service (External)
5	Recon Enumeration (Internal)
6	Intrusion User (Internal)
7	Escalation service (Internal)
8	Intrusion Root (Internal)
9	Goal Pilfering (Internal)

Initially, a hacker can attack an organization's machine on the external portion of the network. Once the external machine has been successfully compromised, the hacker can work their way through the external network until the capability to access internal machines is reached. Once the hacker has infiltrated the internal network, the internal machines can be compromised until the hacker reaches their goal. Figure 21 illustrates the cyber attack process from the internet to a goal on an internal machine.

The simulation model includes automated and user-specified cyber attack generation methods. The automated method utilizes the network specifications and connectivity in combination with a guidance template of the available stages to determine the capabilities of the attacker and vulnerabilities of the network and generate a feasible sequence of attack steps for the cyber attacks. The graph-based guidance template is used to determine which groups of actions (stages) are feasible at different points of the attack. A diagram of the graph-based template that the simulation model currently operates under is shown in Figure 22. The graph is a directed graph, which means that an edge (arc) indicates a feasible transition in the direction that the edge is pointing. Nodes within the same group form a complete graph in which each

node is connected to every other node. This graph-based template is represented as an adjacency matrix of 1's and 0's representing which stages are accessible after which other stages have been performed.

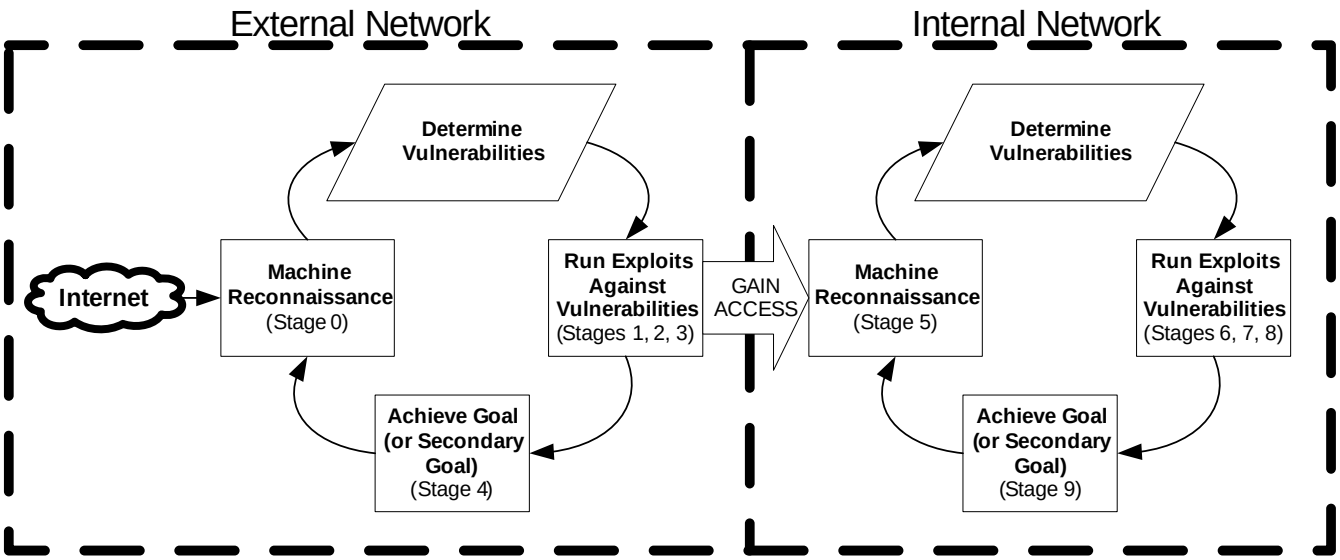


Figure 21: Progression of a Cyber Attack on a Computer Network from the Internet

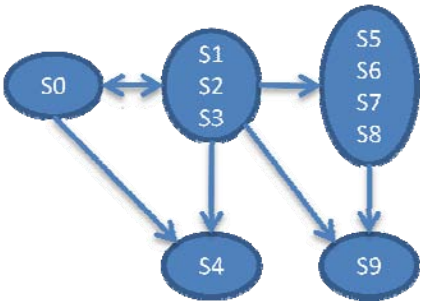


Figure 22: Directed Graph Representing Attack Structure

Given the attack structure (in the form of the guidance template) and the network configuration specified, the user is able to specify a target machine, a goal, and several other attack related parameters through a series of forms. Figure 23 illustrates the automated method that is used to generate the specific multi-stage attack.

In generating the steps (prior to simulating them over a time period), the methodology first defines the attack's final target and goal and maps out a critical path to the target machine. The logic used to generate the sequence of steps first starts at an external (internet) source and locates a machine with external access. The attacker IP address for attacks on external

machines is created randomly since hackers will generally “spoof,” or disguise, their IP address when attacking from the Internet. An attack progression for this external target is determined, and appropriate attack steps are generated that will give the attacker access to the internal network (using Stage 0 through Stage 4). Once an external machine is accessed, the options for the new target are machines with which the current attacker can communicate. If a sampled random value does not exceed the attack efficiency, then the attack chooses a new target that will move the attack to a lower level of the network topology (toward the final target machine) along the critical path. If the efficiency is exceeded, then the new target will be on the same level or on a branch off of the critical path. The logic will then repeat the steps for determining guidance template progression and generating the steps, using Stage 5 through Stage 9. This process continues, moving the attack through the network topology, until the final target machine is reached. At each targeted machine, another sampled value is compared to the goal parameter. If the goal value is exceeded, an additional (intermediate) goal step is setup. Once the attack generation process reaches the target machine, an attack step is generated using the final goal, and the attack is complete. This entire process is used for each attack within the attack scenario that requires automated attack generation.

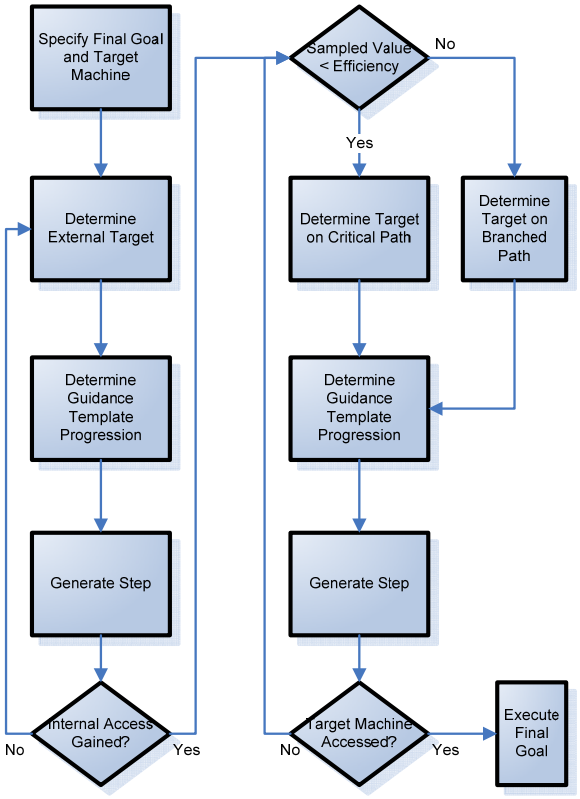


Figure 23: Automated Attack Generation Method

Alternatively, if the user prefers to specify the specific steps of the attack, various levels of automation are provided down to attacks that can be fully specified by the user.

2.9.2.3 Simulating Cyber Attacks and Generating IDS Sensor Alert Data

The general modeling approach for representing the cyber attacks is to model the individual attacks as a collection of attack steps, with the attack steps modeled as entities. One entity for each attack is created at the beginning of the simulation to represent the first step of the attack. These entities are kept on an event order list which manages the order in which the simulation events occur. Each attack only has one active entity at any given time. The event order list keeps track of the simulation time and includes both attack steps and noise with no distinction between the two (both are considered traffic entities). This is necessary to reduce any bias within the simulation. The noise represents false alarms produced by ordinary network activity. The occurrence rate of noise alerts are specified by the user, and generated via a Poisson arrival process.

Each traffic entity is executed one-at-a-time, based on its start time, and routed between the appropriate nodes in the network. Figure 24 displays the methodology used in processing entities on the event order list. In the model, nodes are the abstract representation of both connectors and machines. After the traffic entity is properly routed between all of the nodes from source to destination, the entity is removed from the event order list and the next entity is added. For attacks, the next entity refers to the very next attack step. For noise, the next entity is a randomly generated action between two machines in the network.

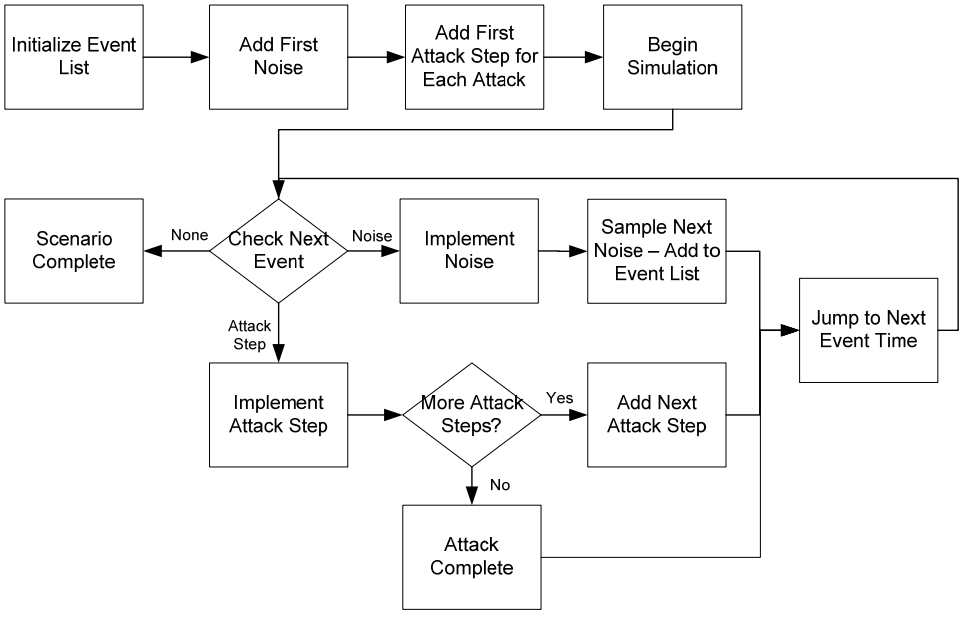


Figure 24: Discrete Event Simulation of an Attack Scenario

When traffic entities are routed to a node in the network, the success of the referenced action is evaluated by comparing the action attributes with the machine or connector details. For a

machine, the type, operating system, and services are of primary concern. The type and operating system must match, and the vulnerability id of the action must be linked with one of the machine's services. Otherwise, the action will fail. For a connector, a list of port permissions along the path that entity takes is the main criteria. Either the action's port and protocol must be allowed by the connector, or the service linked with the action must have its port and protocol allowed by the connector. If the action succeeds and it corresponds with the final goal of an attack, then the attack is considered complete. This entity/event handling process is repeated until no entities are remaining in the event order list.

Also, for each node device that includes an IDS, the appropriate alert information is written to output files. These files include ground truth files and IDS alert files. A ground truth file contains a listing of the hacker attack actions that were executed during each attack as well as an indication of whether each action was successful or not. The IDS alert files include formatted alerts which are dependent on the location of the IDS in the network. The IDS alert files include alerts produced from both attack actions and noise and are representative of the information that a system administrator may receive when using IDSs to monitor network activity.

2.9.3 Cyber Attack Example

This section includes two example attacks that use the automated attack generation process within a fictitious network. The goal of the first attack is to create a backdoor on a machine in the BPN Group subnet, while the goal of the second attack is to perform pilfering on a machine in the Research Group 1 subnet. For both attacks, information is gathered about the external network, and then the VPN server is penetrated. The server is then used as a stepping stone to reach the target machine. Since the automatic attack generator is used, the attack steps are not entered by the user, but instead are generated during the simulation.

The computer network created using the Cyber Attack Simulator is shown in Figure 25, with a summary as follows:

- 1 main web-server;
- 3 main subnet domains;
- 2 subnet domains have further subnets attached;
- Only one external machine; and
- Red dots indicate IDS sensor presence

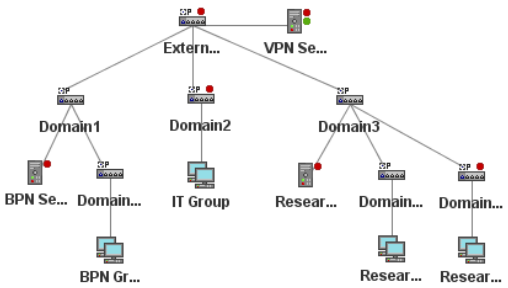


Figure 25: Sample Network

Table 15 illustrates the attack information provided via the auto-attack user interface. The scenario has 150 noise alerts per hour on average where 85% of the noise is reconnaissance, 10% is escalation, and 5% is classified as miscellaneous. This simulation run results in the generation of the steps shown in

Table 16, considered the ground truth. The steps are sorted by the time at which they occur.

Table 15: Auto Attack Parameters

Attack	Target	Goal Type	Efficiency	Stealth	Skill	Delay	Step Time
Attack 1	192.168.5.11	Backdoor	0.6	1.0	1.0	2	3
Attack 2	192.168.6.41	Pilfering	0.4	0.8	0.9	5	4

Table 16: Auto Attack Steps Generated

Attack	Step	Group	Subgroup	Action/Exploit	Source IP	Target IP	Success?
1	1	Recon	Footprinting	SCAN cybercop os PA12 attempt	252.38.110.189	192.168.1.1	Success
1	2	Recon	Enumeration	WEB-IIS /iisadmpwd/aexp2.htr access	174.233.163.182	192.168.1.1	Success
1	3	Intrusion	Other	WEB-CLIENT readme.eml download attempt	36.22.133.111	192.168.1.1	Success
2	1	Intrusion	Other	INFO Connection Closed MSG from Port 80	44.235.206.153	192.168.1.1	Success
1	4	Escalation	Service	EXPLOIT x86 windows MailMax overflow	192.168.1.1	192.168.1.1	Success
1	5	Intrusion	Other	INFO Connection Closed MSG from Port 80	192.168.1.1	192.168.1.1	Success
1	6	Escalation	Service	MS-SQL xp_proxiedmetadata possible buffer overflow	192.168.1.1	192.168.2.1	Success
2	2	Escalation	Service	WEB-COLDFUSION expeval access	205.68.170.171	192.168.1.1	Success
1	7	Intrusion	Other	WEB-CLIENT readme.eml download attempt	192.168.2.1	192.168.2.1	Success
2	3	Intrusion	Root	WEB-CGI pals-cgi arbitrary file access attempt	192.168.1.1	192.168.4.1	Success
1	8	Escalation	Service	DNS EXPLOIT named overflow attempt	192.168.2.1	192.168.5.11	Success
1	9	Goal	Backdoor	BACKDOOR ACKcmdC trojan scan	192.168.5.11	192.168.5.11	Success
2	4	Misc	Other	POLICY HP JetDirect LCD modification attempt	192.168.4.1	192.168.4.1	Success
2	5	Misc	Other	POLICY HP JetDirect LCD modification attempt	192.168.4.1	192.168.4.1	Success
2	6	Recon	Enumeration	WEB-IIS /exchange/root.asp access	192.168.4.1	192.168.4.1	Success
2	7	Misc	Other	POLICY HP JetDirect LCD modification attempt	192.168.4.1	192.168.4.1	Success
2	8	Recon	Enumeration	WEB-MISC ion-p access	192.168.4.1	192.168.6.91	Success
2	9	Goal	Backdoor	BACKDOOR MISC Linux rootkit attempt lrkr0x	192.168.4.1	192.168.6.41	Success

The combination of these attack steps and the noise within the network create a large number of IDS sensor alerts during the simulation run. Two sample alerts produced are displayed in Figure 26.

```
0:17:17.22 [**] [1:1222:1] WEB-CGI pals-cgi arbitrary file access attempt [**] [Classification: web-application-attack] [Priority:6] {TCP} 192.168.1.1 -> 192.168.4.1

0:22:31.98 [**] [1:510:1] POLICY HP JetDirect LCD modification attempt [**] [Classification: null] [Priority:0] {TCP} 192.168.4.1 -> 192.168.4.1
```

Figure 26: Excerpt from generated file of Snort alerts

2.9.4 Conclusions and Future Work

The Cyber Attack Simulator is capable of generating IDS alert and ground truth files based on the specification of a computer network and attacks. The simulator is built with a user interface to allow the creation of various computer network configurations and attack actions. The model also incorporates a method for automated attack generation given the hacker capabilities and the vulnerabilities of the network. Current and future work entails interfacing with information fusion tools, and validation and improvements to the model to represent hacker behavior as accurately as possible.

2.10 Evaluating Fusion System Performance

This section outlines various performance results when testing FuSIA over a set of 10 scenarios created by the Cyber Attack Simulator using the example network in Figure 27. A key problem in assessing the performance is that there are currently not any metrics defined to evaluate a level 2/3 system. TANDI did develop some metrics to test the projection algorithms, which will be used in this analysis. This analysis will only focus on the performance in terms of speed and projection accuracy. It will not focus on assessing the accuracy of the impact scores since it is unclear how to determine such accuracy since it cannot be validated.

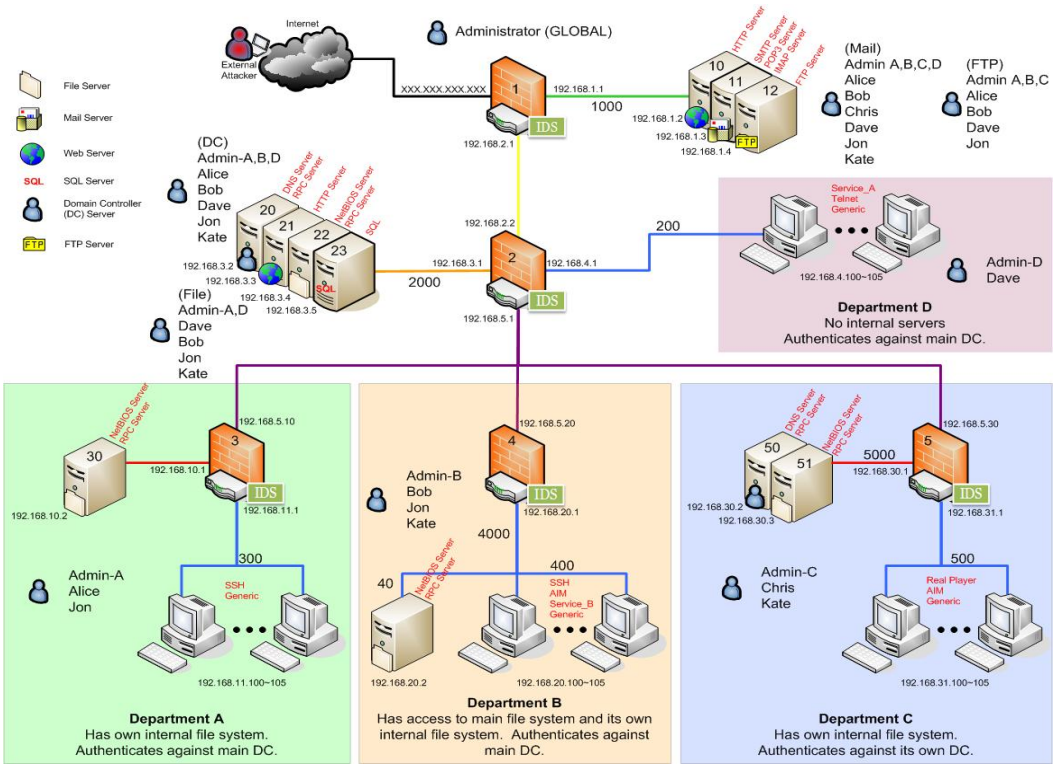


Figure 27: Network used to evaluate fusion system performance. This is the same network as depicted in Figure 3.

Table 17 shows the average processing speed of FuSIA's situation estimation and projection capabilities. Included in FuSIA is the ability to calculate the metrics discussed in this section across many different dimensions. The simulations suggest that FuSIA is able to estimate and project situations at a rate of 277 alerts (or observables) per second with no metrics being calculated and 142 alerts per second with the metrics being calculated. It should be noted that there is a large number of redundant calculations taking place in the metrics, so the metric capturing feature of FuSIA could be optimized greatly. A current effort, ECCARS, is seeking to integrate FuSIA into a cyber security solution. In addition to integration, the optimization of FuSIA code will be a main task since the integration strategy is likely to decrease the current processing rate of FuSIA.

Table 17: Current Estimated FuSIA Alert Processing Rate Running Capability and Opportunity Algorithms

Scenario	2A	2B	2C	2D	2E	2F	2G	2H	2I	2J	Average
Without metrics	263	280	286	290	329	255	268	281	310	331	**277 alerts/sec**
With metrics	138	139	145	155	158	138	125	131	132	158	**142 alerts/sec**

Let S_t be a set of scores (e.g. plausibility scores, reliabilities, impact scores) at time *t* such that

$$range\{s_t^i\} = \{[0,1], unknown\}, \forall i \in [1, |S_t|]$$

Let R be a set of objects such that r_t is an index of a score of interest for that object at time t. Let n_t be the normalized score at time t for r_t:

$$n_t = \begin{cases} \frac{s_t^{r_t}}{\sum_{i=1}^{|S|} s_t^i}, & 0 \leq s_t^{r_t} \leq 1 \\ unknown, & otherwise \end{cases} \quad (25)$$

We can then calculate the following statistics (which are consistent with the definitions set forth by TANDI):

Table 18: Example Projection Metrics

Metric	Calculation
Average Normalized Score (ANS)	$avg(n_t)$
Average Score Percentile (Upper & Lower)	$avg\left[percentile_{S_t}\left(s_t^{r_t}\right)\right]$

The Average Normalized Score (ANS) is the average of the scores a single activity prior to that object being compromised. The Average Score Percentile captures a relative ranking of that score with respect to the other scores. Neither of these metrics, however, can *individually* give a true estimate of accuracy. For example, using an extreme case, assume that all scores were the same (or very close to each other). The ANS and Upper Percentile would be 1, while the lower percentile would be 0. These results show that there is no real distinction between the scores. The main point of this example is that, like ranking situations and plausibility scores, there may not be one single metric to illustrate accuracy.

Table 19 shows the results for the original (Rule-based) and revised (statistical) implementation of the capability algorithm across the 10 simulated scenarios. In this case, all metrics are very low, which obviously indicates a very poor projection capability of the original capability algorithm. Notice that the revised capability algorithm outperformed the original algorithm in all respects. The Average Normalized Plausibility score was only 0.456, but the percentile values were relatively high. This indicates that even though the Normalized Plausibility Scores were low in general, the scores are still ranked relatively high compared to the other values.

Table 19: Original Capability Performance Metrics at Host Level for all Combined Scenarios

	Original Capability	Revised Capability
Average Normalized Plausibility Score	0.248	0.456
Avg Normalized Reliability Score	0.134	0.955
% Unprojected	0.851	0.005
Avg Plausibility Score Percentile	[0.14,0.15]	[0.75,0.80]
Avg Reliability Score Percentile	[0.13,0.15]	[0.34,0.38]

Table 20 shows the projection performance of the revised opportunity and capability algorithms as well as the fused projection score using the SMMR Dempster-Shafer combination rule. It should be noted that due to the fusion of the scores, the fused reliability of the projection was higher than the individual algorithms. Likewise, the fused percentile score indicates that the average plausibility score was in the top 30% of the scores.

Table 20: Current Performance Metrics at Host Level for all Combined Scenarios

	Fused	Opportunity	Capability
ANS	0.498	0.686	0.456
Avg Reliability	0.978	0.713	0.955
% Unprojected	0.000	0.000	0.005
Percentile	[0.70 0.74]	[0.53 0.87]	[0.75,0.80]
Reliability Percentile	[0.63,0.66]	[0.61,0.87]	[0.34,0.38]

2.11 SNePS Integration and Improvements

The previous section described the process by which states were assigned to a given object of interest. This process utilized the virtual terrain in order to determine the state of the object. Before the object states were calculated, one of the tasks in N-CMIF was to simply determine if an attack was successful or unsuccessful. This process actually corresponds to steps 1 and 2 in the previous section. Step 3 was added once the need for determining object state was identified. While the success algorithm was implemented in Java, it was also explored how a logical reasoner such as Semantic Network Processing System (SNePS) would be used to

perform the same task. This section details the work accomplished in developing the success algorithm within SNePS as well as detailing the improvements made to SNePS to improve processing speed.

2.11.1 Overview of SNePS

SNePS is a knowledge representation, reasoning, and acting system that is well suited for representing the reasoning processes of a SME. Some general capabilities that SNePS possesses that can aid in the process are:

- higher-order logical representation of facts and rules;
- backwards and forward inference;
- contradiction detection;
- acting;
- and procedural attachment.

Through the use of higher-order logic SNePS can represent any background knowledge or information sources the SME utilizes when reasoning about network abnormalities, and through the use of rules, any inferential processes needed to draw conclusions about that knowledge and data. However, this process does require translation (and typically the intercession of a knowledge engineer) although graphical user interfaces (GUI) can help remove the knowledge engineer can help remove the knowledge engineer from the process.

SNePS employs a reasoning engine that is capable of performing backwards inference, a form of goal-directed or query driven reasoning, and forward inference, a process of deriving conclusions from new knowledge. Both are useful for reasoning in the cyber security domain. Backwards, because given the existing knowledge about the network and abnormalities one needs to query the reasoner about the abnormality (e.g., a query assessing whether the abnormality is an attack), and forward, because new information is constantly being processed by the reasoner. As part of its reasoning engine, SNePS also can detect contradictions made when reasoning or adding new information to the knowledge base, and then correct the issue with user input.

To help capture the ability of a SME to handle attacks when they are detected, a method for representing and executing planned acts is necessary. SNePS processes an acting system that is integrated with the reasoning system [32]. With this ability a variety of conditional plans can be represented, such that, when the system reasons to some conclusion it can act on the results appropriately (e.g., when an abnormality is detected and reasoned to be an attack, then execute a procedure for notifying the appropriate security analysts).

Finally, procedural attachment is available in SNePS for efficient computation of mathematical predicates, or for performing look-ups in huge data repositories, a feature necessary for reasoning in the cyber security domain, which relies on vast vulnerability databases.

2.11.2 Example Cyber Reasoning in SNePS

An attack track is a report that alleges that some network device, referred to as the target, was attacked, that the attack originated from some source device, and that it exploited some vulnerability of the target. The vulnerability is identified by a signature identifier. (SID). One problem is that some attack tracks report attacks that didn't really happen, or were incidents that did not really constitute attacks. SNePS is being used to try to identify such false positives. Various procedures and knowledge required for identifying attack tracks as false positives were elicited from an SME. As a result, two forms of reasoning were chosen for implementation.

2.11.2.1 Vulnerability Correspondence

The first form of reasoning elicited from a SME involves determining whether the SID of the attack track indeed corresponds with a known vulnerability of the network. The network vulnerabilities are provided as terrain data, which are translated into SNePS by an automatic interpreter written for this purpose. If the SID does not correspond to any vulnerability, then the attack is deemed improbable, and the attack track is flagged as a false positive. Unfortunately, the network terrain data give the vulnerabilities of the network hosts as Common Vulnerabilities and Exposure (CVE) [9] identifiers, which are different from the SIDs reported in the attack tracks. (The terrain data also includes exposure vulnerabilities for hosts, which are basically the English names of the exposures, and don't use a particular identification scheme like CVE or SID.) There is a CVE repository that provides a correspondence between CVEs and Bugtraq identifiers (BIDs), a third form of vulnerability labeling. Finally, files of Snort [13] sensor rules provide a correspondence between BIDs and SIDs.

This technique of identifying false positives is:

1. The attack track, *a* reports that the target device *n* was attacked via its vulnerability *sid*.
2. Use the terrain data to find the CVE identifiers of vulnerabilities that *n* is subject to.
3. Look up the CVE identifiers in the CVE repository to find their corresponding BIDs.
4. Use the SNORT files to determine if any of those BIDs corresponds to *sid*.
5. If not, then *a* is a false positive.

The SID-BID-CVE connection is represented in SNePS with the following two predicates.

- CVE BID Equiv(*cve*,*bid*)
  CVE *cve* denotes the same vulnerability as BID *bid*. (From the CVE repository.)
- SID BID Equiv(*sid*,*bid*)
  SID *sid* denotes the same vulnerability as BID *bid*. (From the SNORT files.)

All the instances of these relations are not stored in the SNePS knowledge base ahead of time. Instead, procedural attachment is used to look up the required information from the appropriate repository when needed.

In addition, the following SNePS predicates are also used.

- PropertyValue(*n,p,v*)
  The property *p* (either CVE or BID) of network device *n* is *v*.
- NetworkDevice(*n,nid,nip*)
  The network id of network device *n* is *nid* and its IP address is *nip*.
- Alert(*a,ip,sid*)
  Attack track *a* reports that the target device that has the IP address *ip* was attacked via a vulnerability with the SID *sid*.
- CVE(*cinst,cid*)
  cinst is an instance of the CVE with the id *cid*.
- PartOf(*v,n*)
  Network device *n* has the particular vulnerability or exposure *v*.
- PossibleAttackSID(*a,sid*)
  Attack track *a* reported an attack via a vulnerability *sid* that exists on the specified host.

Using these predicates, the rules for determining that an attack track is a false positive are:

- If a network device, *ninst,* has a particular instance of a vulnerability type with identifier, *cid*, then we say that *ninst*'s CVE is *cid*.

```
all(ninst,nid,nip)(NetworkDevice(ninst,nid,nip)
                   => all(cinst,cid)(CVE(cinst,cid)
                         => (PartOf(cinst,ninst)
                               => PropertyValue(ninst,CVE,cid))))
```

- A network device has the BID *bid* that corresponds to every CVE id that it has.

```
all(ninst,nid,nip)(NetworkDevice(ninst,nid,nip)
                   => all(cid)(PropertyValue(ninst,CVE,cid)
                         => all(bid)(CVE_BID_Equiv(cid,bid)
                               => PropertyValue(ninst,BID,bid))))
```

- An attack track is possible if its SID corresponds to any BID (or CVE) its alleged target has.

```
all(a,sid,nip)(Alert(a,nip,sid)
                   => all(n,nid)(NetworkDevice(n,nid,nip)
                         => all(bid)(PropertyValue(n,BID,bid)
                               => (SID_BID_Equiv(sid,bid)
                                     => {PossibleAttackSID(a,sid)})))).
```

2.11.2.2 Firewall Rules

Even if the target device does have the vulnerability which the attack track reported that the source exploited, the attack track could be a false positive if there was no way for the source to communicate with the target. This is the second form of reasoning elicited from the SME, and uses connectivity rules for the network -- what network devices are allowed access to which others on specific ports. These rules are supplied by the network terrain data in addition to the vulnerability information. The basic relation is represented by the SNePS predicate,

- `ConnectedByPort(`*`src,dst,port,prot`*`)`
  The source network device, *`src`*, is connected to the destination network device, *`dst`*, on port *`port`* using protocol *`prot`*.

This relation is not directly supplied by the network terrain data, but can be inferred from the data that is supplied. See [33] for the rule that is used. Nevertheless, it still only specifies network devices that are directly connected. Clearly, devices can be indirectly connected. Rather than using standard logical inference to derive these indirect connections, we use SNePS's path-based inference. A SNePS proposition is represented by a slot-filler frame, where the slots specify the argument positions and the fillers are the arguments in those positions. For example, the proposition

```
ConnectedByPort(nd1,nd2,p1,tcp)
```

is represented by the frame

```
(src (nd1) dst (nd2) port (p1) prot (tcp))
```

A SNePS knowledge base can also be viewed as a directed labeled graph, in which each frame is a node and each slot labels a directed arc from the frame it is in to each of the fillers. Path-based inference involves inferring an arc labeled r from a node n to a node m whenever a certain path goes from n to m [34]. Following paths in a SNePS network is more efficient than using normal logical inference, but is not always applicable. It is particularly appropriate, however, for reasoning about transitive relations, which connectedness is, being the transitive closure of `ConnectedByPort`.

To specify that the presence of an arc may be inferred from the presence of a path, a path-based inference rule is specified. We will not explain the syntax of path-based inference rules here. (See [36] for the details.) However, the rules for the relations `src` and `dst` are:

- If *a* has *b* as a src, and *b* has *c* as a src, then *a* has *c* as a src.
  If *a* does not have *c* as a src, but *b* has *c* as a src, then *a* does not have *b* as a src.

```
define-path src
      (or src
            (compose ! src (kstar (compose dst- ! src)))
            (domain-restrict ((compose arg- ! max) 0)
                        (compose src
```

```
                                    (kstar (compose src- ! dst)))))
```

- If *a* has *b* as a dst, and *b* has *c* as a dst, then *a* has *c* as a dst. If *a* does not have *c* as a dst, but *b* has *c* as a dst, then *a* does not have *b* as a dst.

```
define-path dst
      (or dst
            (compose ! dst (kstar (compose src- ! dst)))
            (domain-restrict ((compose arg- ! max) 0)
                  (compose dst
                        (kstar (compose dst- ! src)))))
```

A key part of both these rules is the path constructor `(kstar p)`, which means zero or more occurrences of the path *p*, and allows an arc to be implied by a path of arbitrary length.

2.11.2.3 Using SNePS Reasoning

With the above rules and background knowledge in place SNePS can be plugged into the automatic cyber security management system. This is handled through the use of a SNePS executable and a Java class that provides `tell/ask` methods for interfacing with the executable. Both the executable and Java interface were developed for this project utilizing the ACL JLinker libraries [29] and executable generator. The Java interface provides two methods to the overall system:

- `public InferdSnepsAPI(String snepsExePath, int interfacePort)`
  A constructor that starts and returns a connection to the SNePS executable.
- `public double doFalsePosCheck(Document doc)`
  Returns a double value that represents a false positive measure for the given attack track doc. Currently, the possible values are 0.5 or 0.0, the former indicating a false positive, the latter indicating that a false positive couldn't be determined.

The most important method is `doFalsePosCheck`, which works by asserting attack track information into SNePS, and then querying the system based on the new information. The algorithm is as follows.

1. Given attack track *a* with:
   - Source IP: *sip*
   - Target IP: *tip*
   - Port: *port*
   - Protocol: *prot*
   - SID: *sid*
   - Exposure: *exp*
2. Assert into SNePS the alert information, `Alert(`*a*`,`*tip*`,`*sid*`)`.

3. Query SNePS to determine if it contains any information about the network devices with those identifiers (e.g., `NetworkDevice(?n,?nid,sip)?`), the exposure (e.g. `Exposure(?e, exp)?`), or if the exposure is known to be on the host in question (e.g., `PartOf(eid,tid)?`, where `eid` and `tid` are the identifiers for the exposure and target network device in the knowledge base). If not, return 0.5.
4. Query SNePS to determine if the target host has the vulnerability specified (e.g., `PossibleAttackSID(a,sid)?`). If not, return 0.5.
5. Query SNePS to determine if the two devices are connected (e.g., `ConnectedByPort(sid,tid,port,prot)?`, where `sid` and `tid` are the network identifiers for these devices. This information is acquired from the query made in step (2)). If not, return 0.5.
6. Return 0.0.

2.11.2.4 Results

The algorithm was tested on 6 cases, one test for each possible output. All outputs were as expected given the input files. The reasoner was deemed to be unacceptably slow when running with the rest of the system on a Windows desktop computer, but ran significantly faster on a Department of Computer Science and Engineering computer, nickelback, a Sun Sunfire X4200 with 8.0 GB of main memory and two Dual Core AMD Opteron™ Processor 285s, clocked at 2592 MHz, running RedHat Enterprise Linux 4 (64-bit).

Following are outputs generated by the 6 runs of the system (each `.xml` file contains an attack track).

```
1. Enter an alert file to parse: _1.xml
   Performing false pos check on:
   Alert[source ip: 131.46.41.51
         target ip: 192.168.1.200
         protocol: tcp
         port: 445
         signature: NETBIOS SMB-DS IPC$ unicode share access
         sid: 2559]
   Determining if this host is in the virtual terrain...
   False positive: No target host found with specified IP:192.168.1.200

2. Enter an alert file to parse: _2.xml
   Performing false pos check on:
   Alert[source ip: 100.10.20.4
         target ip: 192.168.1.2
         protocol: icmp
         port: unknown
         signature: ICMP L3retriever Ping
         sid: 466]
   Determining if this host is in the virtual terrain...
   Determining if the signature is on any host in the virtual terrain...
   False positive: Exposure(ICMP L3retriever Ping) does not reference an
   exposure in the knowledge base. No host is known to have it.
```

```
3. Enter an alert file to parse: _3.xml
   Performing false pos check on:
   Alert[source ip: 192.168.1.3
         target ip: 192.168.10.2
         protocol: tcp
         port: 445
         signature: MS-SQL xp_showcolv possible buffer overflow
         sid: 2466]
   Determining if this host is in the virtual terrain...
   Determining if the signature is on any host in the virtual terrain...
   Determining if the signature corresponds to a vulnerability on this
   host...
   False positive: Exposure(MS-SQL xp_showcolv possible buffer
   overflow)does not correspond to a vulnerability on host 192.168.10.2

4. Enter an alert file to parse: _4.xml
   Performing false pos check on:
   Alert[source ip: 100.10.20.9
         target ip: 192.168.1.3
         protocol: tcp
         port: 25
         signature: SMTP RCPT TO overflow
         sid: 252]
   Determining if this host is in the virtual terrain...
   Determining if the signature is on any host in the virtual terrain...
   Determining if the signature corresponds to a vulnerability on this
   host...
   Determining if SID corresponds to a CVE vulnerability...
   False Positive: SID 252 does not correspond to a CVE vulnerability on
   192.168.1.3

5. Enter an alert file to parse: _11.xml
   Performing false pos check on:
   Alert[source ip: 100.10.20.9
         target ip: 192.168.1.2
         protocol: icmp
         port: unknown
         signature: WEB-MISC bad HTTP/1.1 request
         sid: 1650]
   Determining if this host is in the virtual terrain...
   Determining if the signature is on any host in the virtual terrain...
   Determining if the signature corresponds to a vulnerability on this
   host...
   Determining if SID corresponds to a CVE vulnerability...
   Determining if a connection is possible between the source and target,
   according to firewall rules...
   False positive: 100.10.20.9 and 192.168.1.2 are not connected.

6. Enter an alert file to parse: _6.xml
   Performing false pos check on:
   Alert[source ip: 100.10.20.9
   target ip: 192.168.1.2
   protocol: unknown
   port: unknown
   signature: WEB-MISC bad HTTP/1.1 request
   sid: 1650]
   Determining if this host is in the virtual terrain...
   Determining if the signature is on any host in the virtual terrain...
   Determining if the signature corresponds to a vulnerability on this
   host...
   Determining if SID corresponds to a CVE vulnerability...
   Determining if a connection is possible between the source and target,
   according to firewall rules...
   SNePS could not determine a false positive for this attack track.
```

There have been two major criticism levied against the approach discussed above: that SNePS does not provide a service significantly different from that provided by database systems, expert system shells, or ontology reasoners; that SNePS is too slow to be useful. In the cyber security architecture, SNePS serves as a false positive arbiter on the attack tracks generated by the network sensors. The two techniques for determining false positives elicited from the SME did not make use of the expressiveness available in SNePS, which has a much more expressive language than that which is typical of many database systems, expert systems, or ontology reasoners.

The SNePS reasoner was deemed to be too slow when running as part of the overall system on a Windows desktop computer. SNePS was considered important enough to assign us the task of finding some of the reasons for its slowness, and improving it. The success of that task is reported in [33].

SNePS is simultaneously a logic-based, frame-based, and network-based knowledge representation, reasoning and acting system. Its logic-based aspect supports logical reasoning; its frame based aspect supports slot-based reasoning (which wasn't utilized in this project); and its network based aspect supports path-based reasoning. Procedural attachment may be used so that, when instances of certain predicates are required, they can be retrieved from large external data bases or files written in XML or other formats. Although implemented in Common Lisp, SNePS has an API implemented in Java, so that it can be used in a large system with other software packages implemented in other languages. SNePS's ability to interact with programs written in a variety of languages, its ability to use data contained in a variety of file formats, the expressiveness of its knowledge format, and the variety of its reasoning methods make it a useful and valuable component of information fusion systems.

In this project, we used SNePS to represent and reason about network information, attack tracks, and the information in vulnerability databases to aid in the maintenance of cyber security. We elicited and implemented reasoning strategies from a Subject Matter Expert regarding vulnerabilities and firewall settings relevant to the assessment of an abnormality as a network threat or a false positive, and demonstrated the success of this approach.

2.11.3 SNePS Comparison With TopBraid/Pellet

We compare the SNePS knowledge representation and reasoning system, with the Topbraid Ontology Editing Tool [37] using the Pellet OWL DL Reasoner [38]. To compare these two systems we represent two problem domains in each system, and have them reason over the data. The two problem domains are:

- The Jobs Puzzle

- The ParentSally Example

2.11.3.1 The Jobs Puzzle

The Jobs Puzzle is a small logic puzzle involving constraint satisfaction. It is a version of one presented in [39]. The puzzle involves figuring out which of eight jobs each of four people has. The following section discusses the representation of the puzzle's constraints in SNePS and Topbraid, and the results of their respective automated reasoners.

2.11.3.1.1 The Jobs Puzzle in SNePS

To represent the Jobs Puzzle in SNePS, we use SNePS's SNePSLOG, a logical language resembling higher-order logic. The following SNePSLOG terms are used for representing the puzzle:

- `Roberta` - The person named Roberta.
- `Thelma` - The person named Thelma.
- `Steve` - The person named Steve.
- `Pete` - The person named Pete.
- `chef` - The job of chef
- `guard` - The job of guard
- `nurse` - The job of nurse
- `"telephone operator"` - The job of telephone operator
- `"police officer"` - The job of police officer
- `teacher` - The job of teacher
- `actor` - The job of actor
- `boxer` - The job of boxer
- `Male(x)` - The proposition that `x` is a male.
- `Female(x)` - The proposition that `x` is a female.
- `Person(x)` - The proposition that `x` is a person.
- `Job(x)` - The proposition that `x` is a job.
- `HasJob(x,y)` - The proposition that `x` has the job of `y`

The puzzle and its formalization in SNePSLOG is as follows. (Each item shows an English statement from the puzzle, and its translation into SNePSLOG.)

- Roberta, Pete, Thelma, and Steve are people.
  ```
  Person(Roberta, Thelma, Steve, Pete).
  ```
- The jobs are chef, guard, nurse, telephone operator, police officer, teacher, actor, and boxer.
  ```
  Job({chef, guard, nurse, "telephone operator","police
  officer", teacher, actor, boxer}).
  ```

- Each person has exactly two of the eight jobs.
  ```
  all(p)(Person(p) => nexists(2,2,8)(j)(Job(j): HasJob(p,j))).
  ```
- Each job is held by exactly one of the four people.
  ```
  all(j)(Job(j) => nexists(1,1,4)(p)(Person(p): HasJob(p,j))).
  ```
- No female has the job of nurse, actor, or telephone operator.
  ```
  all(w)(Female(w)=> andor(0,0) {HasJob(w, nurse), HasJob(w,
  actor), HasJob(w, "telephone operator")}).
  ```
  This is formulated as .A female is not a nurse, nor an actor, nor a telephone operator., instead of as .The nurse, the actor, and the telephone operator are males. so that SNePS can perform direct reasoning to conclusions of the form `HasJob(`*p*`,`*j*`)` or `~HasJob(`*p*`,`*j*`)` without using the rule of inference of *modus tollens*, which is not implemented in SNePS.
- No male has the job of chef.
  ```
  all(m)(Male(m) => ~HasJob(m, chef)).
  ```
- No person is both the chef and the police officer.
  ```
  all(p)(Person(p) => andor(0,1){HasJob(p, chef), HasJob(p,
  "police officer")}).
  ```
- Roberta and Thelma are female.
  ```
  Female(Roberta, Thelma).
  ```
- Steve and Pete are male.
  ```
  Male(Steve, Pete).
  ```
- Roberta is neither the boxer, nor the chef, nor the police officer.
  ```
  andor(0,0){HasJob(Roberta, boxer), HasJob(Roberta, chef),
  HasJob(Roberta, "police officer")}.
  ```
- Pete is neither the nurse, the police officer, nor the teacher.
  ```
  andor(0,0) {HasJob(Pete, nurse),HasJob(Pete, "police
  officer"), HasJob(Pete, teacher)}.
  ```

With all the constraints specified, we query the system for the job assignments:

```
: HasJob(?p,?j)?wff108!: HasJob(Thelma,boxer)
wff107!: ~HasJob(Thelma,guard)
wff105!: ~HasJob(Thelma,teacher)
wff103!: ~HasJob(Pete,boxer)
wff101!: ~HasJob(Pete,guard)
wff98!: HasJob(Pete,telephone operator)
wff96!: HasJob(Pete,actor)
wff95!: ~HasJob(Steve,boxer)
wff93!: ~HasJob(Steve,guard)
wff91!: ~HasJob(Steve,teacher)
wff89!: ~HasJob(Steve,telephone operator)
wff87!: ~HasJob(Steve,actor)
wff84!: HasJob(Steve,nurse)
```

```
wff82!: HasJob(Roberta,guard)
wff80!: HasJob(Roberta,teacher)
wff78!: ~HasJob(Thelma,police officer)
wff76!: ~HasJob(Steve,chef)
wff75!: ~HasJob(Pete,chef)
wff72!: ~HasJob(Roberta,nurse)
wff71!: ~HasJob(Roberta,actor)
wff70!: ~HasJob(Roberta,telephone operator)
wff69!: ~HasJob(Thelma,nurse)
wff68!: ~HasJob(Thelma,actor)
wff67!: ~HasJob(Thelma,telephone operator)
wff32!: HasJob(Thelma,chef)
wff28!: HasJob(Steve,police officer)
wff23!: ~HasJob(Pete,nurse)
wff22!: ~HasJob(Pete,police officer)
wff21!: ~HasJob(Pete,teacher)
wff20!: ~HasJob(Roberta,boxer)
wff19!: ~HasJob(Roberta,chef)
wff18!: ~HasJob(Roberta,police officer)
```

This is the complete solution to the puzzle: Pete is the actor and the telephone operator; Steve is the police officer and the nurse; Thelma is the boxer and the chef; and Roberta is the guard and the teacher. In addition, the 4 * 6 = 24 negative conclusions are found: who doesn't hold which jobs.

2.11.3.1.2 The Jobs Puzzle in TopBraid/Pellet

The Topbraid Ontology Editor [37] can create and edit ontologies, load them from, and save them as RDF or OWL files. It has a user interface which allows a user to organize classes into a class hierarchy (the upper left frame displays this hierarchy), and edit properties of those classes in the class form frame (the upper middle frame). Standard OWL Full RDF schema features [24] are provided, and new properties can be specified by the user in the properties frame (the upper right frame).

To represent the Jobs Puzzle in Topbraid, we created two subclasses of `owl:Thing`, the top-level OWL class: `Job`, and `Person`. `Job` has as mutually distinct subclasses the eight jobs (`Actor`, `Boxer`, `Chef`, `Guard`, `Nurse`, `Police Officer`, `Teacher`, and `Telephone Operator`). `Person` is partitioned into the two disjoint and exhaustive subclasses, `Female Person` and `Male Person`. `Female Person` is partitioned into the subclasses `Thelma` and `Roberta`, while `Male Person` is partitioned into the subclasses `Pete` and `Steve`. Each of the leaf subclasses, `Actor`, `Roberta`, *etc.* has a single instance (e.g. `Ind Roberta` and `Actor Pete` are the only instances of `Roberta` and `Actor`, respectively). The use of

names as classes was done because one cannot place contraints on the instances in Topbraid, only the classes; and SWRL rules, which allow for reasoning to be done on instances, proved insufficient for the reasoning needed by this domain. With the hierarchy in place, the only components necessary for formalizing constraints are relationships. We've made `hasJob` a relation between a `Person` and `Job`, and `heldBy` its converse relationship (its `owl:inverseOf`).

To formalize the constraint that all jobs are held by one person, we placed the condition on the `Job rdfs:subclassOf` property that

```
owl:Thing and heldBy exactly 1 and heldBy all Person
```

and on the `Job owl:equivalentClass` property that

```
Actor or Boxer or Chef or Guard or Nurse or Police_Officer
or Teacher or Telephone_Operator
```

To formalize the constraint that all people have two jobs and that no person holds both the job of the `Police Officer` and `Chef`, the `Person rdfs:subclassOf` property was given the condition

```
owl:Thing
and ((hasJob some Police_Officer) and not hasJob some Chef)
or ((hasJob some Chef) and not hasJob some Police_Officer)
or (not hasJob some Chef and not hasJob some Police_Officer)
and hasJob exactly 2
and hasJob all Job
```

The constraint that the jobs of `Nurse`, `Actor` and `Telephone Operator` are not held by a female is specified by placing on `Female Person rdfs:subclassOf` the two restrictions

```
Person
(Person
 and not hasJob some Nurse
 and not hasJob some Actor
 and not hasJob some Telephone-Operator)
```

A similar constraint is placed on `Male Person rdfs:subclassOf` to indicate that males cannot be the chef:

```
Person
not hasJob Chef
```

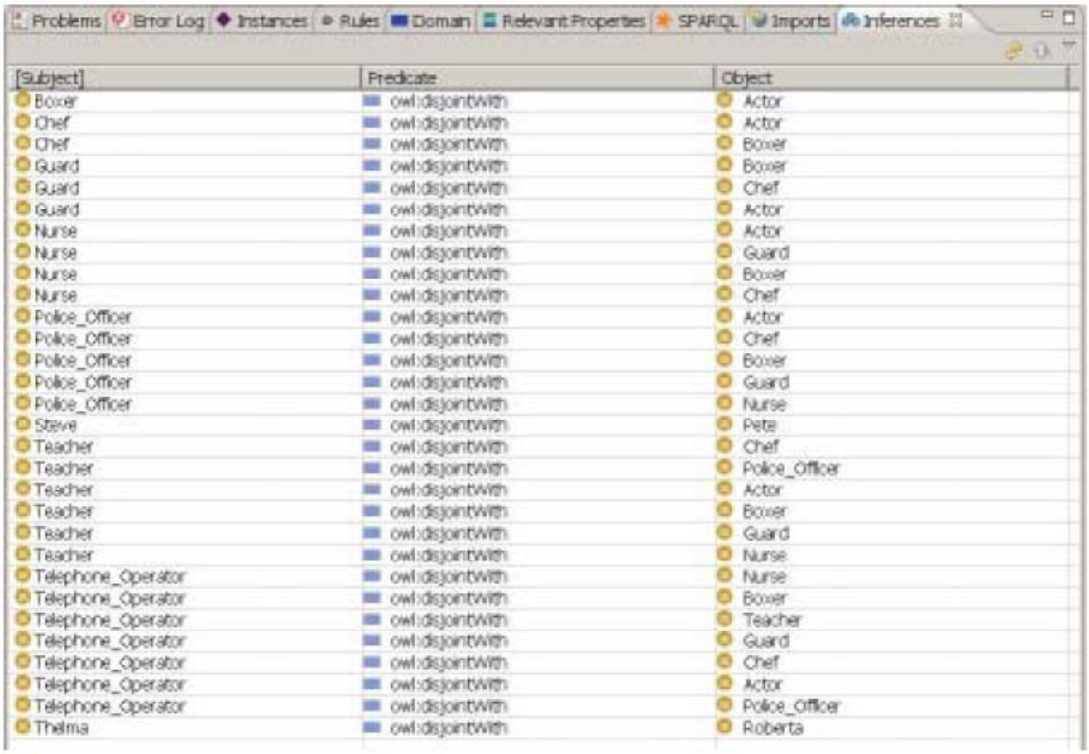
To indicate the Jobs the individuals cannot hold, similar restriction are placed on their `rdfs:subclassOf` restriction sets. For `Roberta`:

```
Female_Person
not hasJob some Boxer
and not hasJob some Chef
and not hasJob some Police_Officer
```

For `Pete`:

```
Male_Person
not hasJob some Teacher
and not hasJob some Nurse
and not hasJob some Police_Officer
```

With all the constraints represented, the Pellet reasoner is invoked. However, after finishing the reasoning process none of the individual constants have their `hasJob` or `heldBy` relationships filled. The reasoner does draw conclusions, such as several `owl:disjointWith` relationships between classes that weren't asserted directly, but these do not suffice for a solution to the puzzle. The inferred relationships are shown in Figure 28. Though Topbraid and Pellet were unable to provide a solution to the puzzle automatically, we were able to check various combinations of solutions by running the consistency checker after entering values for the `hasJob` relationships for each individual. If a particular combination was inconsistent, the consistency checker would report it. The correct values caused no inconsistency.



[Subject]	Predicate	Object
Boxer	owl:disjointWith	Actor
Chef	owl:disjointWith	Actor
Chef	owl:disjointWith	Boxer
Guard	owl:disjointWith	Boxer
Guard	owl:disjointWith	Chef
Guard	owl:disjointWith	Actor
Nurse	owl:disjointWith	Actor
Nurse	owl:disjointWith	Guard
Nurse	owl:disjointWith	Boxer
Nurse	owl:disjointWith	Chef
Police_Officer	owl:disjointWith	Actor
Police_Officer	owl:disjointWith	Chef
Police_Officer	owl:disjointWith	Boxer
Police_Officer	owl:disjointWith	Guard
Police_Officer	owl:disjointWith	Nurse
Steve	owl:disjointWith	Pete
Teacher	owl:disjointWith	Chef
Teacher	owl:disjointWith	Police_Officer
Teacher	owl:disjointWith	Actor
Teacher	owl:disjointWith	Boxer
Teacher	owl:disjointWith	Guard
Teacher	owl:disjointWith	Nurse
Telephone_Operator	owl:disjointWith	Nurse
Telephone_Operator	owl:disjointWith	Boxer
Telephone_Operator	owl:disjointWith	Teacher
Telephone_Operator	owl:disjointWith	Guard
Telephone_Operator	owl:disjointWith	Chef
Telephone_Operator	owl:disjointWith	Actor
Telephone_Operator	owl:disjointWith	Police_Officer
Thelma	owl:disjointWith	Roberta

Figure 28: Results of TopBraid's Job Puzzle

It is clear that both Topbraid and SNePS are capable of representing the constraints of the Jobs Puzzle. SNePS uses a higher-order logic notation with specialized logical operators (such as `nexists` and `andor`) to accomplish this task, while Topbraid uses the OWL Full restrictions. Where the two systems differ is in the reasoning. SNePS is capable of reasoning to the puzzle's solution, while the Pellet reasoner in Topbraid cannot. This is because Pellet was primarily designed to reason over OWL ontologies, which doesn't allow for inferring the values of a particular instance's relationship slots from negative information (i.e. by asserting the relationship is not of some class, the reasoner cannot conclude it is of another). Though there exists a logical notation for OWL called SWRL [24] that can fill in relation slots, an attempt at representing the puzzle using this import proved unsuccessful, as SWRL rules lack negation. As discussed previously, an automatic solution could be built around Pellet that would reason to the solution by trying various possibilities for the `hasJob` relationship by utilizing the Java API, but this is beyond the scope of this study. These reasoning issues with Pellet are easily handled in SNePS using `nexists`, which was created precisely to handle reasoning from negative information (Shapiro 1979).

2.11.3.2 The ParentSally Example

The ParentSally Example is a domain of our creation that was designed to illustrate Description Logic reasoning. The domain includes the classes of Sex and Animal. The Animal type has as its subclasses Person, Cattle, and Dog. Cattle has only one subclass, Cow, while Person is divided

into Man, Woman, and Parent subclasses. All classes are considered subclasses of the class Thing, which is the top-level class. Various properties are ascribed to each subclass, and will be discussed in the following sections.

2.11.3.2.1 The ParentSally Example in TopBraid/Pellet

Topbraid as a description logic framework was built to represent classification hierarchies. As such there is little effort required to represent the hierarchy and populate it with instances. Figure 29 shows the hierarchy. All the classes needed, plus the additional `PersonWithAtMost1Child`, which will be discussed later, are shown in the leftmost frame. The rightmost frame shows the relationships used; `hasChild` (and its `owl:inverseOf` relationship `isChildOf`, and `hasSex` (and its `owl:inverseOf` relationship `isSexOf`). Finally, the middle frame displays the `Woman` class, which specifies that it is `owl:disjointWith` the `Man` class. In addition to the classes: `Fred` is created as an instance of `owl:Thing`; `Elsie` as an instance of `Cattle`; `Lucy`, `Pete`, `Tom`, and `Sally` as instances of `Person`; and `male` and `female` as instances of `Sex`. With the hierarchy established, we begin creating restrictions on the classes using the relationships. Restrictions in the `rdfs:subClassOf` and `owl:equivalentClass` restriction sets represent necessary, and necessary and sufficient properties for being members in those classes respectively. The class of `Man` is declared to be an

`rdfs:subclassOf`

```
Person and hasSex has male
```

`Woman` is an `rdfs:subclassOf`

```
Person and hasSex has female
```

`Cow` is the `rdfs:subclassOf` and `owl:equivalentClass` of

```
Cattle and hasSex has female
```

We give the class `Parent` the necessary and sufficient conditions that a parent is a person with at least one child by placing a restriction in its `rdfs:subClassOf` and `owl:equivalentClass` of:

```
Person and hasChild min 1
```

and we add to `Pete`'s representation

```
Pete rdfs:type Parent
Pete hasChild Fred
```

Topbraid/Pellet concludes that `Fred IsChildOf Pete`.

Then, we set the isSexOf relationships for the two Sexes as

```
male isSexOf Fred
female isSexOf Elsie
```

Topbraid adds Fred hasSex male to Fred's representation, and Elsie has Sex female to Elsie's.

The results of then running the Pellet reasoner is that Fred remains in the category of Thing (Figure 30), while Elsie is inferred to be a Cow (Figure 31).

To make the reasoner conclude that Fred is a person we specify that all the children of parents are people, by adding to the Parent rdfs:subclassOf and owl:equivalentClass restrictions, so that they are now

```
Person and hasChild min 1 and hasChild all Person
```

Notice that this also means that any person all of whose children are people is a parent.

Running the Pellet reasoner on this allows the system to conclude Fred is a person (because Fred is Pete's child), as depicted in Figure 32.

Finally, we want to establish that Sally is a parent by asserting she has one child that is a person. To do this we add to Sally's hasChild relationship the value of Lucy, who is also a person. Performing Pellet reasoning on this does not conclude that Sally is a person, because Pellet operates under the open world assumption: Sally might have some as yet unknown children that aren't people. To solve this, we create a new subclass of Person named PersonWithAtMost1Child and give it a rdfs:subClassOf restriction of:

```
Person and hasChild max 1
```

We then add to Sally's rdfs:type field the class of PersonWithAtMost1Child, asserting that Sally is an individual of this class. Invoking Pellet reasoning now causes the reasoner to conclude that Sally is a parent, as depicted in Figure 33.

Figure 29: Topbraid - ParentSally Example Classification Hierarchy

Figure 30: Topbraid - Fred's Resource Description after Pellet inference, showing that he's still just a Thing.

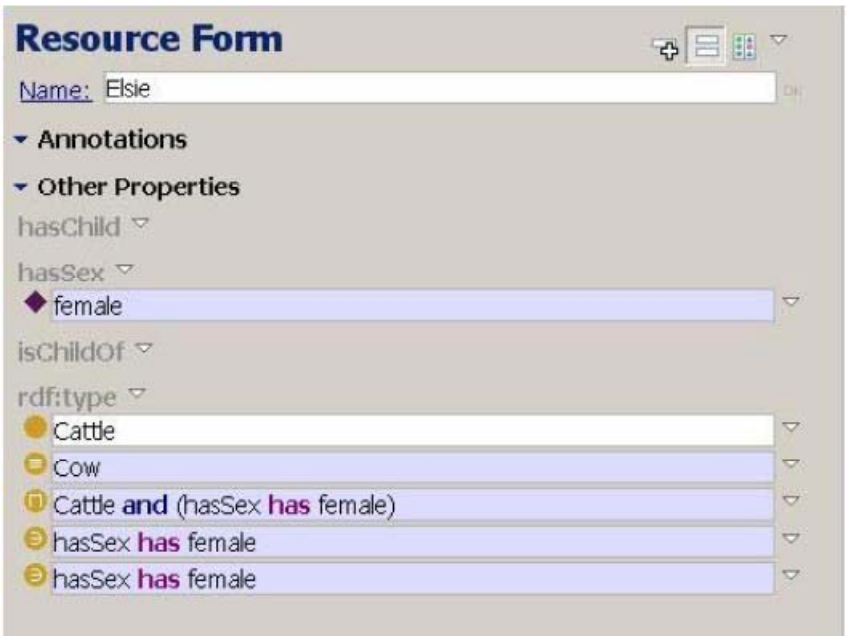


Figure 31: Topbraid - Elsie's Resource Description after Pellet inference, showing that she's a Cow.

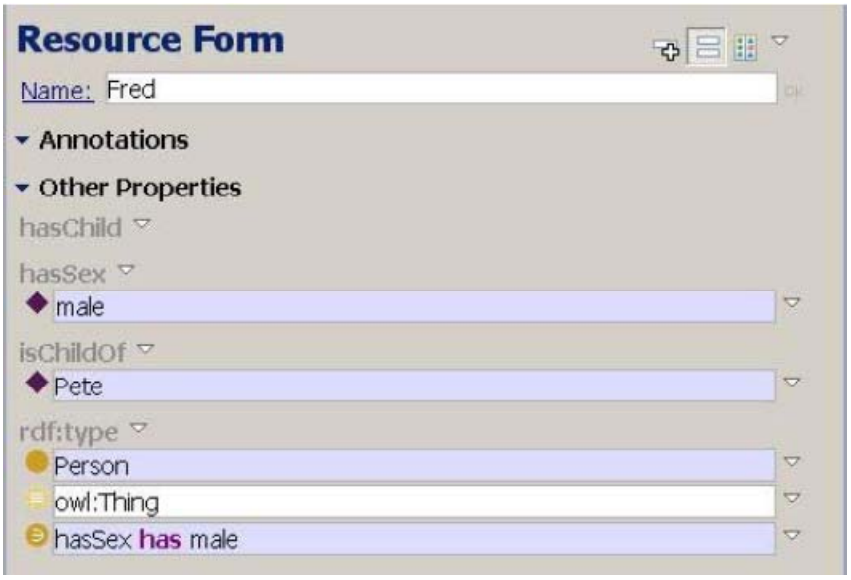


Figure 32: Topbraid - Fred's Resource Description after a second Pellet inference infers that he's a Person.

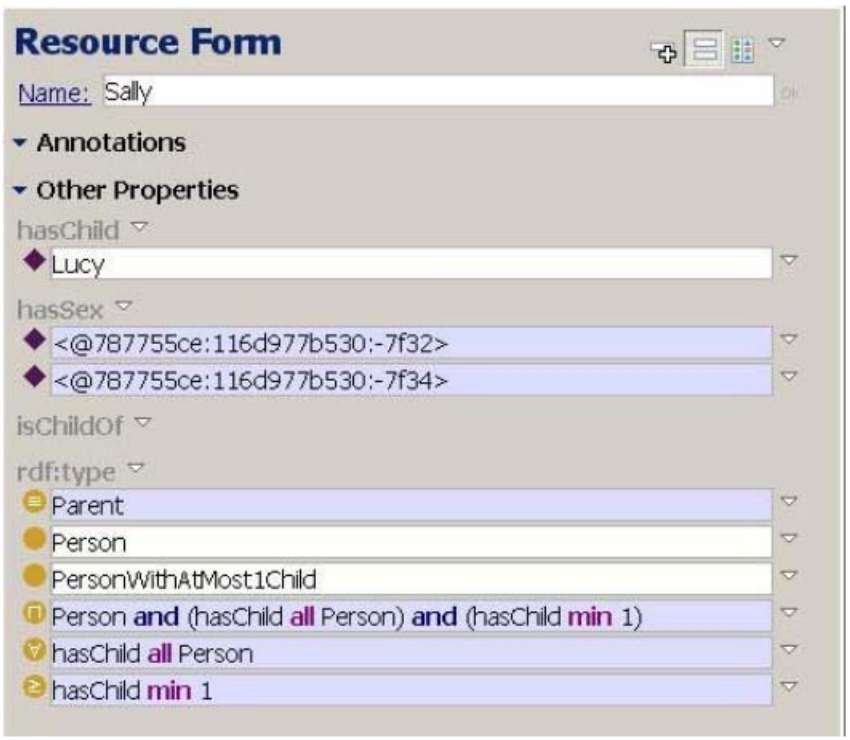


Figure 33: Topbraid - Sally's Resource Description after Pellet infers that she's a parent.

2.11.3.2.2 The ParentSally Example in SNePS

To represent the structure of a classification hierarchy in SNePSLOG we use the two terms:

- `Isa(x,y)` - The proposition that `x` is a member of class `y`
- `Ako(x,y)` - The proposition that `x` is a subclass of `y`

To reason about the hierarchy, the following path-based rules [36] are used:

```
(a) define-path superclass
     (or superclass
          (compose ! superclass (kstar (compose subclass- !
          superclass)))
          (domain-restrict ((compose arg- ! max) 0)
               (compose superclass
                    (kstar (compose superclass- !
                    subclass)))))

(b) define-path subclass
     (or subclass
          (compose ! subclass (kstar (compose superclass- !
          subclass)))
          (domain-restrict ((compose arg- ! max) 0)
               (compose subclass
                    (kstar (compose subclass- ! superclass)))))
```

```
(c) define-path class
     (or class
          (compose ! class (kstar (compose subclass- !
          superclass)))
          (domain-restrict ((compose arg- ! max) 0)
               (compose class
                    (kstar (compose superclass- ! subclass)))))
```

The details of SNePS' path-based inference are not important. The above essentially establishes when the system can create a new path between nodes in the SNePS network, thus, creating a believed proposition. Path-based rule (a) allows the system to reason that if some class *c1* has *c2* as a superclass, and *c2* has *c3* as a superclass, then *c1* has *c3* as a superclass. Path-based rule (b) allows the system to reason that if some class *c1* has *c2* as a subclass, and *c2* has *c3* as a subclass, then *c1* has *c3* as a subclass. In other words, (a) and (b) establish *Ako* transitivity. Path-based rule (c) allows the system to reason that some instance *i* of class *c1* is a member of *c2*, if *c1* is a subclass of *c2*. In other words, if *Isa* holds between an instance and its class, it also holds between that instance and that class' superclasses. What is significant about SNePS' path-based reasoning is that it is more efficient that using the equivalent rules

```
all(c1,c2,c3)({Ako(c1,c2), Ako(c2,c3)} => Ako(c1,c3)).
all(c1,c2,c3)({~Ako(c1,c2), Ako(c3,c2)} => ~Ako(c1,c3)).
all(i,c1,c2)({Isa(i,c1), Ako(c1,c2)} => Isa(i,c2)).
all(i,c1,c2)({~Isa(i,c1), Ako(c2,c1)} => ~Isa(i,c2)).
```

The above is a general SNePS axiomatization for reasoning about class hierarchies. Terms used for the ParentSally example are:

- `Thing` - the class of all things
- `Sex` - the class of sexes
- `Animal` - the class of animals
- `Person` - the class of people
- `Dog` - the class of dogs
- `Cattle` - the class of cattle
- `Man` - the class of men
- `Woman` - the class of women
- `Parent` - the class of parents
- `Cow` - the class of cows
- `male` - the individual male
- `female` - the individual female
- `Fred` - the individual Fred
- `Elsie` - the individual Elsie

- `Lucy` - the individual Lucy
- `Pete` - the individual Pete
- `Sally` - the individual Sally
- `Tom` - the individual Tom
- `childOf(x)` - A child of `x`
- `hasSex(x,y)` - the proposition that `x` has the sex `y`
- `hasChild(x,y)` - the proposition that `x` has the child `y`

The class hierarchy is established by making the assertions:

```
Ako({Sex, Animal}, Thing).
Ako({Person,Dog,Cattle}, Animal).
Ako({Man,Woman,Parent}, Person).
Ako(Cow, Cattle).
Isa(Fred, Thing).
Isa(male,Sex).
Isa(female, Sex).
Isa(Elsie, Cattle).
Isa({Lucy, Pete, Sally, Tom}, Person).
```

The ParentSally example requires various constraints:

- The classes of *Man* and *Woman* are disjoint.
  `all(x)(andor(0,1)Isa(x,Man),Isa(x,Woman)).`
- Every animal has exactly one sex.
  `all(x)(Isa(x,Animal) => nexists(1,1,2)(s)(Isa(s,Sex): hasSex(x,s))).`
- Every man is male. `all(x)(Isa(x,Man) => hasSex(x,male)).`
- Every woman is female.
  `all(x)(Isa(x,Woman) => hasSex(x,female)).`
- Every cow is a female cattle.
  `all(x)(Isa(x,Cow) => Isa(x,Cattle), hasSex(x,female)).`
- Every female cattle is a cow.
  `all(x)(Isa(x,Cattle) => (hasSex(x,female) => Isa(x,Cow))).`
- Every parent is a person who has a child.
  `all(x)(Isa(x,Parent) => fIsa(x,Person), hasChild(x,childOf(x))).`

We enter the assertions that Pete is a parent whose child is Fred,

```
Isa(Pete,Parent).
hasChild(Pete,Fred).
```

and that Fred is male and Elsie is female.

```
hasSex(Fred,male).
hasSex(Elsie,female).
```

Then we ask SNePS for the classes that Fred and Elsie are instances of.

```
: Isa(Fred,?x)?
wff5!: Isa(Fred,Thing)
: Isa(Elsie,?x)?
wff45!: Isa(Elsie,Cow)
wff38!: Isa(Elsie,Thing)
wff37!: Isa(Elsie,Animal)
wff8!: Isa(Elsie,Cattle)
```

As in the Topbraid/Pellet run, the system still has Fred as just a thing, but infers that Elsie is a cow. So, again as we did in the Topbraid/Pellet run, we assert that

- Every child of a parent is a person.
  `all(x)(Isa(x,Parent) => all(y)(hasChild(x,y) => Isa(y,Person))).`
- Every person all of whose children are people is a parent.
  `all(x)(Isa(x,Person)=> (all(y)(hasChild(x,y) => isa(y,Person))=> Isa(x,Parent))).`

and again ask for the classes that Fred is an instance of:

```
: Isa(Fred,?x)?
wff35!: Isa(Fred,Person)
wff25!: Isa(Fred,Animal)
wff5!: Isa(Fred,Thing)
```

As in the Topbraid/Pellet run, this is now successful.

The culmination of the ParentSally example is to assert that

1. Lucy is Sally's child.
   `hasChild(Sally,Lucy).`

and ask if Sally is a parent:

```
: Isa(Sally,Parent)?
```

The lack of response indicates that SNePS can neither conclude that `Isa(Sally,Parent)` nor `~Isa(Sally,Parent)`,

even though it has that both Sally and Lucy are people:

```
: Isa(Sally,Person)?
wff60!: Isa(Sally,Person)
: Isa(Lucy,Person)?
wff62!: Isa(Lucy,Person)
```

SNePS does not conclude that Sally is a parent, even though one might expect it to be able to from the rule

```
all(x)(Isa(x,Person)=> (all(y)(hasChild(x,y) => Isa(y,Person))=>
Isa(x,Parent))).
```

There are two reasons for the absence of this inference.

1. SNePS currently does not have implemented an introduction rule that would allow it to infer universally quantified implications like `all(y)(hasChild(Sally,y) => Isa(y,Person))`
2. Even if SNePS had this rule, it does not follow logically from the current knowledge base that an arbitrary child of Sally is a person. This reason is similar to the reason that Pellet couldn't infer that Sally was a person until we said that Sally had at most one child.

To solve this problem, we employ a form of limited closed world assumption using SNeRE, the SNePS acting system [36]. The following assertion gives a plan for concluding that, for any person x who is known to have a child who is a person, if every child they are known to have is a person, then x is a parent.

```
all(x,y)({Isa(x,Person), hasChild(x,y), Isa(x,Person)}
=> ActPlan(parentIfAllKnownChildrenArePeople(x),
snsequence4(believe(Maybe(Isa(x,Parent))),
withall(y, hasChild(x,y),
snif({if(Isa(y,Person), noop()),
else(disbelieve(Maybe(Isa(x,Parent))))}),
noop()),
snif({if(Maybe(Isa(x,Parent)),
believe(Isa(x,Parent))),
else(believe(~Isa(x,Parent)))}),
disbelieve(Maybe(Isa(x,Parent))))).
```

This plan for a person x with a known child who is a person is:

```
1. Believe, as a temporary default, that x may be a parent;
2. With every y such that y is a child of x
if y is a person do nothing
else disbelieve that x may be a parent
but if x has no children, do nothing (but this cannot be)
3. If it is still believed that x may be a parent
believe that x is a parent
else believe that x is not a parent
```

4. Disbelieve that x may be a parent.

The reason for the temporary default belief that Maybe(Isa(x,Parent))[4] instead of a temporary belief that Isa(x,Parent) is that once the system believed that Isa(x,Parent), it would infer that all x's children were people, defeating the check that all x's children are already known to be people. Since we want to say that Sally is a parent if all her known children are people, we perform this act on Sally:

```
: perform parentIfAllKnownChildrenArePeople(Sally).
```

and then again ask if Sally is a parent:

```
: Isa(Sally,Parent)?
wff64!: Isa(Sally,Parent)
```

Sally is now believed to be a parent.

To make sure that the system is not overgeneralizing, let us introduce Ted, a person with two children, only one of

whom is known to be a person:

```
Isa(Ted,Person).
: hasChild(Ted,{Betty,Jean}).
: Isa(Betty,Person).
```

and see if Ted is inferred to be a person:

```
: perform parentIfAllKnownChildrenArePeople(Ted).
: Isa(Ted,Parent)?
wff129!: ~Isa(Ted,Parent)
```

The system infers that Ted is not a parent.

2.11.3.2.3 The ParentSally Example Comparison

Topbraid/Pellet and SNePS are both able to represent the ParentSally Example, and to perform the required reasoning. The techniques, of course, are different. The most noticeable difference is in the way they handle the inference that Sally is a parent. Topbraid/Pellet essentially does the following.

1. Sally is a person with a child, Lucy, who is a person.
2. Sally is an instance of PersonWithAtMost1Child.

3. All instances of `PersonWithAtMost1Child` have at most 1 child.
4. Therefore Lucy is Sally's only child.
5. Therefore all Sally's children are people.
6. Therefore Sally is a parent.

The crucial step is (4), which is a version of circumscription (McCarthy 1980), using, specifically, a number restriction on the role, `hasChild`. Number restrictions, and reasoning according to them, are among the earliest features of Description Logics, and .are sometimes viewed as a distinguishing feature of Description Logics[40].

SNePS inferred that Sally is a parent by following this line of reasoning:

1. Sally is a person with a child, Lucy, who is a person.
2. So maybe Sally is a person.
3. Consider all Sally's children.
    a. Lucy is a child of Sally's, and she is a person, so Sally still may be a person.
    b. That's all the children of Sally that I know of.
4. Sally still may be a parent.
5. So Sally is a parent.

SNePS uses introspective acting to: consider all the children of Sally it knows; disbelieve that Sally may be a parent if one of them is not a person; believe that Sally is a parent if it still believes that she may be one after considering all her children. Unlike Topbraid/Pellet, SNePS never concludes that Lucy is Sally's only child.

Both Topbraid/Pellet and SNePS use a form of closed-world reasoning restricted to the set of Sally's children. If Topbraid/Pellet later learned that Sally had another child, that would contradict the conclusion that Sally is an instance of `PersonWithAtMost1Child`. If SNePS later learned that Sally had another child, say Dave, who was not known to be a person, it would conclude that Dave is a person, because Sally is now believed to be a person.However, if we had SNePS first `disbelieve` that Sally is a parent, when we subsequently had it perform `parentIfAllKnownChildrenArePeople (Sally)` again, it would then believe that `~Isa(Sally,Parent)`.

2.11.4 Modifications to Improve SNePS Efficiency

As a result of studying the tests reported above, several modifications were made to SNePS 2.7.0. The modified version of SNePS is SNePS 2.7.1. The following sections describe the modifications in two groups:

1. Modifications to reduce the space requirements of SNePS in order to:
    - Lessen the impact of the garbage collector.

- Increase the size of knowledge bases that might be stored without running out of memory.
- Incidentally reduce the running time.

2. Modifications primarily to reduce the running time of SNePS.

Other major improvements to the design of SNePS have been scheduled for the implementation of SNePS 3. Those improvements are discussed in Section 2.11.4.3.

2.11.4.1 Modifications to Reduce Space

Several of the data structures used in SNePS are composed of a fixed number of components, and are implemented with straight-forward constructors and selectors. Some of them that were implemented in SNePS 2.7.0 as lists have been changed to vectors. A vector with n elements uses half the space of a list with n elements, and moreover its elements are faster to access.

A SNePS assertion has a set of supports, which are used by SNeBR, the SNePS Belief Revision System [36]. Each support includes an origin set, a set of hypotheses which were used to derive the assertion, and a restriction set, a set of sets of hypotheses, each of which, when unioned with the origin set, forms a minimal nogood. A minimal nogood is a set of hypotheses known to be inconsistent such that no subset of it is known to be inconsistent [41]. Restriction sets have now been eliminated in favor of a global set of minimal nogoods. Since SNeBR was not invoked in any of the tests discussed in this paper, this modification is mainly relevant for the space it saves in the support set of each assertion.

2.11.4.2 Modifications to Reduce Running Time

In SNePS 2.7.0, a query or subquery is matched against "all the nodes in the KB except base nodes (individual constants)". In SNePS, every functional term or predicate of arity greater than 0 is represented by an assertional frame. The set of slots of the frame, called a caseframe, constitutes a higher order function (or predicate). Comparing the caseframe of the query with the caseframe of a potential match could eliminate the potential match without actually performing the match routine. This is complicated by the fact that slots themselves can be inferred via path-based inference [34]. Nevertheless, we implemented a filter in SNePS 2.7.1 that considers the caseframes and the possibility of path-based inference so that the match routine is not even tried on nodes for which it is relatively easy to tell that it would not possibly succeed. In SNePS 3, nodes will be indexed by caseframes (considering path-based inference) to further reduce the number of nodes on which the match routine will be tried.

2.11.4.3 Modifications Planned for SNePS 3

In SNePS 2.7.0, sets of nodes are represented by ordered lists, and set operations, such as intersection and union, are implemented using hashsets. It would be more efficient to

represent the sets as hashsets also, but knowledge of the implementation as lists is too pervasive in the code to make that change. In SNePS 3, sets of nodes are being represented by hashsets.

Which rule is more efficient between

```
all(x,y,z)(A(x,y) => (B(y,z) => C(x,z)))
```

and

```
all(x,y,z)(B(y,z) => (A(x,y) => C(x,z)))
```

depends on whether `x` or `z` is bound by the query. In SNePS 3, the rule will be expressed either as

```
(&=> (setof (A x y) (B y z)) (C x z))
```

or as

```
(C (any x (A x (any y))) (any z (B y z)))
```

(*see* (Shapiro 2004)), neither of which specify which antecedent should be tried first. We intend to implement SNePS 3 to consider which variables are free or bound, and to order antecedents accordingly.

2.11.4.4 Post-Modification Reruns

A set of 10 tests were performed to determine any efficiency improvements of SNePS 2.7.1 over SNePS 2.7.0. Figures Figure 34-Figure 43 show these results. Each of these figures show the total times, in milliseconds, of the test for SNePS 2.7.0 and for SNePS 2.7.1. Note, especially that:

- In test 1, SNePS 2.7.1 actually ran slower for 7,000 extraneous propositions than SNePS 2.7.0, but still took less than 4 ms. For 8,500 extraneous propositions, SNePS 2.7.1 took about 1/3 the time of SNePS 2.7.0.
- In tests 4, 6, and 8, SNePS 2.7.0 suffered from garbage collection spikes, but these spikes have been eliminated in SNePS 2.7.1.
- In tests 5, 7, and 9, SNePS 2.7.0 reached out of memory conditions (after 4,000, 5,500, and 7,000 extraneous propositions, respectively), but SNePS 2.7.1 was able to run with 8,500 extraneous propositions.
- In tests 2 and 3, SNePS 2.7.1 performed significantly faster than SNePS 2.7.0 after 7,000 (in test 2) or 2,500 (in test 3) extraneous propositions.
- In test 10, SNePS 2.7.1's performance was exactly the same as SNePS 2.7.0's, both of which took only 10 ms to run with 8,500 extraneous propositions.

- In the Firewall Rules Inference test, SNePS 2.7.1 performed better than SNePS 2.7.0 regardless of the form of the rule.

Various timing tests were performed on the SNePS 2.7.0 knowledge representation and reasoning system [36]. The tests varied on the format of some simple reasoning rules, and on the size of the knowledge base when the tests were run. The size of the knowledge base was varied by introducing various numbers of propositions that, although extraneous to the results of the reasoning rules, were involved in the matching of goals and subgoals. For most tests, the number of extraneous propositions was increased in steps up to 8,500. However, in three tests the available heap was exceeded before 8,500 extraneous propositions could be entered. In one case this happened between 4,000 and 5,000 extraneous propositions. In three tests, the run times were affected by garbage collection spikes. These six tests showed the need for improvements in the amount of space used to store SNePS knowledge bases.

The time to perform the tests when the maximal number of extraneous propositions were present varied from 10 ms to 2,204 secs, showing the need for improvements in the speed of various SNePS procedures.

After profiling SNePS performance, several basic software engineering improvements were made to SNePS data structures and procedures. The results were that all tests could be run with 8,500 extraneous propositions without incurring a heap overflow, garbage collection spikes were eliminated, and running times were improved. Additional modifications in the design of the data structures and procedures of SNePS are planned for SNePS 3.

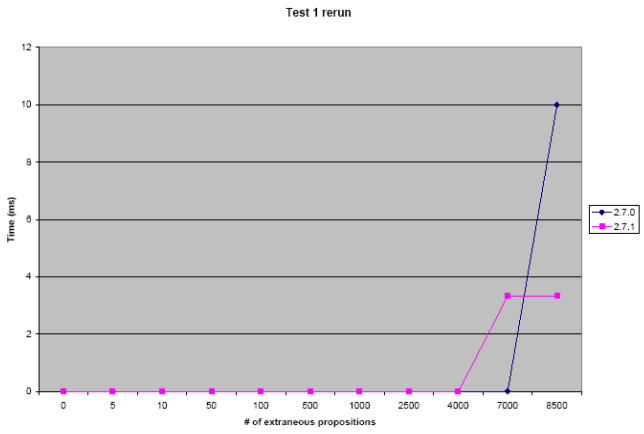


Figure 34: Comparing SNePS 2.7.0 to 2.7.1 using Test 1

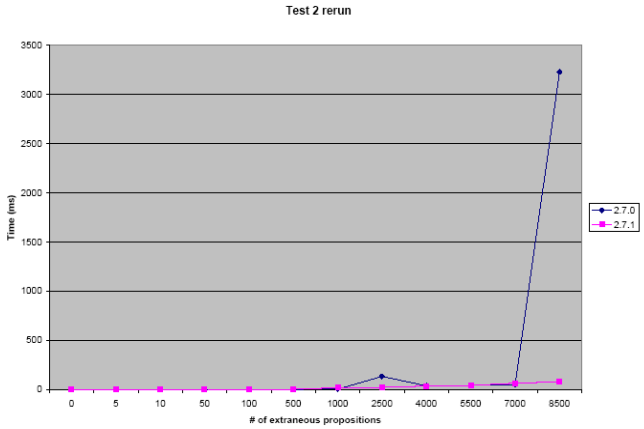


Figure 35: Comparing SNePS 2.7.0 to 2.7.1 using Test 2

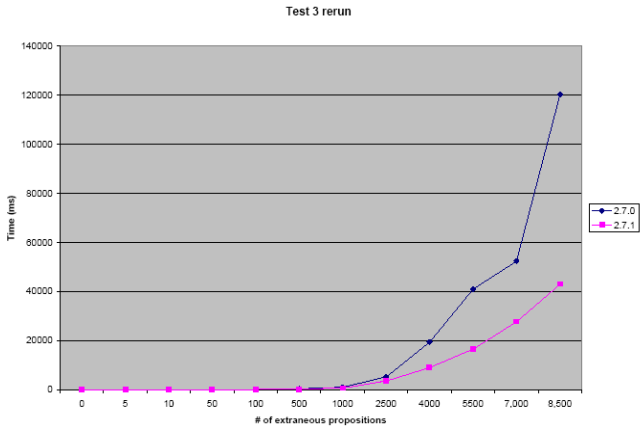


Figure 36: Comparing SNePS 2.7.0 to 2.7.1 using Test 3

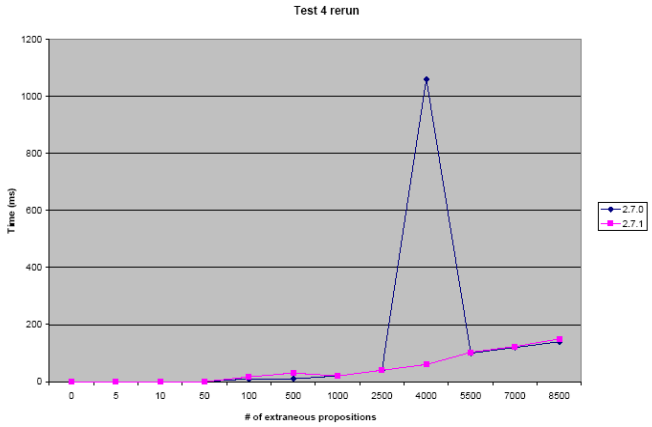


Figure 37: Comparing SNePS 2.7.0 to 2.7.1 using Test 4

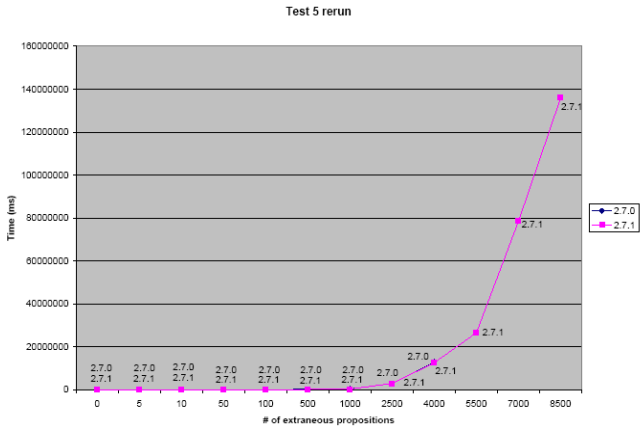


Figure 38: Comparing SNePS 2.7.0 to 2.7.1 using Test 5

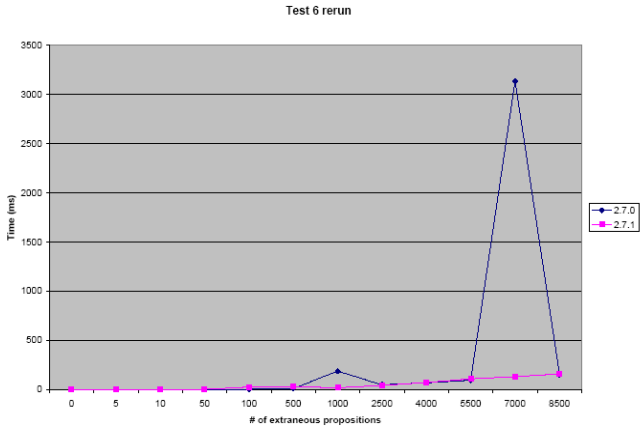


Figure 39: Comparing SNePS 2.7.0 to 2.7.1 using Test 6

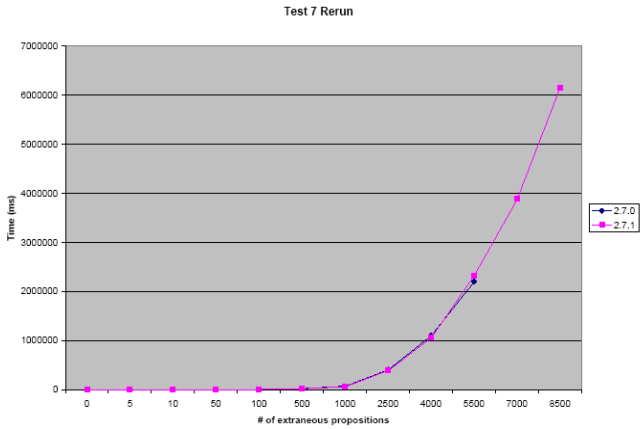


Figure 40: Comparing SNePS 2.7.0 to 2.7.1 using Test 7

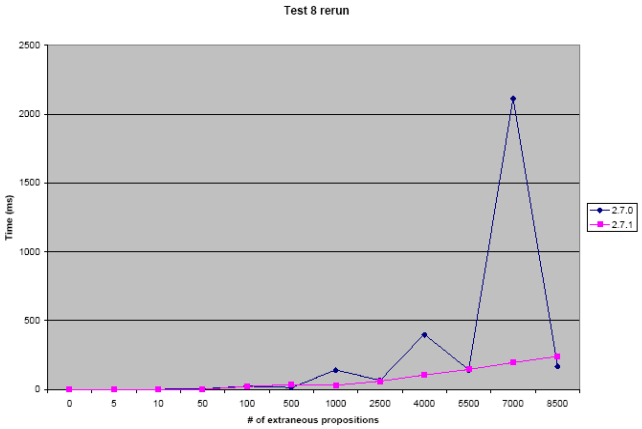


Figure 41: Comparing SNePS 2.7.0 to 2.7.1 using Test 8

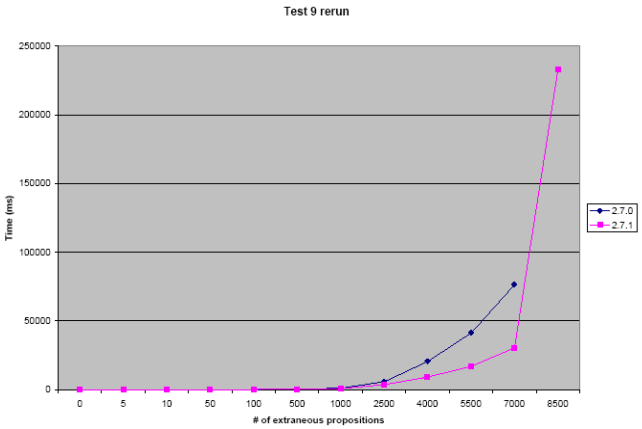


Figure 42: Comparing SNePS 2.7.0 to 2.7.1 using Test 9

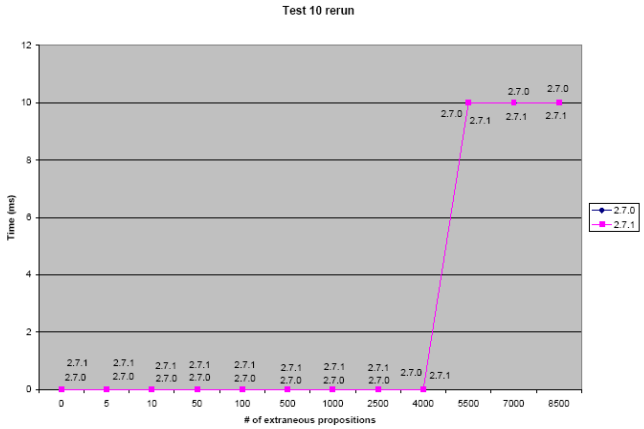


Figure 43: Comparing SNePS 2.7.0 to 2.7.1 using Test 10

3 Information Fusion Virtual Library (4.1.2)

Task 4.1.2 involved the creation of an information fusion virtual library and is accessible at http://www.infofusion.buffalo.edu/. Figures Figure 44-Figure 51 illustrate some example screenshots of the virtual library and how to access it from the CMIF website.

Figure 44: CMIF Research Activity Page

Figure 45: Representative Research Projects Page

Figure 46: CMIF Sponsored Workshops Page

Figure 47: CMIF Publications Page

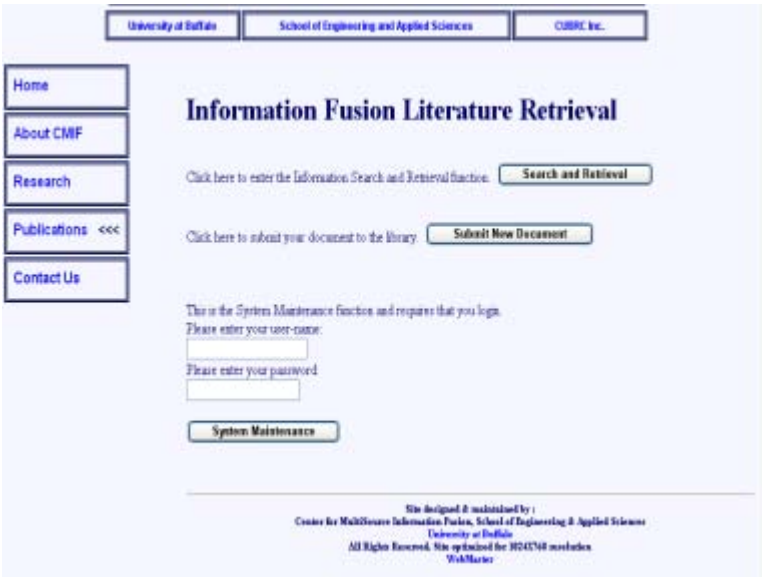


Figure 48: Literature Retrieval Page 1 of 2

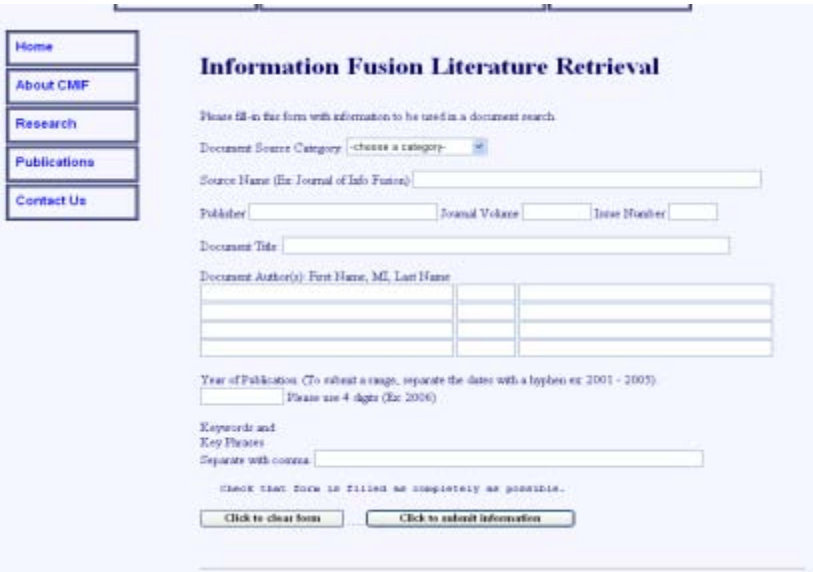


Figure 49: Literature Retrieval Page 2 of 2

Figure 50: Document Upload Page

Figure 51: Document Upload Page File Selection

4 Information Fusion Workshops and Conferences (4.1.3)

N-CMIF researchers published papers at, made presentations at, and attended various conferences throughout the course of the project. In addition, an online journal article was published. This section provides bibliographical information outlining all of the publications prepared under N-CMIF.

Yang, S., Stotz, A., Holsopple, J., Sudit, M., Kuhl, M.,"High Level Information Fusion for Tracking and Projection of Multistage Cyber Attacks", Information Fusion, In Press, Accepted Manuscript, Available online 28 June 2007.

Stotz, A., Sudit, M., Nagi, R., Hirsch, M., Sambhoos, K. "Graph Dissemination for Distributed Enterprises (GraDDE)", Proceedings of the NSSDF Conference, June 2007.

M. Kandefer, S. C. Shapiro, A. Stotz, and M. Sudit, "Symbolic Reasoning in the Cyber Security Domain", Proceedings of the NSSDF Conference, June 2007.

Hall, D. B. Hellar, M. McNeese and J. Llinas, "Assessing the JDL model: a survey and analysis of decision and cognitive process models and comparison with the JDL model," Proceedings of the National Symposium on Sensor Data Fusion, June 2007.

Hall, D, B. Hellar and M. McNeese, "Rethinking the data overload problem: closing the gap between situation assessment and decision making," Proceedings of the National Symposium on Sensor Data Fusion, June 2007.

A. Stotz, M. Sudit, "INformation Fusion Engine for Real-time Decision-making (INFERD): A Perceptual System for Cyber Attack Tracking", Proceedings of the ISIF Conference, Quebec City, QE, CA, July 2007.

Yang, S., Holsopple, J., Argauer, B., Fava, D., "Terrain and Behavior Modeling for Projecting Cyber Attacks", Proceedings of the ISIF Conference, Quebec City, QE, CA, July 2007.

Llinas, J., "New Challenges for Defining Information Fusion Requirements", Plenary Address, Proceedings of the ISIF Conference, Quebec City, QE, CA, July 2007.

Kuhl, M., Kistner, J., Constantini, K., Sudit, M., "Cyber Attack Modeling and Simulation for Network Security Analysts", Proceedings of the Winter Simulation Conference, December 2007.

Holsopple, J., Yang, S. Argauer, B. "Virtual terrain: a security-based representation of a computer network", Proceedings of the SPIE Defense and Security Conference, Orlando, FL, March 2008.

Yang, S., Argauer, B. “VTAC: Virtual Terrain Assisted Impact Assessment for Cyber Attacks”, Proceedings of the SPIE Defense and Security Conference, Orlando, FL, March 2008.

Holsopple, J., Yang, S. “Fusia: Future Situation and Impact Awareness”, Proceedings of Fusion 2008: the International Conference on Information Fusion, Cologne, Germany, July 2008.

Hall, D., Llinas, J., McNeese, M., and Mullen, S., “A framework for dynamic hard/soft fusion,” Proceedings of Fusion 2008: the International Conference on Information Fusion, Cologne, Germany, July 2008.

Ballora, M. Lee Hong, S., Panulla, B., and Hall, D., ”Information through sound: understanding data through sonification,” Proceedings of ISEA2008, The 14th International Symposium on Electronic Art, July 25 – August 2, 2008, Singapore, pp. 47 – 49, 2008.

5 Software and Hardware (4.1.4) and Reporting (4.1.5)

All developed source code, compiled code, and demos are provided on the N-CMIF Deliverables CD. The CD also includes all publications from the previous section and associated conference presentations where applicable. In addition, the CD also provides electronic copies of the presentations of the quarterly reviews of the project.

No special hardware is required to run any of the software developed for N-CMIF. With the exception of visualization, all software was developed using Java 6 and other free open-source software packages. Visualization was developed with Adobe Flex. Development of Java code was performed using Eclipse (www.eclipse.org). The source code is stored on the CD such that it can be imported as an Eclipse project. Documentation is provided on the CD on how to execute the demos.

6 Nationbuilding

The final N-CMIF task was to shift the focus away from cyber attacks and to perform model elicitation on nationbuilding models previously developed by AFRL. These tasks focus on extracting and understanding data from different modules in the National Operational Environment Model (NOEM) architecture, which focus on different aspects of nationbuilding (such as Health, Economics, Infrastructure, etc.). These modules are represented in a simulation tool called Ptolemy II [43].

There were four main tasks conducted under N-CMIF and are each described in their own sections below:

1. Ptolemy/NOEM Model Analysis – to create an understanding of the Ptolemy simulation data structures [43] and to understand the modules being used in the NOEM.
2. GraphML Representation – to create an XML and graph-based data structure to store the data associated with the NOEM models in order to simplify the data representation and use various graph algorithms to elicit the model.
3. Petri Net Representation – to explore how petri nets could be used for model elicitation
4. Markovian Analysis – to explore how Markov chains could be used for model elicitation

N-CMIF was intended to act as a catalyst for CUBRC/UB/RIT's Graph and Network Object Model Elicitation (GNOME) contract, which continued the initial nationbuilding research conducted in N-CMIF.

6.1 Ptolemy Model Analysis

6.1.1 Granularity

An important attribute of the NOEM model is the varying degrees of granularity of its actors. The granularity of any synchronous dataflow program has important effects on the efficiency of that program. In the data flow model of computation, complex operations can be built from a large number of simple processing units, or one complex processing unit. The granularity of the model is considered course if the representative operations are implemented as single, highly complex, actors or fine granularity if the same operations are represented as a collection of many, individually less complex actors. In a data flow program, larger granularities result in less overhead [46]. However, due to their more abstract representation, models with comparatively course granularity also contain less structural information in their actor graphs.

6.1.2 Time – Recurrence

The NOEM simulation model is implemented as a "synchronous data flow" program, an actor oriented programming framework. In synchronous data flow programs, execution is controlled by the passing of messages between connected individual processing entities. These types of programs can conveniently represent models described by a system of difference equations, which can be used to describe discrete dynamical systems, or the manner in which certain phenomena develop over time [45]. Difference equations are widely used tools in both the social sciences and engineering, with some more specific examples being their application to control theory problems, econometric models, queuing problems, and behavior learning [44].

Cycles and delays in SDF can result in varying importance of an input over time. In Figure 52, for example, assume the initial output of node *f* is *f(X)* and node *g*'s output is *g(Y,0)*. Note that node *g* has two input arcs, one of which is a self loop. Assuming *X* and *Y* remain constant, the values output in the second iteration for *f* and *g* are *f(X)* and *g(Y,g(Y))*, respectively. If *Y* were not constant, call Y_0 and Y_1 the initial and second iteration values of *Y* respectively, the value of *g* at the second iteration could be more thoroughly described as $g(Y_2,g(Y_1))$. In general, the cycle around *g* in this example *g*'s output on any given iteration will be partially dependent upon its output from the previous iteration, which is itself partially dependent on node *g*'s output from the prior iteration. This dependency continues recursively back to the first iteration, with the result being that node *g*'s output is dependent upon all historical values of *Y*. This recursive dependency is one definition of a difference equation; *x(n+1) = f(x(n))* [45].

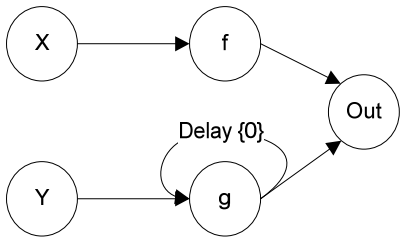


Figure 52: A SDF model with one self loop.

In this way, SDF programs may be used to simulate systems of difference equations; the behavior of which can lead to highly dynamic behavior. This becomes important to the NOEM model, as anywhere a directed cycle exists there is likely to be a complex time-dependent relationship between variables. While methods exist for finding closed form solutions to certain difference equations [44], the application of these solution methods to large SDF programs would be extremely difficult in practice, due in part to the varying levels of granularity and transparency in SDF models. Although a node may contain a process to implement some

function "*f*" to map a set of inputs *X* to a set of responses *Y*, no guarantee is made that "*f* " itself is retrievable from the program.

6.1.3 Review of SDF Graph Analysis Literature

Since their structures are strongly graph based, synchronous data flow programs lend themselves well to graph theoretic based analysis. Of specific interest in analysis is the ability to determine which source actors, or inputs, have a causal impact on other actors in the model. A further question is, given a set of source actors that have an effect on a given node, which of those effects are of most importance.

The first question, which inputs have an effect on which outputs, can be addressed using "causality interfaces" [47]. A causality interface, as defined by Zhou & Edward states the dependency that an actor's output signal has on input signals. If the causality interfaces are known for all actors in an actor graph, it is possible to determine the causality interface for the entire network, or composition of actors. In a SDF model consisting of actors where all inputs to an actor have a causal impact on all outputs from that same actor, a simple form of causality can be determined by the reachable matrix of the actor graph. Under this simplifying assumption, if actor *A* is connected to actor *B*, and actor *B* is connected to actor *C*, then the output of actor *A* should have a causal effect on actor *C*.

Just having a causal effect does not necessarily indicate that effect has a meaningful magnitude. Given a set of inputs that are known to contribute to an output's value, the individual effect of some inputs may be overwhelmed by the effect of others. To date, there does not appear to be any literature on ranking or identifying important effects using the actor graph's structure alone.

6.1.4 Inputs / Outputs

In general, the inputs to a data flow program are the values produced by source nodes. Outputs can be values recorded by sink nodes, or values generated by any processing node in the graph.

In practice, two types of inputs exist:

1) Inputs that determine the value of source token production. Source actors themselves are not necessarily inputs, but any parameters controlling the value of the tokens produced by a source actor are. This distinction is required to account for instances where an input parameter is used to model a relationship between numerous source actors. For instance, each population group in the NOEM model may have an individually configurable incidence rate for a given disease, but all of these incidence rates may point to the same global variable. This effectively allows for the configuring of each population group through the same variable.

2) Inputs that are attributes of a processing node and determine the behavior of such nodes. In data flow simulation models, such inputs often take the form of coefficients used in regression functions.

In the scope of this work, input factors will be defined as parameters that are settable attributes of the model. This definition of an input differs from simply considering source actors, and is required to represent the two types of inputs detailed above. A consequence of this definition is that only source nodes that draw their values from model parameters will be considered inputs. Any nodes that read a value from an attribute that is only in scope for that specific node will not be considered an input to the model. This assumption is made that such source nodes are modeled in this explicitly inflexible manner specifically to exclude them from consideration as inputs. This type of input shows up in the NOEM model, usually with respect to "Constant (Ptolemy.actor.lib.const)" actors. A constant actor is a source node that produces tokens of constant value. These actors can either produce tokens with values taken from a parameter in the model, in which case the referenced parameter will be considered an input, or from a hidden attribute of the constant actor, in which case the source will not be considered an input.

Outputs can be defined as the values of an output stream from any actor in the model. It is not enough to only consider values received by sink actor nodes, as the simulation analyst may be interested in intermediate outputs. By this definition, every non-source actor node in the model may be considered an output. In practice, there is usually a very small subset of total outputs that are meaningful and worth examining. It is these meaningful outputs that are the target of interest to the simulation analyst. Which individual outputs are meaningful is a question that is should be addressed by a domain expert.

6.2 GraphML

The initial phase of the Nation building project focused on developing a consistent and slightly simplified graph-based data structure in which to represent the full Ptolemy models. This structure, referred to as GNOME (Graph and Network Objects for Model Elicitation), allows for a greater degree of elicitation, manipulation, and analysis of the models.

The intent of the GNOME data structure is to store the NOEM modules in a clearly defined, neutral graphical form that can be input into other applications where analysis can be performed. This document identifies the architecture and format of the GNOME data structure.

6.2.1 Architecture

The GNOME data structure will serve as a bridge between applications within the GNOME architecture. Figure 53 indicates how the data structure will interact with various applications.

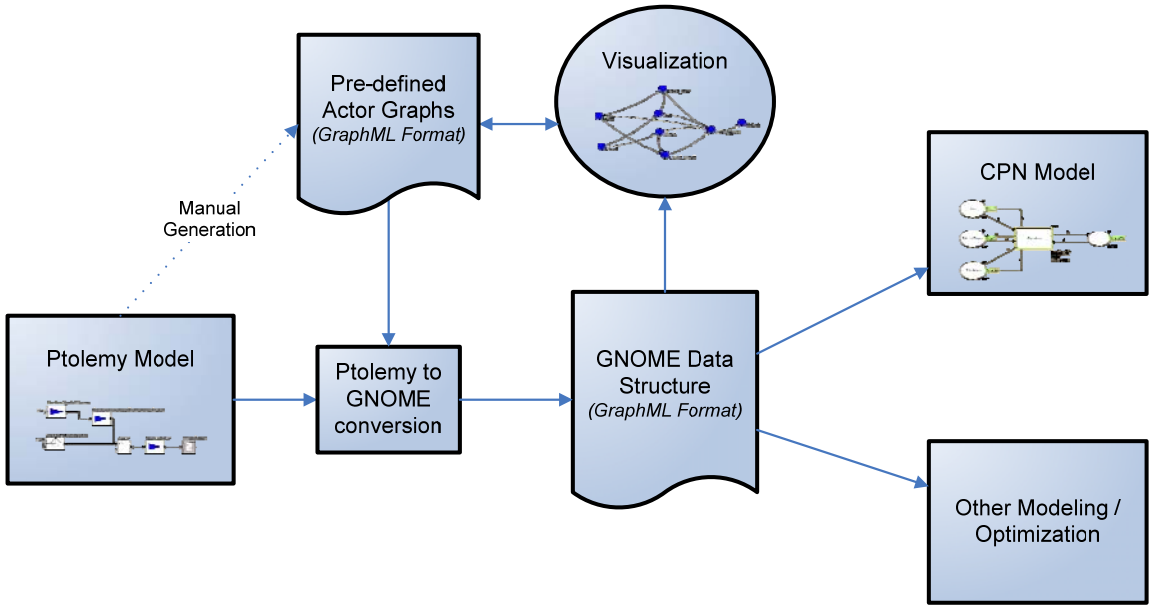


Figure 53: Data Structure Architecture

Currently, the data for NOEM is stored using Ptolemy. Ptolemy offers a strong simulation environment for running the model. However, the Ptolemy models are stored in a custom XML format that is not easily ported to other applications. For this reason, the GNOME data structure is used to provide a more interoperable and easy-to-use graphical XML format to store the NOEM data.

Several common XML formats currently exist for representing graphs. One very flexible format is GraphML, where custom data can easily be represented. Also, programming packages are available that both parse and construct GraphML documents. Since GraphML has these advantages, the GNOME data structure is primarily based on the GraphML format. Storing the NOEM data in this manner makes the process of transforming or loading the data into other applications much simpler.

Information and schemas for GraphML are available at http://graphml.graphdrawing.org/.

6.2.2 GraphML Format

The GraphML format for XML documents uses basic graph-oriented elements to depict a graphical structure. The primary elements are *graph*, *node*, and *edge*. The *graph* element stores a set of nodes and edges. The *node* element represents a point in the graph, and the *edge* element links two nodes in a directed or undirected manner. A basic GraphML document is depicted in Figure 54.

```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
     http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="n0"/>
    <node id="n1"/>
    <node id="n2"/>
    <node id="n3"/>
    <node id="n4"/>
    <node id="n5"/>
    <node id="n6"/>
    <node id="n7"/>
    <edge id="e1" source="n0" target="n2"/>
    <edge id="e2" source="n1" target="n2"/>
    <edge id="e3" source="n2" target="n3"/>
    <edge id="e4" source="n3" target="n5"/>
    <edge id="e5" source="n3" target="n4"/>
    <edge id="e6" source="n4" target="n6"/>
    <edge id="e7" source="n6" target="n5"/>
    <edge id="e8" source="n5" target="n7"/>
  </graph>
</graphml>
```

Figure 54: Example GraphML Document

To make a more complex graph, several other elements are available for use. These include *key*, *data*, and *port*. The *key* element and *data* element work together to define custom data that belongs to a specific category. The *key* element is specified outside of any graphs to establish a data category with an id, name, type, and element usage (what type of elements the data is for). With a key in place, *data* elements are specified within other elements (such as *node* and *edge*) and reference keys to provide specific data values for elements.

The *port* element is used to provide a node with multiple connection points for edges. This allows different types of connections to be appropriately classified when needed. The *edge* elements will then indicate the source and target ports along with the source and target nodes. *Port* elements can also contain data elements.

The GraphML documents used for the GNOME data structure contain a standard set of keys to reflect the data of interest in GNOME. Also, the set of data keys is intentionally designed to closely match the structure of a Colored Petri-Net (CPN) model, since this type of model is one of the primary applications of the GNOME data structure. This similarity makes for an easy transformation between the GNOME data structure and the CPN input.

The XML document below shows an example graph using the GNOME data structure and includes the standard set of data keys used for all documents. Figure 55 illustrates the process of squaring a parameter using a generic “Multiply” node.

```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
       http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <key id="nm" for="node" attr.name="name" attr.type="string"/>
  <key id="dtype" for="port" attr.name="data_type" attr.type="string"/>
  <key id="ptype" for="port" attr.name="port_type" attr.type="string"/>
  <key id="pval" for="port" attr.name="port_value" attr.type="int"/>
  <key id="fn" for="node" attr.name="function" attr.type="string"/>
  <key id="fnprim" for="node" attr.name="function_primary_var"
       attr.type="string"/>
  <key id="inval" for="node" attr.name="initial_value" attr.type="string"/>
  <key id="invaltype" for="node" attr.name="initial_value_type"
       attr.type="string"/>
  <key id="wt" for="edge" attr.name="weight" attr.type="int"/>
  <graph id="Ex" edgedefault="directed">
    <node id="p1">
      <data key="nm">Parameter</data>
      <data key="inval">3</data>
      <data key="invaltype">double</data>
    </node>
    <node id="m1">
      <data key="nm">Multiply</data>
      <data key="fn">product</data>
      <port name="i">
       <data key="ptype">input</data>
       <data key="dtype">double</data>
      </port>
      <port name="j">
       <data key="ptype">input</data>
       <data key="dtype">double</data>
      </port>
      <port name="k">
       <data key="ptype">output</data>
       <data key="dtype">double</data>
      </port>
    </node>
    <node id="o1">
      <data key="nm">Output</data>
    </node>
    <edge id="e1" source="p1" target="m1" targetport="i"/>
    <edge id="e2" source="p1" target="m1" targetport="j"/>
    <edge id="e4" source="m1" target="o1" sourceport="k"/>
  </graph>
</graphml>
```

Figure 55: Example GraphML file for multiplication

Figure 56 provides a visualization of the example shown in Figure 55.

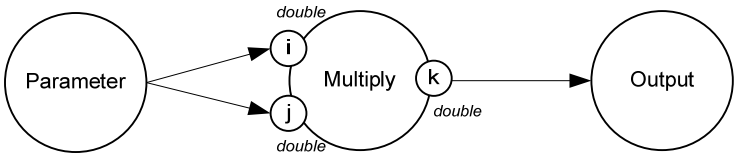


Figure 56: Visualized GraphML Graph

In the case where a node within a graph acts as a sub-graph (represents another graph), the node will contain a *graph* element that references another graph within the document. This nested graph element can also contain additional nodes that act as properties of the sub-graph. The XML document below displays an example of this sub-graph referencing approach. Parts of the document not relevant to the sub-graph referencing approach are not included and denoted by "…".

```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  …
  <graph id="Model" edgedefault="directed">
    …
    <node id="n4">
      <data key="nm">AverageIncome</data>
       <graph id="Average">
         <node id="a1">
           <data key="nm">source</data>
           <data key="inval">income</data>
         </node>
      </graph>
    </node>
    …
  </graph>
  <graph id="Average" edgedefault="directed">
    …
  </graph>
</graphml>
```

Figure 57: Example of a sub-graph implemented using GraphML

6.2.3 Data Transformation

The majority of effort in implementing the GNOME data structure is the transformation of the initial Ptolemy model into a GraphML document. The transformation to other applications using the data structure is fairly straightforward.

Although Ptolemy stores models in a custom XML format (known as MOML), not all of the needed information is available through this document. The program itself uses actors to perform much of the processing that needs to occur. These actors are simply referenced in the XML tags by name without any indication of what each actor actually performs. Because of this, the transformation from the Ptolemy MOML structure to the GNOME data structure requires a predefined set of graphs in GraphML format that represent each actor's functionality. For the most part, these graphs will need to be manually defined.

The *Average* actor within Ptolemy has been represented in GraphML format to illustrate the transformation process. This actor, which is a java class, basically keeps a running sum and count of all input it receives, and it uses the ratio of the sum to the count to output an average value. There is also an additional input to reset both the sum and count. Figure 58 depicts the

graphical structure (nodes, edges, and ports) of the GraphML representation of the Average actor.

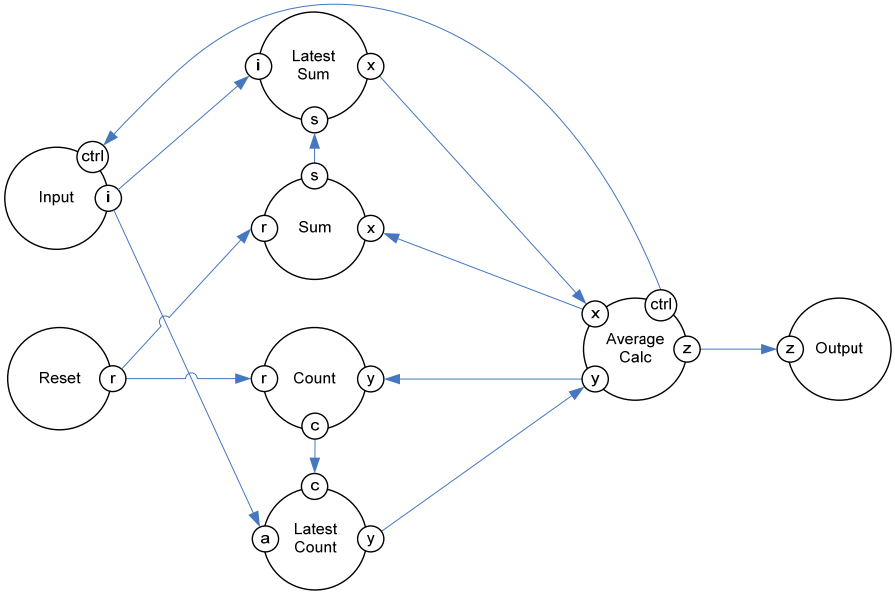


Figure 58: GraphML Graph for Average Actor

The GraphML representation of the Average actor can be easily translated into a CPN model by taking into account both the graph structure and the data elements for the various components of the graph. Figure 59 displays a CPN model (see Section 6.3) manually generated from some simple algorithms used on the GraphML structure.

Although the example of the Average actor uses mostly manual data transformation, the final process will utilize Java-based applications to perform this transformation.

An application has been developed to transform the Ptolemy MOML model into the GNOME data structure. This application can reference a database of GraphML graphs whenever specific actors need to be represented. These graph snippets will be treated as sub-graphs within the document. Currently, only a few graphs are available in this transformation process. This application uses Apache XMLBeans [50] tools to easily parse XML documents conforming to the Ptolemy schema and generate XML documents conforming to the GraphML schema. The Health, Demographics, and Crime modules have been converted to the GraphML format from a saved Ptolemy model (down to the Actor level only) using this automatic transformation process.

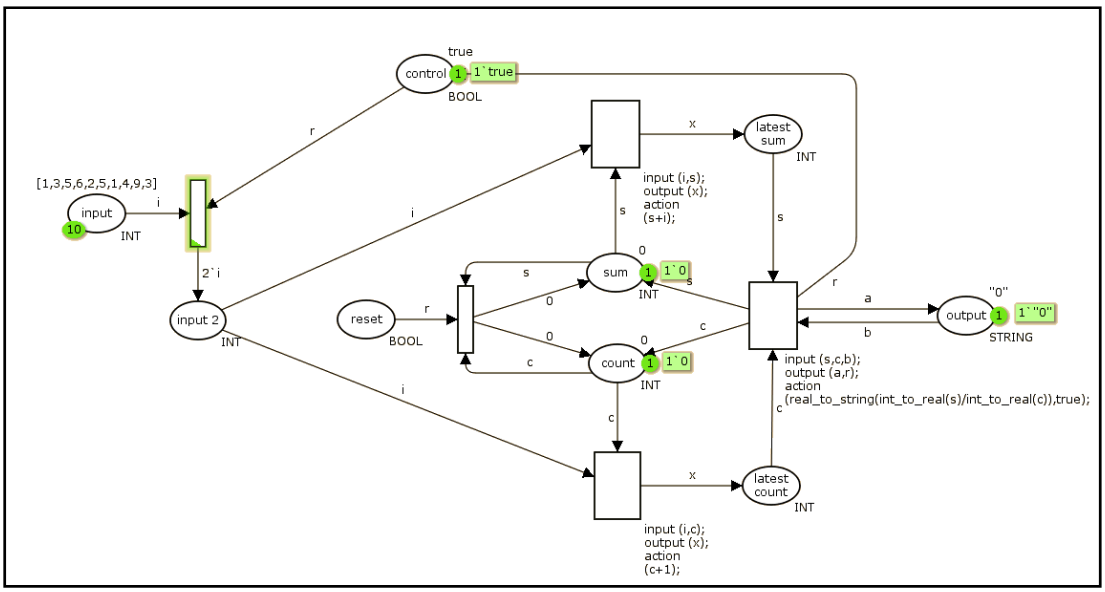


Figure 59: CPN Model for Average Actor

6.2.4 Visualization

In order to visualize and interact with the graphs contained within the data structure, a GraphML Viewer has been developed. This application utilizes the Jung graphical package for Java. The Jung package already has some built in functionality to parse and write simple GraphML files. This capability has been extended in order to parse and write GraphML files that are consistent with the GNOME data structure. Development of this GraphML viewer has continued through a separate GNOME contract.

Figure 60 shows a screenshot of the current GraphML Viewer. This viewer provides a panel to graphically display the nodes and edges of a graph. When a node is selected, the attributes, ports (if any), and other graphical properties are displayed in the remaining panels.

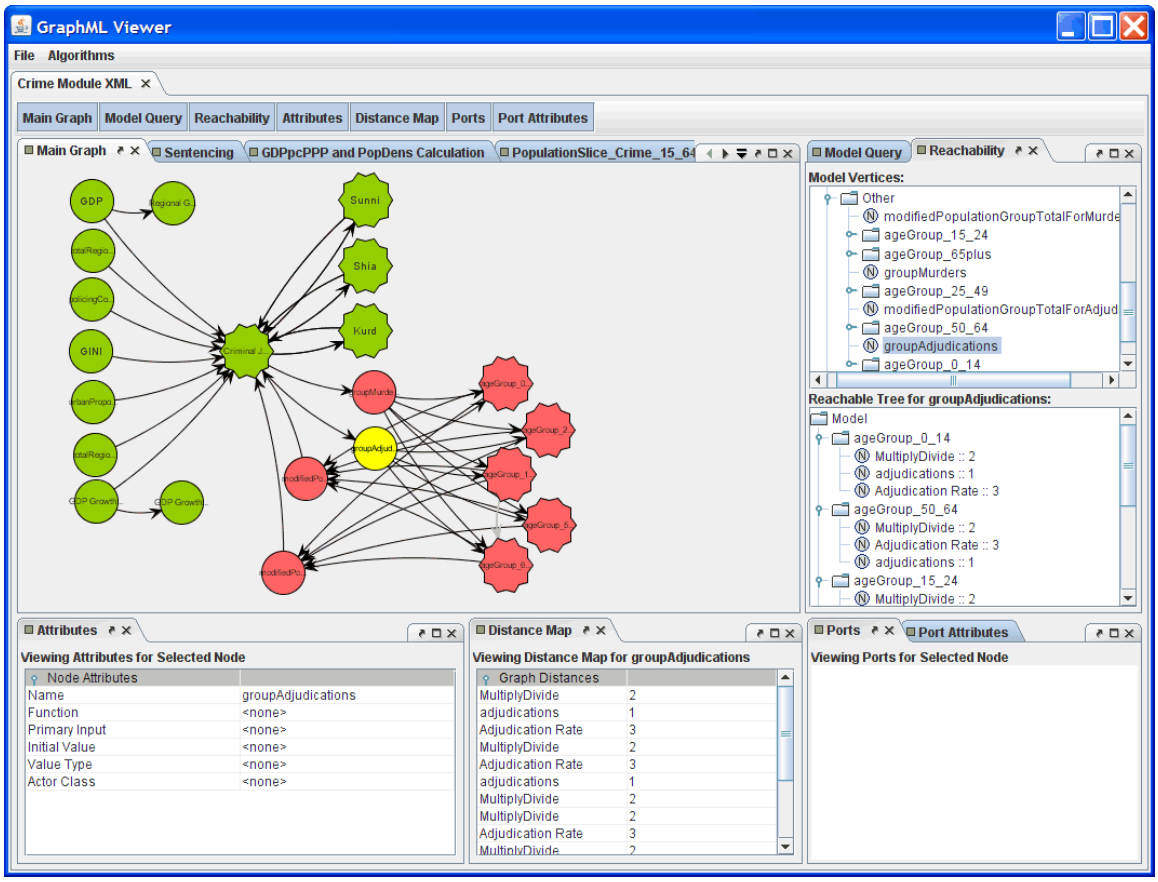


Figure 60: GraphML Viewer

6.3 Petri Nets

Currently, nation-building problems are intensively studied based on simulation approaches, and a lot of data will be generated by running simulations for further data analysis. The main disadvantage of this method is the low efficiency because it is very time consuming to generate the large amount of data. Another disadvantage is that the simulation model is not straightforward for model analysis. Therefore, we need to solve the two problems as follows:

(1) Can we improve the simulation efficiency?

(2) Can we get a new model which is easier for model analysis?

To solve these problems, we need to find a new tool and methodology for both modeling and model analysis.

There are three objectives for this research work:

(1) Finding new simulation tools;

(2) Developing and validating new models based on the new tool;

(3) Conducting model analysis.

By achieving the research objectives above, we can gain the following benefits:

(1) Improving simulation efficiency;

(2) Conducting more effective model analysis.

The main technical findings are as follows:

(1) A Java program has been finished to convert Ptolemy model to Petri net model automatically;

(2) The Petri net model has been validated by comparing the computational results from Ptolemy model;

(3) A simplified Petri net model has been developed and several new methods (incidence matrix analysis, reachability analysis, etc.) have been found for model analysis purpose.

6.3.1 Overview

Currently, several different tools are used to develop the nation-building model for different purposes. The comparison is as follows:

Table 21: Comparison of different models

	Colored Petri Nets	Basic Petri Nets	GraphML	Ptolemy Model
Structural Representation	Simple	Complex	Complex	Complex
Purpose	System Properties Analysis	Simulation/ Analysis	Graph Analysis	Simulation
Data for Study	Model Structure Parameters/ Markings	Simulation Input/Output	Graph Topology	Simulation Input/Output
Approach	Incidence Matrix Analysis/Reachability Analysis/Stochastic Process Analysis	Detailed PN properties/ Statistics/DOE	Algorithms in Graph Theory (e.g. shortest path)	Statistics/DOE

6.3.2 Basic (Black-white) Petri net model

Basically, a ***Petri net*** is a weighted bipartite graph *PN= (P, T, A, w, x)*, where

- ***P*** is a finite set of ***places,*** $P=\{p_1,\ldots,p_n\}$;
- ***T*** is a finite set of ***transitions***, $T=\{t_1,\ldots,t_m\}$;
- ***A*** is the set of ***arcs***, represented by (p_i, t_j) or (t_j, p_i), from places to transitions and from transitions to places;
- ***w*** is the ***weight function*** on arcs;
- ***x*** is the ***marking vector*** $x=[x_1,\ldots,x_n]$ represents the number of tokens in each place.

The basic Petri net model is mainly used for simulation, and it has a higher efficiency compared with the old Ptolemy model. A Petri Net model is developed by first decomposing the Ptolemy model into basic units. A Petri-net is then designed to represent each of these basic units. These Petri Nets of the basic units are then combined into a single Petri Net representing the entire Ptolemy model. Some snapshots of the Petri net are shown as follows:

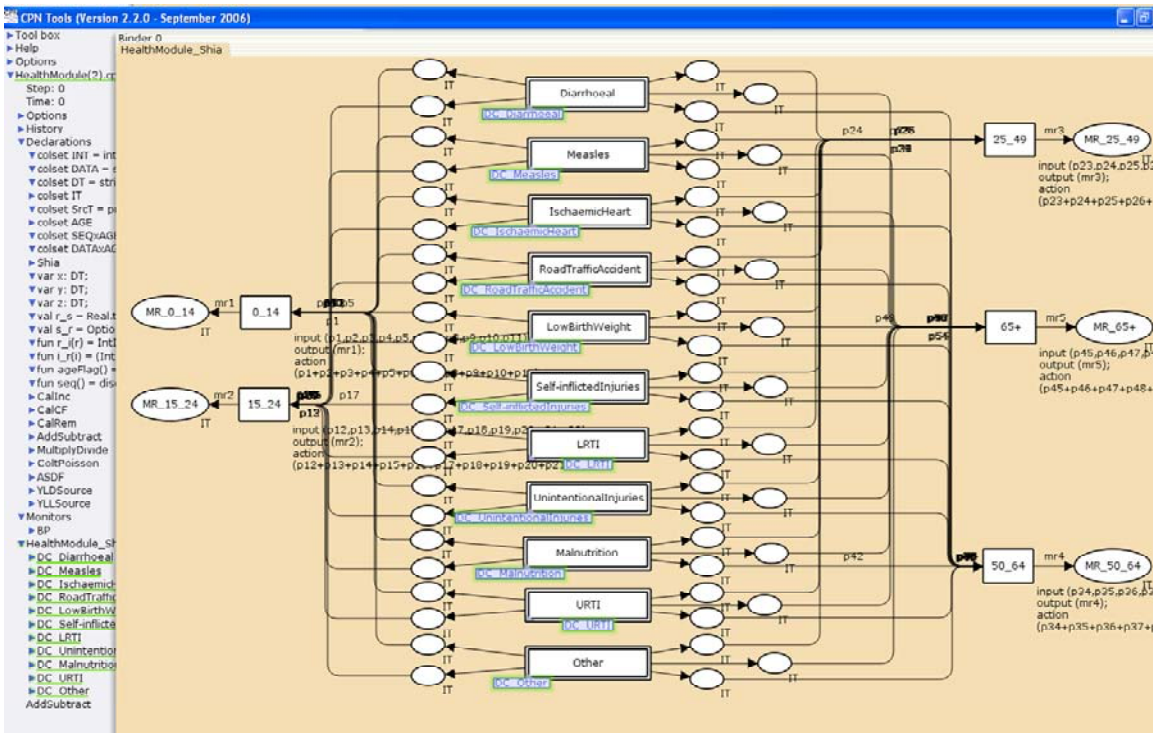


Figure 61: Overall structure for "Health" module in basic Petri net model

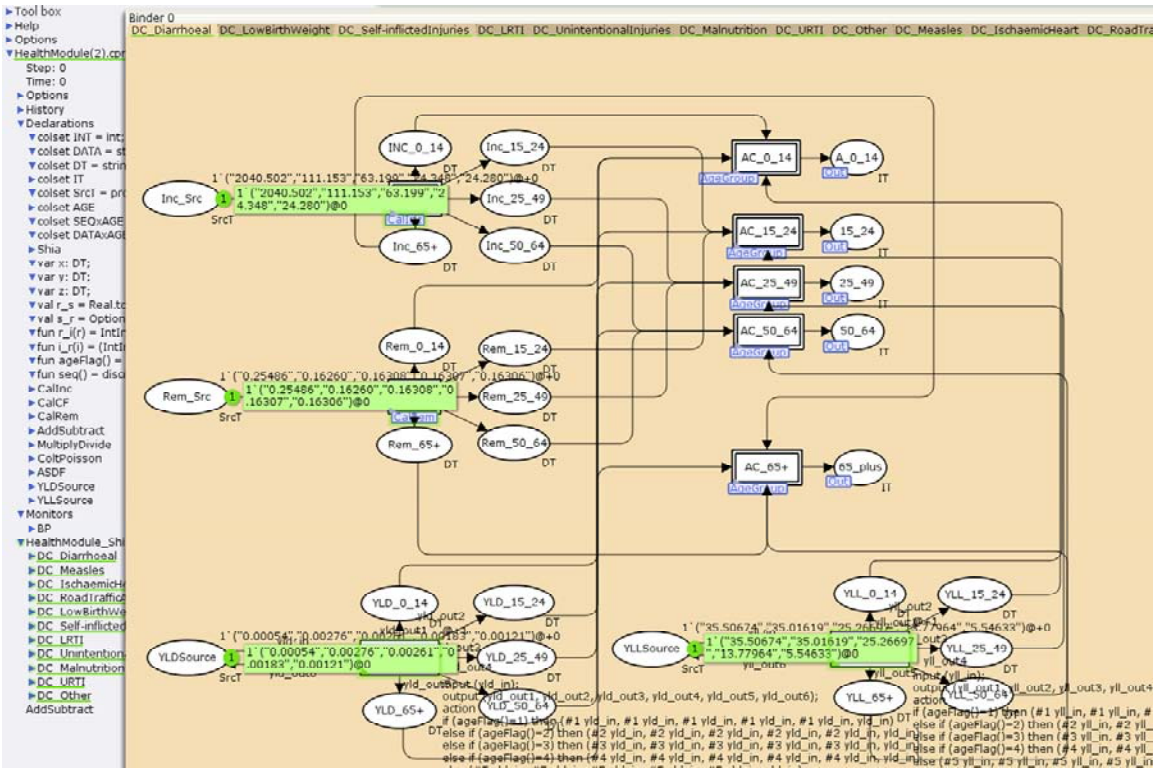


Figure 62: Basic module for diseases in "Health" module

The basic Petri net model has been validated by comparing the computational results from the Ptolemy model. To do this, we run simulations on two models individually, and then compare the outputs and analyze the similarity.

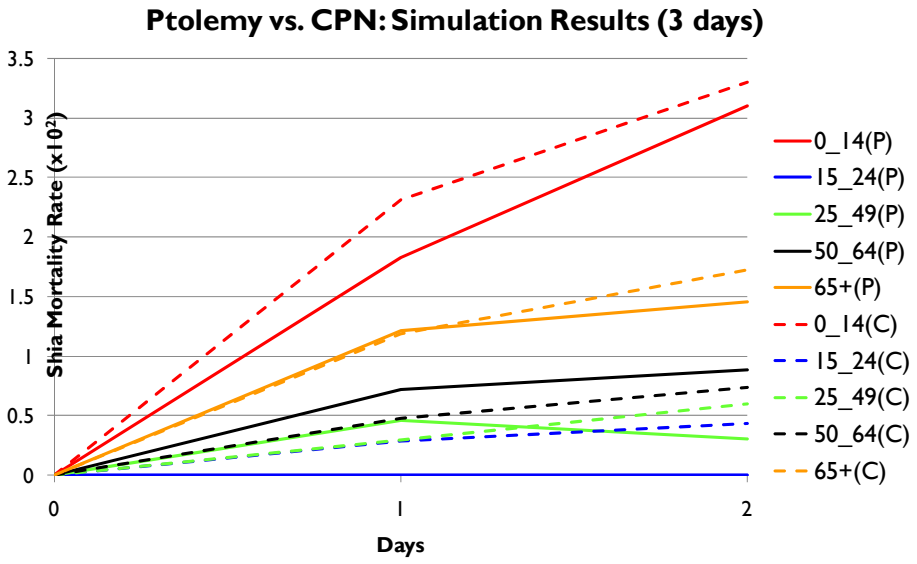


Figure 63: Comparison 1 (simulation time: 3 days)

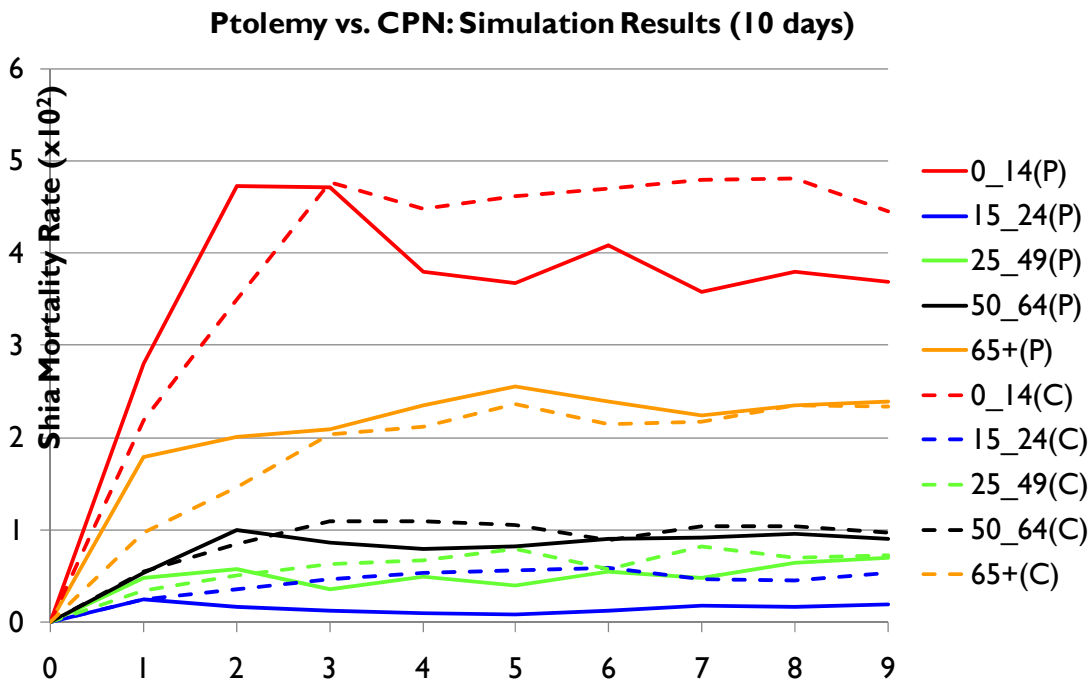


Figure 64: Comparison 2 (simulation time: 10 days)

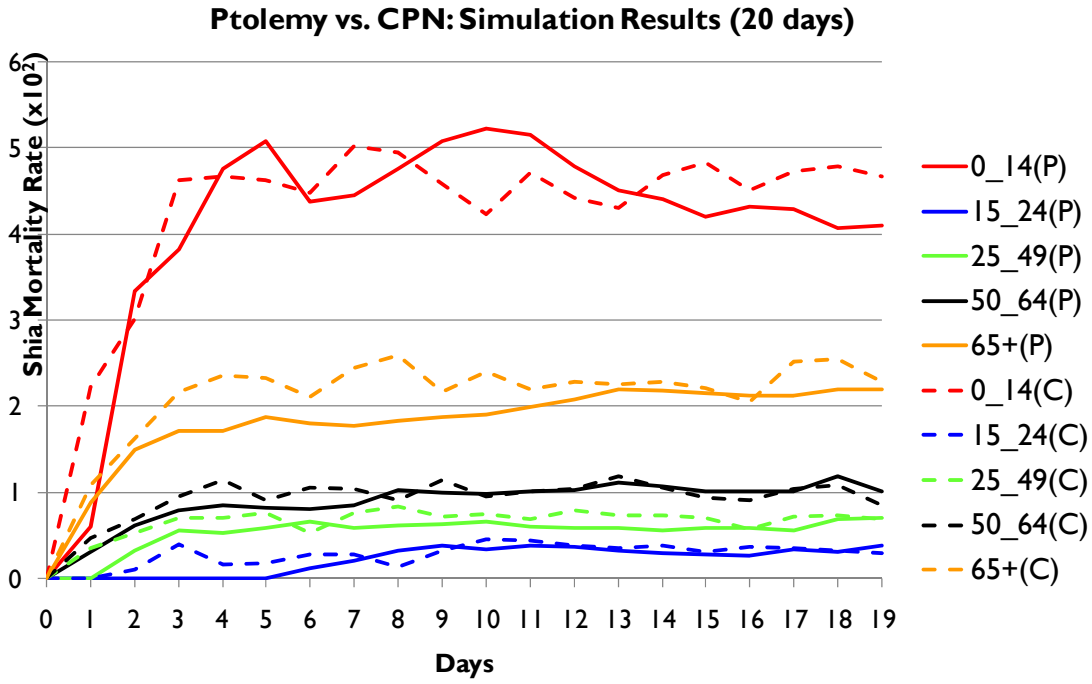


Figure 65: Comparison result 3 (simulation time: 20 days)

Figure 63 to 65 above are comparison results from "Health" module with varying simulation times. In a figure, x-axis is the simulation time, and y-axis is the mortality rate. In "Health" module, an ethnic group is divided into five age groups: "0-14", "15-24", "25-49", "50-64", "65+". Both models will calculate the mortality rate for each age group. To differentiate the results from different model, solid lines are used to represent the results from Ptolemy model, and dashed lines are used to represent the results from Petri net model. Each of the three figures illustrate that the Petri Net results are very similar to the Ptolemy results.

6.3.3 Colored Petri net model

The colored Petri net model is developed for efficient model analysis. The goal is to find out the key components of the model to predict simulations results without running a large number of simulation replications. The structure of the colored Petri net model is as follows:

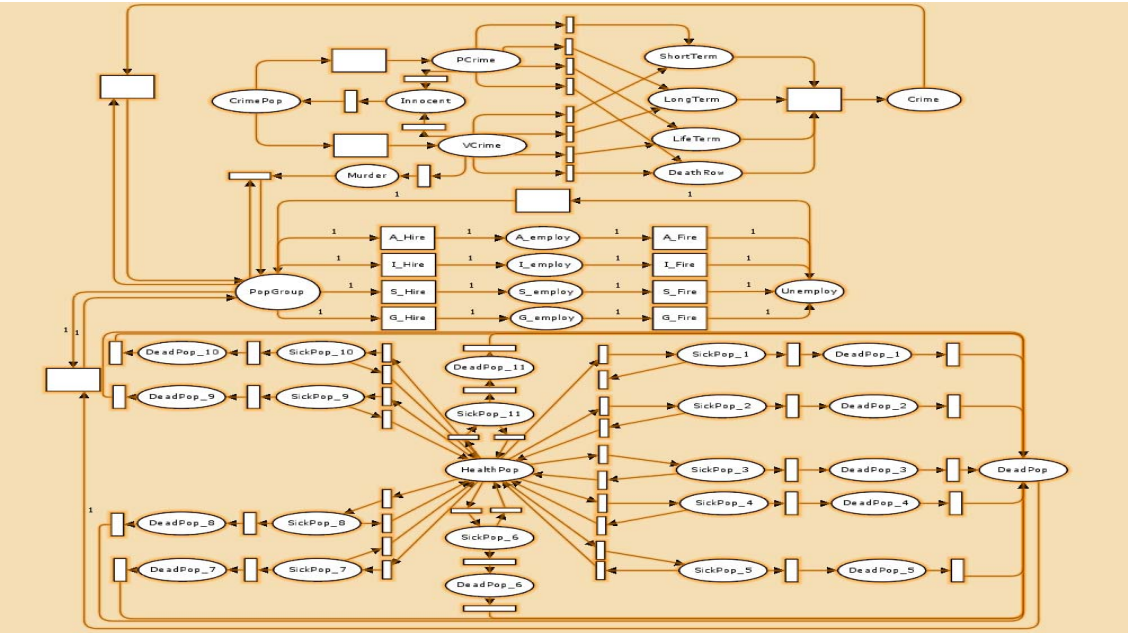


Figure 66: Model structure for simplified Petri net model

6.3.4 Model Analysis Approaches

Several model analysis approaches have been developed to conduct model elicitation based on analyzing the model structure. A Petri net can be uniquely represented by a matrix called the incidence matrix. Therefore, the model analysis techniques are developed based on analyzing incidence matrix.

6.3.4.1 Chain Analysis

The purpose is to find the chains which indicate series of consequences in the system. To do this we need to find the matrix blocks characterized by the form of:

$$\begin{bmatrix} a_1 & 0 & 0 & 0 & 0 \\ -b_1 & a_2 & 0 & 0 & 0 \\ 0 & -b_2 & 0 & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & 0 & -b_{n-1} & a_n \\ 0 & 0 & 0 & 0 & -b_n \end{bmatrix}$$

The benefits of the chain analysis are as follows:

(1) Finding series of consequences, which implies the relationship between elements in the system;

(2) Indicating that stochastic process (e.g. Markov Chain) may apply;

(3) Improvement over GraphML approach.

6.3.4.2 Coupling Analysis

The purpose is to analyze the interdependency of different modules and find the connection factors. The module interdependency is shown in Figure 67.

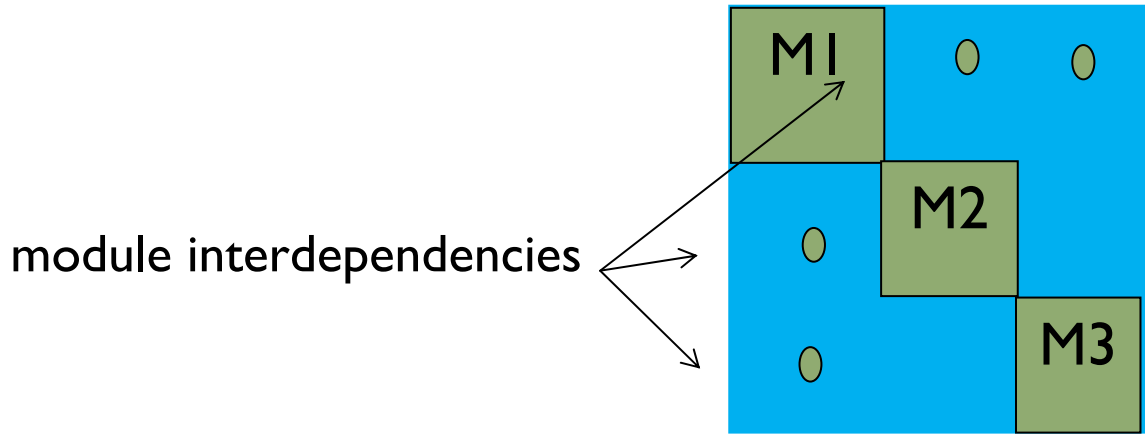


Figure 67: Illustration of Module interdependency

The procedure for coupling analysis is as follows:

(1) Partition (block-diagonalize) the incidence matrix to different blocks based on the model structure;

(2) Finding out the overlapping entries indicating connection factors;

(3) Calculate the degree of connection based on some rules, e.g. the summation of the number of overlap entries.

Some conclusions from coupling analysis are as follows:

(1) The coupling indices between Health(**H**), Crime(**C**) and Demographic(**D**) module are: C(**H**,**C**) =1, C(**H**,**D**) =2, C(**C**,**D**) =4; Therefore,
(2) The dependency between Health and Crime module is less than that between Health and Demographic module;
(3) The dependency between Health and Crime module is less than that between Crime and Demographic module;
(4) The dependency between Health and Demographic module is less than that between Crime and Demographic module.

6.3.4.3 Reachability Analysis

The definition for reachability is as follows:

- *Definition:* For a net system $\boldsymbol{S}=<\boldsymbol{N}, \mathbf{m}_0>$, the set
  $RS(\boldsymbol{S}) = RS(\boldsymbol{N}, \boldsymbol{m_0}):=\{\mathbf{m} \mid \exists\, w \in T^*, \mathbf{m}_0 \rightarrow \mathbf{m}\}$

is the ***reachability set***.

$FS(\boldsymbol{S}):=\{w \in T^* \mid \exists\, w \in T^*, \mathbf{m}_0 \rightarrow \mathbf{m}\}$ is the set of occurrence-transition sequences (or firing-sequence set) of ***S***.

The objective of reachability analysis is to check if there exists a sequence $\boldsymbol{\sigma}$ of transitions which leads to a given marking ***M*** starting from an initial marking $\boldsymbol{M_0}$, i.e. $\boldsymbol{M_0} \rightarrow \boldsymbol{M}$. For nation-building, it can be used to study whether the stabilization defined by some criteria can be achieved given initial conditions, or predict the approximate time needed for stabilization.

The procedure for Reachability analysis is as follows:

(1) Calculate counting vector by matrix computation;

(2) Finding firing sequences of transitions from the counting vector;

(3) Calculating the time duration for the firing sequence.

There are three types of matrices in a CPN:

(1) Constant

$\boldsymbol{p}$ = # of places, $\boldsymbol{t}$ = # of transitions, $\boldsymbol{c}$ = # of colors

(2) Marking Matrix $\boldsymbol{Mp \times c}$, in which entry $\boldsymbol{m_{i,j}}$ = # of tokens with color ***j*** in each place ***i***

(3) Model Structure Matrix ***Uc×p×t***

$$u_{i,j,k} = \begin{cases} -w_{i,j,k}, \ if \ j = {}^0k \ \text{(To token color } i\text{, \# of tokens taken away from place } j \text{ when transition } k \text{ fires)} \\ w_{i,j,k}, \ if \ j = k^0 \ \text{(To token color } i\text{, \# of tokens added to place } j \text{ when transition } k \text{ fires)} \\ 0, \ \text{otherwise} \end{cases}$$

Counting Vector ***Vσ*** is a vector in which v_i = # of transition ***i*** 's in the firing sequence ***σ***

The procedure to calculate the counting vector is as follows:

(1) For any color ***i***, calculate counting vector ***Vσ(i)*** by

$$U_{i\times p\times t} \cdot V_\sigma(i) = M_{p\times c}(*,i) - M^0_{p\times c}(*,i)$$

(2) Comparing all ***Vσ(i)'s*** ($1 \le i \le c$) and finding out whether they are compatible;

(3) If they are compatible, searching firing sequence ***σ***; otherwise there is no such a firing sequence.

Firing sequences of transitions can be found from counting vectors by solving the IP model as follows:

$$\text{Let } x_{t,q} = \begin{cases} 1, & \text{if transition } t \text{ is fired at order } q \\ 0, & \text{otherwise} \end{cases}$$

$m_{p,c} = \#$ of tokens of color c in place p initially

$$u_{c,p,t} = \begin{cases} -w_{c,p,t}, \ if \ p = {}^0t \ \text{(To token color } c\text{, \# of tokens taken away from place } p \text{ when transition } t \text{ fires)} \\ w_{c,p,t}, \ if \ p = t^0 \ \text{(To token color } c\text{, \# of tokens added to place } p \text{ when transition } t \text{ fires)} \\ 0, \ \text{otherwise} \end{cases}$$

So the IP model is

$$\min \sum_{t,q} x_{t,q}$$

s.t.

$$m_{p,c} + \sum_{t,q} u_{c,p,t} \cdot x_{t,q} \ge 0, \quad \forall p,c,q$$

$$\sum_q x_{t,q} = V_\sigma(t), \quad \forall t$$

The procedure for calculating time duration of a firing sequence is as follows:

(1) Finding all the chains in the firing sequence;

(2) Compute the time duration for each chain by summing up the firing time of transitions;

(3) The maximum value is the time duration of the firing sequence.

An example of extended scenario from "Health" module is shown to illustrate the application of reachability analysis on nation-building. The model structure is as follows:

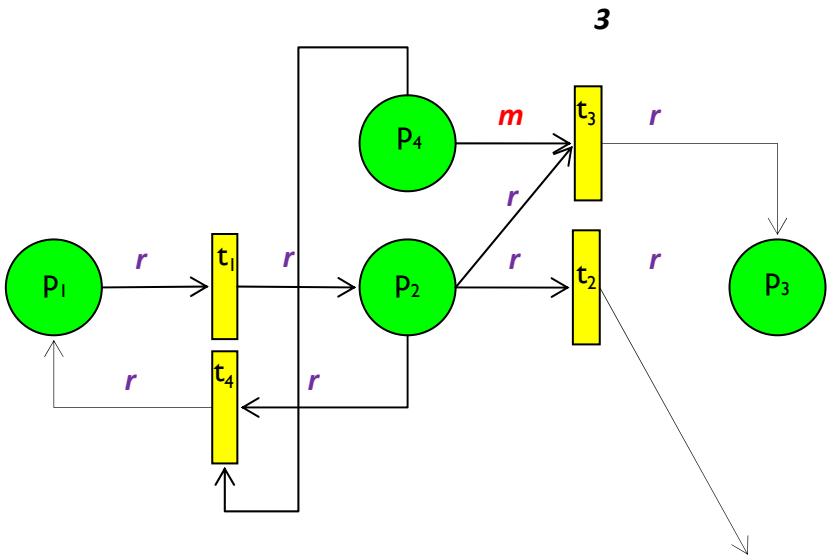


Figure 68: CPN model for the example

The scenario is explained as follows:

(1) Token color: ***r***---people, ***m***---medicine;

(2) Place: p_1---healthy people, p_2---sick people, p_3---dead people, p_4---medicine for the disease;

(3) Transition: t_1--- health people get sick, t_2, t_3 ---sick people die, t_4---sick people get well;

(4) Assuming that the firing time for t_2 is less than t_3.

The data and calculation is as follows:

(1) $p = 5$, $t = 4$, $c = 2$

(2) Incidence Matrix:

$$U_{1\times p\times t}=\begin{bmatrix}-1 & 0 & 0 & 1\\ 1 & -1 & -1 & -1\\ 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} \qquad U_{2\times p\times t}=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & -1 & -3\end{bmatrix}$$

(3) Markings

$$M^{0}_{p\times c}=\begin{bmatrix}1000 & 0 & 0 & 0\\ 0 & 0 & 0 & 50\end{bmatrix}^{T} \qquad M_{p\times c}=\begin{bmatrix}800 & 150 & 50 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}^{T}$$

(4) Solution

For token color 1:

$$\begin{bmatrix}-1 & 0 & 0 & 1\\ 1 & -1 & -1 & -1\\ 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}=\begin{bmatrix}800\\ 150\\ 50\\ 0\end{bmatrix}-\begin{bmatrix}1000\\ 0\\ 0\\ 0\end{bmatrix} \Rightarrow$$

x_1-x_2=200 (1)

x_2+x_3=50 (2)

For token color 2:

$$\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & -1 & -3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\\ 0\end{bmatrix}-\begin{bmatrix}0\\ 0\\ 0\\ 50\end{bmatrix} \Rightarrow$$

x_3+3x_4=50 (3)

Since (1), (2) and (3) are compatible, the solution is $\boldsymbol{x_1}$=200+3$\boldsymbol{x_4}$, $\boldsymbol{x_2}$=3$\boldsymbol{x_4}$, $\boldsymbol{x_3}$=50-3$\boldsymbol{x_4}$. To satisfy this, a specific solution is $\boldsymbol{x_1}$=200, $\boldsymbol{x_2}$=0, $\boldsymbol{x_3}$=50, $\boldsymbol{x_4}$=0.

6.3.5 Model Converter

A Java program is developed to convert the Ptolemy model to Petri net model automatically. The mechanism is to read and extract key information from Ptolemy model in a XML file, and then generate a Petri net model by an XML file in its format. Generally speaking, the procedure can be represented by the flow chart in Figure 69.

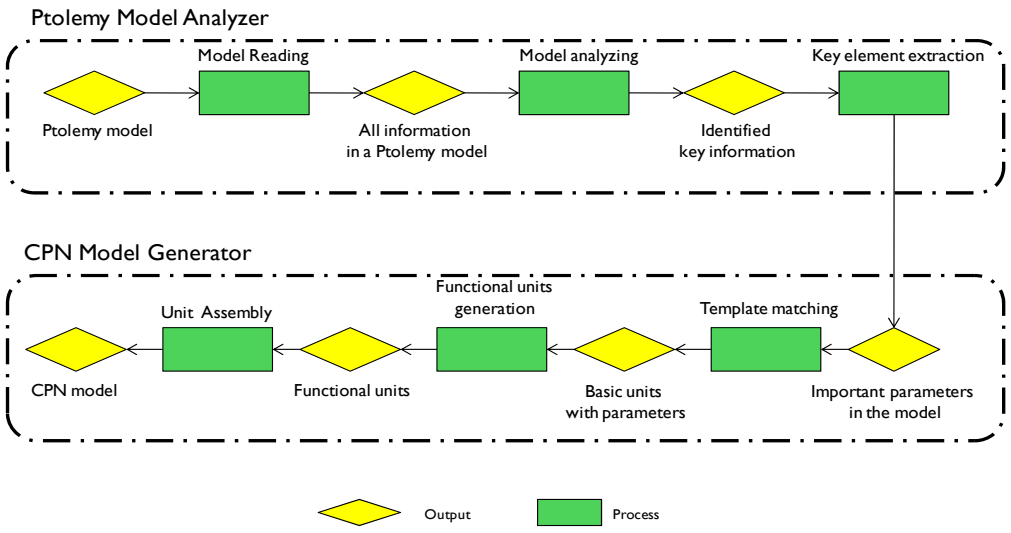


Figure 69: Flow Chart for Automatic Petri net Model Converter

6.4 Markov Chain Analysis

We use Markov chains for developing a model that quantifies the effect of investment in training and infrastructure on the per-capita contribution to the economy. We present the basic building blocks for the model, along with a plan to generate a computer program (in MATLAB) that can be used in optimizing the investment decisions.

6.4.1 Economics Module

We will assume that a limited amount of money is available for investment in multiple channels, e.g., education, building roads and bridges, ensuring power supply, water supply maintenance etc. We will also model each person in the population to be in a state that is defined by his/her employment status and nature of employment. The change that can occur for any individual from one status to another is a stochastic process in any such demographic system, and we will use a probabilistic model to describe these transitions. Clearly, these transitions will depend on the amount of money invested in the various channels. We will develop a model that needs as inputs:

1. The amount of money invested in each channel
2. The transition probability functions associated with each dollar amount,

and delivers as outputs:

1. The proportion of population that in the long run will be in a given employment status
2. The contribution in terms of dollars to the economy.

This model could be simulation-based, but a Markov chain model will be easier to optimize. Hence we will pursue a Markov chain model. Once this model is up and running, we can then optimize this model to determine the amounts of money that should invested in each channel in order to obtain a desired output.

6.4.1.1 Markov Chain Model

For the sake of simplicity, we begin with two channels of investment: 1) education & training and 2) infrastructure development (water, electricity, security). The number of channels of investment can be increased depending on the nature of the model and available data. The other assumption here will be that each person will be in one of the following states:

The economics model can be represented by the states $E1, \dots, E4$, and NE, where NE denotes not employed and Ei denotes employed in the *i*th job-type. The probability of going from one state to another will be assumed to be Markovian, which is a reasonable assumption in demographic studies, and these probabilities will be a function of the money invested in each channel. Using the Markov chain model we can determine the long-run proportion of population in each employment category from the invariance equations of the underlying Markov chain. Furthermore, if an estimate of the population is available, the economic contribution of each investment strategy can be measured in terms of dollars, provided there is a way to quantify the contribution of a given employment category to the economy.

Consider Figure 70 below. Let Π_i denote the invariant probability of being in state i. Let $P_S(i,j)$ denote the probability of transiting from state *i* to *j* under investment strategy S. Then one can use the invariance equations to determine the invariant probabilities. From the invariance probability distribution, one can also generate a model for the economic contribution of each strategy. The invariance equations are as follows:

$$\vec{\Pi}^T \boldsymbol{P}_S = \vec{\Pi}^T \tag{26}$$

$$\sum_{i=1}^{n} \Pi(i) = 1 \tag{27}$$

where the first equation is the so-called invariance equation in which $\vec{\Pi}$ denotes the column vector of the invariant (limiting) probabilities, T denotes the transpose, and P_s denotes the transition probability matrix associated with strategy S. Note that the element in the *i*th row and *j*th column of this matrix is $P_s(i,j)$. The second equation above is the normalizing equation in which $\Pi(i)$ denotes the *i*th element of the vector $\vec{\Pi}$ and n denotes the number of elements in the Markov chain.

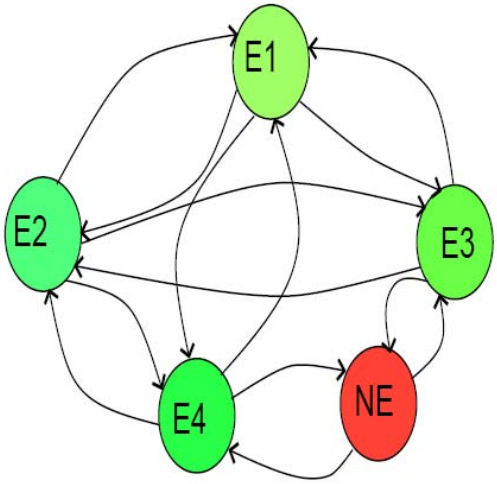


Figure 70: A schematic showing the Employment Markov Chain. Along the arrows, one typically has transition probabilities, which depend on the strategy on hand.

6.4.1.2 Optimization

The economic implication of each strategy can be formulated as:

$$E = N \cdot \sum_{i=1}^{n} r_i \Pi_S(i) \tag{28}$$

where N denotes the population and r_i denotes the economic contribution of each person in employment state i. Clearly, if the transition probabilities are a function of the investment ratios, using gradient descent, one can optimize the above revenue with respect to the investment ratios.

6.4.2 Health Module

We assume (consistent with the Ptolemy model) that for each population-group (Shia, Sunni, Kurd, and other), age-group, and disease, there are Poisson processes that account for infection (I) and recovery (R). Each person can be in any of the 3 states when observed: Healthy (H), Sick (S) and Dead (D). We can model this system with a discrete-time Markov chain (MC) in which the state D is an absorbing state, and S and H are transient states. We will assume (consistent with Ptolemy model) that each individual starts out as healthy. Using transient Markov chain theory, one can find the amount of time (transitions of the Markov chain) an individual spends before being absorbed into the absorbing state. This time is equal to the longevity (L) of an individual. If N denotes the initial population, the mortality rate (number of deaths per unit time) can be given by:

$$MR = N/L \tag{29}$$

We will choose the unit of time to be day. Thus the time for one transition of the Markov chain will be 1 day. Now, we need to set up the transition probabilities of the MC.

6.4.2.1 MC model

Let λ_I denote the Poisson rate of infection per day, let λ_R denote the Poisson rate of recovery per day, and let λ_X denote the Poisson rate of death per day for a sick person. We assume that a person has to go through the sick stage before death. Let $P(i,j)$ denote the one-step transition probability of going from state i to j. Here the states are: H, S, and D. Let I denote the time to get infected for a healthy person, R the time to recover for a sick person and X the time to die for a sick person. Then:

$$P(H,S) = Pr(I < 1) = 1 - e^{-\lambda_I \cdot 1} = 1 - e^{-\lambda_I} \tag{30}$$

Clearly, hence $P(H,H) = 1 - P(H,S)$, since $P(H,D) = 0$. For going from sick to heathy in 1 day, there are two possible ways:

Both R and X are less than 1 day but $R < X$.

$R < 1$ and $X > 1$.

Hence,

$$P(S,H) = Pr(R < X < 1) + Pr(R < 1)Pr(X > 1) \tag{31}$$

Now, $\Pr(R < 1) = 1 - e^{-\lambda_R}$ and $Pr(X > 1) = e^{-\lambda_X}$. For $Pr(R < X < 1)$ we need to do some additional work. We know that for any continuous non-negative random variables,

$$\Pr(R < X < a) = \int_0^a F_R(x) f_X(x) dx \tag{32}$$

where $f(\cdot)$ denotes the pdf, $F(\cdot)$ the cdf, and a is a scalar. For the exponential distribution (which is the distribution for R, X, and I) of a random variable Y , $f_Y(a) = e^{-\lambda_Y} a$ and

$F_Y(a) = 1 - e^{\lambda_y a}$. Hence:

$$\begin{aligned}\Pr(R < X < 1) &= \int_0^a F_R(x) f_X(x) dx \\ &= \int_0^a \left(1 - e^{-\lambda_R x}\right) \lambda_x e^{-\lambda_X x} dx \\ &= \left[1 - e^{\lambda_X}\right] - \frac{\lambda_x}{\lambda_X + \lambda_R}\left[1 - e^{-(\lambda_X + \lambda_R)}\right]\end{aligned} \tag{33}$$

Thus:

$$\Pr(R < X < 1) = \left[1 - e^{\lambda_X}\right] - \frac{\lambda_x}{\lambda_X + \lambda_R}\left[1 - e^{-(\lambda_X + \lambda_R)}\right] \tag{34}$$

Then:

$$\Pr(S, H) = \left[1 - e^{\lambda_X}\right] - \frac{\lambda_x}{\lambda_X + \lambda_R}\left[1 - e^{-(\lambda_X + \lambda_R)}\right] + \left(1 - e^{-\lambda_R}\right) e^{-\lambda_X} \tag{35}$$

Similar to Equation (34), we can work out the following:

$$\Pr(R < X < 1) = \left[1 - e^{\lambda_R}\right] - \frac{\lambda_R}{\lambda_X + \lambda_R}\left[1 - e^{-(\lambda_X + \lambda_R)}\right] \tag{36}$$

Note that for going from being sick to dying in 1 day, there are two possible ways:

1. Both R and X are less than 1 day but $X < R$.
2. $R > 1$ and $X < 1$.

Hence,

$$P(S, D) = Pr(X < R < 1) + Pr(R > 1) Pr(X < 1) \tag{37}$$

Similarly, then we have:

$$\Pr(S, D) = \left[1 - e^{\lambda_R}\right] - \frac{\lambda_R}{\lambda_X + \lambda_R}\left[1 - e^{-(\lambda_X + \lambda_R)}\right] + \left(1 - e^{-\lambda_X}\right) e^{-\lambda_R} \tag{38}$$

Clearly:

$$P(S, S) = 1 - P(S, H) - P(S, D) \tag{39}$$

Then, we have that the transition probability matrix is:

$$P = \begin{bmatrix} P(H,H) & P(H,S) & P(H,D) \\ P(S,H) & P(S,S) & P(S,D) \\ P(D,H) & P(D,S) & P(D,D) \end{bmatrix} \quad (40)$$

where the elements are defined above; note that $P(D,D) = 1$, and $P(D,H) = P(D,S) = 0$.

6.4.2.2 Mean time in a state

If $\boldsymbol{P_T}$ denotes the truncated transition probability matrix with only the transient states, then:

$$S = [\boldsymbol{P_T} - \boldsymbol{I}]^{-1} \quad (41)$$

where $\boldsymbol{I}$ denotes the identity matrix. Here $S(i,j)$ which denotes the term in the ith row and jth column of $\boldsymbol{S}$ is equal to the mean time spent in state j before entering the absorbing state if the system starts in i. For our model,

$$\boldsymbol{P_T} = \begin{bmatrix} P(H,H) & P(H,S) \\ P(S,H) & P(S,S) \end{bmatrix} \quad (42)$$

and the longevity will be:

$$L = S(H,H) + S(H,S) \quad (43)$$

7 References

7.1 Information Fusion

[1] White, F. A Model for Data Fusion. In Proceedings of 1st National Symposium on Sensor Fusion, 1988

[2] A. Stotz, M. Sudit, "INformation Fusion Engine for Real-time Decision-making (INFERD): A Perceptual System for Cyber Attack Tracking", Proceedings of the ISIF Conference, Quebec City, QE, CA, July 2007.

[3] Argauer, B. VTAC: Virtual Terrain Assisted Impact Assessment for Cyber Attacks. Master's Thesis. Rochester Institute of Technology, 2007.

[4] Bass, T. "Intrusion detection systems and multisensor data fusion," Communications of the ACM 43, April 2000.

[5] Holsopple, J., Yang, S. J. and Sudit, M. TANDI: Threat assessment for networked data and information. In Proceedings of SPIE, Defense and Security Symposium, vol. 6242, April 2006.

[6] Phillips, C. and Swiler, L. A graph-based system for network vulnerability analysis. In Proceedings of the 1998 workshop on new security paradigms, pages 71–79, New York, NY, USA, 1998. ACM Press.

[7] Jajodia, S., S. Noel and B. O'Berry. Topological analysis of network attack vulnerability. Managing Cyber Threats: Issues, Approaches, and Challenges. 2003

[8] Lippman, R. and Ingols, K. An annotated review of past papers on attack graphs. Technical Report ESC-TR-2005-054. Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA. March 2005.

[9] Mitre, "Common vulnerabilities and exposures (CVE dictionary)." http://cve.mitre.org, 2007.

[10] Ning, P. and Xu, D. Learning attack strategies from intrusion alerts, In the Proceedings of the 1-th ACM Conference on Computer and Communications Security, New York: ACM Press, pp. 200-209.

[11] OPNET Technologies Inc., http://www.opnet.com.

[12] Schneier, B. Attack Trees. Dr Dobb's Journal. December 1999.

[13] Sourcefire, "Snort: an open source network intrusion prevention and detection system." http://www.snort.org, 2007.

[14] M. Sudit, A. Stotz, and M. Holender, "Situational awareness of a coordinated cyber attack," in Proceedings of SPIE, Defense and Security Symposium, pp. 114–129, March 2005.

[15] T. N. S. Inc., "Nessus vulnerability scanner." http://www.nessus.org/, 2007.

[16] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," IEEE Transactions on Dependable and Secure Computing 1, pp. 146 – 169, July-Sep 2004.

[17] Vidalis, S. and Jones A. Using vulnerability trees for decision making in threat assessment. Technical Report CS-03-2, University of Glamorgan, School of Computing, June 2003.

[18] Yang, S. J., et al. Terrain and Behavior Modeling for Projecting Multistage Cyber Attacks. In proceedings of International Data Fusion Conference. Quebec, Canada. July 2007.

[19] Hall, D., Llinas, J., McNeese, M., and Mullen, S., "A framework for dynamic ha*rd/soft fusion,"* Proceedings of Fusion 2008: the International Conference on Information Fusion, Cologne, Germany, July, 2008

[20] Hall, D. B. Hellar, M. McNeese and J. Llinas, "Assessing the JDL model: a survey and analysis of decision and cognitive process models and comparison with the JDL model," in the *Proceedings of the National Symposium on Sensor Data Fusion*, June 2007

[21] Hall, D, B. Hellar and M. McNeese, "Rethinking the data overload problem: closing the gap between situation assessment and decision making," in the *Proceedings of the National Symposium on Sensor Data Fusion*, June 2007

[22] Ballora, M. Lee Hong, S., Panulla, B., and Hall, D., "Information through sound: understanding data through sonification," *Proceedings of ISEA2008, The 14th International Symposium on Electronic Art*, July 25 – August 2, 2008, Singapore, pp. 47 – 49, 2008

[23] Sudit, M. Stotz, A., Holender, M. Tagliaferri, W., Canarelli, K., and Dasarathy, B., (2006) "Measuring situational awareness and resolving inherent high-level fusion obstacles", *Proceedings of the SPIE*, 19-20 April, Kissimmee, FL

[24] Web Ontology Language: OWL. http://www.w3.org/2004/OWL/

[25] ArcSight, "ArcSightESM," http://www.arcsight.com (accessed March 2008).

[26] E. Little, G. Rogova, and A. Bourry-Brisset, "Theoretical foundations of threat ontology for data fusion applications," DRDC-Valcartier, Tech. Rep. TR-2005 -269, November 2005.

[27] G. Shafer, A Mathematical Theory of Evidence. Princeton University Press, 1976.

[28] R. Haenni, Shedding new light on zadeh's criticism of dempster's rule of combination," in Proceedings of 8th International Conference on Information Fusion, July 2005.

[29] Franz Inc. (2008). jLinker - a dynamic link between Lisp and Java. http://www.franz.com/support/documentation/8.1/doc/jlinker.htm.

[30] Kandefer, M., Shapiro, S., Stotz, A., and Sudit, M. (2007). Symbolic reasoning in the cyber security domain. In *Proceedings of MSS 2007 National Symposium on Sensor and Data Fusion.*

[31] Kandefer, M. and Shapiro, S. C. (2008). Comparing SNePS with Topbraid/Pellet. SNeRG Technical Note 42, Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY.

[32] Kumar, D. and Shapiro, S. C. (1994). Acting in service of inference (and *vice versa*). In Dankel II, D. D., editor, *Proceedings of The Seventh Florida AI Research Symposium (FLAIRS 94)*, pages 207.211. The Florida AI Research Society.

[33] Seyed, A. P., Kandefer, M., and Shapiro, S. C. (2008). SNePS ef_ciency report. SNeRG Technical Note 43, Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY.

[34] Shapiro, S. C. (1991). Cables, paths and .subconscious. reasoning in propositional semantic networks. In Sowa, J., editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 137.156. Morgan Kaufmann, Los Altos, CA.

[35] Shapiro, S. C. (2000). SNePS: A logic for natural language understanding and commonsense reasoning. In Iwa´nska. M. and Shapiro, S. C., editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*, pages 175.195. AAAI Press/The MIT Press, Menlo Park, CA.

[36] Shapiro, S. C. and The SNePS Implementation Group (2008). *SNePS 2.7 User's Manual*. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY. Available at http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf.

[37] Top Quadrant Inc. 2007. Topbraid Composer. http://www.topbraidcomposer.com/.

[38] Clark & Parsia, LLC. 2007. Pellet: The Open Source OWL DL Reasoner. http://pellet.owldl.com/.

[39] Wos, L.; Overbeek, R.; Lusk, E.; and Boyle, J. 1984. *Automated Reasoning: Introduction and Applications*. Englewood Cliffs, NJ: Prentice-Hall.

[40] Nardi, D., and Brachman, R. J. An introduction to description logics. 1.43.

[41] Forbus, K. D., and de Kleer, J. 1993. *Building Problem Solvers*. Cambridge, MA: MIT Press.

[42] XML Tutorial, http://www.w3schools.com/xml/default.asp.

7.2 Nationbuilding

[43] The Ptolemy Project. http://ptolemy.eecs.berkeley.edu/

[44] Goldberg, S. (1958). *Introduction to Difference Equations.* John Wiley & Sons, Inc.

[45] Elaydi, S. (1996). *An Introduction to Difference Equations.* New York: Springer-Verlag.

[46] Lee, E., & Messerschmitt, D. G. (1987). Synchronous Data Flow. *Proceedings of the IEEE* , 1235 - 1245.

[47] Zhou, Y., & Edward, L. A. (2008). Causality Interfaces for Actor Networks. *ACM Transactions on Embedded Computing Systems , 7* (3), 2-35.

[48] L.K. Nozick, M.A. Turnquist, D.A. Jones, C.R. Davis, and C.R. Lawton. Assessing the performance of interdependent infrastructures and optimizing investments. In *Proceedings of the 37th Hawaii International Conference on System Sciences - 2004*, pages 1-7. IEEE, 2004.

[49] S. M. Ross. *Introduction to Probability Models*. Academic Press, San Diego, CA, USA,1997.

[50] Apache XMLBeans. http://xmlbeans.apache.org/.

8 List of Acronyms

AFRL – Air Force Research Lab
AFRL/IF – Air Force Research Lab Information Fusion Division
BID – Bugtraq Identifier
UB – State University of New York at Buffalo
CPN – Colored Petri Net
CVE – Common Vulnerabilities and Exposures
CUBRC – CUBRC, Inc.
DST – Dempster-Shafer Theory
ECCARS – Event Correlation for Cyber Attack Recognition System
EOI(s) – Event(s) of Interest
FuSIA – Future Situation and Impact Awareness
GNOME – Graph and Network Objects for Model Elicitation
HIDS – Host Intrusion Detection Sensor
HIPS – Host Intrusion Prevention Sensor
IDS – Intrusion Detection Sensor
JDL – Joint Director's Laboratory
INFERD – Information Fusion Engine for Real-Time Decision-Making
N-CMIF – National Center for Multi-source Information Fusion
NOEM - National Operational Environment Model
NIDS – Network Intrusion Detection Sensor
NIPS – Network Intrusion Prevention Sensor
OWL – Web Ontology Language
PN – Petri Net
PSU – Penn State University
RIT – Rochester Institute of Technology
SID – Signature Identifier
SNePS – Semantic Network Processing System
TANDI – Threat Assessment of Network Data and Information
VLMM – Variable Length Markov Models
VT – Virtual Terrain
VTAC – Virtual Terrain Assisted Cyber Attack Impact Assessment
XML – Extensible Markup Language