Course of Action Generation for Cyber Security Using Classical Planning

Mark Boddy, Johnathan Gohde, Thomas Haigh, Steven Harp

Adventium Labs

The Problem

Finding and closing (or monitoring) attack vulnerabilities

For example:

1. Attacker sends an email message, spoofed to be from a colleague, with a new screensaver as an attachment.
2. Attachment is an executable that enables remote login, and captures and relays the users password.
3. Attacker logs into the machine and executes a buffer overflow attack, gaining root (admin) privileges.
4. …

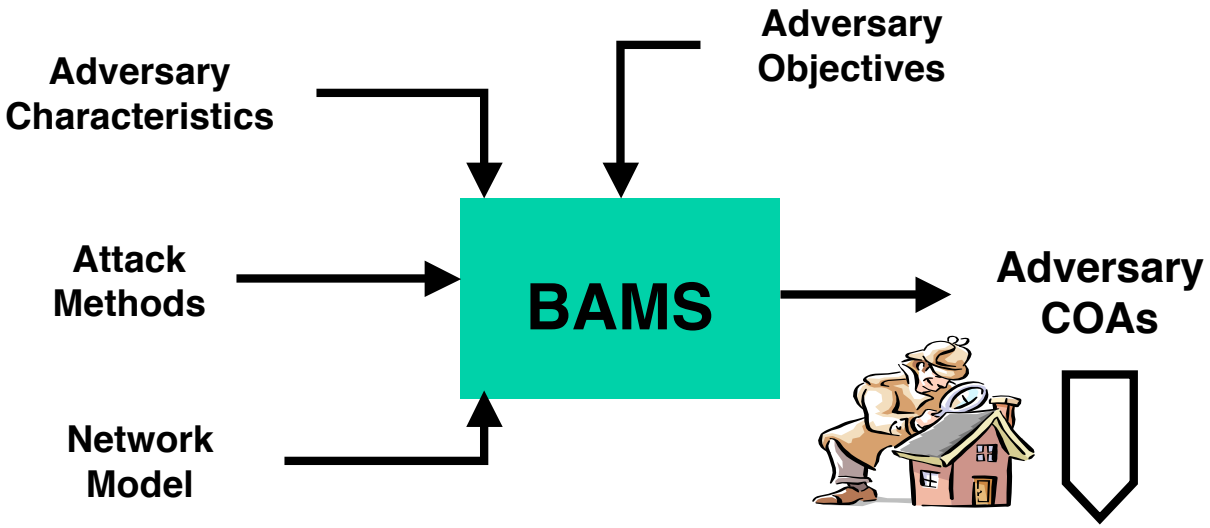
Current approaches

- User Modeling
  - Psychological models
  - Example attacks
- Exhaustive analysis of “attack graphs”, e.g., [Sheyner, et al., 2002]
- “Network Hardening” [Noel et al., 2003]

Why is this hard?

- Network and system scale, complexity, and dynamism
- Attackers are stealthy
- Many steps in any given attack may be legitimate.
- Some exploits involve actions taken outside the network.
- Some exploits are impossible or expensive to detect.
- Limited supply of experts

Behavioral Adversary Modeling System

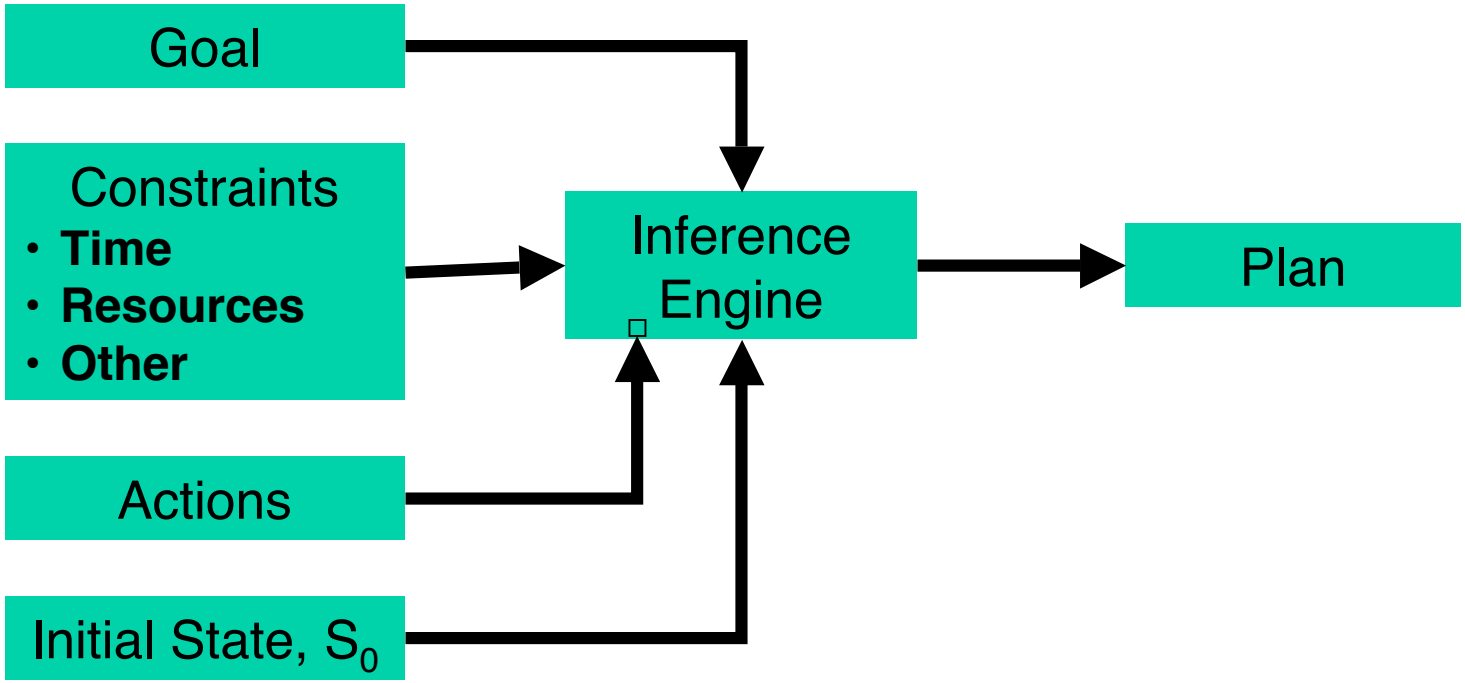


- Probability of success
- Probability of detection
- Likely side-effects
- Choke-points for prevention and detection
- Possible coalitions

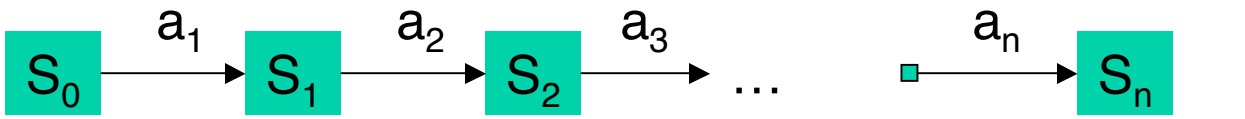
Proposed solution: use classical planning

Classical Planning

- Origins in theorem-proving and dynamic logic
- Time is treated as a succession of *states*
- States have associated facts, or *fluents.*
- *Operators* map from state to state:
  - Action: (move a b)
  - Preconditions: (clear b), (clear a)
  - Effects: (on a b)
- Complications: frame and ramification problems
- Extensions: durative actions, context-dependent effects, resource usage, uncertain effects, knowledge-gathering actions, …



Plan:



Goal (S_n) = True

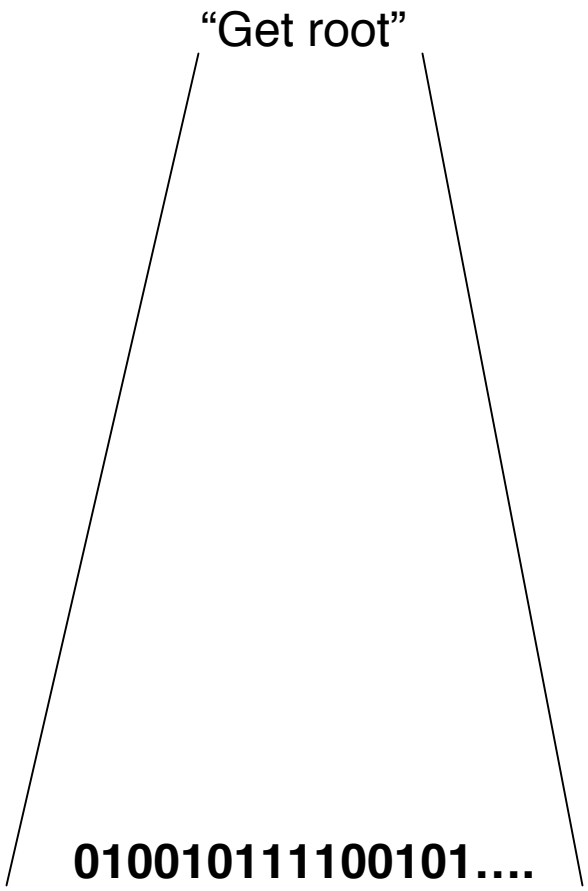
Why Classical Planning?

- Domain characteristics
  - Propositional representation fits well
  - Time is not very important
  - Nor are resources
- State of the art
  - Recent (IPC-02, IPC-04) results demonstrating that current planning systems will scale
  - Expressive extensions: STRIPS -> ADL -> PDDL
- Style of inference
  - No *a priori* assumptions about operator sequences
  - Richer state representation
  - Rigorous determination of infeasibility

Mapping COA Generation to a Planning Model

- Adversary characteristics
  - Risk tolerance
  - Available resources
- Attack methods
  - Operators
- Network (domain) model
- Adversary objectives
  - Goals

Modeling the Domain



Finding a middle ground:

- Preconditions, actions, effects relevant to user
  - Sending email
  - Logging in
  - Creating/modifying files
- Useful representation of preconditions and effects

Examples: Facts

- (insider bob)
- (in_room bob bobs_office)
- (can_unlock key1 lock1)
- (knows bob root_password)
- (accessible s_iexplore sherpa)
- (can_read_email ms_outlook)
- (trusts_instructions greg adam)

Examples: Goals

(:goal (knows bob secret_info))
(:metric minimize (detection_risk))

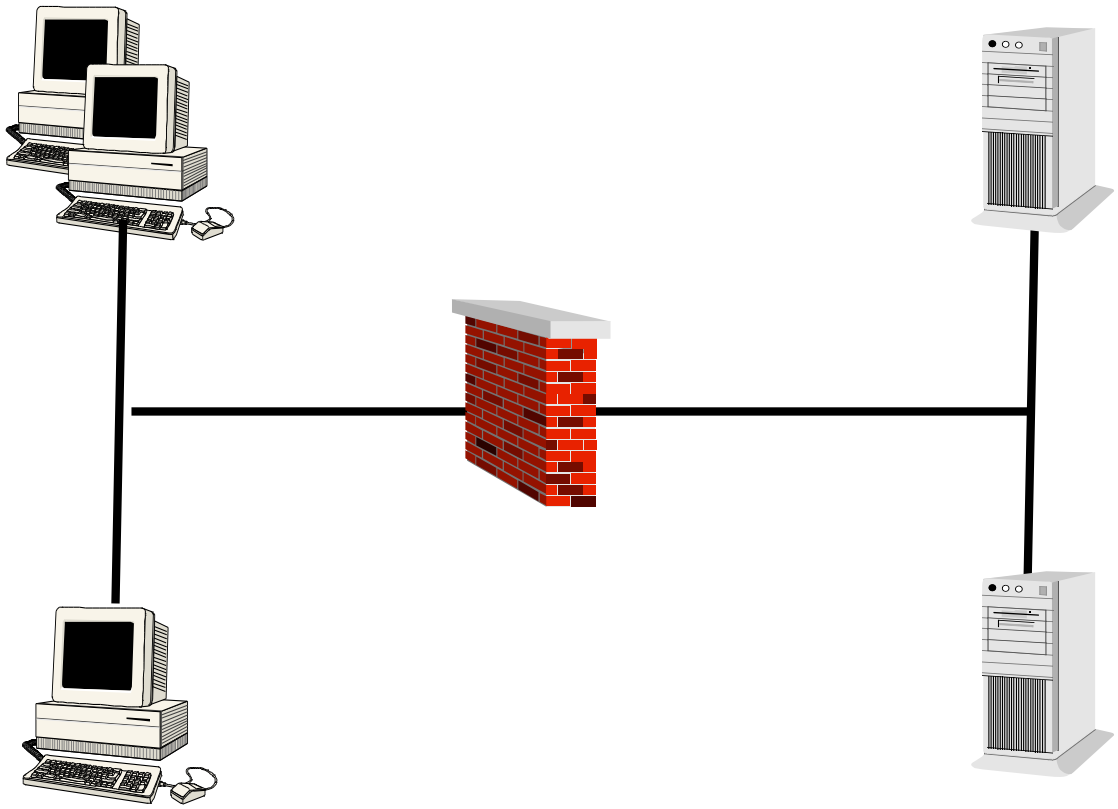
(and (knows bob secret_info)
(<= (detection_risk) 5))

Examples: Actions

```
(action DMS_ADD_GROUP_ALLOW
    :parameters (?admin - c_human
                 ?chost - c_host
                 ?shost - c_host
                 ?doc - c_file
                 ?gid - c_gid)
    :precondition
        (and (nes_admin_connected ?chost ?shost)
             (at_host ?admin ?chost)
             (insider ?admin)
     :effect (and (dmsacl_read ?doc ?gid)))
```

End-users

Mail Server



Sys Admin
- Password protected account
- Manages user accesses

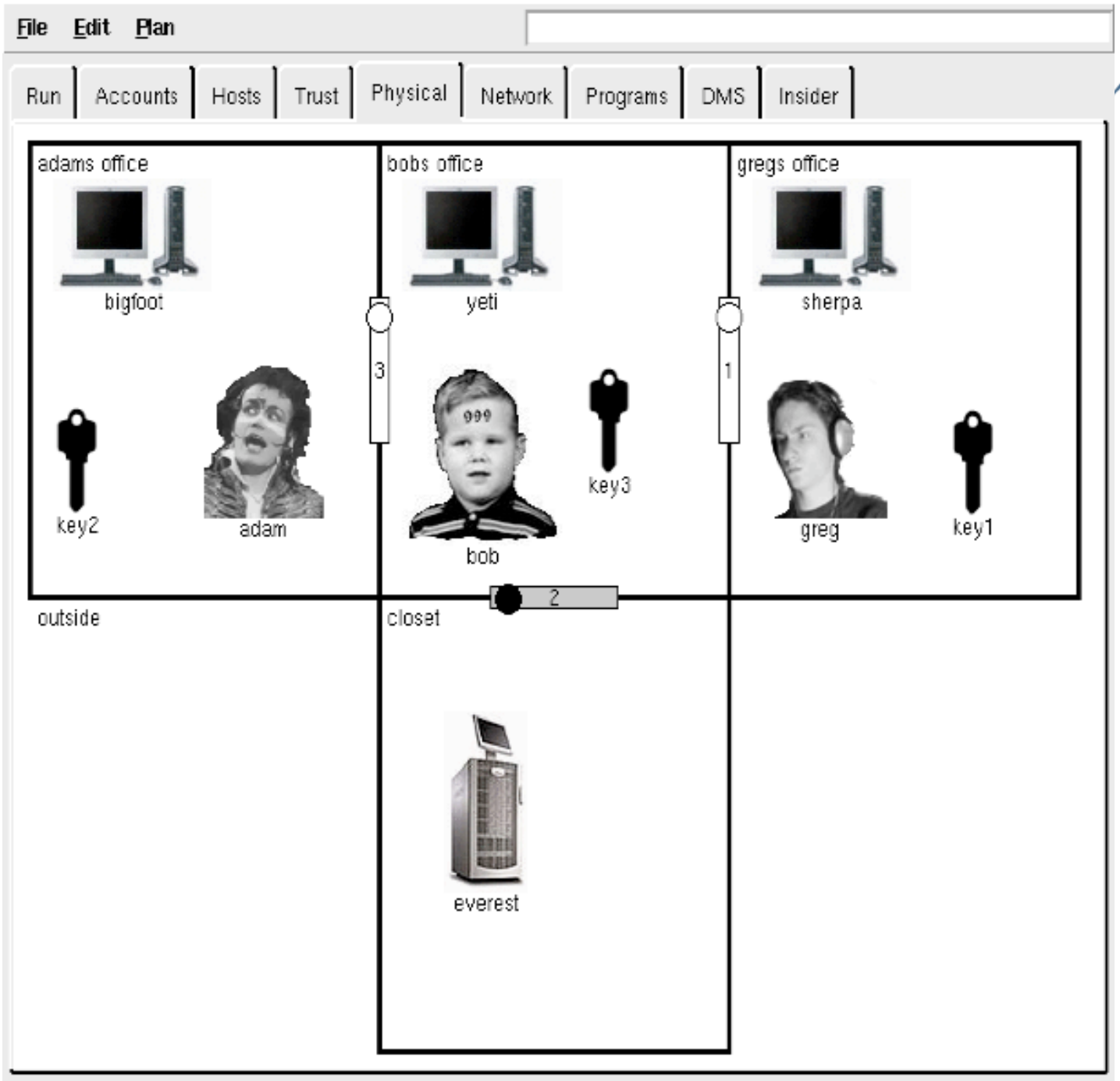
COI Web Server
- SSL with fixed passwords
- ACLs

Domain Features

- Document Management System based on the”Community of Interest” model
- Three adversaries
  - Moderately sophisticated loner
  - Sophisticate working with foreign intelligence organization
  - System administrator
- Cyber, physical, and social exploits

Domain Features

- Cyber defenses:
  authentication (2 forms), access permissions, controlled change of access permissions, firewalls, detectability, hubs and switches
- Cyber exploits:
  manipulation of access permissions, direct attacks against a workstation, password hacks, mis-directed trust (multiple aspects), host and network sniffing, spoofing, e-mail viruses, misdirected information,
- Physical system and exploits:
  location, shoulder surfing, hardware keystroke logger
- Social behavior:
  various forms of trust, social engineering, tolerance for risk, coalitions of attackers



Adventium LABS

mysterioso

What skills and tools does this malicious insider possess?

Hacking Skills

- Low
- Medium
- High

Social Engineering Skills

- Low
- Medium
- High

- [x] Has a network packet sniffer
- [ ] Has a hardware keystroke logger
- [x] Has a browser-infecting custom virus
- [ ] Has a windows-trojaning buffer overflow exploit

< Prev | Cancel | Next >

Scale of a Typical BAMS Problem

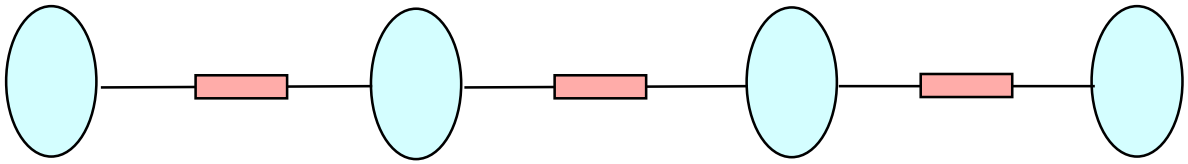
PROBLEM: `NESACL'
Defined classes: 28
Defined predicates: 123
Number of objects: 100
Number of facts: 189
Number of goals: 1
Number of actions: 56

Inference

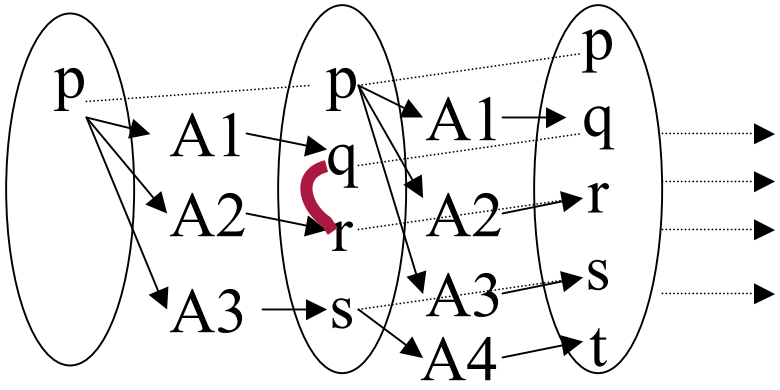
We use Hoffmann's Metric-FF:

- Forward heuristic planner, using a relaxed plan graph to compute a distance heuristic
- On failure reverts to A* search, using the same heuristic
- Fairly complete PDDL parser
  - Quantification, conjunction, disjunction in preconditions
  - Context-dependent effects
  - Metric values associated with actions

Planning Graphs



Traditional state-action planning



Planning Graph

Forward Heuristic Search

- Add to the end of a partial plan.
- Possible additions are the applicable operators
- Distance heuristic from a *relaxed* planning graph.
  - Ignores mutexes
  - Very effective for many domains
  - Not effective for:
    - functional relationships (e.g., logins)
    - required sequences of true, false (e.g., going through a door and closing it behind you).

A Plan

0 : ADAM sits down at BIGFOOT
1 : ADAM enters ADAM_UID as user name for login on host BIGFOOT
2 : ADAM enters password ADAM_PWD for login at host BIGFOOT
3 : Shell B_WEXPLORE is launched on host BIGFOOT for user ADAM_UID
4 : Program WEXPLORER on host BIGFOOT forks a child process
5 : Contents of file B_IEXPLORE begin executing as uid ADAM_UID on host BIGFOOT
6 : BOB sits down at YETI
7 : BOB enters BOB_UID as user name for login on host YETI
8 : BOB enters password BOB_PWD for login at host YETI
9 : Shell Y_WEXPLORE is launched on host YETI for user BOB_UID
10 : Program WEXPLORER on host YETI forks a child process
11 : Contents of file Y_ETHEREAL begin executing as uid BOB_UID on host YETI
12 : ETHEREAL starts sniffing the networks on YETI
13 : ADAM logs onto dms admin server EVEREST from BIGFOOT
14 : BOB reads the sniffer thus learning NES_ADMIN_PASS

Plan, Continued

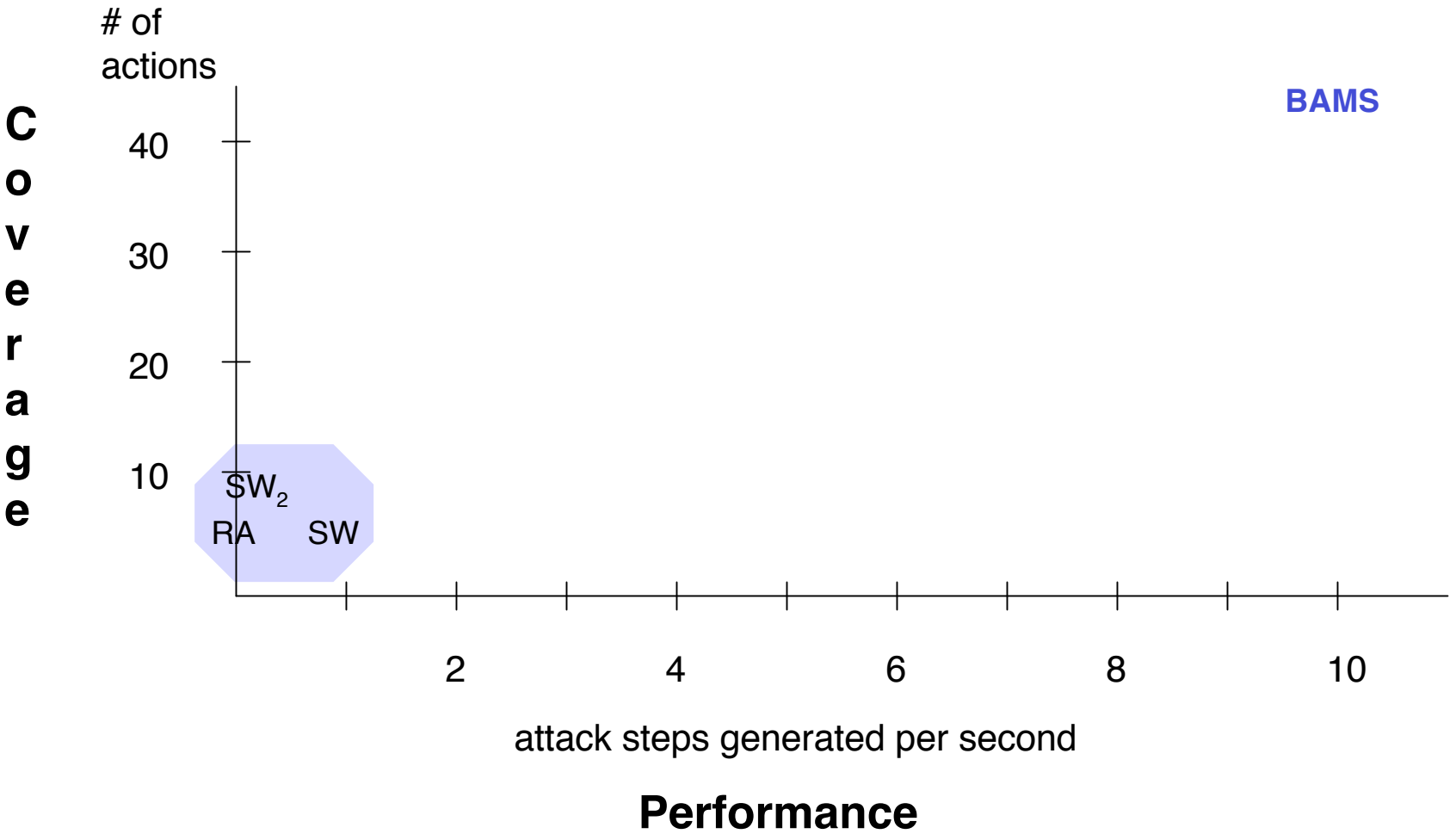
15 : Program WEXPLORER on host YETI forks a child process
16 : Contents of file Y_IEXPLORE begin executing as uid BOB_UID on host YETI
17 : BOB logs onto dms admin server EVEREST from YETI
18 : DMS session DMSS1 has begun
19 : BOB begins a DMS session on YETI
20 : Connect DMS session DMSS1 to server NES on EVEREST
21 : A route from YETI to DMS server EVEREST exists
22 : BOB enters password BOB_DMS_PWD for the DMS session.
23 : Authenticate BOB_UID in dms session DMSS1 with EVEREST using BOB_DMS_PWD
24 : BOB adds an acl to allow read access of E_SECRET_DOC to the EAST_GID group
25 : BOB begins a DMS request at YETI in session DMSS1
26 : Document E_SECRET_DOC is requested in session DMSS1
27 : Document E_SECRET_DOC is sent and displayed on YETI in session DMSS1
28 : BOB reads E_SECRET_DOC and learns SECRET_INFO

Generating Plans

	Steps	Time
Direct Client Hack	25	0.67
Misdirected Email	32	0.67
Shoulder Surfing	18	0.69
Email Trojan	37	0.71
Spoofed Email Trojan	37	0.73
Spoofed Instructions	36	0.79
Administrator ACL Change	23	1.20
Sniff Administrator Password	28	1.62
Sniff Password from Email	44	4.77

BAMS vs. Other Approaches

(based on **very** limited published data)



Other BAMS Advantages

1. COAs generated from system model and adversary profile and objective
2. Does COA generation well
   a. Richness (and scale) of systems and adversaries modeled
   b. COAs are at useful level of detail
   c. Easy to change inputs to study impact of counter measures
3. Supports a rich adversary model
   - Covers a broad range of adversary traits and exploits
4. Supports rapid exploration of alternative adversary and network models

Pragmatic Issues

- Performance (esp. memory consumption)
  - Optimizing grad-ware
  - Rewriting the model to avoid “hard actions”
  - Rewriting to minimize the size of the propositional expansion
- Representing processes (e.g., composing and sending email).
- Entities that are created or destroyed
- Derived predicates
- Maintaining large domain models

Rewriting the model to avoid "hard actions"

Metric-FF compiles away much of PDDL's expressive power:

- Quantification is expanded on the domain.
- Conjunction and disjunction are rewritten.
- Context-dependent effects are not removed.

- "Hard actions" appear to be those whose preconditions are not in DNF. So, we can rewrite

  (and foo (or bar baz)

  to be

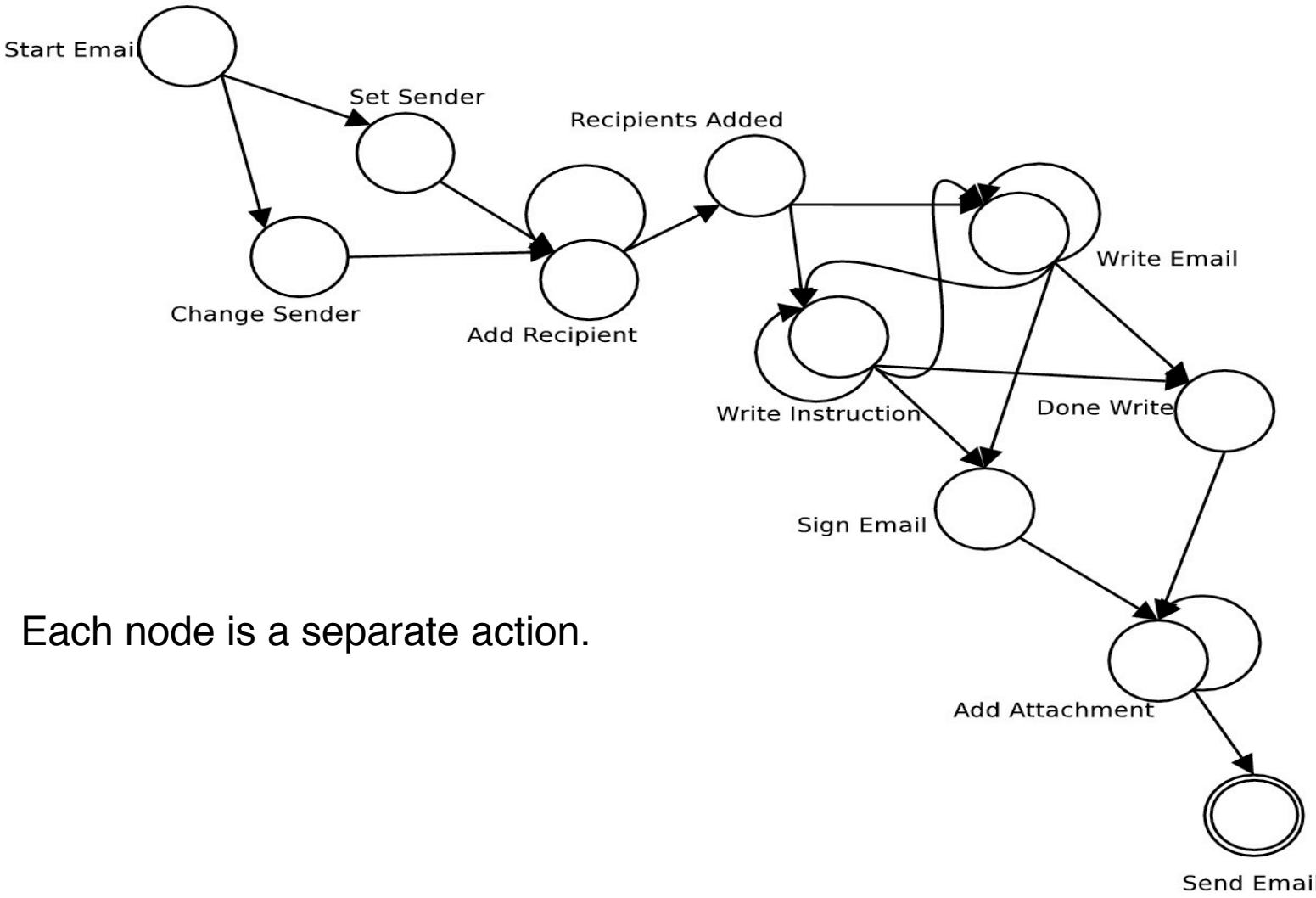
  (or (and foo bar) (and foo baz))

Rewriting to minimize the size of the propositional expansion

- Each action is used to generate a set of propositional operators, by instantiating all possible values for each parameter.
- The number of resulting operators is exponential in the number of parameters.
- Some actions had nine parameters.

Solution: factor the action into smaller actions, which must then occur in an uninterrupted sequence.

Representing processes (e.g., composing and sending email).



Each node is a separate action.

Created Entities

In a cyber domain, there are numerous “handles” whose specific value is unimportant, many of which are created on the fly.

- Process IDs
- File IDs
- Sockets, sessions, etc…

A propositional planner is not smart enough to know when trying a different ID might help, and when it won’t.

Derived predicates

- Some action preconditions are usefully defined in terms of other domain propositions.

- For example, a file is “readable” by a user with a given UID, just in case:
  - UID has access to the directory and read permission on the file, or
  - is in a group GID with those permissions, and
  - the user is logged in on the appropriate host, or
  - on a host that has the appropriate volume mounted.

Modular Domains in PDDL

PDDL input consists of:

- A *domain model,* specifying object types, predicates, and actions
- A *problem statement,* specifying all objects, the initial state and a goal.

A more natural way to specify a complex domain is in separate modules, but this aggregation is inconsistent with the PDDL spec.

m4

Powerful macro language, used in configuration management.

1. DEFMODAL: augment operators with special pre and post-conditions to enforce sequence.
2. “Gensym” -- force ID creation to draw from an ordered pool, in such a way that it will only succeed once.
3. Define derived predicates as macro substitutions into action preconditions.
4. Compile multiple domain modules into global domain model and problem statement.

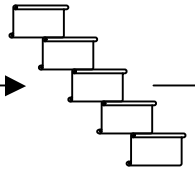
Process

Adversaries
- Skills
- System access
- Risk tolerance

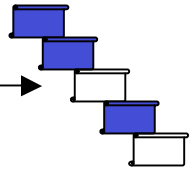
Model Files
- Independent
- Composable

Problem

M4 Macro Language

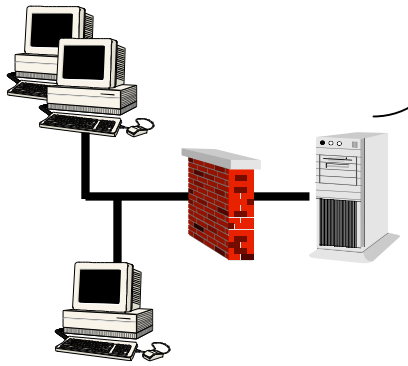


GUI



Single LAN DMS

End-users

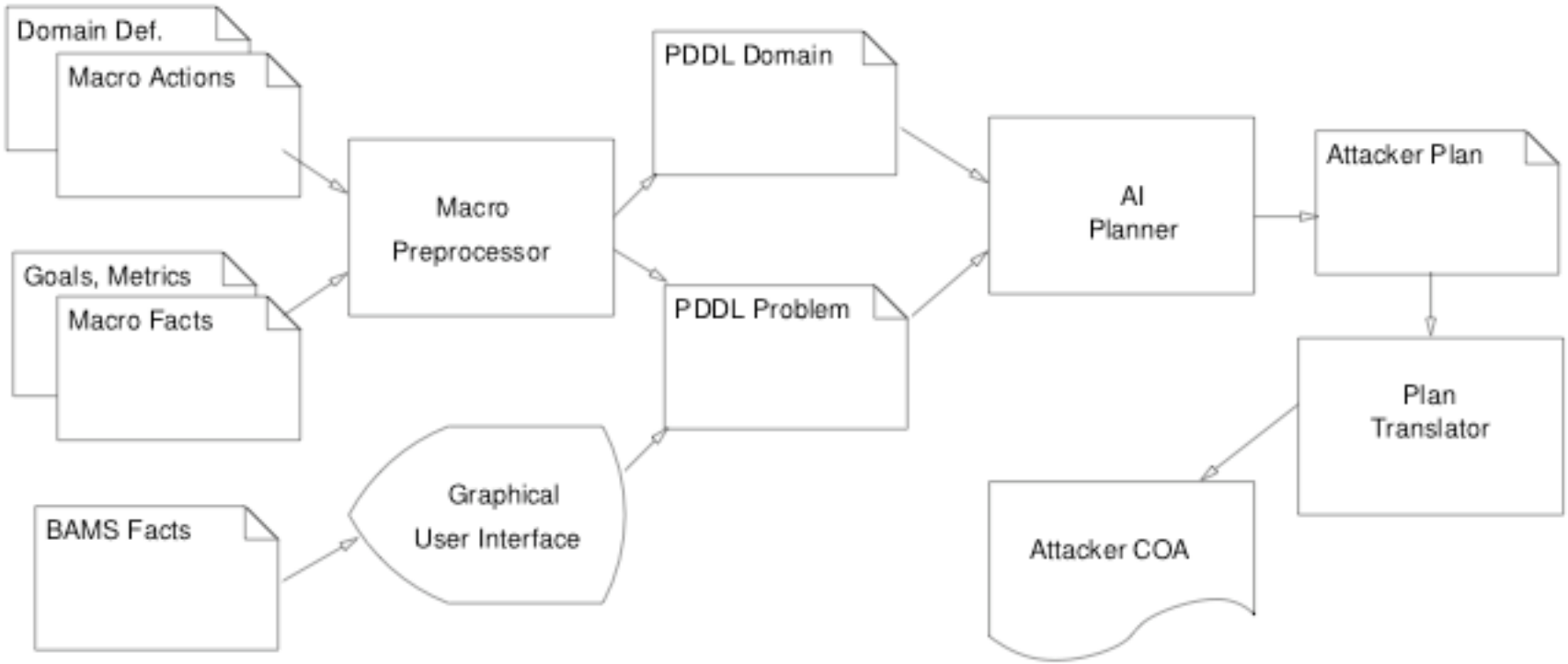


COI Web Server and Mail
- SSL with fixed passwords or digital certificates
- ACLs

Adversary Course of Action

Sys Admin
- Password protected account
- Manages user accounts

Information Flows



Future Work

- Planner Technology
  - Efficient generation of multiple plans
  - Improvements in performance and scalability, including more extensive use of metrics
- Modeling Tools and Techniques
  - Make it easier for domain experts to extend and maintain the model
  - Compile user model into performance-tuned PDDL
- Analytic Capabilities
  - Bottleneck analysis
  - Probabilistic or uncertain reasoning
- IC Specific Models
  - Drives the work in the first three areas
- Comparative analysis
  - Head-to-head
  - Planning Competition

Generation of Multiple Plans

- Current planner (FF) does not generate multiple *interesting* plans efficiently

- Backward planners (PEGG) can do this
  - much slower
  - do not parse as much of PDDL.

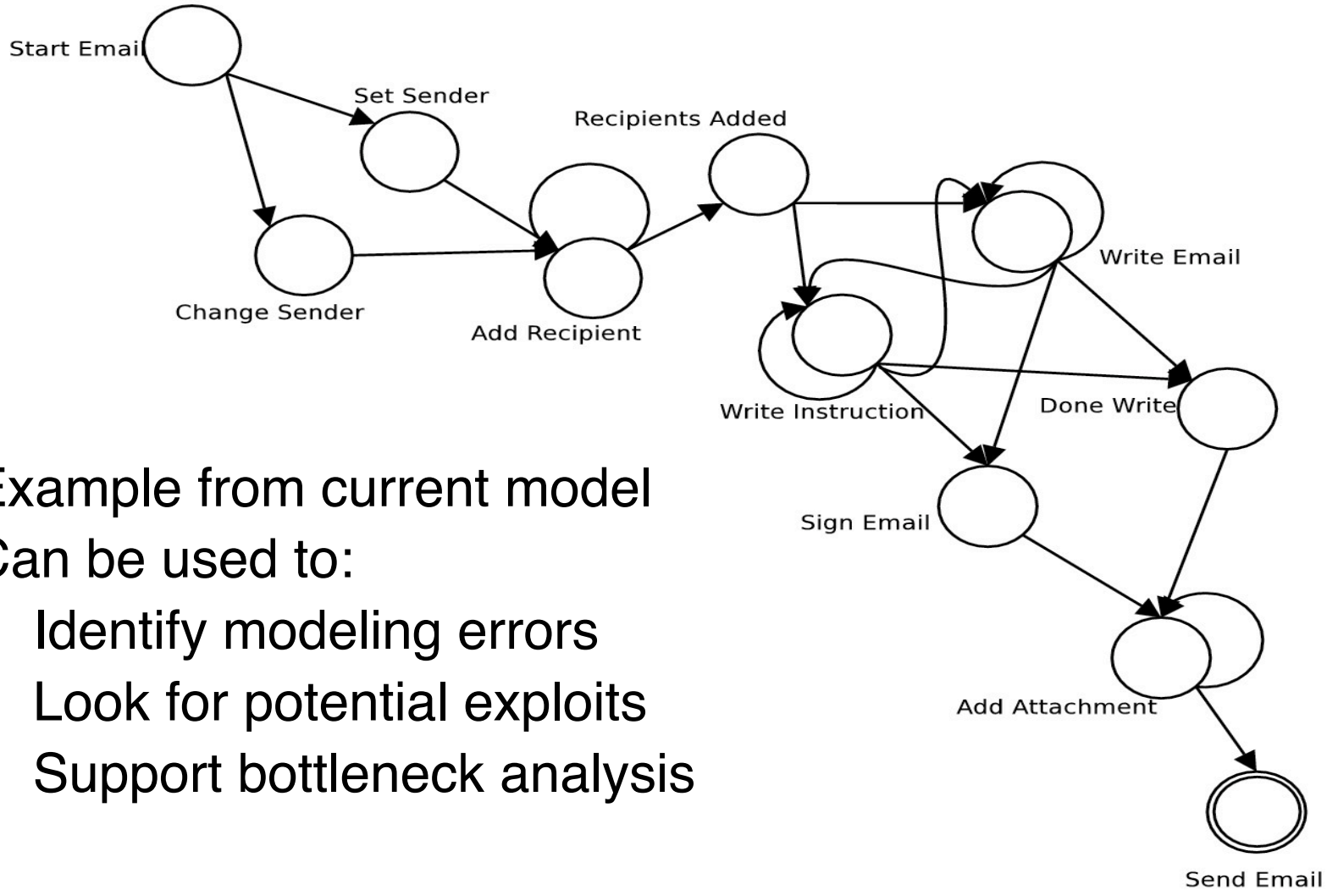
Performance and Scalability

- Motivation
  - Expect an order of magnitude increase in size of domain model
  - Lesser increase in length of plans
  - Need better performance for metric information
- Avenues of attack:
  - Improving heuristics and tuning for this domain
    - Object creation and deletion
    - Context dependent effects
  - Compilation to a well-behaved subset of PDDL
  - Wait for the next planning competition…

Modeling Tools and Techniques

- Maintaining an existing domain model (adding new objects, modifying features of existing ones)
  - Existing GUI, with some maturation
  - Opportunities for integration with 3rd-party tools
- Modifying the domain model (new *types* of objects)
  - Class editor, with automatic generation of editors for the new object classes. (e.g. DOME)
  - Input parsers and checkers
  - Abstract ontology that facilitates generating new classes through specialization. (e.g. COIL)

Example Checker



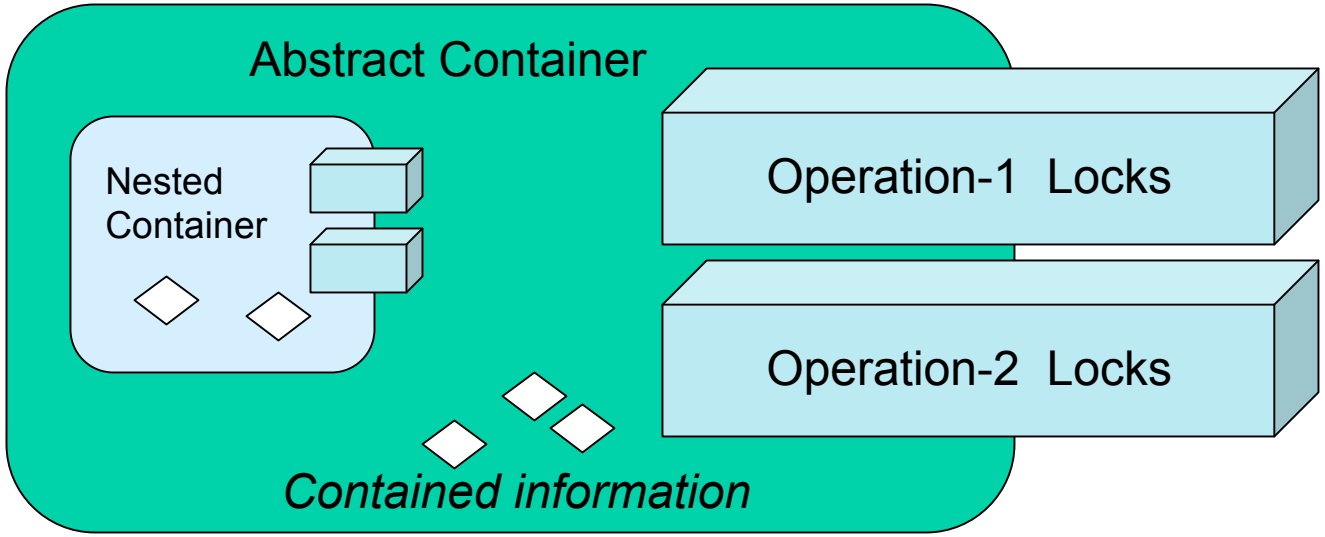
Example from current model

Can be used to:

- Identify modeling errors
- Look for potential exploits
- Support bottleneck analysis

Containers, Objects, Information, and Locks (COIL)

- Container = directory, file, DB record, attacker's head, paper, …
- Lock = acl, lock on door, buffer overflow, …
- Ontology of Generic Operations that open locks and add/remove data in containers
- Templates to instantiate different classes of containers, locks, and actions such as e-mail, file system, web, IM
- Automated compilation into PDDL



Analytic Tools

- Bottleneck Analysis
  - Post plan generation tools
  - Identify common attack steps in a set of plans
  - Determine best points for adding counter measures
- Reasoning with uncertainty
  - Use metric capability of planner to compute “probabilities” of success or detection for a plan
  - Move to a full probabilistic planner
  - Estimate probabilities over multiple plans
  - Manipulate planner to reason about hypothetical actions.

Conclusions

- Classical planning is an effective way to model cyber attacks
- The current state of the art in planning is (barely) sufficient to handle the required scale and complexity.
- The current state of the art in modeling tools is even more marginally adequate.

- We're pushing the state of the art in planning, in the process of providing new results in research on cyber security.

- Lots of work left to do.