Creating Objects in Flexible Authorization Framework

Nicola Zannone,[1 2] Sushil Jajodia,[2] Duminda Wijesekera[2]

[1] Dep. of Information and Communication Technology, University of Trento

[2] Center for Secure Information Systems, George Mason University

Talk Outline

- Creating objects in Access Control Framework
  - Conditions for creating objects
  - Authorizations on derived objects
- Flexible Authorization Framework
- Information Flow Control
- Conclusion

Access Control

- Essential for building secure information systems
- Protect the confidentiality of information
- An authorization is a triple of the form *(o, s,<sign>a)*
  - *(o, s, +a)*: subject s is allowed to execute action a on object o
  - *(o, s, –a)*: subject s is denied to execute action a on object o
- Authorization frameworks manage access to data
  - Is a subject entitled to execute an action on an object?
- But...

Creating Objects

- Information systems support data processing for manipulating information
- The output of data processing can be seen as a new object
- Represent data processing as function
  - e. g., $f(s, o_1, \ldots, o_m) = o$
    - s is the subject who wants to perform the data processing
    - $o_1, \ldots, o_m$ are the objects required by the data processing
    - o is the output of the data processing
- In order to protect data, we should answer
  - Is the subject s entitled to create the derived object o?
  - Who is authorized to access the derived object o?

Conditions for Creating Objects

- Subjects need to access objects required by data processing
- Specific domain conditions for determining who is entitled to execute data processing
  - e.g. users that play a certain role or belong to a certain group
- Make explicit the conditions under which a subject can perform data processing
  - $$f(s, o_1, \ldots, o_m) = \begin{cases} o & \text{if } C \text{ is true} \\ \perp & \text{otherwise} \end{cases}$$
    - C represents the condition that must be satisfied
    - $\perp$ means that object o cannot be created

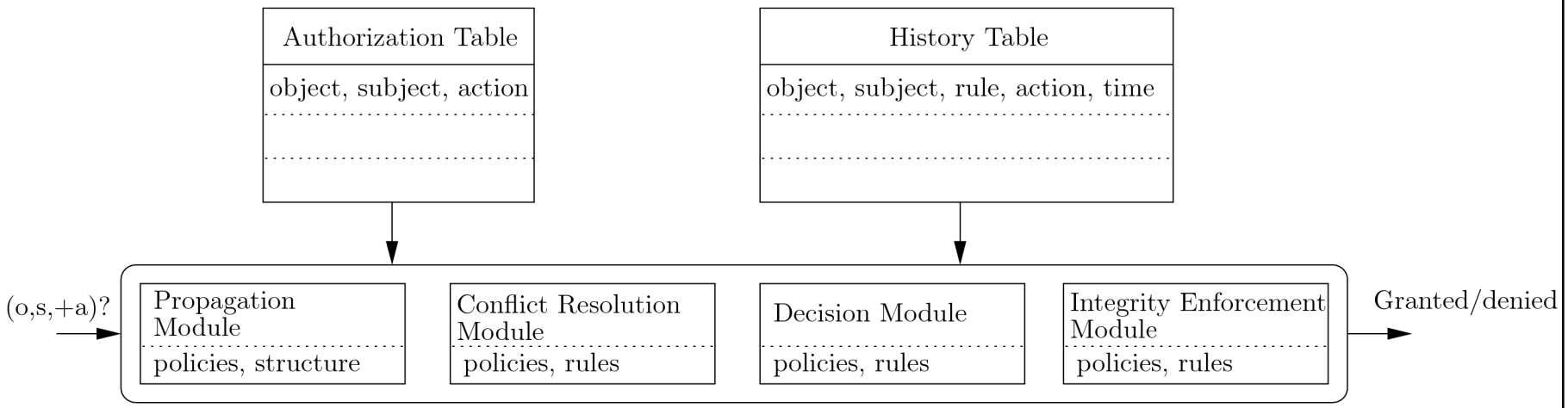
Authorizations on Derived Objects

- Once an object has been created, an access control policy should be associated with the object
- The derived object is not independent from the objects used to derived it
- The access control policy is defined on the basis of the authorizations associated with the objects used to derive it

Flexible Authorization Framework (FAF)

- Proposed by Jajodia et al. [TODS 2001]
- Logic-based framework developed to manage access to data
  - Determine if a user can execute an action on an object
  - For any access request, exactly one decision (allowed/denied) is provided
- Based on a language through which users can specify security policies to be enforced on specific accesses
  - allow specification of positive and negative authorizations
  - incorporate notions of authorization propagation, conflict resolution, and decision strategies

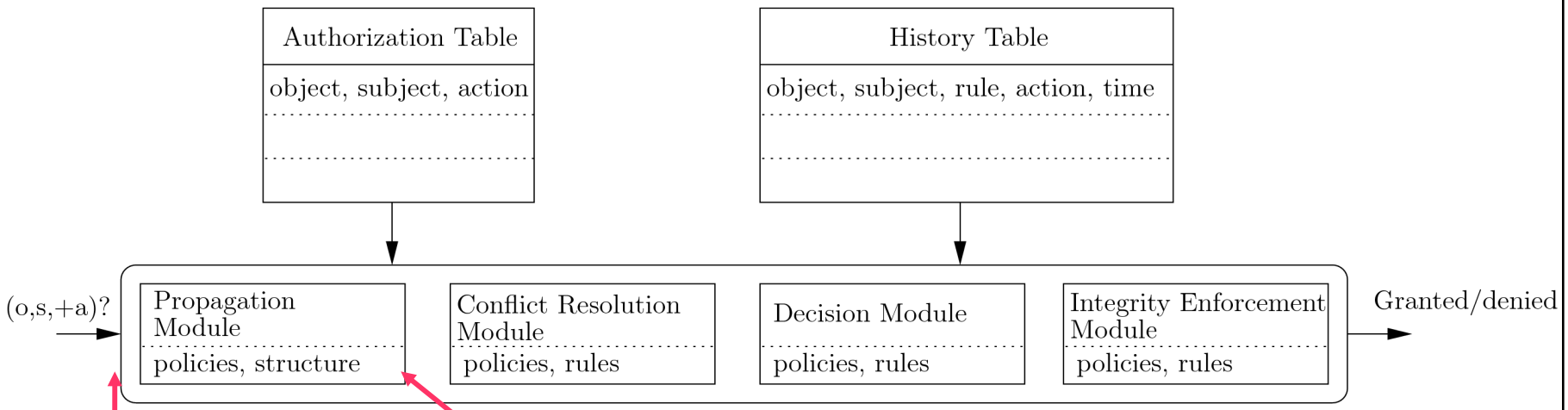
FAF Architecture



Semantics & Materialization

- A FAF specification forms a locally stratified logic program
  - It has a unique stable model
- Access requests should be authorized or denied very fast
  - Materialization algorithm reconstructs the unique stable model
- Computed in polynomial time on data complexity

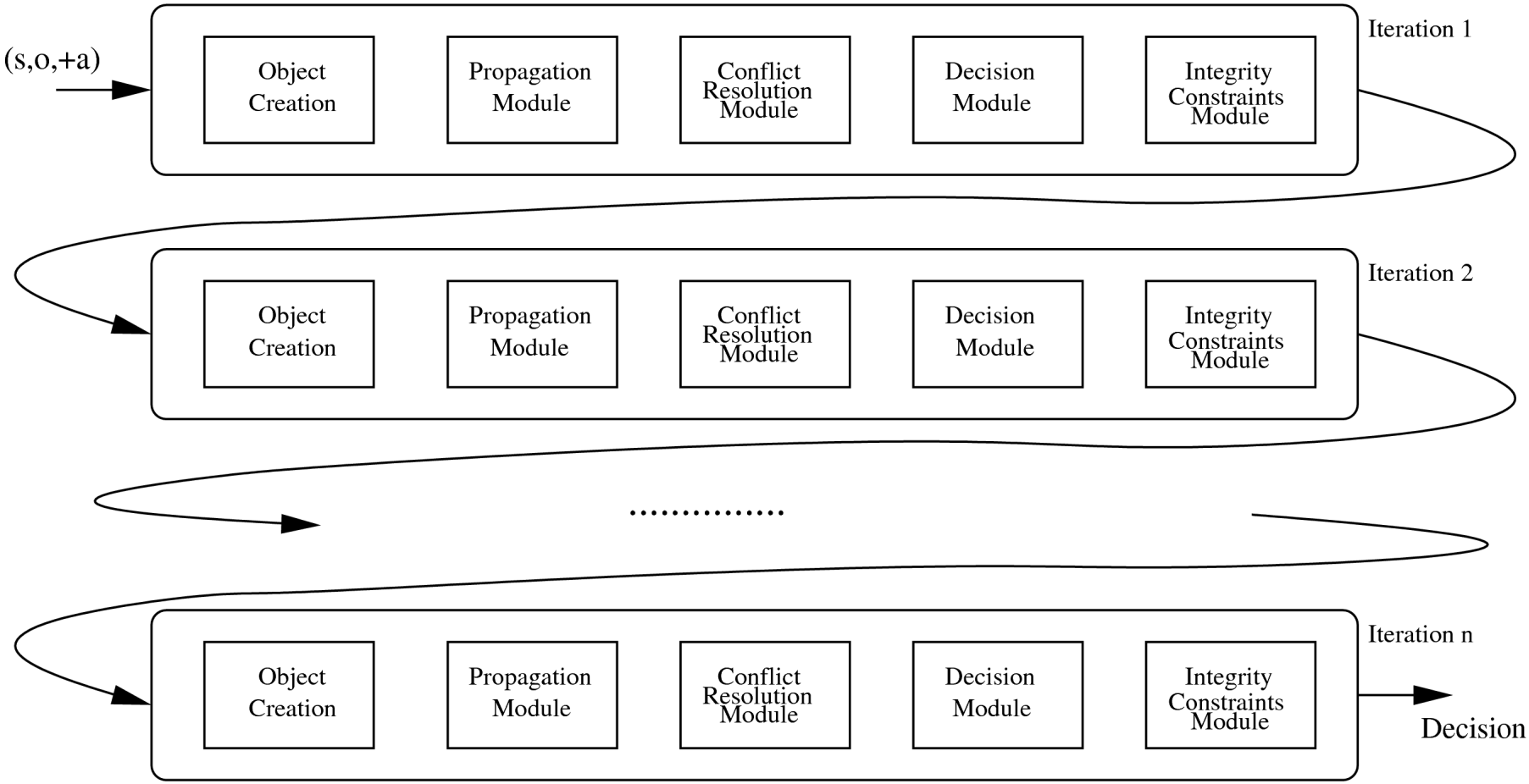
When should objects be created?



objects cannot be created since required authorizations might be not yet computed

authorizations on derived objects are not propagated

A New Architecture



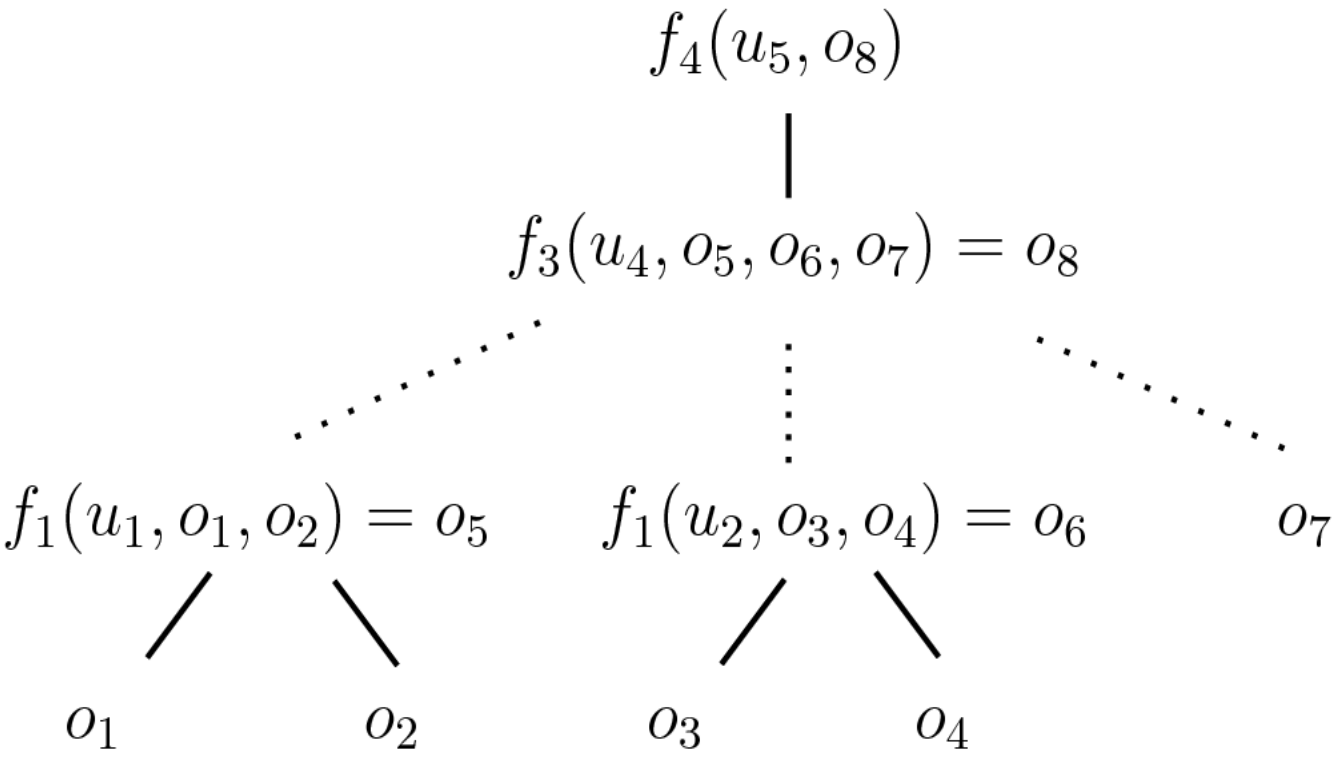
Computational Issues

- The new architecture does not preserve the locally stratified structure of the specification, but...
- Introducing syntactic constraints
  - Verify existence of objects
  - Distinguish the iteration in which objects are derived (i.e., materialization view of each iteration)
- Every authorization specification is a locally stratified program
  - $M(AS^{i-1}) \subseteq M(AS^{i})$
  - literals in $M(AS^{i}) / M(AS^{i-1})$ refer to objects created at the i-th iteration
- Preserve FAF computational results
  - Stable model can be computed in polynomial time

Derivation Tree

- Information systems manipulate information
  - The outcome of a data processing can be seen as a new object
- The outcome of a data processing may be used as input for other data processing
- The process to derive an object can be seen as a tree
  - Root is the “final” object
  - Leaves are primitive objects (i.e. objects not derived by using functions)
  - Edges keep trace of the process used to create objects

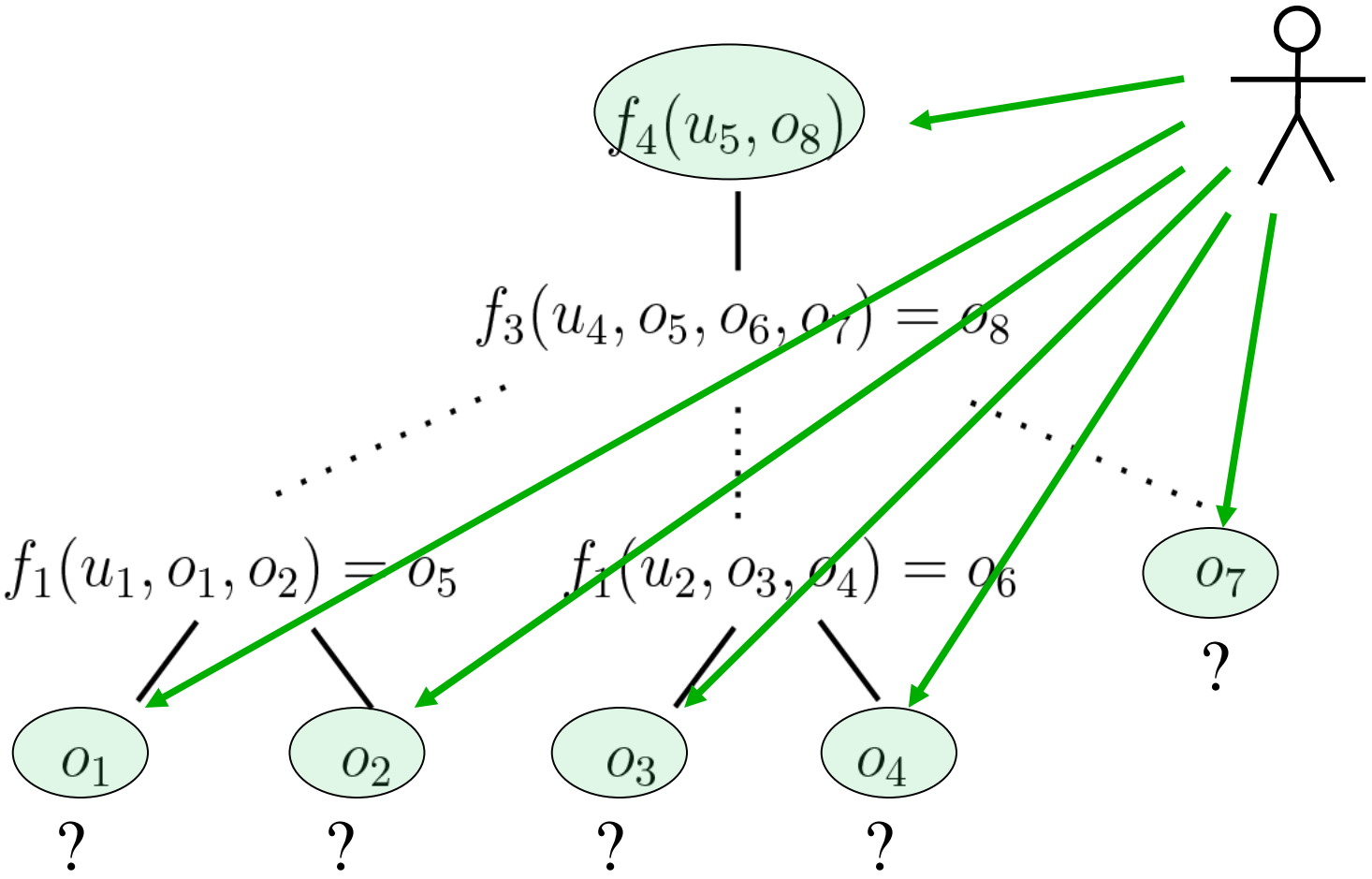
Derivation Tree



Information Flow Control

- Information systems may release information as part of their functionalities
  - Derived objects contain information belonging to the objects used to derive it
  - So they disclose information about the objects used to create it
- Need to introduce information flow control
  - Ensure that information are not disclosed to unauthorized entities
- The policy associated with the object is the intersection of the policies associated with the objects used to derive it
- Integrity constraint module
  - Define warning constraints to detect unauthorized information flow

Information Flow Control



Some Exceptions

- Some information must be disclosed for satisfying availability requirements
  - Privacy Act allows an agency to disclose data without the consent of the data owner to those officer and employees of the agency who need the data to perform their duties
- The system administrator may be perfectly happy with a system that does not satisfy all warning constraints
  - Notice that Warning ≠ Error

Verification Process

1. The authorization framework spots warnings when leaks are detected
2. The system administrator has to decide if
   - the leak complies with system requirements or
   - the leak corresponds to a system vulnerability
3. The system administrator will fix system vulnerabilities

Conclusion

- Support for object creation
  - Verify permissions for creating objects
  - Automatically derive access control policies for derived objects
- Formalization of the process for enforcing access control policies concerning derived objects
- Mechanisms for detecting information leakage