Access Control Policy Tool (ACPT), an assurance tool that combines symbolic model checking with combinatorial coverage

Access Control Policy

Presently access control policies are hand crafted by administrators, and difficult to check for correctness. We need a tool for:

- Composing policy by structure framework
- Detecting conflicts in policy rules
- Efficient testing of implementation
- Policy code generation

Outline

Access Control Policy Tool (ACPT) Overview

Approaches
- Model specification and composition
- Property verification
- Policy testing
- XACML generation

Related work

Future work

ACPT Overview - Functions

Composition

Allows specification of policy combinations, rules and properties through model and rule templates.

Verification

Allows testing and verification of policies against specified properties and reports problems that may lead to security holes.

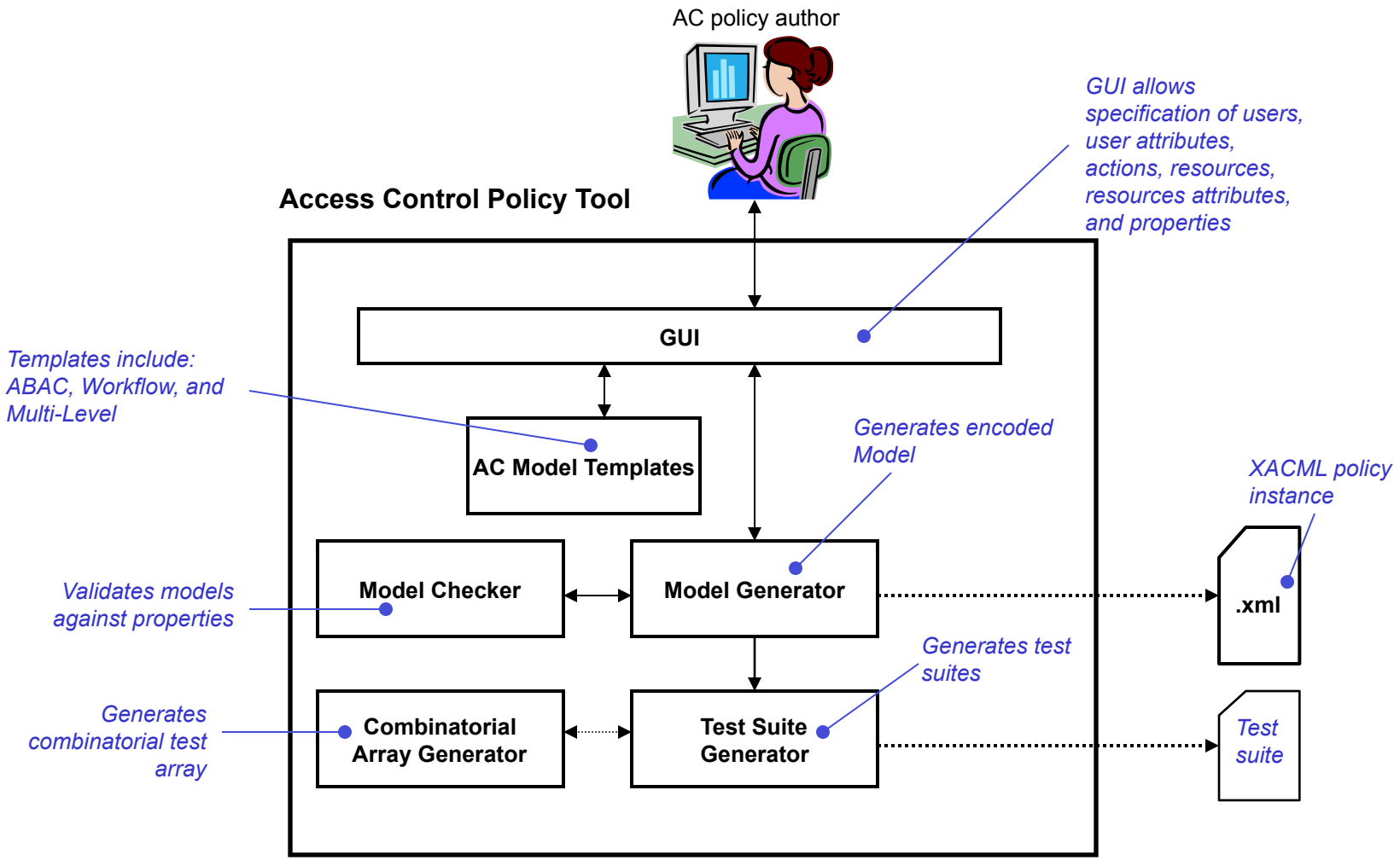
Testing

Generates efficient test suites (by applying NIST's combinatorial testing technology) for testing of access control implementation, test suites can be applied to any access control implementation.

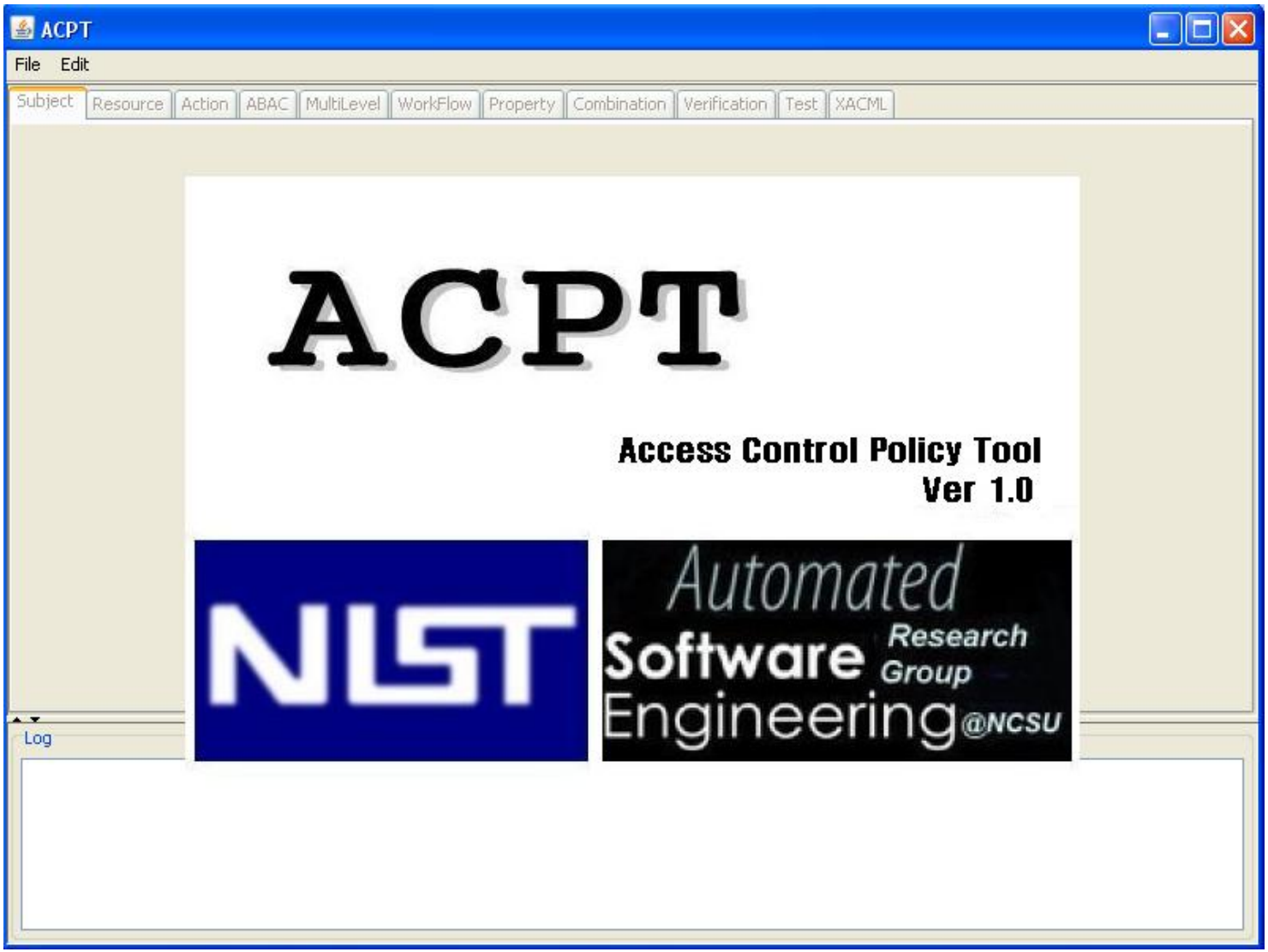
Policy

XACML policy generation.

ACPT Overview - Architecture



ACPT Overview

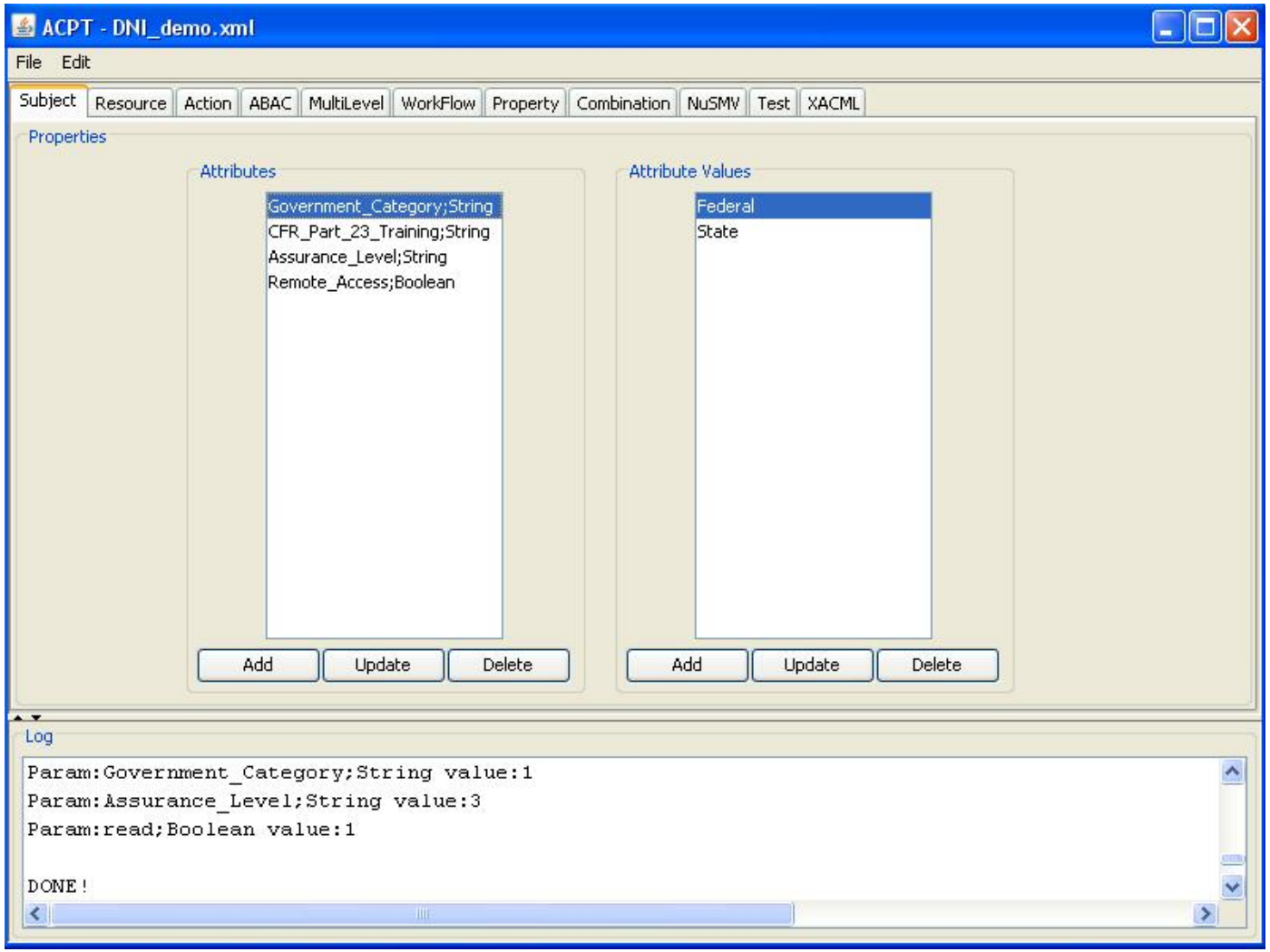


Approaches: AC Model Specification and Composition

Allow to conveniently specify mandatory AC models (as well as AC rules) through pre-defined model templates

- Create various models by specifying attribute values e.g., role subjects, resources, and actions for RBAC, user and resources ranks for MLS.
- Combine different AC models or rules into a composed one e.g., combine RBAC with Multi-Level models.
- Configure model priority for combining models or rules.

Approaches: AC Model Specification and Composition - Example



Approaches: Property Verification

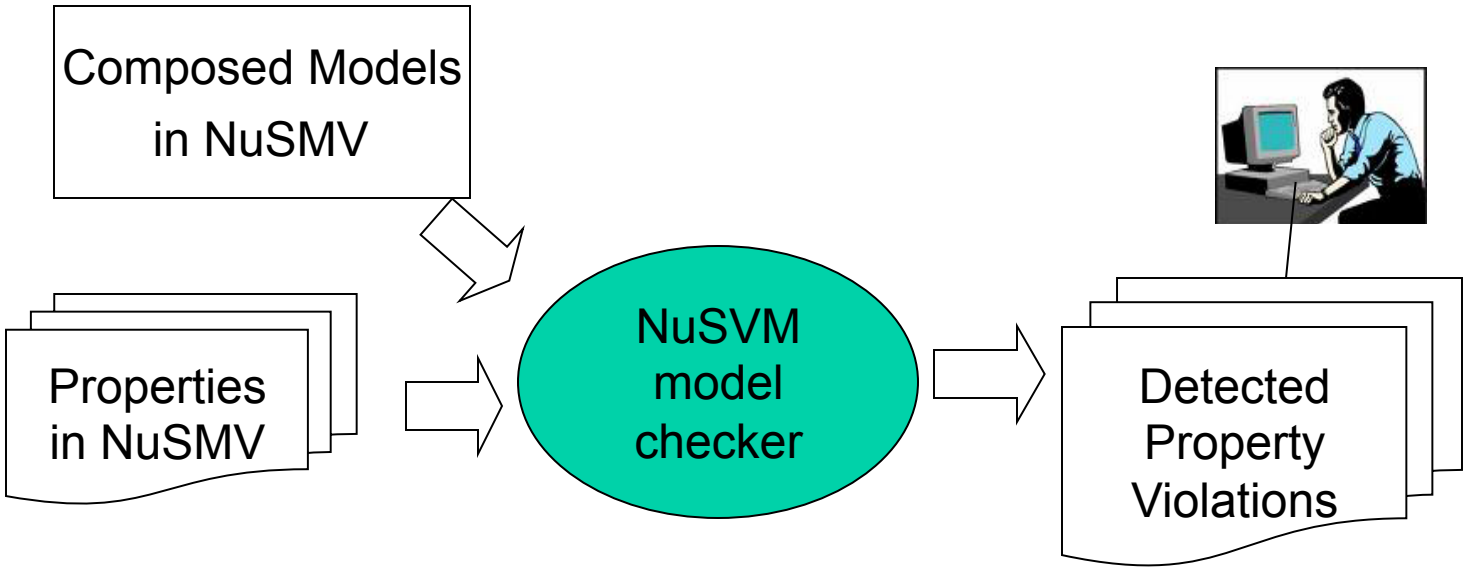
Conflicts among policy entities and their complexity may leak unauthorized or prohibit authorized access privileges.

Convert composed models and user-specified properties to input models and properties for the model checker - NuSMV.

Verify models against specified properties, and report detected property violations.

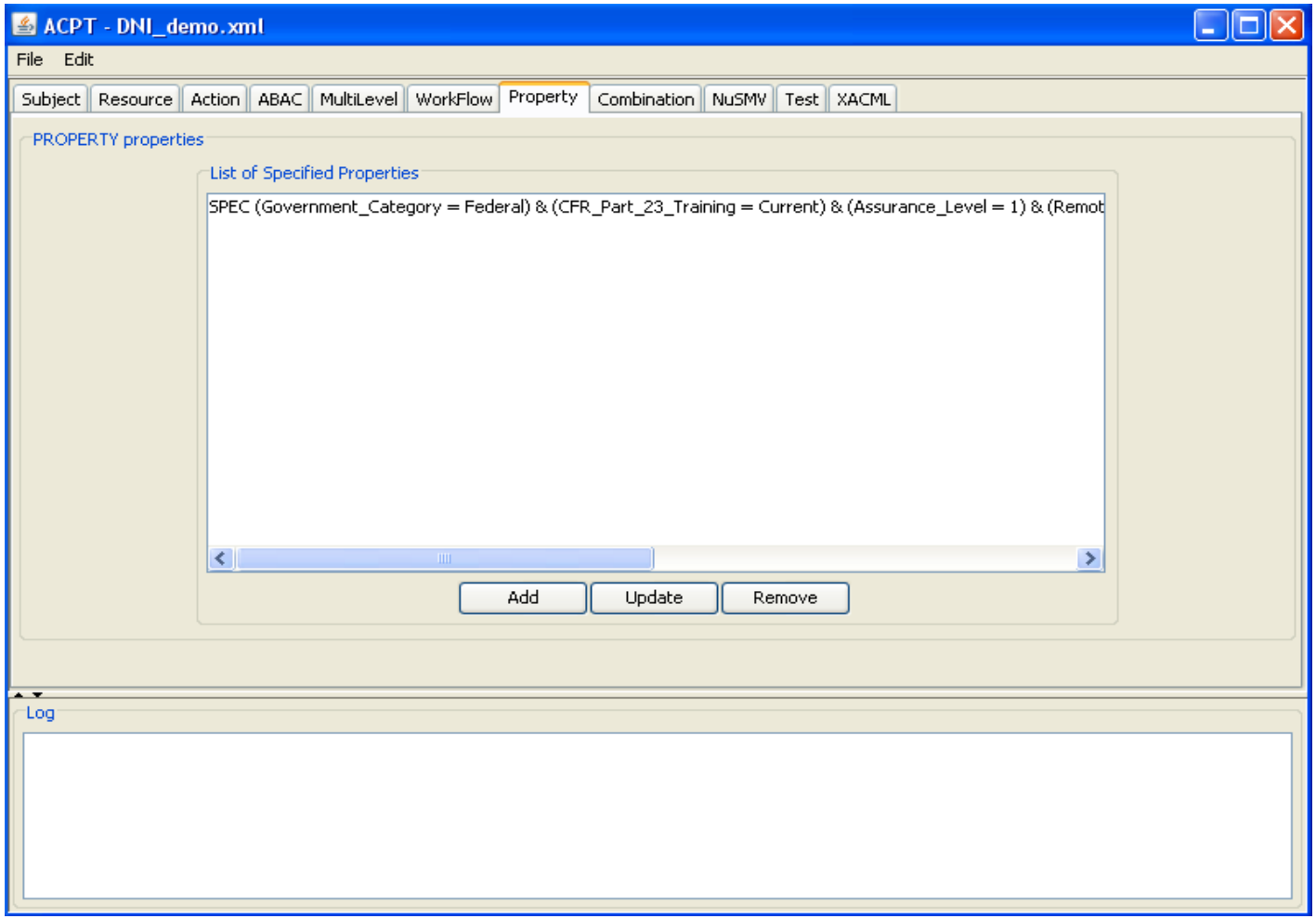
Approaches: Property Verification cont.

ACPT uses the NuSMV model checker, a well-structured, flexible, and efficient tool (supporting CTL and LTL model checking)



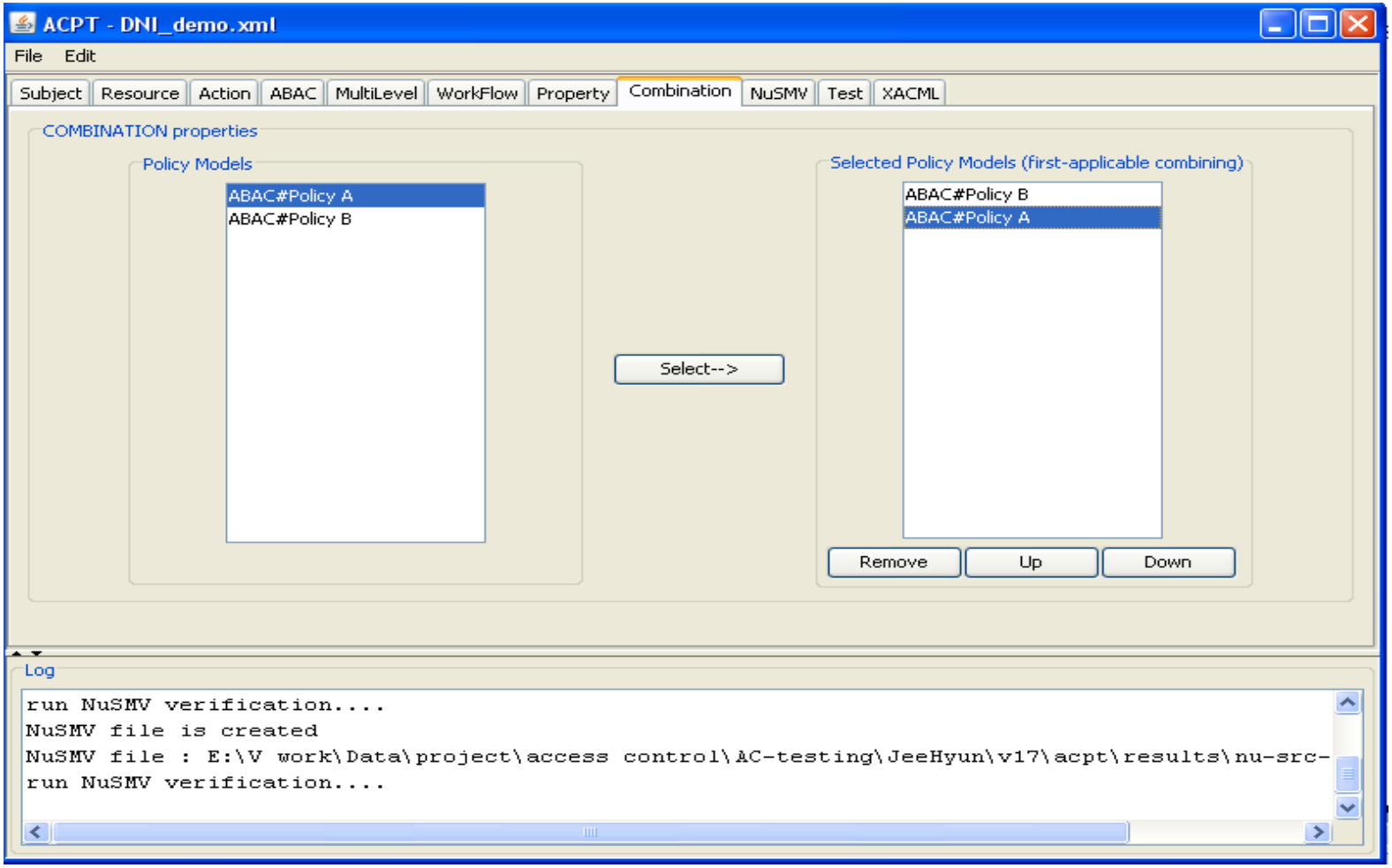
Approaches: Property Verification - Example

Property specification in ACPT



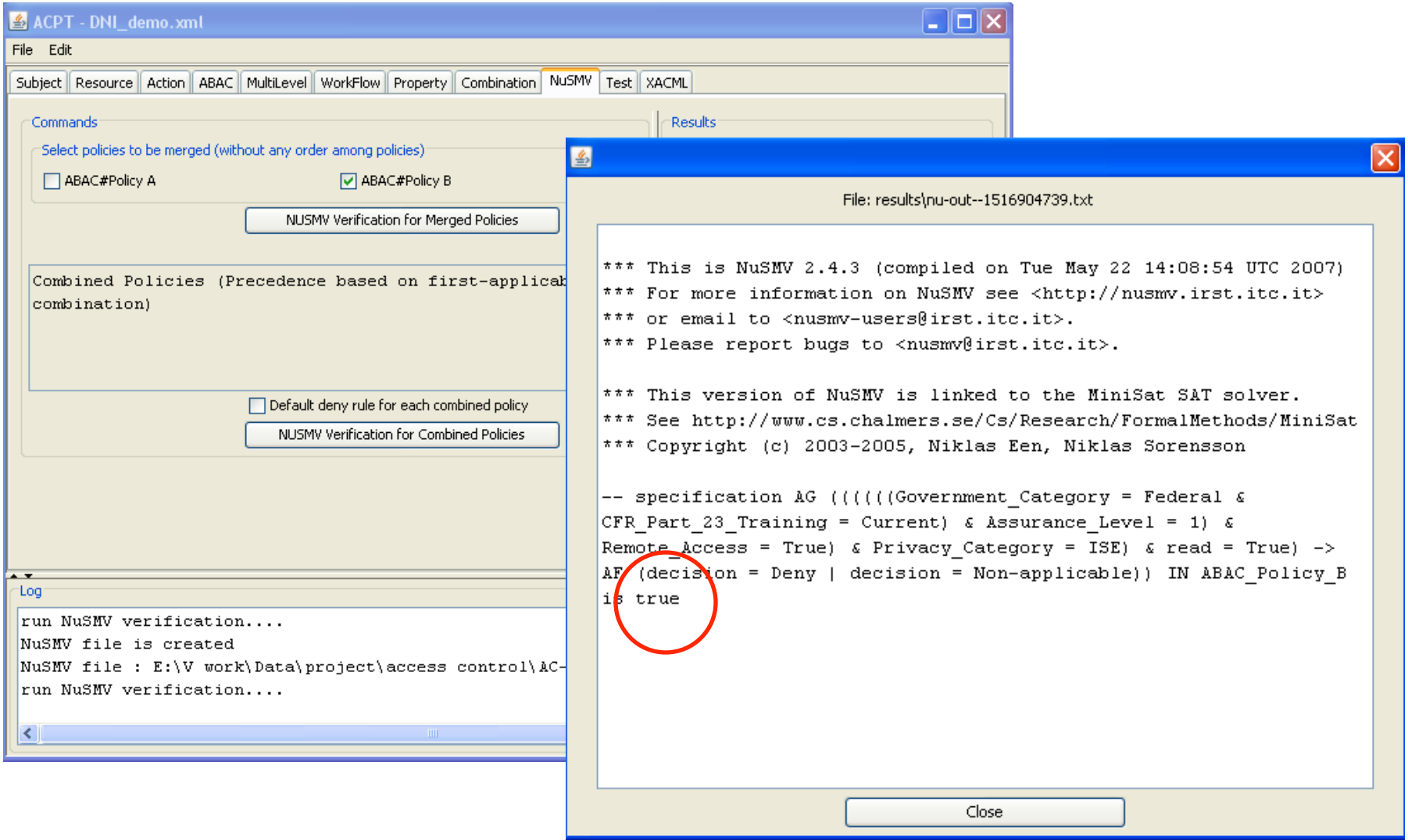
Approaches: Property Verification – Example cont.

Test the property against Policy A **combined** with Policy B. Combined polices has the priorities of the combined rules. This slide shows the combination of policies, where Policy B has higher priority than policy A



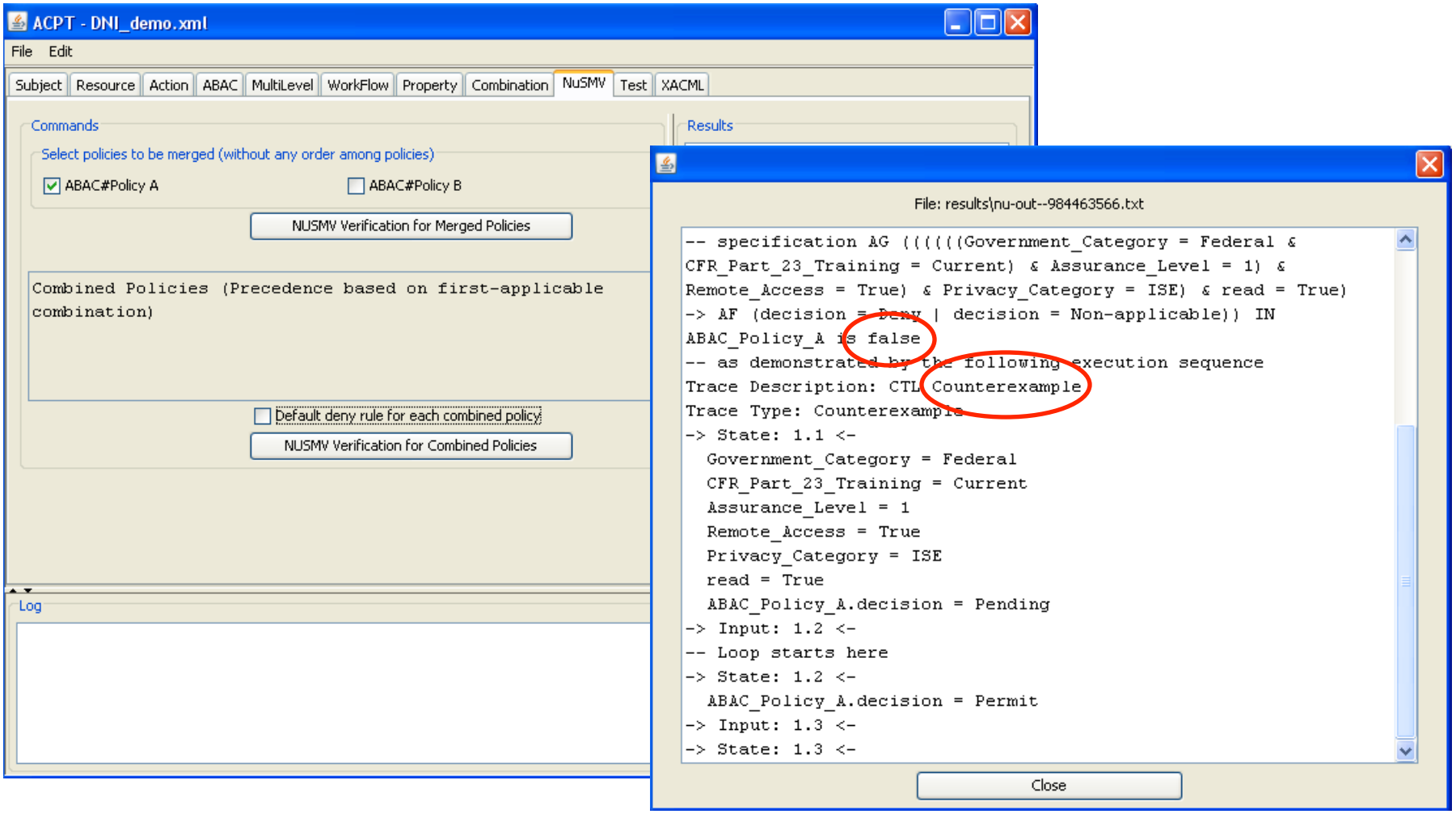
Approaches: Property Verification – Example cont.

Test the property against Policy B, the result return *true*.

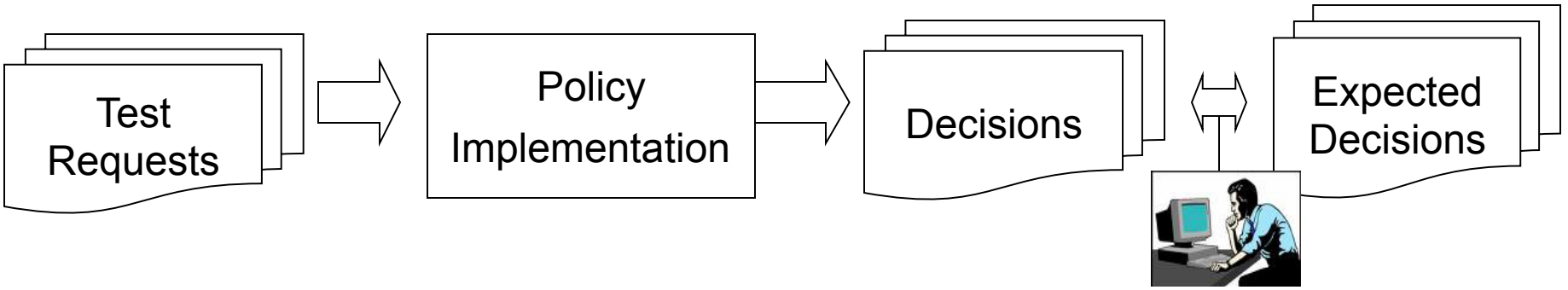


Approaches: Property Verification – Example cont.

Test the property against Policy A, the result return *false* with counterexample.



Approaches: Policy Testing



Assure correct policy implementations by

- Test Generation: Generate *test requests.*
- Test Execution: Evaluate test requests (against policy implementations) and produce their decisions.
- Test-Result Evaluation: Check if the decisions are consistent with expected decisions (from properties or manual inspection, etc.).
  - If inconsistent, implementation faults are revealed.

Approaches: Policy Testing – Combinatorial Testing

Exhaustive testing is impractical (esp. for large number of AC entities).

Generating efficient and effective test suites (from AC models) using t-way covering array generation tool, NIST ACTS.

Generated test suites can be applied to any access control implementations in practice to find implementation faults

Approaches: Policy Testing - Combinatorial Test cont.

Collect domain variables in AC models and generate ***efficient*** test suite automatically to detect faults using NIST combinatorial testing tool (ACTS)

- inputs: a domain of variables
- outputs: t-way covering arrays as tests

For example, with 34 on-off switches, we have 2^{34} = 17 billion possible inputs:

Approaches: Policy Testing - Combinatorial Test cont.

- Combinatorial tests based on 2-way interactions: 11 tests
- Combinatorial tests based on 3-way interactions: 33 tests
- … 4-way interactions: 85 tests
- 5-way interactions: 211 tests
- 6-way interactions: 522 tests

Rationale: empirically-derived Interaction Rule, that failures involve only a small number of conditions interacting (≤ 6 in real-world reports)

Approaches: Policy Testing - Combinatorial Test cont.

Access control rules example,

Domain of variables:

- 2 subjects: Faculty and Student
- 2 actions: write and view
- 2 resources: grades and records

Given the domain, 4 and 8 tests are generated for 2-way and 3-way interactions, respectively

<Faculty, grades, write>, <Faculty, records, view >, …

Approaches: Policy Testing - Combinatorial Test cont.

- Combinatorial tests based on 2-way interactions

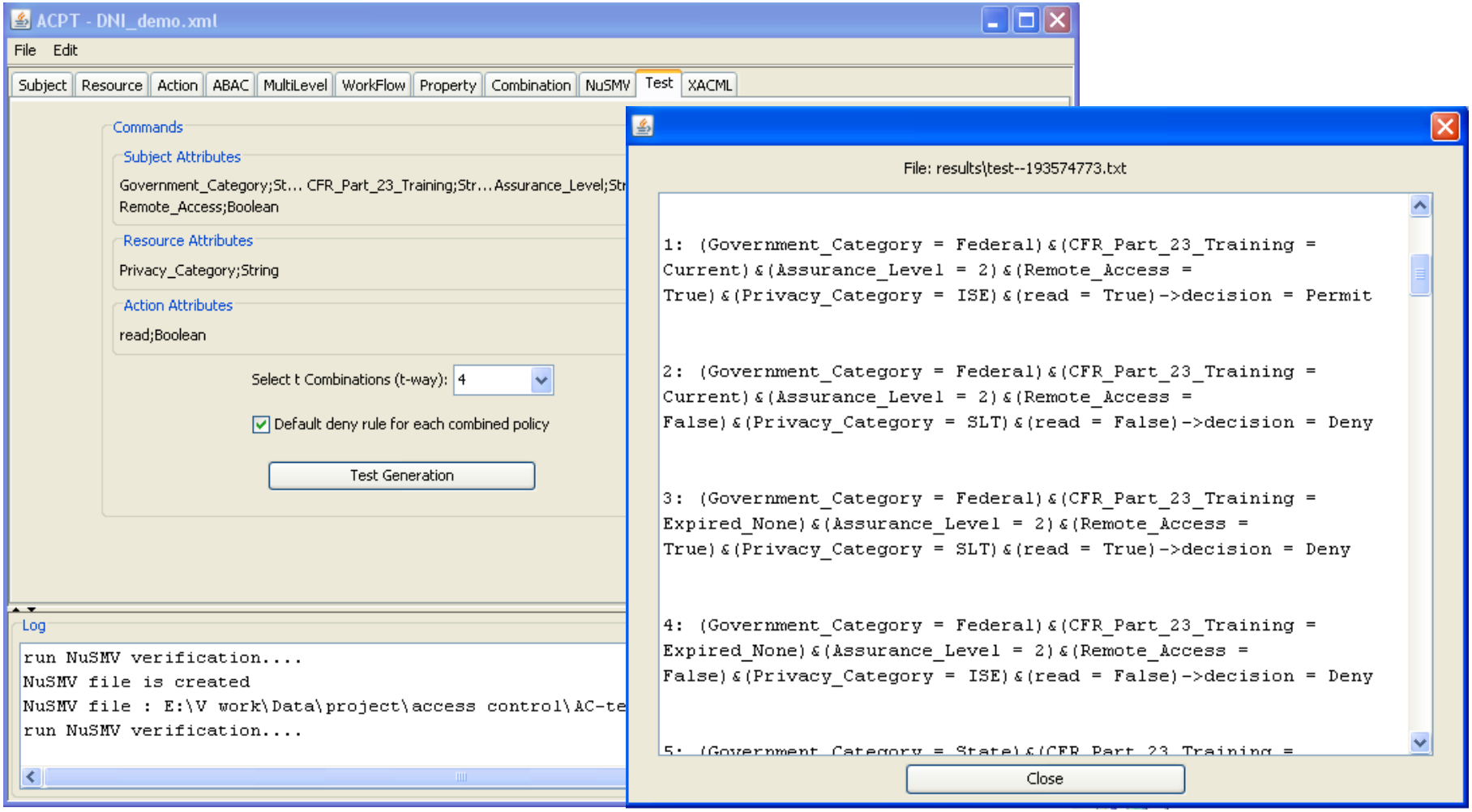
	SUBJECTS	RESOURCES	ACTIONS
1	Faculty	grades	write
2	Faculty	records	view
3	Student	grades	view
4	Student	records	write

- Combinatorial tests based on 3-way interactions (being exhaustive tests)

	SUBJECTS	RESOURCES	ACTIONS
1	Faculty	grades	write
2	Faculty	grades	view
3	Faculty	records	write
4	Faculty	records	view
5	Student	grades	write
6	Student	grades	view
7	Student	records	write
8	Student	records	view

Approaches: Policy Testing – Example

Test cases generation:

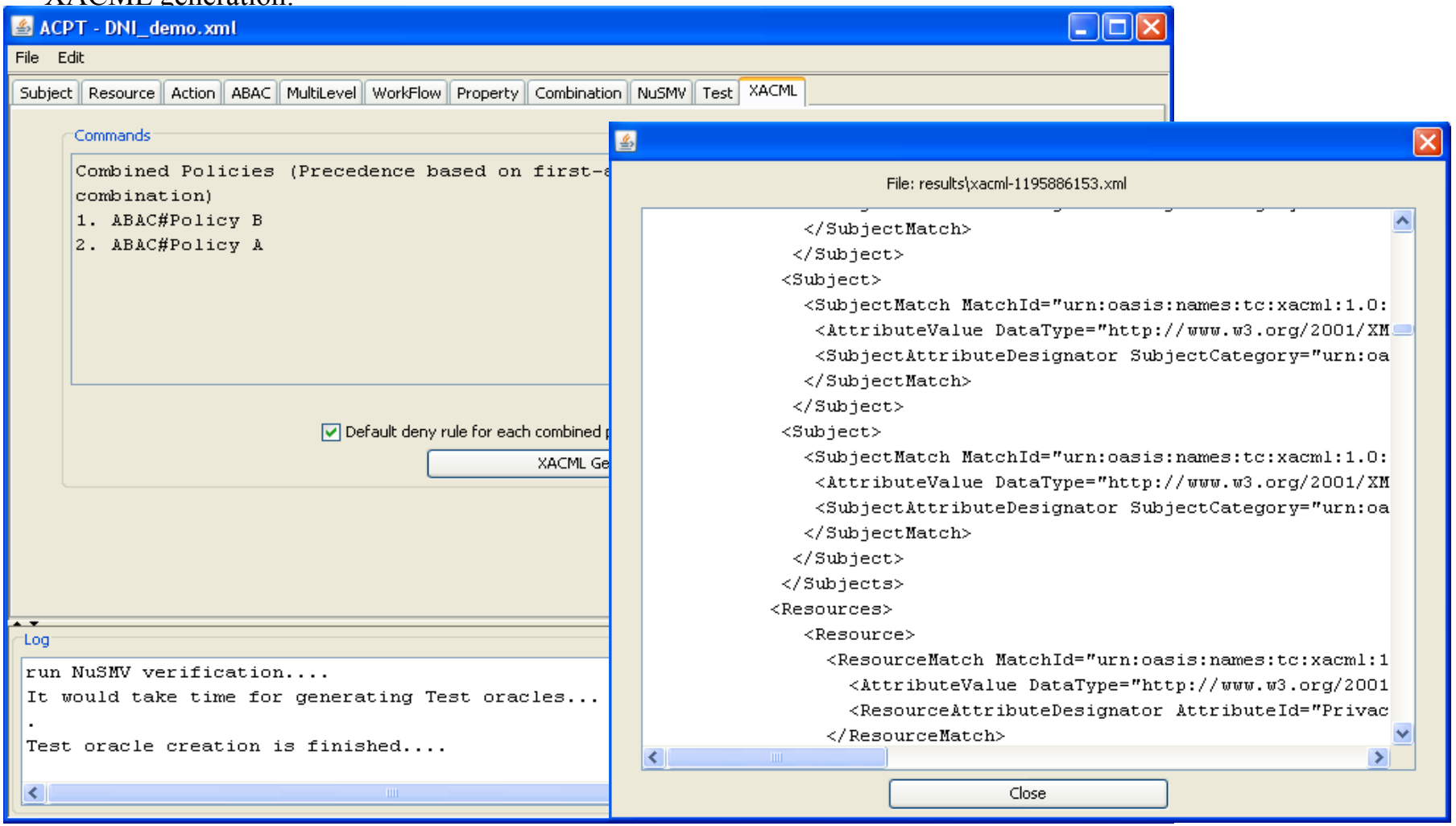


Approaches: XACML Generation

Generate XACML policy based on the verified (combined or individual) models and rules.

Approaches: XACML Generation – Example

XACML generation:



Approaches: XACML Generation – Example cont.

```
<PolicySet PolicySetId="n" PolicyCombiningAlgId="First-Applicable">
  <Target/>
  <Policy PolicyId="RBAC_school" RuleCombinationAlgId="First-Applicable">
    <Target/>
    <Rule RuleId="1" Effect="Deny">
      <Target>
        <Subjects><Subject>    Student    </Subject>
                  <Subject>    Secretary  </Subject></Subjects>
        <Resources><Resource> Grades     </Resource></Resources>
        <Actions><Action>       Change     </Action></Actions>
      </Target>
    </Rule>
    <Rule RuleId="2" Effect="Permit">
      <Target>
        <Subjects><Subject>    Professor  </Subject>
                  <Subject>    Lecturer   </Subject>
                  <Subject>    Secretary  </Subject></Subjects>
        <Resources><Resource> Grades     </Resource>
                   <Resource> Records    </Resource></Resources>
        <Actions><Action>       Change     </Action>
                                           </Actions>
      </Target>
    </Rule>
  </Policy>
  <Policy PolicyId="ABAC_school" RuleCombinationAlgId="First-Applicable">
    <Target/>
    <Rule RuleId="3" Effect="Permit">
      <Target>
        <Subjects><Subject>    Jim        </Subject></Subjects>
        <Resources><Resource> Records    </Resource></Resources>
        <Actions><Action>       Change     </Action>
                 <Action>       Read       </Action></Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
```

Rule 1:
A student or secretary can not change grades.

Rule 2:
A professor, lecturer, or secretary can change grades or records.

RBAC_school policy

Rule 3:
Jim can change grades or records.

Policy rules

Related Work: Compare with Commercial AC Tools

Commercial AC policy management tools do not have all the following capabilities that NIST ACPT has:

- **AC model templates** for specifying models/polices: ABAC, Multi-Level, and Workflow.
- **Composition of multiple AC models** into a composed one, e.g., combine RBAC with MLS models.
- **AC property verification** to detect faults in models/policies. Some have only limited SOD (Separation of Duty) check.
- **Test-suite generation** for testing AC implementations in real operation environment to detect faults in implementations.

Future Work

- White-box model/properties verification to verify coverage and confinement of AC rules.
- Additional AC policy templates including dynamic and historical access control models.
- API or mechanism for acquiring or consuming information about users, attributes, resources, etc.
- More flexible user interfaces for composing AC rules and properties.
- AC property sets for specified AC requirements.

Questions?

http://csrc.nist.gov/groups/SNS/acpt/index.html
vhu@nist.gov
d.kuhn@nist.gov