2nd Workshop on Security of Systems & Software resiLiency
Besançon, France
November 20-23, 2012
(iThings'12, GreenCom'12, CPSCom'12)

Heidelberger Institut für Theoretische Studien

HITS

A conflict detection approach for XACML policies on hierarchical resources

University Heidelberg / HITS

Xiaofeng Xia

20. 11. 2012

Agenda

- ✓ Problem definition
- ✓ Related concepts
- ✓ Specifying authorization and restrictions
- ✓ The approach to static conflict analysis
- ✓ Testing of the approach
- ✓ Problems and future work

HITS

1. Problem Definition

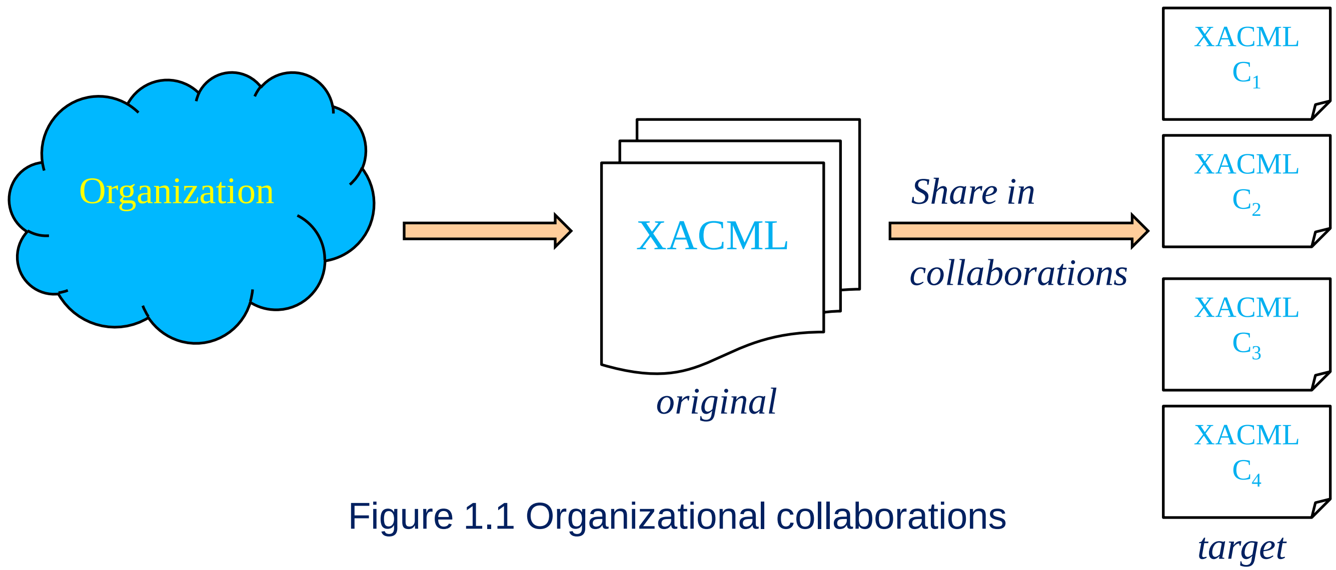
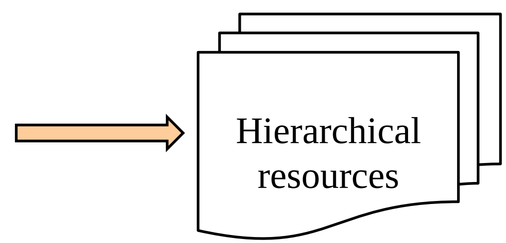


Figure 1.1 Organizational collaborations

P_1: Authorization conflicts

P_2: Conditional conflicts (attribute-based)



P_3: Handling large number of resources

2. Related concepts

2.1 Hierarchical resources

- A resource organized as a hierarchy may be:

  Tree | DAG | Polyarchy (Forest)

- Why hierarchical ?

  Authorization and constraint granularity

2.2 XACML elements

PolicySet Policy Rule

Condition Target Combining algorithm

2. Related concepts

2.2 XACML elements (ct.)

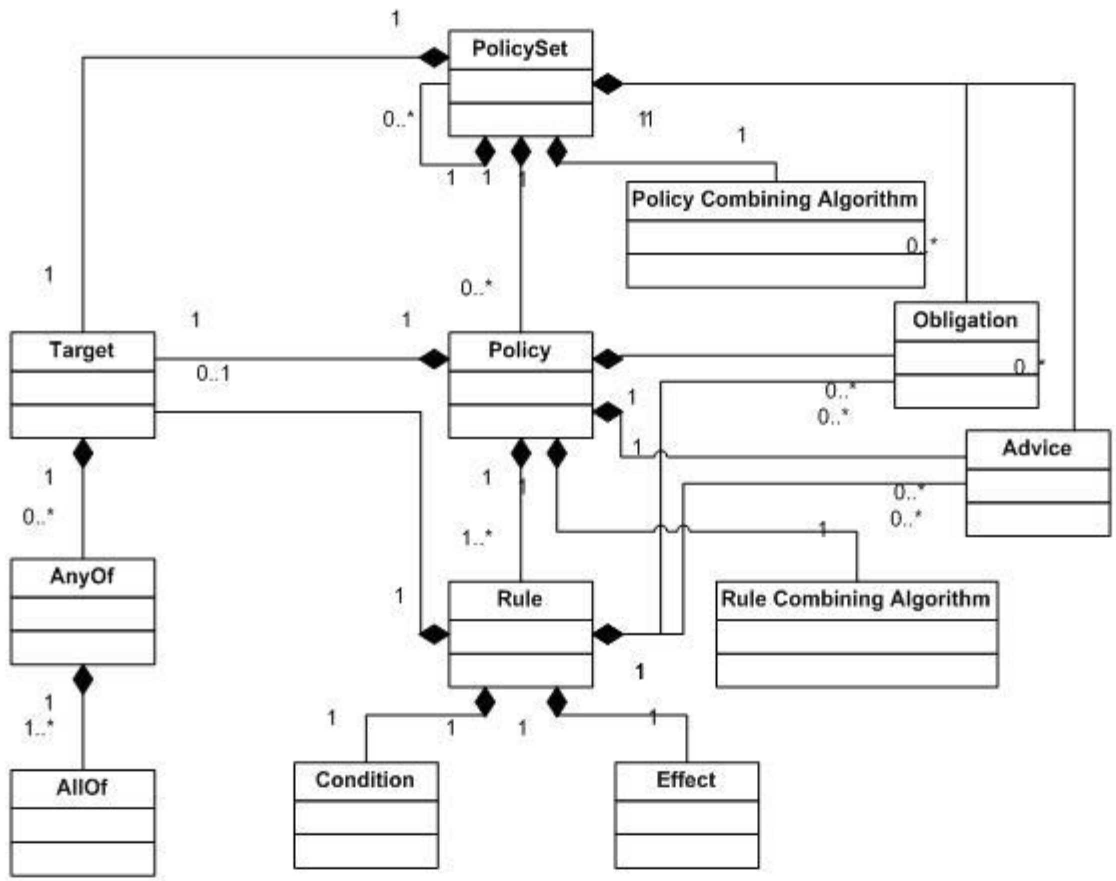


Figure 2.1 XACML policy language model

3. Specifying authorization and restrictions

3.1 Resource graph

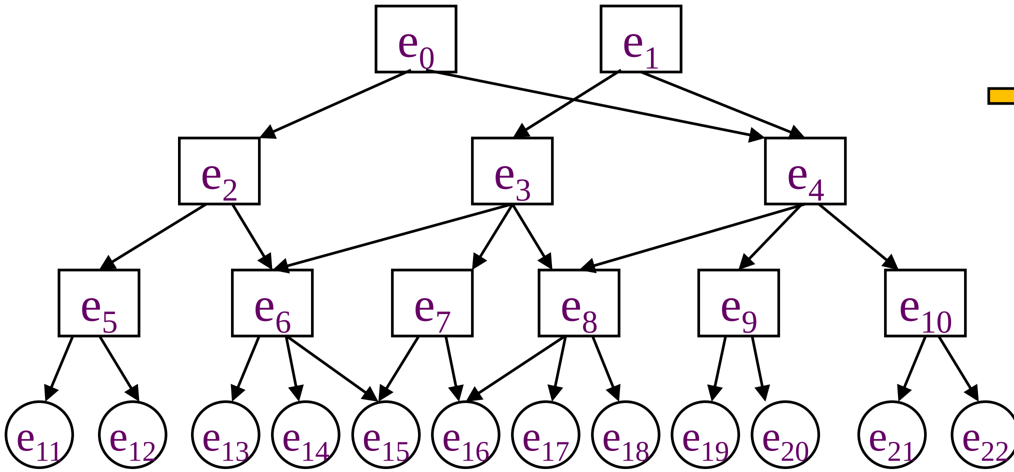
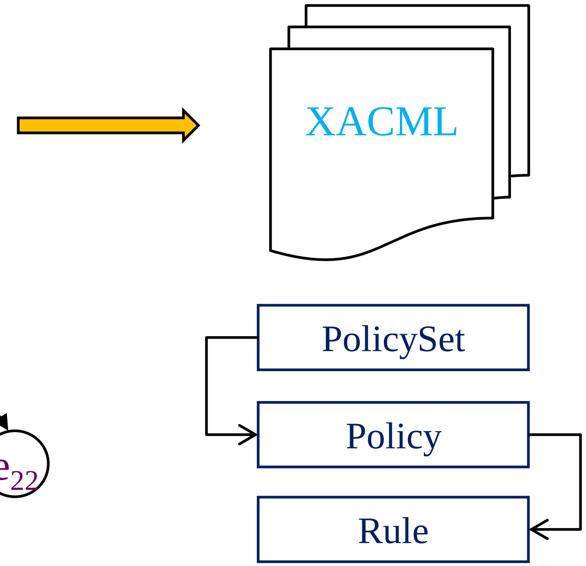


Figure 3.1 hierarchical resource graph (DAG)



- The hierarchical relations of DAG can be mapped into XACML policies with the hierarchical relations of XACML elements.

- How to represent authorization and constraint granularity

3. Specifying authorization and restrictions

3.2 Specifying with XACML

- An important element: (Policy/Rule) CombiningAlgorithm

  Used for making decision on multiple (PolicySet/Policy/Rule) evaluations

  Two typicals are used: PermitOverrides (PO)
  DenyOverrides (DO)

- For each resource node it corresponds with a „PolicySet“ element.

- Each resource node has 2 „PolicySet“ as sub-elements:

  Condition and Connector

- Each atomic resource node has 3 action types: Read/Write/Execute

3. Specifying authorization and restrictions

3.2 Specifying with XACML (ct.)

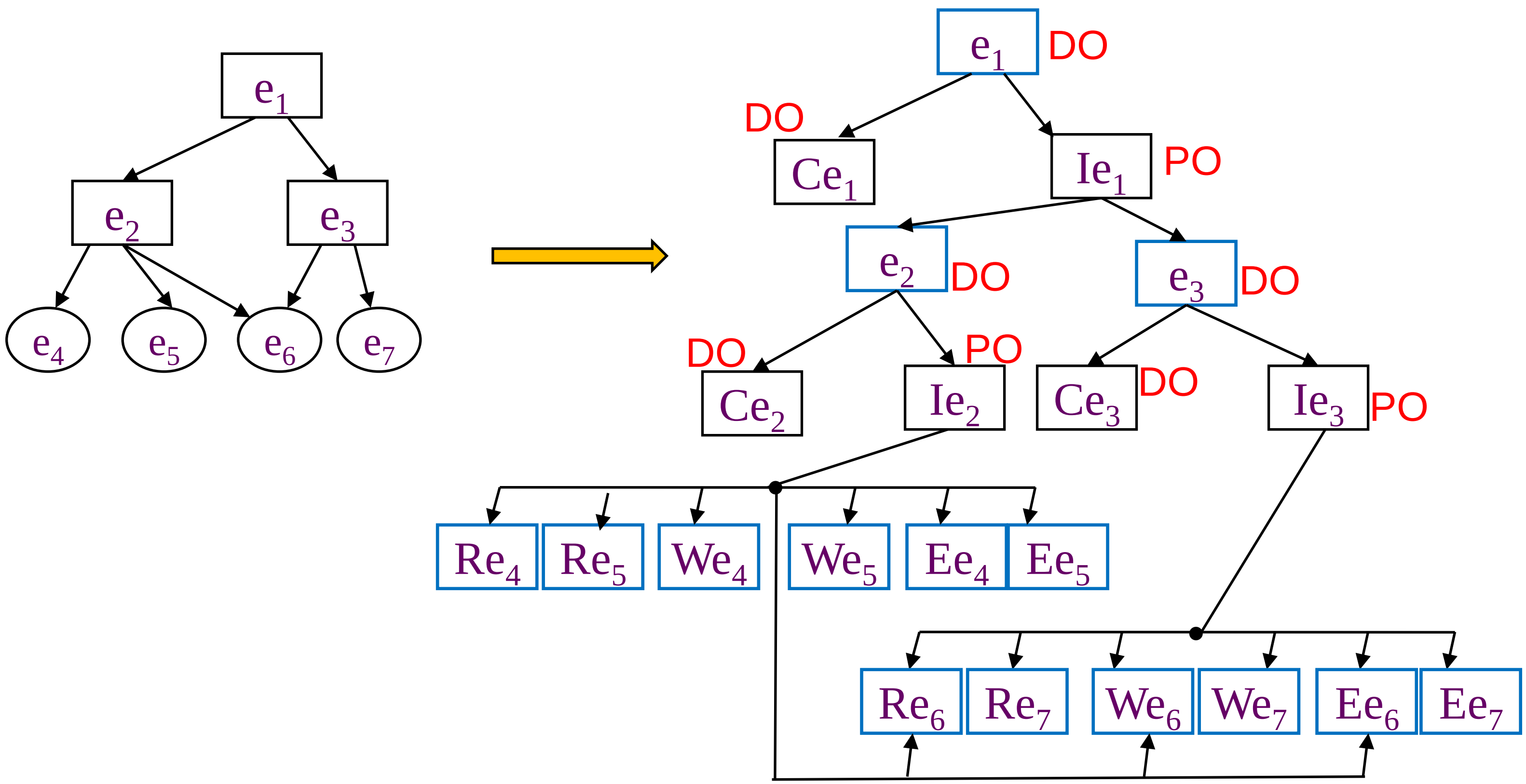


Figure 3.2 XACML specification of resources

4. The approach to static conflict analysis

4.1 The framework of approach

1. **Simulation**
   - Generating DAG for resources
   - Generating XACML specifications for DAG
2. **XACML parsing**
   - Parsing role authorizations
   - Parsing resource graph and restrictions for original settings
   - Parsing resource graph and restrictions for target settings
3. **Graph decomposition**
   - Setting the bound of resource nodes involed in a FSM
   - Algorithm for handling two typical cases
4. **Building FSM**
   - FSM state transitions
   - FSM specifications
5. **Model checking**
   - Handling authorization conflicts
   - Handling conditional conflicts

4. The approach to static conflict analysis

4.2.1 XACML parsing

- Original and target XACML specifications are based on same resource structure, but have possibly different constraints on resource nodes
- The constraints must be mapped onto corresponding node
- The constraints must be „pushed down" to descendant nodes

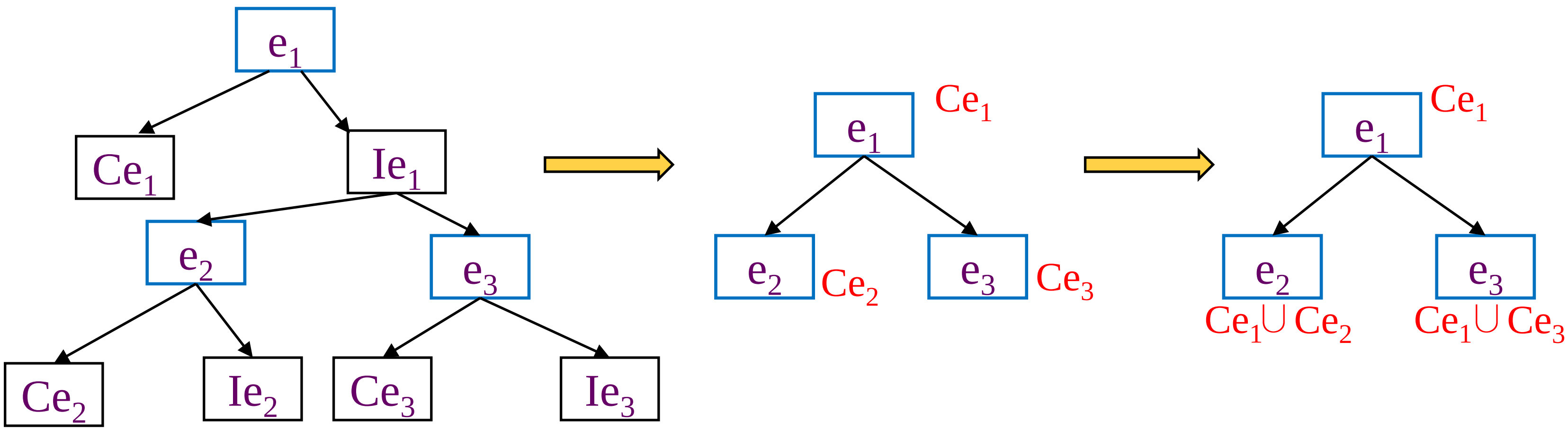


Figure 4.1 „pushing down " constraints in XACML parsing

4. The approach to static conflict analysis

4.2.2 Graph decomposition

- Graph decomposition for DAG is feasible by setting a bound of decomposing
- Algorithm for handling two typical cases by the number of descendants

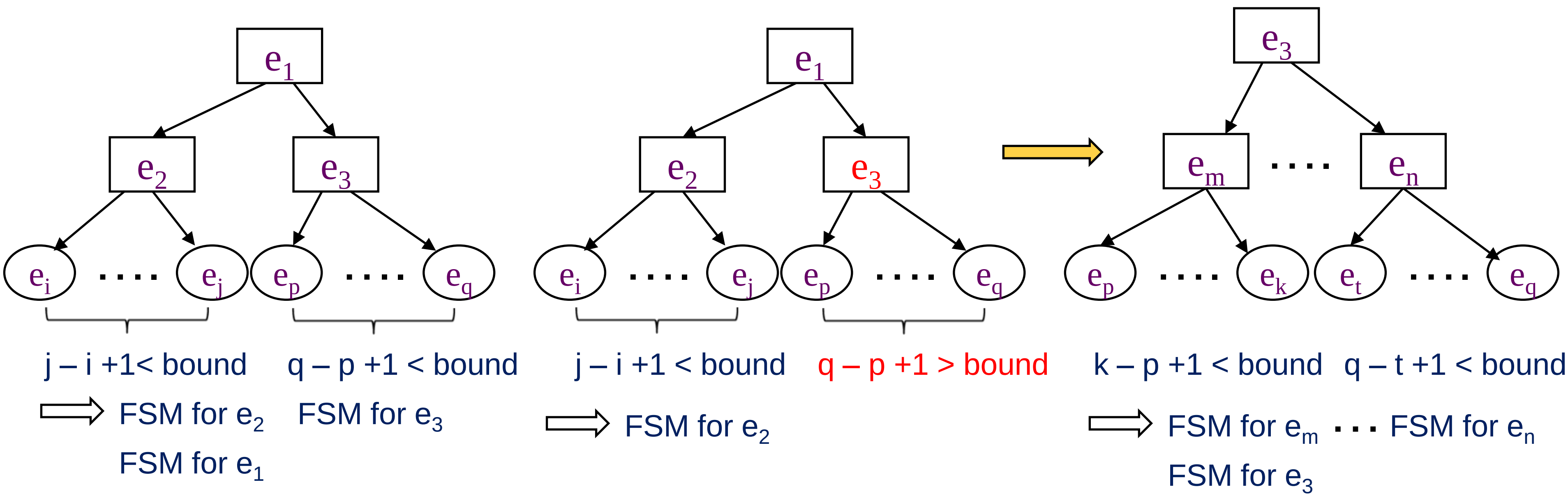


Figure 4.2 Two typical cases in graph decomposition

4. The approach to static conflict analysis

4.2.3 Building FSM and model checking

State variables
Resource nodes
Actions (R, W, E)
Conditions

State transitions
Resource nodes
Actions
Conditions

Specifications
Resource nodes
Actions
Conditions

FSM

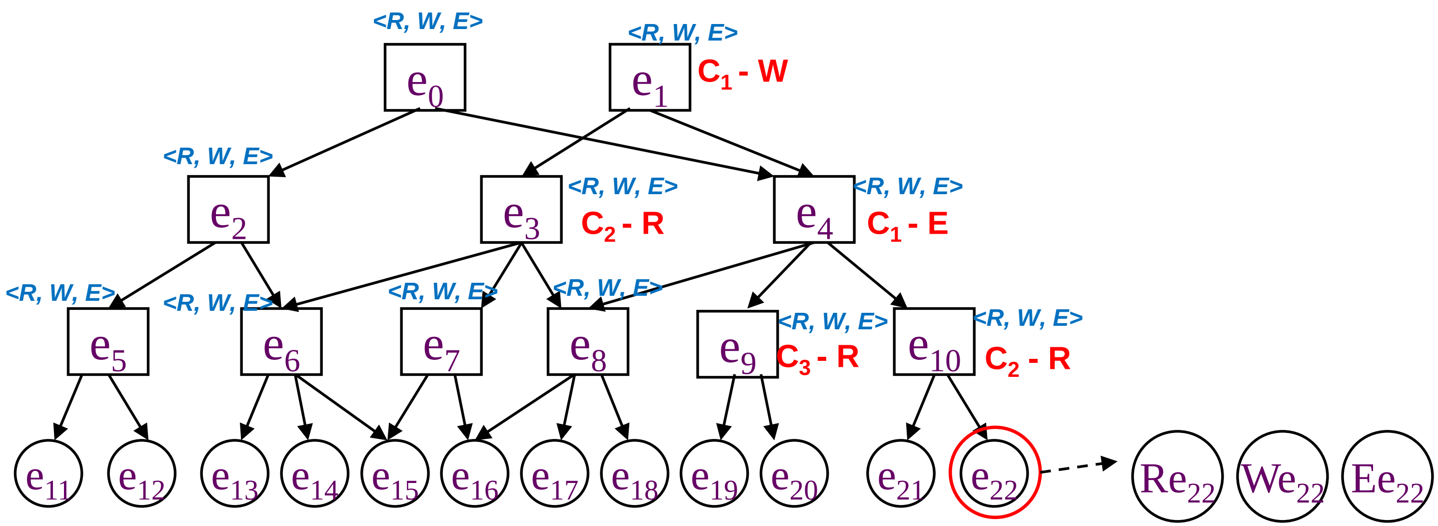
- Each FSM handles only one action type
- Authorization and conditional conflicts are separately handled

- Resource node relations are built as state transitions
- Specifying conditions in TARGET XACML for action variables

- Resource nodes are randomly selected
- Specifying conditions in ORIGINAL XACML for action variables

4. The approach to static conflict analysis

4.2.3 Building FSM and model checking (ct.)



- Authorization conflicts

  Finding out a path from original authorization nodes to target nodes

- Conditional conflicts

  Precisely finding out which nodes have conflicts in target XACML

4. The approach to static conflict analysis

4.2.3 Building FSM and model checking (ct.)

- Authorization conflicts

  E.g. Assuming a role „r“ has following authorizations:

Original spec. : $<r, e_3>, <r, e_9>$

Target spec. : $<r, e_7>, <r, e_{10}>$

$AG\ ((L=e_3) \rightarrow EF(L=e_7))$ T

$AG\ ((L=e_3) \rightarrow EF(L=e_{10}))$ F

$AG\ ((L=e_9) \rightarrow EF(L=e_7))$ F

$AG\ ((L=e_9) \rightarrow EF(L=e_{10}))$ F

- Conditional conflicts

  E.g. The selected checking node is „e_4“:

$< e_4, e_4>$ $< e_4, e_9>$ $< e_4, e_{10}>$ $< e_9, e_{19}>$

$< e_9, e_{20}>$ $< e_{10}, e_{21}>$ $< e_{10}, e_{22}>$

The checking node „e_4“ has condition for „Execute“: **C_1 - E**

4. The approach to static conflict analysis

4.2.3 Building FSM and model checking (ct.)

- Conditional conflicts (ct.)

  E.g. The selected checking node is „e_4“:

$$
\begin{aligned}
&AG\,((L=e_4) \text{ -> } AG((L=e_4) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))\\
&AG\,((L=e_4) \text{ -> } AG((L=e_9) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))\\
&AG\,((L=e_4) \text{ -> } AG((L=e_{10}) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))\\
&AG\,((L=e_9) \text{ -> } AG((L=e_{19}) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))\\
&AG\,((L=e_9) \text{ -> } AG((L=e_{20}) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))\\
&AG\,((L=e_{10}) \text{ -> } AG((L=e_{21}) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))\\
&AG\,((L=e_{10}) \text{ -> } AG((L=e_{22}) \wedge \neg C_1\text{-}E \wedge \neg E \text{ -> } \neg EX(E)))
\end{aligned}
$$

Identifying the conflict nodes

5. Testing of the approach

5.1 Testing with increasing # of roles

# of resource nodes	# of roles	# of BDD nodes	Time(Sec)
1000	5	17583	0.5
1000	15	19266	0.59
1000	25	20341	0.65
1000	35	21170	0.73
1000	45	21967	0.79
1000	55	22304	0.91
1000	65	23343	0.94
1000	75	23734	1.04
1000	85	24082	1.07
1000	100	24710	1.18

5. Testing of the approach

5.2 Testing with increasing # of authorizations

# # of resource nodes	# of Auth.	# of BDD nodes	Time(Sec)
1000	5	30367	0.87
1000	10	31366	0.85
1000	15	32651	1.10
1000	20	33381	1.10
1000	25	34026	1.35
1000	30	35033	1.37
1000	35	34859	1.78
1000	40	35653	2.10
1000	45	35984	2.25
1000	50	36285	2.37

5. Testing of the approach

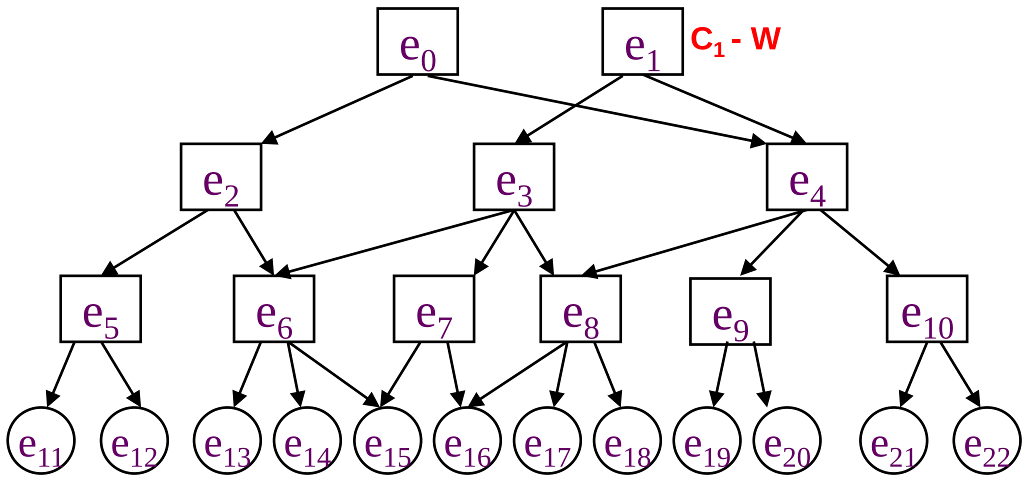
5.3 Testing with increasing # of resource nodes

# of resource nodes	# of conditions	# of BDD nodes	Time(Sec)
1000	0-3	130947	1.86
2000	0-3	242229	7.10
3000	0-3	374896	18.97
4000	0-3	481634	28.55
5000	0-3	650088	60.74
6000	0-3	762472	76.10
7000	0-3	853760	92.69
8000	0-3	1000578	111.57
9000	0-3	1204934	226.41
10000	0-3	1564373	270.12

5. Testing of the approach

5.3 Testing with increasing # of resource nodes (ct.)

# of resource nodes	# of conditions	# of BDD nodes	Time(Sec)
20000	0-3	2945227	143.66
30000	0-3	4294527	222.3
100000	0-3	15465862	2862.72



6. Problems and future work

6.1 Current work and problems

- Improving the algorithm for graph decomposition
- Hierarchical resources and XACML policies

6.2 Future work

- Try to find realistic system policies to improve this approach
- Conflicts detection in various collaboration patterns

Thank you!