Policy Conflict Analysis Using Free Variable Tableaux for Access Control in Web Services Environments

10th May 2005

Hiroaki Kamoda, Masaki Yamaoka, Shigeyuki Matsuda
NTT DATA COPORATION

Krysia Broda, Morris Sloman
Imperial College London

Agenda

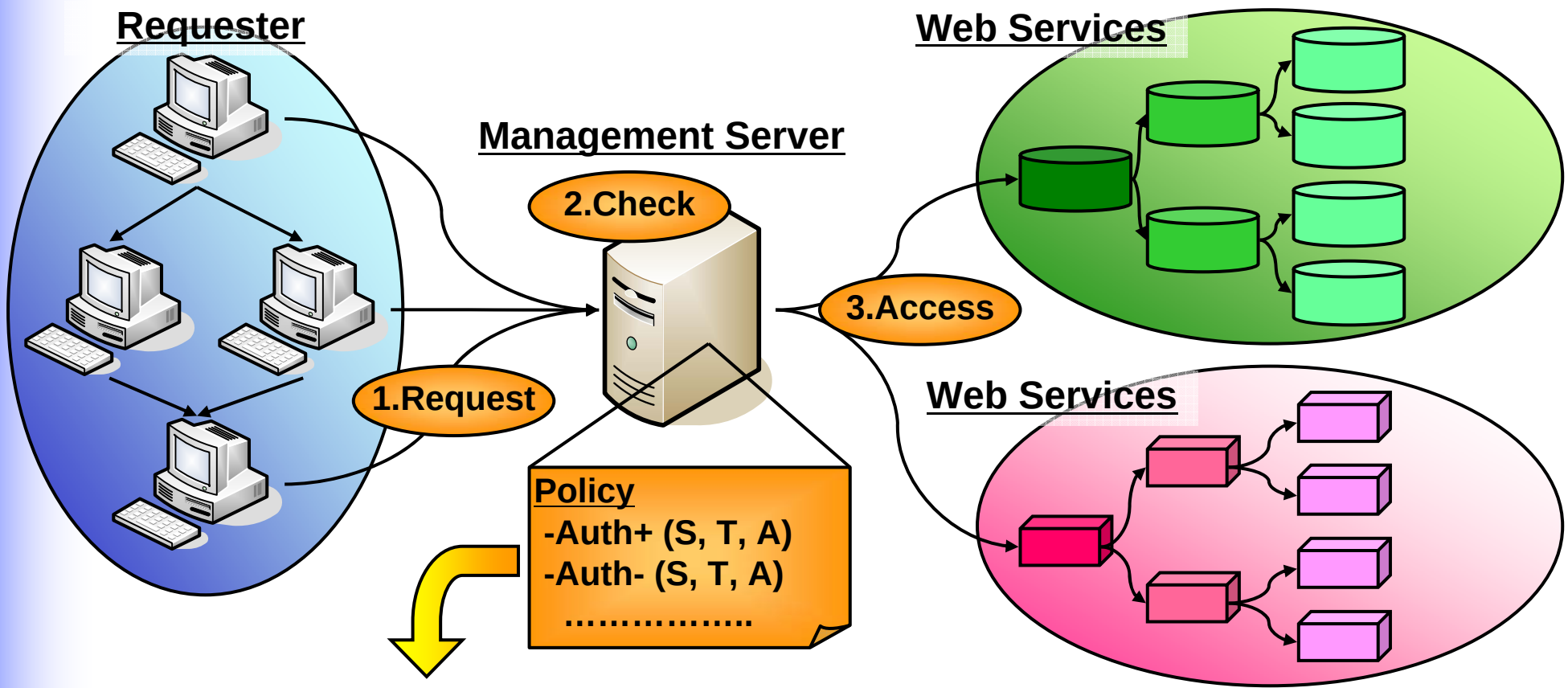
- Background
- Problem
  - Access Control Policy and Conflict
- Approach
  - Static Conflict Detection Method
- Conclusions and Future Work

Agenda

- Background
- Problem
  - Access Control Policy and Conflict
- Approach
  - Static Conflict Detection Method
- Conclusions and Future Work

Web Services and Access Control Policy

- Aggregation Web Services Model
  - A management server integrates several Web Services and their resources and provides a common services interface for requesters.
  - Ex. Travel agency Web Services.



Problem : Access Control Policy may include conflicting policy.
Goal : Establish a static conflict detection method.

Agenda

- Background
- Problem
  - Access Control Policy and Conflict
- Approach
  - Static Conflict Detection Method
- Conclusions and Future Work

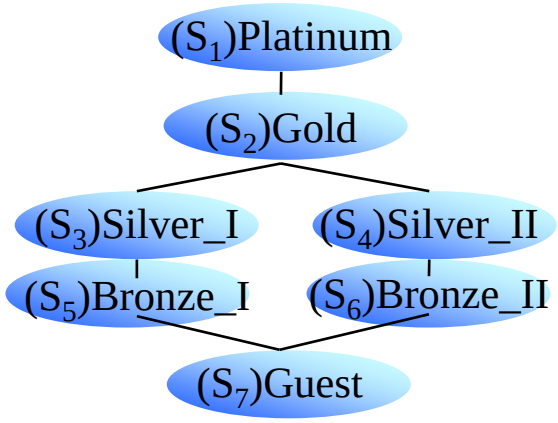
Access Control Policies and Roles

- **Policies** used for the Web Services
  - **Authorization Policy**
  - **Obligation Policy**

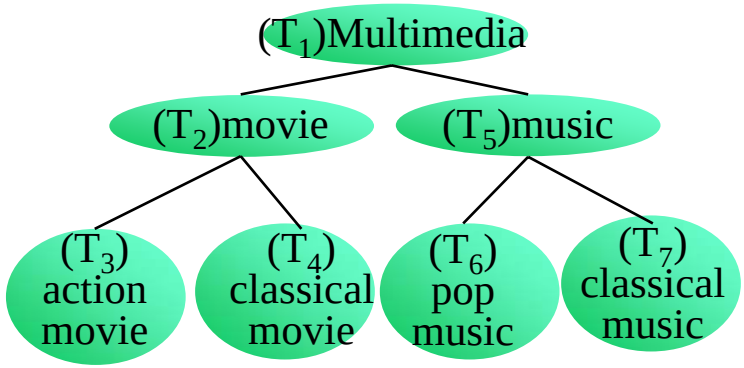
  Basic Policy
  - **Propagation Policy**

  Propagation Policy
  - **Action Composition Policy**
  - **Chinese Wall Policy**
  - **Separation of Duty Policy**
  - **Time Constraint Policy**

  Constraint Policy
- **Roles**
  - A role is a named collection of privileges.
  - Partial order relation is defined among roles.



(a) Subject Role Structure



(b) Target Role Structure

Basic Policy and Explicit Modality Conflict

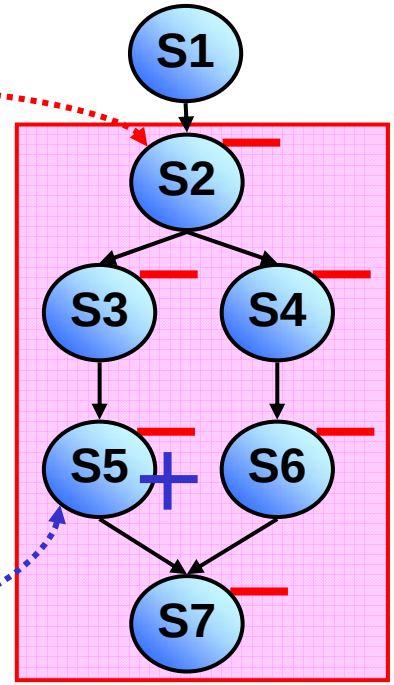
- **Authorization Policy**
  - `Auth+ (S, T, A) / Auth- (S, T, A)`
    - A positive/negative authorization policy defines the action "A" that a subject role "S" is permitted/prohibited to perform on a target role "T".
    - Ex. **`Auth+ (Bronze_I, movie, play)`**
    - Ex. **`Auth- (Gold, movie, play)`**
- **Obligation Policy**
  - `Obli+ (E, S, T, A) / Obli- (E, S, T, A)`
    - A positive/negative obligation policy defines the action "A" that a subject role "S" must/must not perform on a target role "T" when an event "E" occurs.
    - Ex. **`Obli+ (Play, Guest, fillout, questionnaire)`**
    - Ex. **`Obli- (Sunday, Guest, login, WebServices)`**

- **Explicit Modality Conflict**
  - Following three pairs of policies are defined as explicit modality conflict.
  - `Auth+ (S, T, A) / Auth- (S, T, A)`
  - `Obli+ (E, S, T, A) / Obli- (E, S, T, A)`
  - `Obli+ (E, S, T, A) / Auth- (S, T, A)`

Propagation Policy

- `Prop (Auth+|-, SRS|TRS, UP|DOWN)`
  A propagation policy defines how an authorization policy propagates in accordance with the partial order of the role structures.
- Examples
  - `Prop (Auth-, S ∈ SRS, DOWN)`
  - `Auth-(S2, T2, play)`

  These two policies derives following policies.
  - `Auth-(S3, T2, play) / Auth-(S4, T2, play)`
  - `Auth-(S5, T2, play) / Auth-(S6, T2, play)`
  - `Auth-(S7, T2, play)`



Subject Role Structure : S

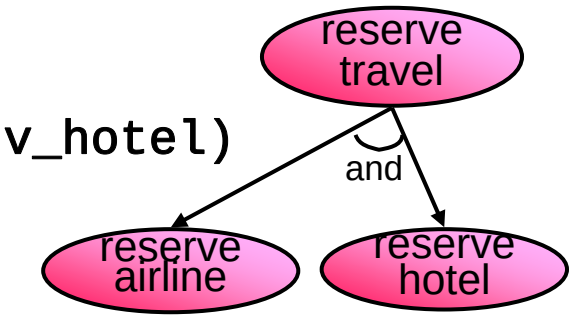
Implicit Modality Conflict

- Implicit Modality conflict is a modality conflict that occurs between the explicitly defined authorization policies and an authorization policies implicitly derived by the propagation policy.
- Examples:
  - `Prop (Auth-, S ∈ SRS, DOWN)`
  - `Auth-(S2, T2, play)`
  - `Auth+(S5, T2, play)`

Constraint Policy and Constraint Conflict (1/2)

- **Action Composition Policy**
  - Policies may be defined in terms of more than one action.
  - Example:
    `ac(reserv_travel = reserv_airline ∧ reserv_hotel)`
  - This action composition policy specifies that two actions `reserv_airline` and `reserv_hotel` are needed to complete the request `reserv_travel` .

- **Constraint Conflict**
  - An action composition policy may lead to constraint conflict.
  - Examples:
    - `ac(reserv_travel=reserv_airline∧reserv_hotel)`
    - `Auth+ (Guest, TR, reserve_travel)`
    - `Auth- (Guest, TR, reserve_hotel)`



Action composition policy

Constraint Policy and Constraint Conflict (2/2)

- **Chinese Wall Policy**
  - `CW (Subject, {T1, T2}, Action)`
    A Chinese wall policy defines two mutually exclusive target roles.
    Ex. **`CW (Guest, {Bank_A, Bank_B}, view_account)`**

- **Separation of Duty Policy**
  - `SoD (Subject, Target, {A1, A2})`
    A Separation of Duty policy defines two mutually exclusive actions.
    Ex. **`SoD (Guest, Auction, {sell, buy})`**

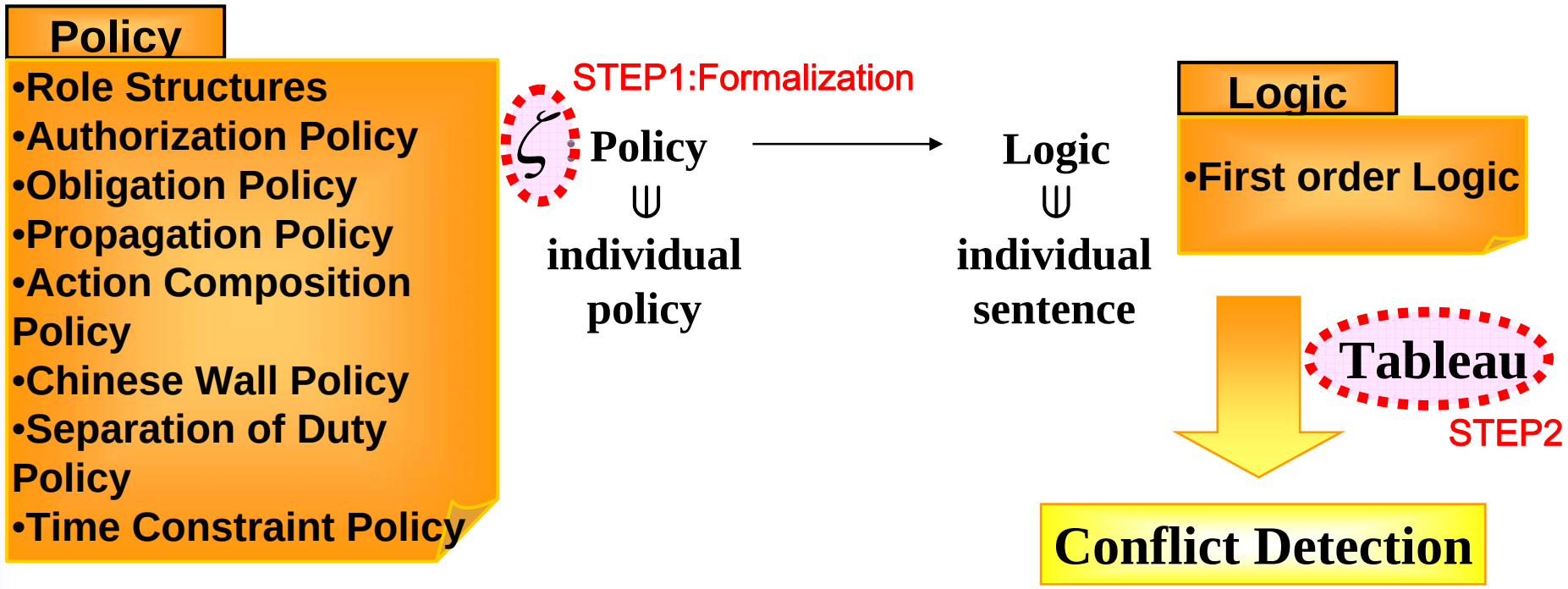
- **Constraint Conflict**
  - A Chinese wall policy and separation of duty policy may lead to constraint conflict.
  - Examples:
    - **`CW (Guest, {Bank_A, Bank_B}, view_account)`**
    - **`Auth+ (Guest, Bank_A, view_account)`**
    - **`Auth+ (Guest, Bank_B, view_account)`**

Agenda

- Background
- Problem
  - Access Control Policy and Conflict
- Approach
  - Static Conflict Detection Method
- Conclusions and Future Work

Formalization and the Free Variable Tableaux

- **Access Control Policy Formalization**



- **Free Variable Tableau**:
  - is the algorithm to detect an inconsistency of the set of sentences.
  - is sound and complete method.
  - has optimized implementations.
  - can abduce helpful information to resolve a conflict.

STEP1: Access Control Policy Formalization

- Authorization and Obligation Policy

$$
\begin{aligned}
\zeta(\texttt{Auth+}(S_1, T_1, A_1)) &:= \forall x(E_x \rightarrow P(S_1, T_1, A_1)) \\
\zeta(\texttt{Auth-}(S_1, T_1, A_1)) &:= \forall x(E_x \rightarrow \neg P(S_1, T_1, A_1)) \\
\zeta(\texttt{Obli+}(E_1, S_1, T_1, A_1)) &:= E_1 \rightarrow O(S_1, T_1, A_1) \\
\zeta(\texttt{Obli-}(E_1, S_1, T_1, A_1)) &:= E_1 \rightarrow R(S_1, T_1, A_1)
\end{aligned}
$$

  - “P” can be read as subject role S1 is permitted to carry out action A1 on target role T1.
  - “O” can be read as S1 must carry out action A1 on target role T1.
  - “R” can be read as S1 must not carry out action A1 on target role T1.
  - “Ex” can be read as an event Ex occurs.

- Axioms

$$
\begin{aligned}
\text{Ax1} &: \forall s, t, a(O(s, t, a) \rightarrow P(s, t, a)) \\
\text{Ax2} &: \forall s, t, a(\neg(O(s, t, a) \wedge R(s, t, a)))
\end{aligned}
$$

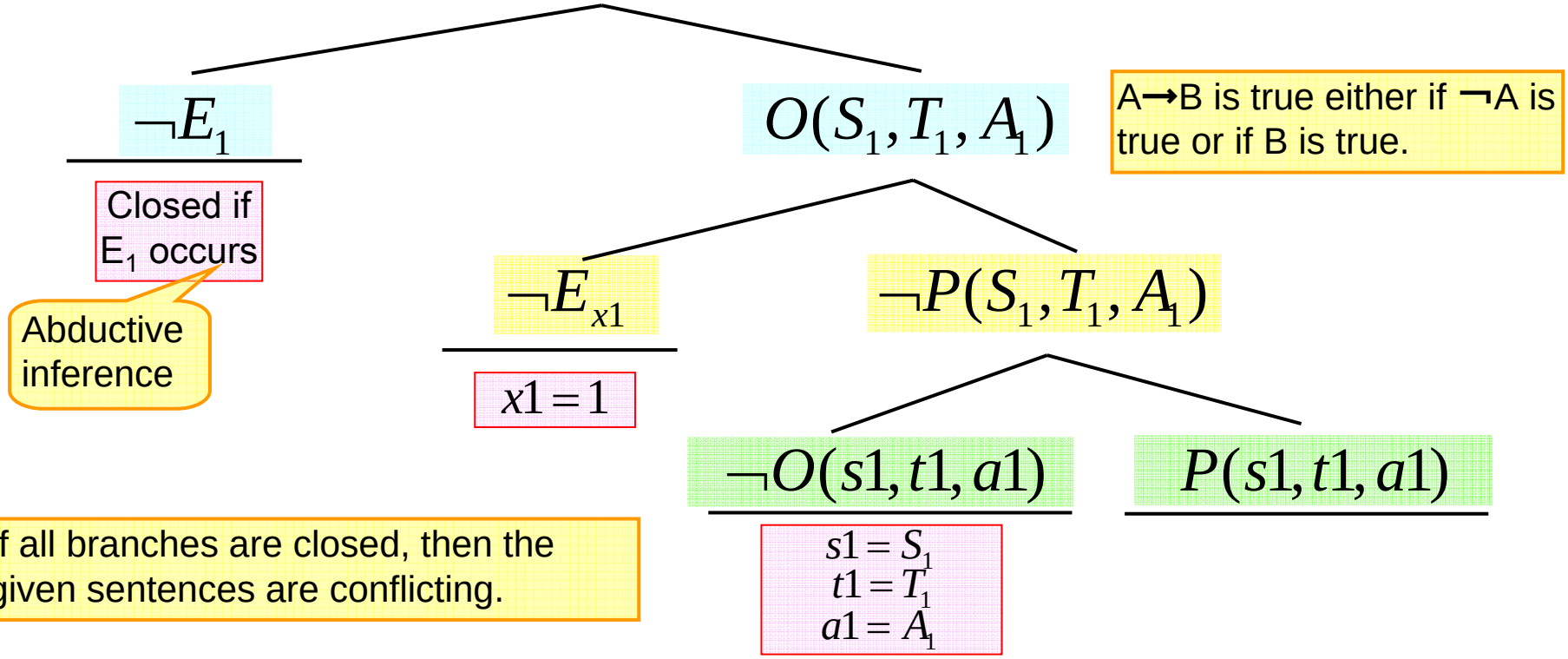
  - Ax1 is used to detect conflicts involving both authorization and obligation policies.
  - Ax2 is used to detect conflicts between positive and negative obligation policies.

STEP2: Explicit Modality Conflict Detection

`Obli+(E1,S1,T1,A1)` $\mapsto$ $E_1 \rightarrow O(S_1, T_1, A_1)$

`Auth-(S1,T1,A1)` $\mapsto$ $\forall x(E_x \rightarrow \neg P(S_1, T_1, A_1))$

`Ax1` $\mapsto$ $\forall s,t,a(O(s,t,a) \rightarrow P(s,t,a))$



This result shows that Obli+ policy and Auth- policy become a conflicting policy if event E_1 occurs.

STEP1: Access Control Policy Formalization

- **Propagation Policy**
  - There are 8 types of propagation policies.

prop1 : prop(Auth+,R∈SRS,UP)
prop2 : prop(Auth-,R∈SRS,DOWN)
prop3 : prop(Auth+,R∈SRS,DOWN)
prop4 : prop(Auth-,R∈SRS,UP)
prop5 : prop(Auth+,R∈TRS,UP)
prop6 : prop(Auth-,R∈TRS,DOWN)
prop7 : prop(Auth+,R∈TRS,DOWN)
prop8 : prop(Auth-,R∈TRS,UP)

$$\zeta(\mathrm{prop1}) = \zeta(\mathrm{prop2}) := \forall x, y, z, a(P(x, y, a) \wedge H_{\mathcal{R}}(z, x) \rightarrow P(z, y, a))$$
$$\zeta(\mathrm{prop3}) = \zeta(\mathrm{prop4}) := \forall x, y, z, a(P(x, y, a) \wedge H_{\mathcal{R}}(x, z) \rightarrow P(z, y, a))$$
$$\zeta(\mathrm{prop5}) = \zeta(\mathrm{prop6}) := \forall x, y, z, a(P(x, y, a) \wedge H_{\mathcal{R}}(y, z) \rightarrow P(x, z, a))$$
$$\zeta(\mathrm{prop7}) = \zeta(\mathrm{prop8}) := \forall x, y, z, a(P(x, y, a) \wedge H_{\mathcal{R}}(z, y) \rightarrow P(x, z, a))$$

where, $H_R(x,y)$ means that x is a senior role of y.

STEP2: Implicit Modality Conflict Detection

Arbitrary event $\mapsto E_1$

Subject Role Structure $\mapsto H_R(S_2, S_3), H_R(S_3, S_5), \cdots$

Auth+(S5,T2,A1) $\mapsto \forall x(E_x \rightarrow P(S_5, T_2, A_1))$

Auth-(S2,T2,A1) $\mapsto \forall x(E_x \rightarrow \neg P(S_2, T_2, A_1))$

Propagation Policy $\mapsto \forall x, y, z, a(P(x, y, a) \wedge H_R(z, x) \rightarrow P(z, y, a))$

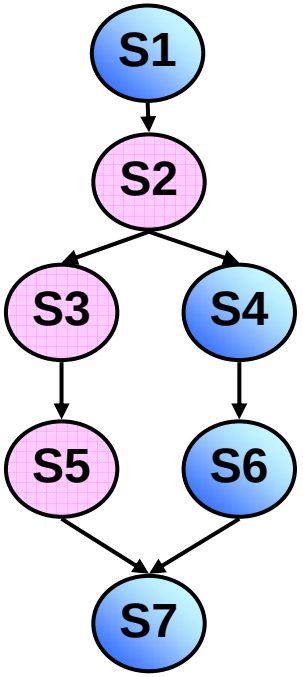
$$\frac{\neg P(x1, y1, a1) \vee H_R(z1, x1) \qquad P(z1, y1, a1)}{x1 = S_5 \quad z1 = S_3 \quad y1 = T_2 \quad a1 = A_1}$$

Abductive inference

$P(S_3, T_2, A_1)$

$$\frac{\neg P(x2, y2, a2) \vee H_R(z2, x2) \qquad P(S_2, T_2, A_1)}{x2 = S_3 \quad z2 = S_2 \quad y2 = T_2 \quad a2 = A_1}$$

Abductive inference



This result indicates that conflict is caused by the propagation {S2→S3→S5}

STEP1: Access Control Policy Formalization

- Action composition Policy

$$\zeta(A_1 = \Gamma(A_2, \cdots, A_n)) := \forall x, y(P(x, y, A_1) \leftrightarrow \Gamma(P(x, y, A_2), \cdots, P(x, y, A_n)))$$

–Examples:
A1=A2∧A3 is translated into
∀x,y (P(x,y,A1) ↔P(x,y,A2) ∧ P(x,y,A3)

- Chinese Wall Policy

$$\zeta(\mathrm{cw1}) := \forall x, y \neg(P(x, T_1, y) \wedge P(x, T_2, y))$$

- Separation of Duty Policy

$$\zeta(\mathrm{sod1}) := \forall x, y(\neg(P(x, y, A_1) \wedge P(x, y, A_2)))$$

STEP2: Constraint Conflict Detection

Arbitrary event $\mapsto E_1$

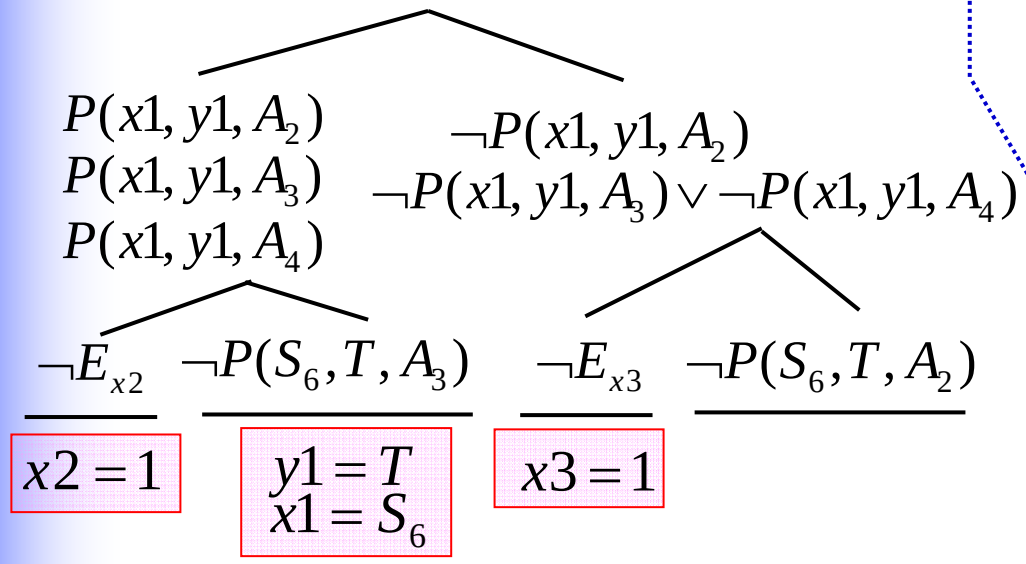
Auth+(S6,T,A2) $\mapsto \forall x(E_x \rightarrow P(S_6, T, A_2))$

Auth+(S6,T,A3) $\mapsto \forall x(E_x \rightarrow \neg P(S_6, T, A_3))$

Auth-(S6,T,A4) $\mapsto \forall x(E_x \rightarrow \neg P(S_6, T, A_4))$

A2=A3∧A4 $\mapsto$

$\forall x, y(P(x, y, A_2) \leftrightarrow P(x, y, A3) \wedge P(x, y, A_4))$



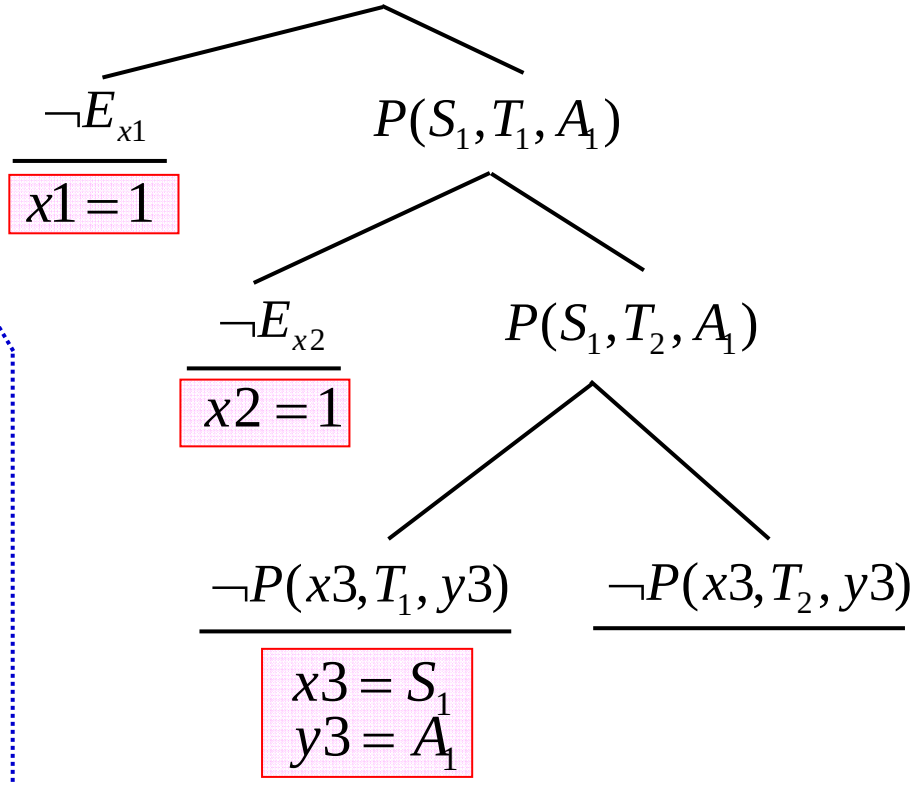
Conflict caused by an action composition policy is detected.

Arbitrary event $\mapsto E_1$

Auth+(S1,T1,A1) $\mapsto \forall x(E_x \rightarrow P(S_1, T_1, A_1))$

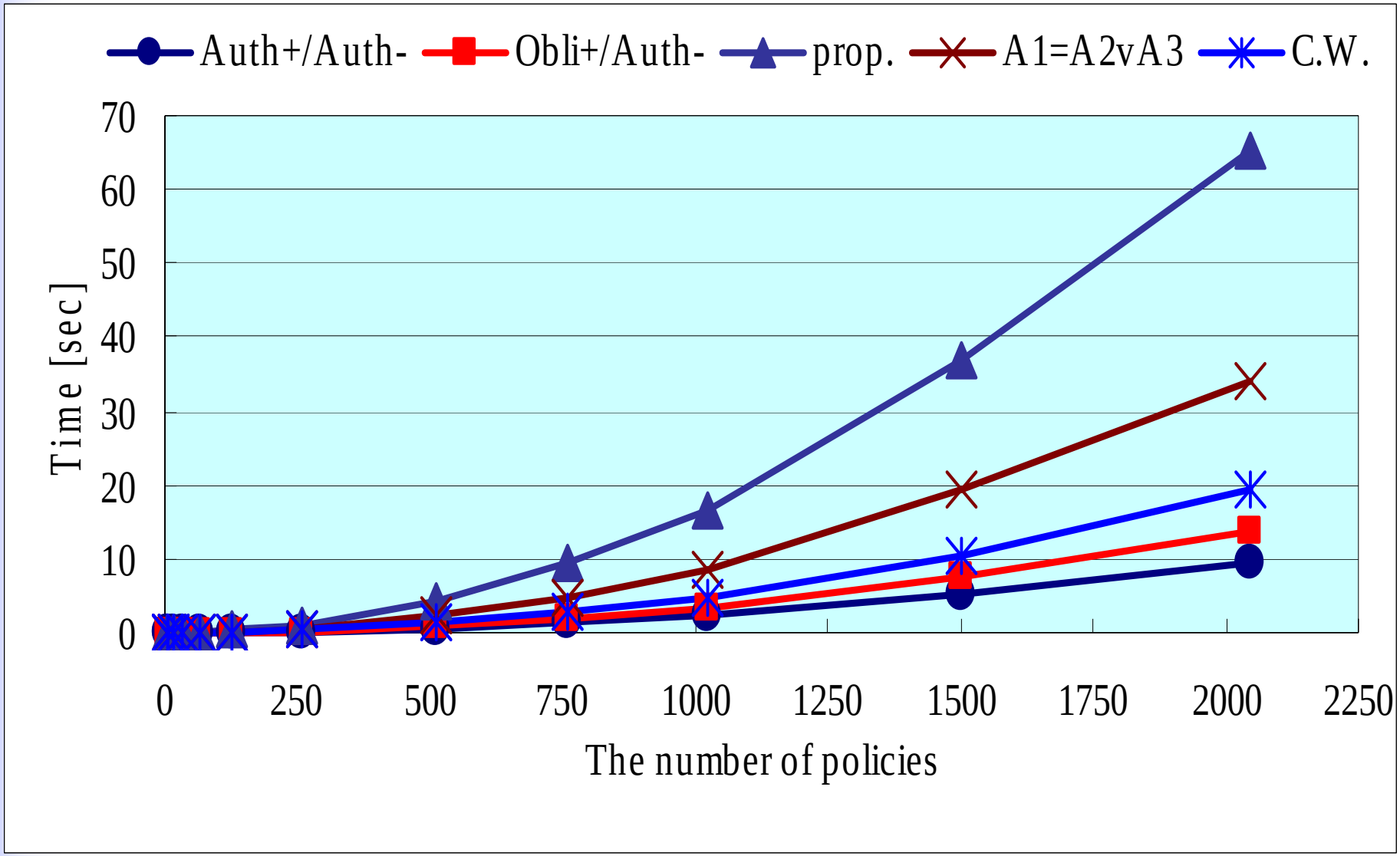
Auth+(S1,T2,A1) $\mapsto \forall x(E_x \rightarrow \neg P(S_1, T_2, A_1))$

cw(*,{T1,T2},*) $\mapsto \forall x, y \neg(P(x, T_1, y) \wedge P(x, T_2, y))$



Conflict caused by Chinese Wall policy is detected.

Computational Time for Conflict Detection



Conclusions and Future work

- **Conclusions**
  - We have presented an approach to statically detect a conflicting policy by using the free variable tableaux.
  - It is realized by translating each access control policy into first order logic.
  - The method can detect not only modality conflict but also constraint conflicts all in a uniform way.
  - Also it can provide helpful information to resolve the conflict by using abductive inference.
  - It has advantage that it can be applied to various policies written in different policy definition languages.

- **Future Work**
  - Extension
    - We will consider the formalization of more complex policies such as delegation policies.
  - Implementation
    - We will implement the approach and use it to develop a tool that detects conflicting policies written in such as XACML or Ponder.