Antecedent and Consequent Semantics

When applying a policy to a system instance:

1. If the antecedent applies:
2. check the consequent to see if the policy was upheld

For the following:

- ❏ let S be a system instance
- ❏ let P be a policy in effect on that system
- ❏ let $A_p(s)$ be true iff s satisfies the antecedent of p
- ❏ let $C_p(s)$ be true iff s satisfies the consequent of p

Antecedent and Consequent Semantics [2]

- ❏ A policy is relevant to a system instance if the antecedent is satisfied.
  - **relevant(P,S)**= $A_p(S)$
- ❏ A policy is upheld on a system instance if it is relevant to the instance and if its consequent is satisfied.
  - **upheld(P,S)**= relevant(P,S) $\wedge$ $C_P(S)$ = $A_P(S) \wedge C_P(S)$
- ❏ A policy is not violated if it either is not relevant or is upheld.
  - **no_violation(P,S)**= ¬relevant(P,S) $\vee$ upheld(P,S) = $\neg A_P(S) \vee (A_P(S) \wedge C_P(S)) = (\neg A_P(S) \vee A_p(S)) \wedge (\neg A_P(S) \vee C_P(S)) =$ $A_P(S) \Rightarrow C_P(S)$
- ❏ A policy is considered to be violated if it is relevant but its consequent is not satisfied.
  - **violation(P,S)**= relevant(P,S) $\wedge \neg C_P(S) = \neg(\neg A_P(S) \vee C_P(S)) =$ $\neg(A_P(S) \Rightarrow C_P(S)) =$ ¬no_violation(P,S)

Composing Policies

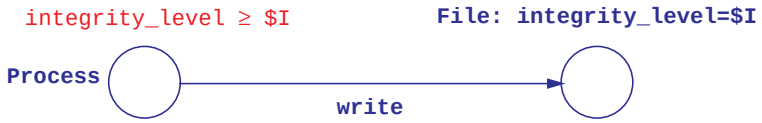
Composed policy:

- the policy consisting of the constraints enforced by two or more policies that are in effect
- semantics of policy composition:
  - a policy violation if and only if system instance violates any of the set of policies
- S is a system instance and P is a set of policies:
  - violation(P,S)= $\exists$ p $\in$ P: violation(p,S)
  - no_violation(P,S)= $\forall$ p $\in$ P: no_violation(p,S)

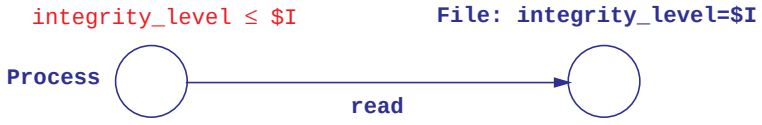
Biba Integrity Policy Constraint Graphs

Two constraints for Biba Integrity policy:

- only a process with higher or equal integrity level can write to a file



- only a process with lower or equal integrity level can read a file

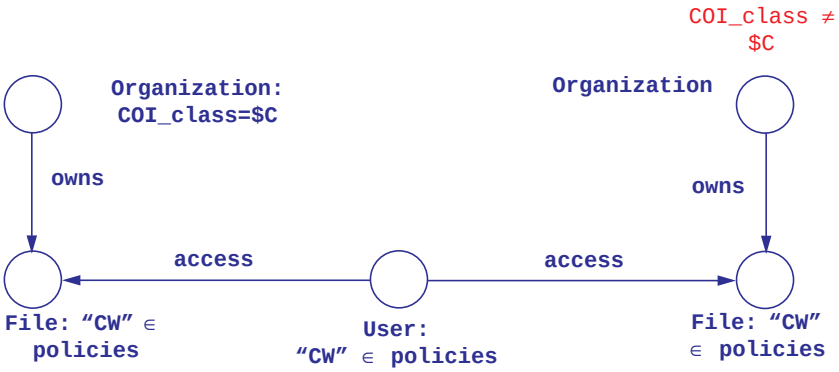


Blue lines and predicates are part of the antecedent

Red lines and predicates are part of the consequent

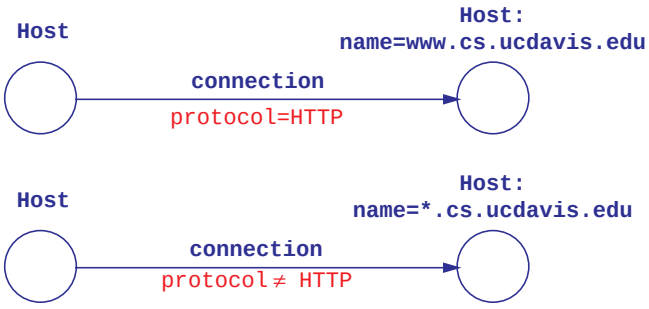
Chinese Wall A/C Constraint Graph

Chinese Wall: If a consultant has accessed protected data from two companies, then one company cannot be in the same conflict of interest class as the other.



Key: **antecedent is blue;** consequent is red

Contradicting Policy Example



Here the contradiction is because:

- ❏ the antecedent can apply at same time
- ❏ the consequents are opposing

The contradiction could be more subtle, i.e., if the second policy had consequent "transport_protocol=UDP", which implies "protocol ≠ HTTP", or only contradict part of the time

Advantages to this Approach

It is expressive:

- ❏ language is independent of the semantics of the entities and relationships
  - nodes are independent of the specific entity
  - edges can represent any relationship

It is formal:

- ❏ can reason about policies expressed in the language
- ❏ can enforce all policies in the same way

It is separate from the system model