Application-Oriented Policies and their Composition

Virgil D. Gligor and Serban I. Gavrila

April 16, 1998

- **Systems and Applications**
- **Property Types; Dependencies**
- **Policy Structure**
- **Policy Composition**

Systems

- *state machine*
  STATES, SUBJECTS, USERS, OPERATIONS, OBJECTS

- *state transitions*
  - commands: $op(s_1, S, obj, s_2)$
  - command sequence: $op_1(s_0, S_1, obj, s_1)op_2(s_1, S_2, obj_2, s_2)\ldots$,
  - tranquil commands: do not alter security attributes

- *system:* a set of command sequences with start states s_0 in STATES_0.

- *secure state, commands*: those that satisfy properties

- *reachable state*: a state appearing in a command sequence of a system

- *secure system*: all state transitions and reachable states are secure

- Ω : set of all command sequences of a secure system

Applications and Executability

- application: App = [ObjSet, OpSet, Plan]

  - **plan: a finite set of pairs $\{(obj_i, op_i)\}$**
  - ordered plan: an ordered set of pairs $\{(obj_i, op_i)\}$
  - plans with "operation bracketing" (e.g., least-privilege princ.)

    . . .

- $App_1 \cup App_2 =$

  $[ObjSet_1 \cup ObjSet_2,\ OpSet_1 \cup OpSet_2,\ Plan_1 \cup Plan_2]$

- command sequence σ *executes* App if for any pair (obj_i, op_i) in Plan there is a command $op_i(s_k, S, obj_i, s_{k+1})$ in σ

Property Types

P = Attribute (AT) properties $\wedge$

Access Management (AM) properties $\wedge$

Access Authorization (AA) properties

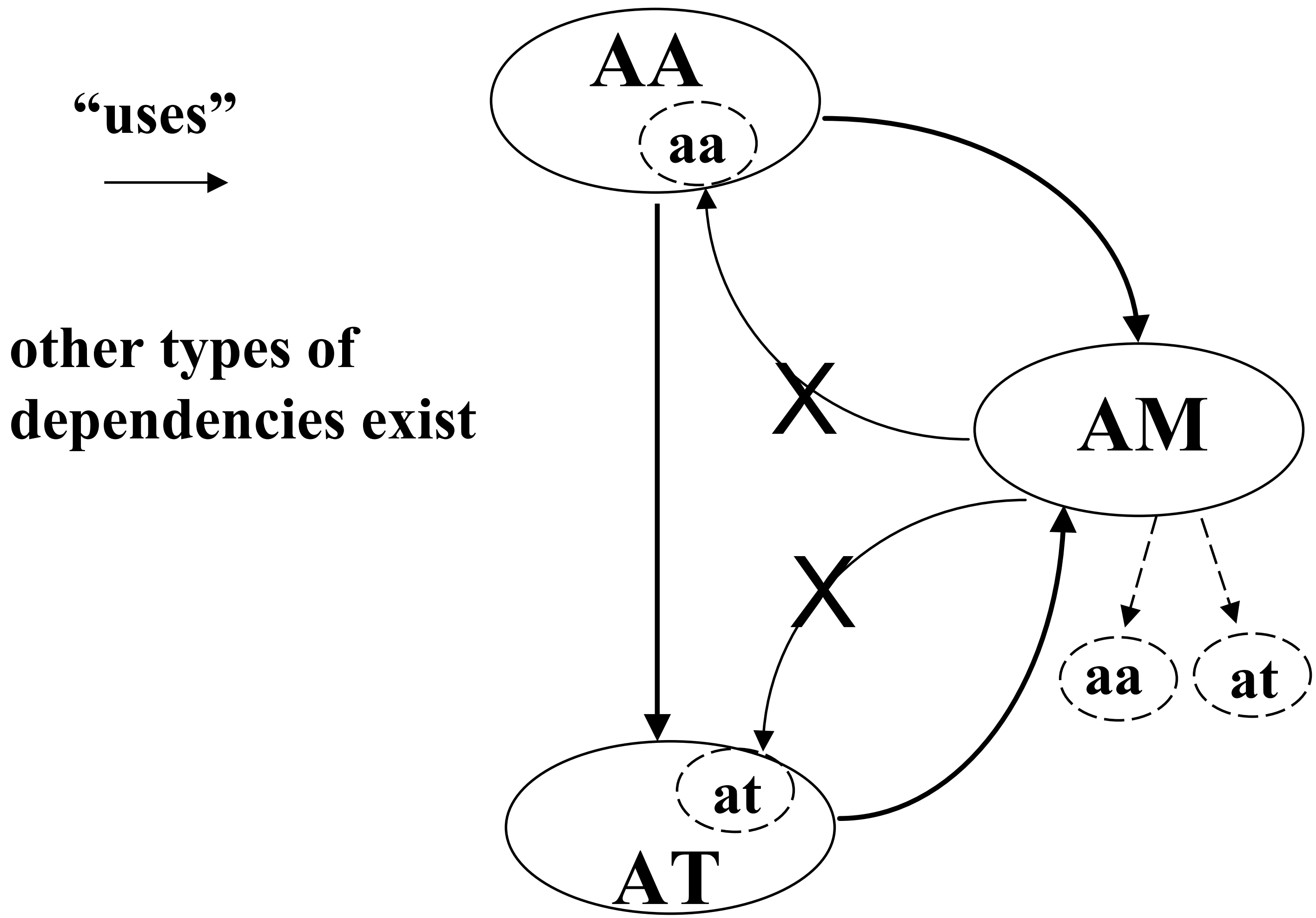
Examples of Property Types

- Attribute (**AT**) Properties
  - security (integrity) levels, partial order, lattice property
  - roles, hierarchy, permissions, membership, inheritance

- Access Management (**AM**) Properties
  - distribution, review, revocation of permissions
    - selectivity, transitivity, independence ...
  - object / subject creation and destruction
  - object encapsulation

- Access Authorization (**AA**) Properties
  - required subject and object attributes for access
    - BLP, Biba, RBAC, UNIX ...

Property Dependencies



Individual policy properties cannot be composed independently

Policy Structure

$$P = P \wedge \text{Admin}(P) \wedge \text{Compat}(P, App)$$

- access management
- access authorization
- attribute properties

Safety Properties

Safety or Liveness Properties ?

Admin(P)

P: a set of tranquil command sequences with the start state in $STATES_0$

for all

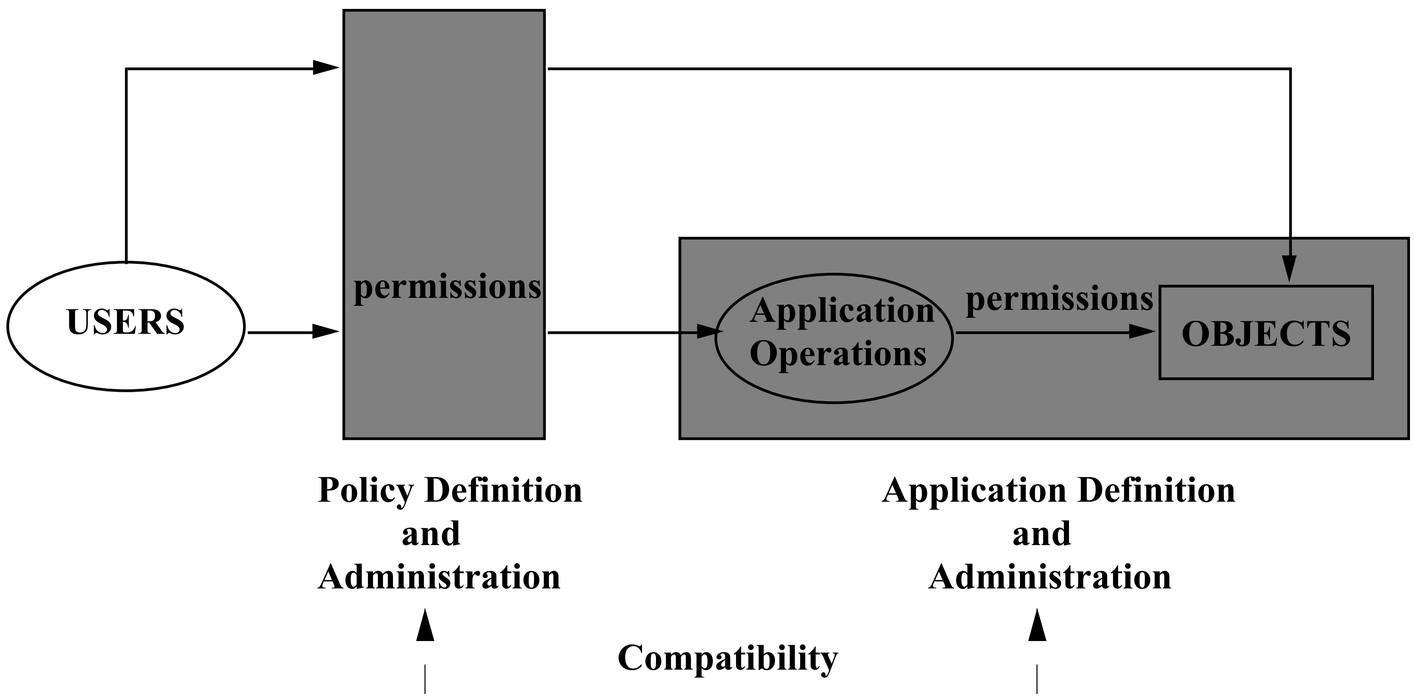
Admin(P) = “for each s in STATES, *there exists* $s_0 \in STATES_0$,
there exists $\omega \in \Omega$ such that: ω starts in s, and
ω reaches s_0 and
s_0* is in P”

Compat(P, App)

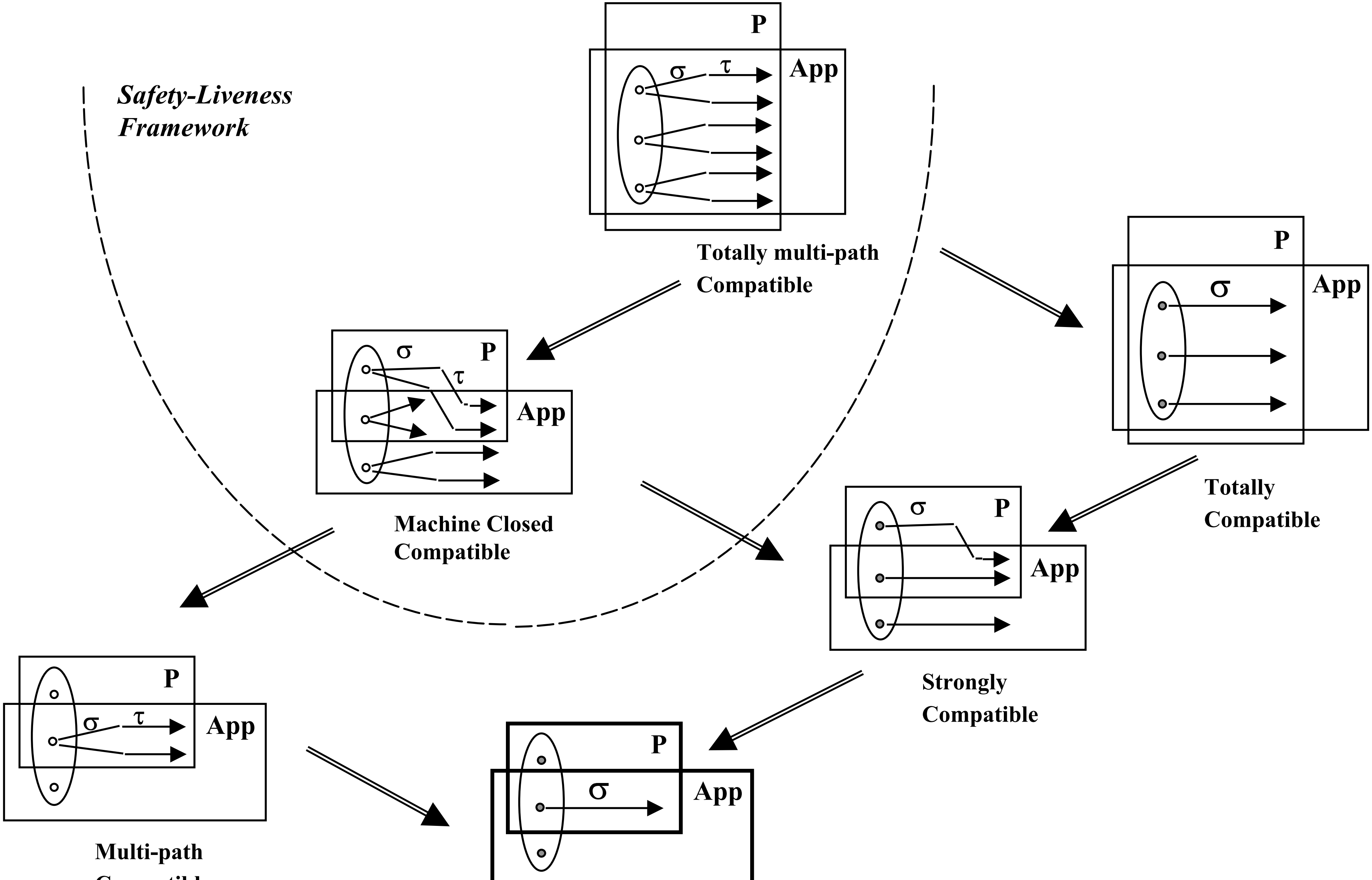
Compat(P) = “there exists $s_0 \in STATES_0$ and $\sigma \in P$ starting in s_0
such that σ executes App”

.... neither Safety nor Liveness ....

Mandated Compatibility



Types of Compatibility



Compat(P, App)

Totally Multi-path Compatible

For each start state s_0 there is a comand sequence σ in P starting in s_0 , and for each finite command sequence σ in P there is a command sequence τ such that $\sigma\tau$ is in P and executes *App*.

Machine-Closed Compatible

For each finite command sequence σ in P there is a command sequence τ such that $\sigma\tau$ is in P and executes *App*.

Multi-path Compatible

There is a start state s_0 such that for each finite command sequence σ in P starting in s_0 there is τ such that $\sigma\tau$ is in P and executes *App*.

Totally Compatible

For each start state s_0 there is a command sequence σ in P starting in s_0 such that σ executes *App*.

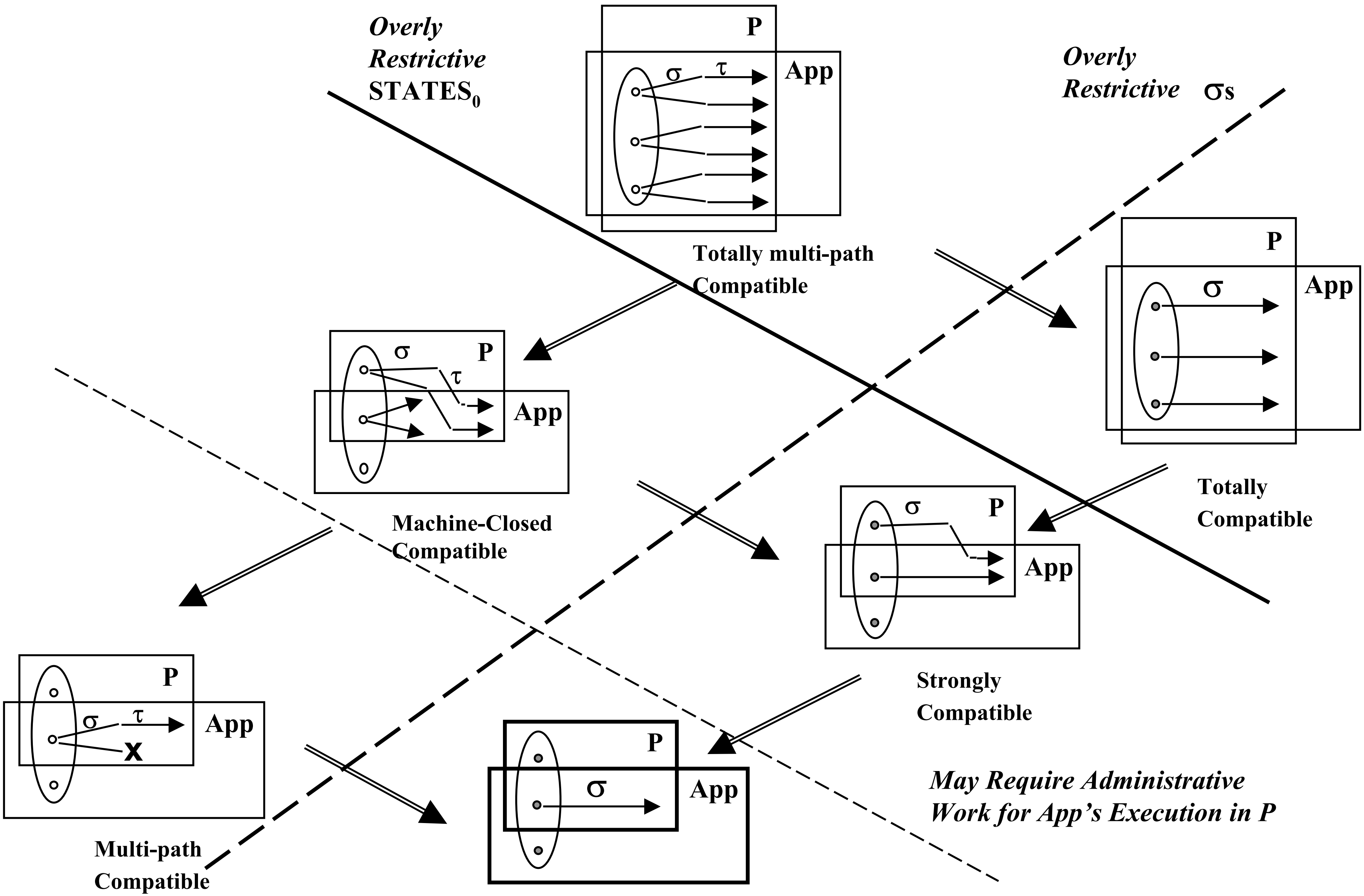
Strongly Compatible

For each start state s_0 such that s_0* is in P, there is a command sequence σ in P starting in s_0 that executes *App*.

Compatible

There is a start state s_0 and a command sequence σ in P starting in s_0 that executes *App*.

Types of Compatibility



Compat(P, App)

Overly Restrictive σ_s

Example:

$App = [\ \{obj\}, \{op_1, op_2\}, plan];\ \ plan = \{(obj, op_1), (obj, op_2)\}$

P : "u_1 and u_2 are the only users who may execute App *and*
a user may not execute two distinct operations on the same object"

Compat(P, App) is true

$$\sigma = \quad s_0 \xrightarrow{S_1:(op_1,\ obj)} s_1 \xrightarrow{S_2:(op_2,\ obj)} s_2$$

u_1: (op_1: obj), S_1 = subject

u_2: (op_1, op_2 : obj), S_2, S_2\`= subjects

Compat_M(P, App) is false

$$\sigma` = \quad s_0 \xrightarrow{S_2`:(op_1,\ obj)} s_1` \qquad \tau = \ S_2`:(op_2,\ obj) \dashrightarrow \text{X}$$

$$s_1` \dashrightarrow \text{X} \qquad S_1:(op_2,\ obj)$$

u_1: (op_1: obj), S_1

u_2: (op_1, op_2 : obj), S_2, S_2'

Policy Composition

$$\mathcal{P}_1 = P_1 \wedge \text{Admin}(P_1) \wedge \text{Compat}(P_1, \text{App}_1)$$

$$\mathcal{P}_2 = P_2 \wedge \text{Admin}(P_2) \wedge \text{Compat}(P_2, \text{App}_2)$$

Let CS($\mathcal{P}_1$), CS($\mathcal{P}_2$) denote sets of command sequences

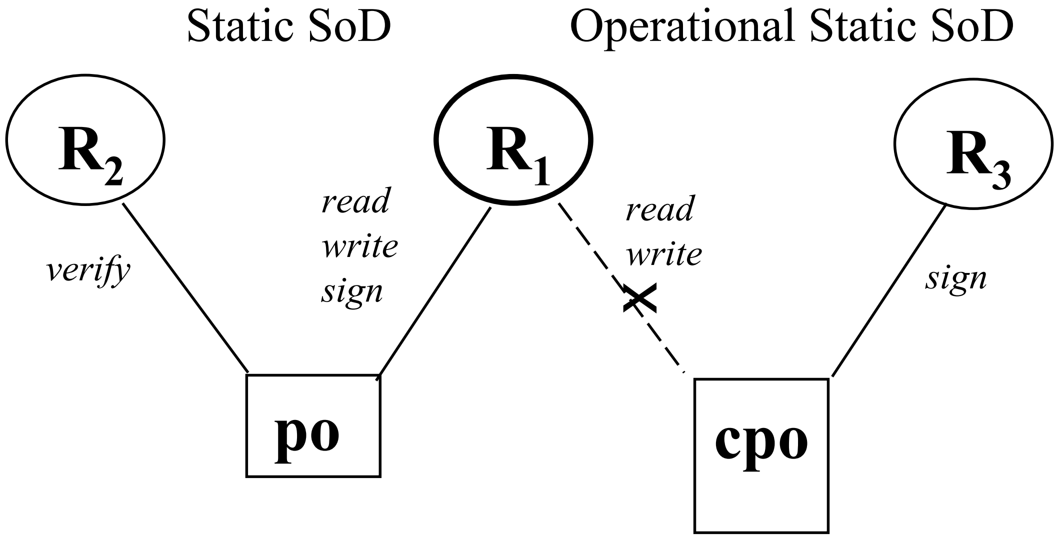
$\mathcal{P}_1$, $\mathcal{P}_2$ are composable if and only if

CS($\mathcal{P}_1 \cap \mathcal{P}_2$) $\neq \phi$ whenever CS($\mathcal{P}_1$), CS($\mathcal{P}_2$) $\neq \phi$

Emerging policy **$\mathcal{P}_1 \circ \mathcal{P}_2$ =**

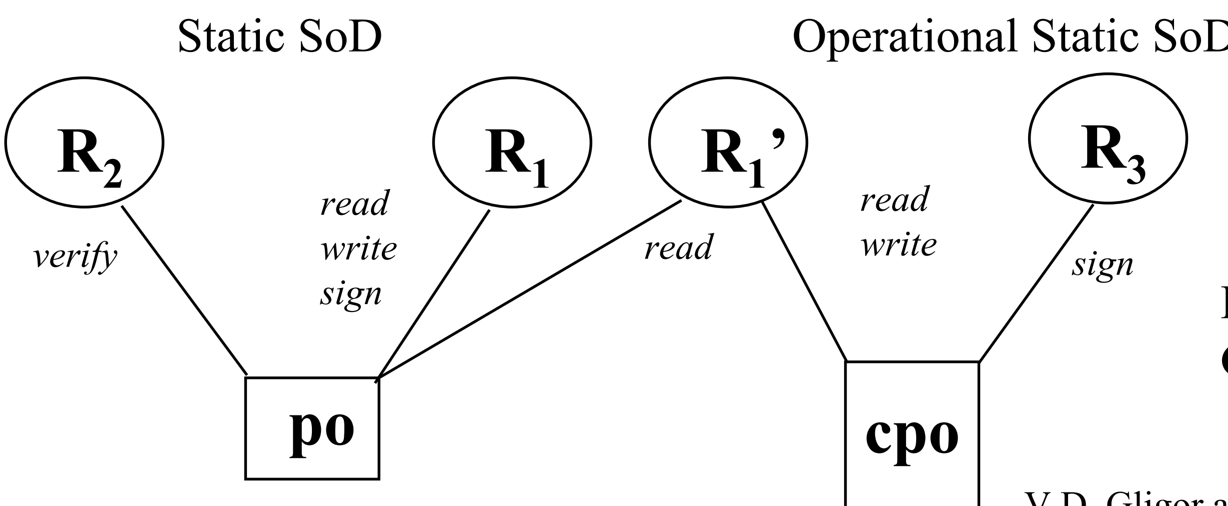
$$= P_1 \wedge P_2 \wedge \text{Admin}(P_1 \wedge P_2) \wedge \text{Compat}(P_1 \wedge P_2, \text{App}_1 \cup \text{App}_2)$$

Example: Non-Composable Separation-of-Duty Policies



Purchasing Staff Department

Purchasing Staff Central Administration



Policy-Management Change