Assessing the Risk of an Information Infrastructure through Security Dependencies

F.Baiardi [1], S.Suin [1], C.Telmon [1], M.Pioli [2]

(1) Dip. di Informatica, Università di Pisa

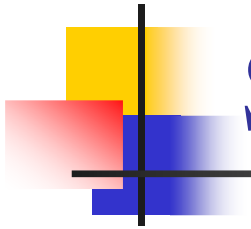
(2) Enel Distribuzione, Roma

Our long term goals

- A formal methodology to integrate the various steps of a risk assessment
  - Each step may exploit alternative strategies
  - A set of programming tools to support both the various steps and their integration
- Ranking of countermeasures to define
  - cost effective risk mitigation plans
  - return of investment in countermeasures

Key elements of the solution

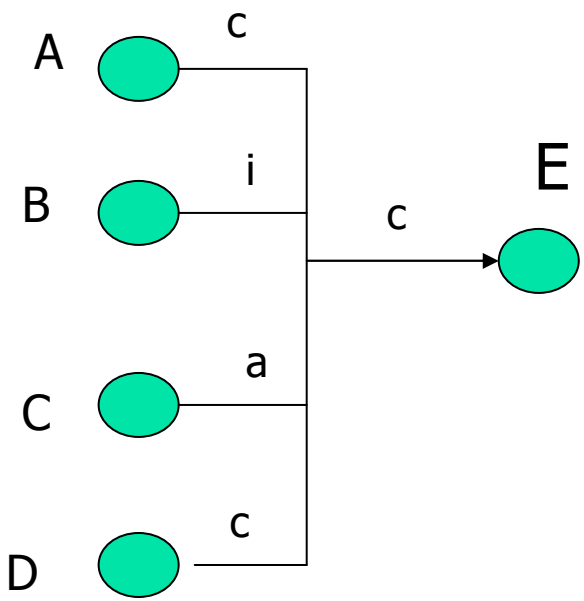
1. Modular description of the information infrastructure
2. Security attributes of components (among others)
   a) confidentiality
   b) integrity
   c) availability
3. Dependencies among attributes of distinct components that are related to component attributes
3. Risk mitigation plan as a sequence of sets of countermeasure
4. Logic programming to define programming tools (backtracking because of NP complete problems)



Standard approaches to …

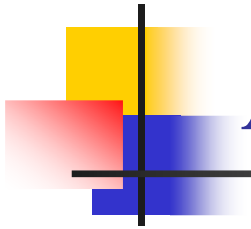
- Vulnerabilities
- Attacks: requires some resources
- Threats/Users: the attack it can attempt depends upon resources it can access
- Goals (intelligent attacks)
- Impacts
- Countermeasures

Dependency = Hyperarc



Anyone that controls
- A confidentiality
- B integrity
- C availability
- D confidentiality

also controls E confidentiality



Attribute Control

- Confidentiality control

  read access to the inner state of the component

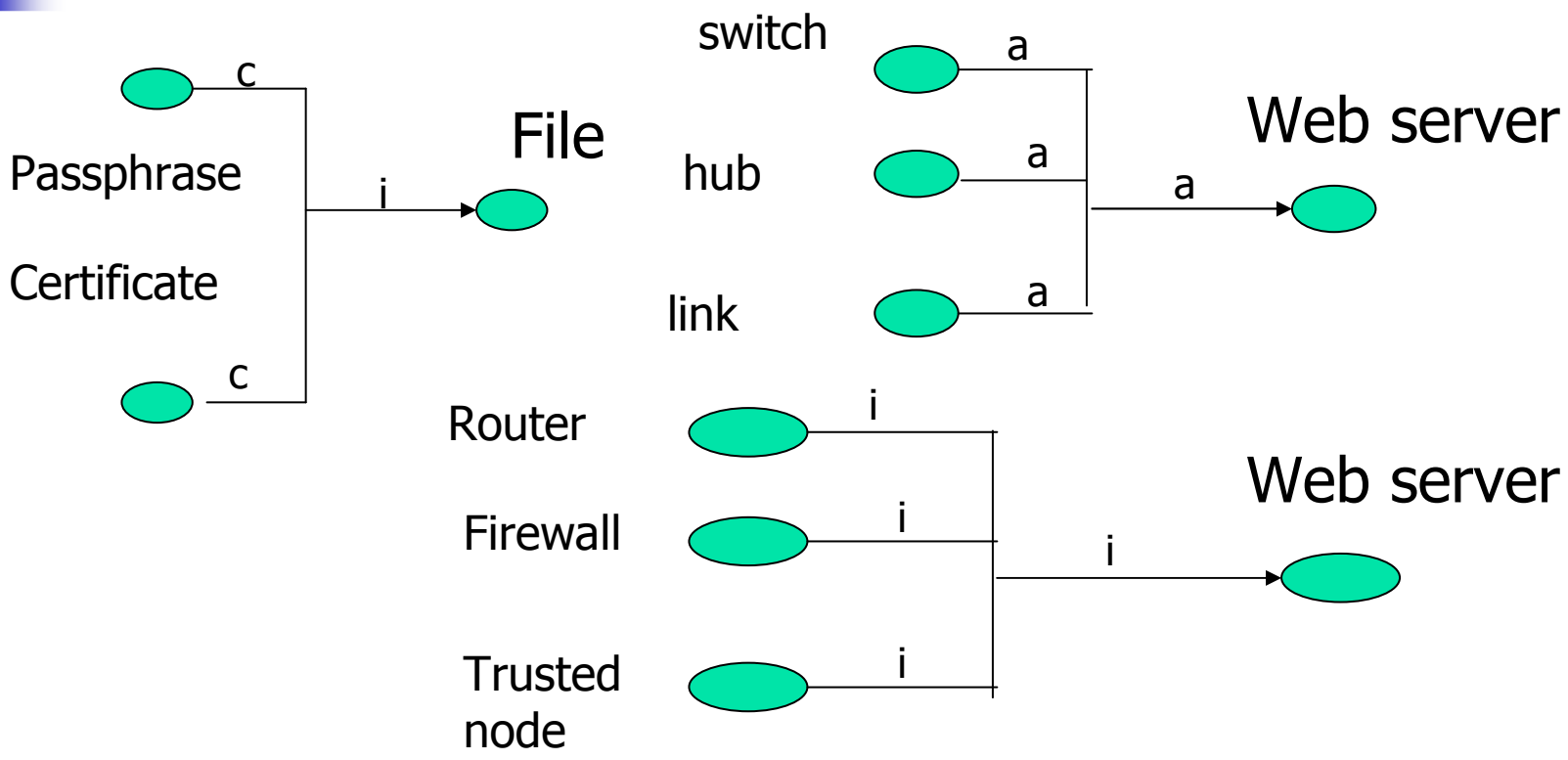
- Integrity control =

  write access to the inner state of the component

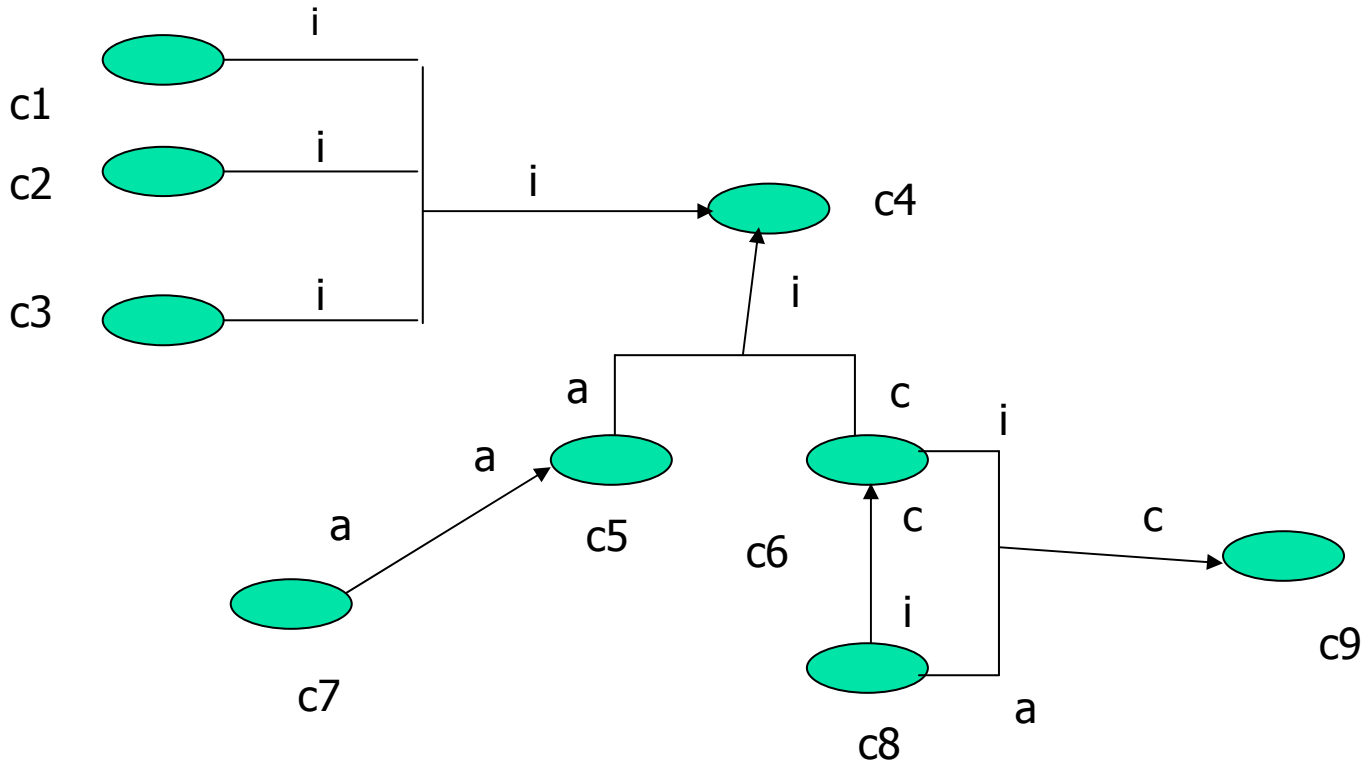
- Availability control

  who can invoke the component operations =
  who can control the component confidentiality and availability (fundamental step to model DOS)

Some trivial examples

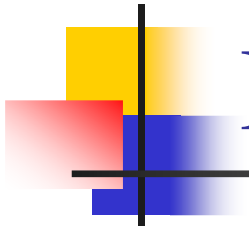


Hypergraph



Rights, Attacks and Goals

- User rights are expressed as set of pairs <component, attribute> = extended right (exr)
- For each attack *At* we determine
  - Resources *At* requires
  - Precondition = exrs to implement *At*
  - Postcondition = exrs gained if *At* is successful
- Each threat goal can be expressed as a set of exrs
- For each goal and a threat, an impact can be computed



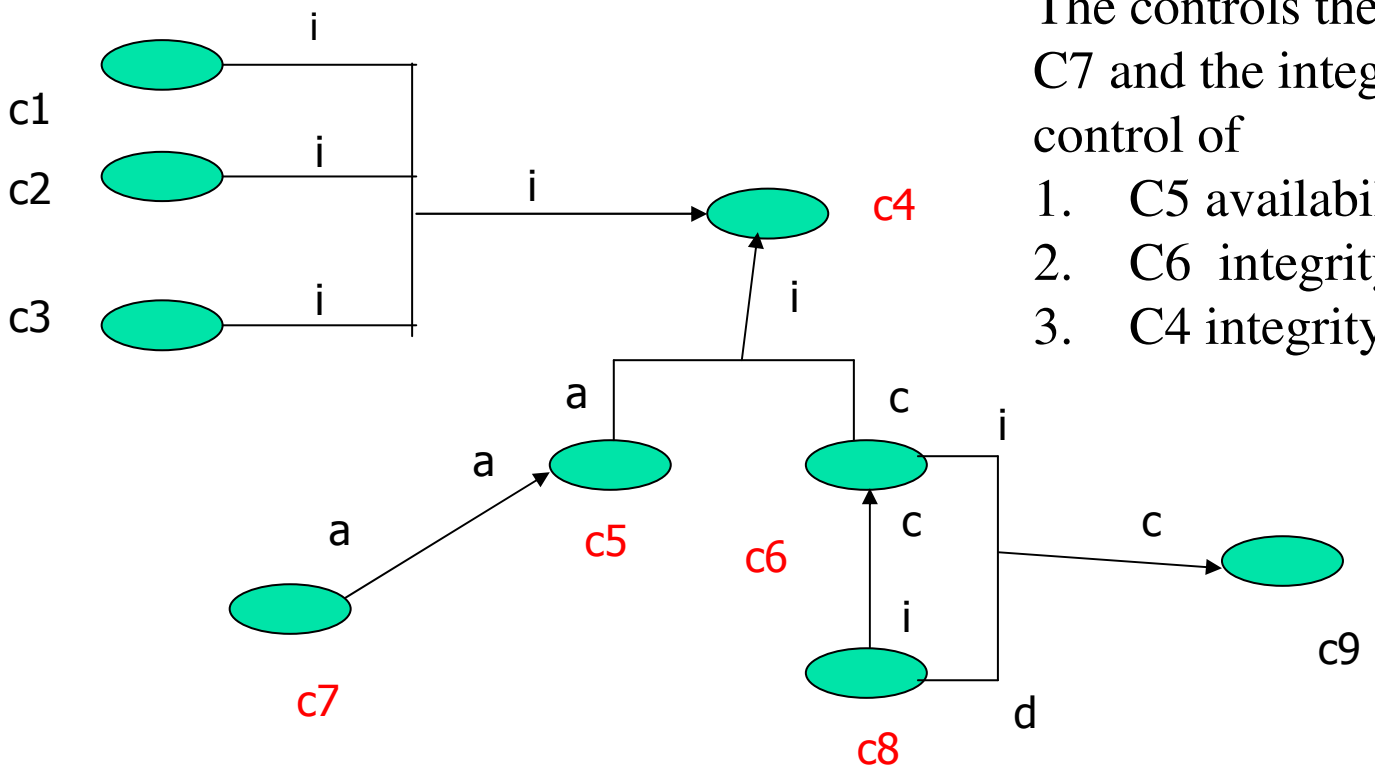
Detail vs abstraction level

- The model support the description of the infrastructure at distinct abstraction and detail levels
- To increase the detail level, at a any abstraction level, component operations may be introduced
  - Extended rights are expressed in terms of operations
  - The component attributes a user control depends upon the operations a user can invoke

Transitive closure

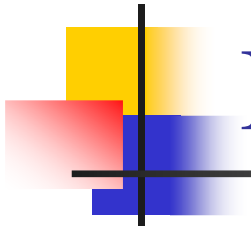
- After any attack can compute
  - the closure of the rights =
  - reachable hypergraph nodes
- The closure propagates the exr the threat owns after the attacks to discover all the exrs that have been acquired because of dependencies
- If the transitive closure does not include one of its goals, the threat will attempt other attacks till a goal has been achieved

Transitive closure



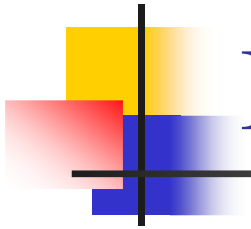
The controls the availability of C7 and the integrity of C8 imply the control of

1. C5 availability,
2. C6 integrity
3. C4 integrity



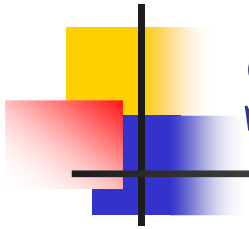
Dependencies vs Attack Tree

- An hypergraph describes both And and Or decompositions of complex attacks
- Automatic and formal discovery of And-Or attack trees by an analysis of
  - Structural dependencies among infrastructure components
  - Component vulnerabilities and the attacks they enable
  - Pre and Post conditions of each attack



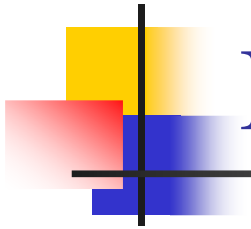
Infrastructure evolution

- An infrastructure state includes a set of users and a set of exrs for each user
- An evolution due to user U is a sequence of states
  - starting in the one where only legal exrs are ownend
  - ending in one where U has achieved one of its goals
- Any transition is due to a useful attack enabled by a component vulnerability
- Evolutions can be described through attack automata or paths of an attack graph



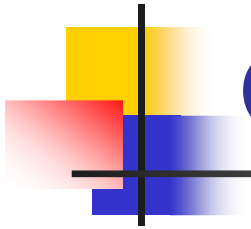
Some definitions

- Two evolutions are
  - equivalent if they enable a threat to reach the same goal
  - disjoint if they exploit distinct attacks
- Concurrent evolutions are due to non interfering attacks of distinct users
- Collusion evolution where cooperating users can grant exrs to each other
- Competitive evolution where some users try to revoke the exr of other users
- Monotone evolutions= Concurrent + Collusion



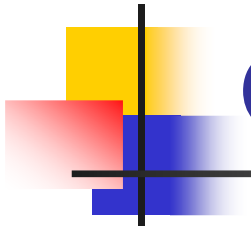
Evolution vs Planning

- In the case of intelligent threats, the notion is strongly related to that of goal oriented planning
- A large amount of know how from other fields can be exploited
- The main difference is due to the fact that security require the discovery of all evolutions (= all plannings) while in other fields the discovery of one optimal or nearly optimal plan sufficies



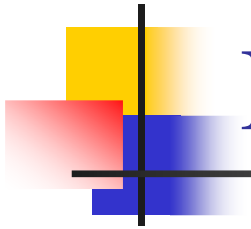
Countermeasures

- A countermeasure for an attack *At* stops *At* by any combination of the followings
  - remove one of the vulnerabilities that enable *At*
  - update dependencies to prevent threats that execute *At* from achieving some exrs
  - update the initial rights of some users
  - increase the resources that *At* requires so that some threat cannot implement it
- A countermeasure stops an evolutions if it stops one of the evolution attacks



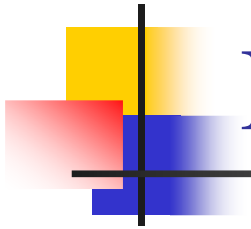
Complete and minimal sets

- A set of countermeasures is complete if it stops any evolution
- A complete set is minimal if no of its subsets is complete
- Computation of minimal sets is an NP complete problem
- We can consider the cost of countermeasures in a set rather than their number



Ranking of vulnerabilities

- Any minimal set represent a distinct way to stop any infrastructure evolution
- Some attacks may be successful but no threat will be able to achieve its goal
- A ranking of vulnerabilities can be defined according to the percentage of minimal sets that include a countermeasure that remove a vulnerability



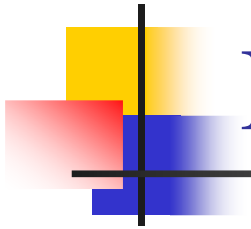
Ranking of countermeasures

The previous ranking strategy is useful

- to decide whether some vulnerabilities can be accepted because of cost effectiveness
- to evaluate a newly discovered vulnerability

Non redundant subsets

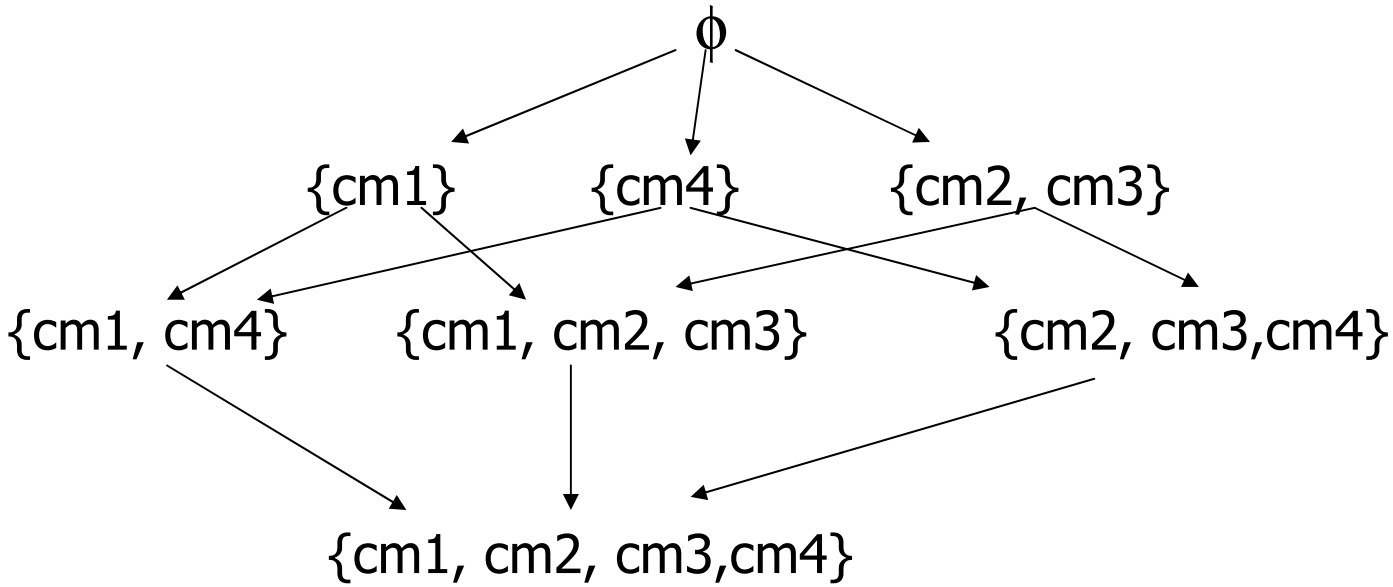
- A subset S of a minimal set is non redundant if
  - no subset of S stops the same evolutions
  - if S stops an evolution, then it stops also any equivalent one
- Only non redundant sets are cost effective because otherwise there is a smaller (and cheaper) subset that has the same benefit



Risk mitigation plans

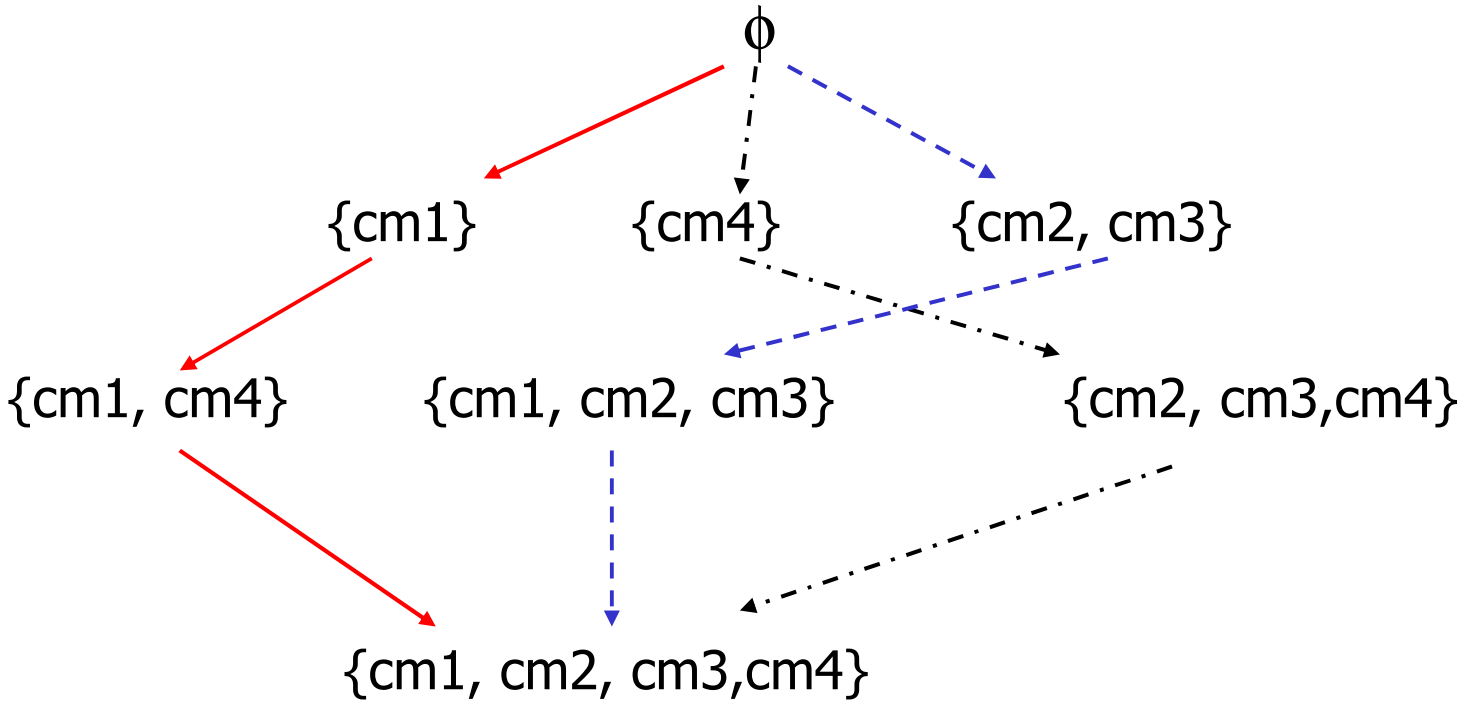
- Cost effective risk mitigation plan implements one of the minimal set of countermeasures
- The implementation may be distributed in time because it may be too expensive to implement all the countermeasures simultaneously
- Non redundant subsets of the minimal sets define alternative schedulings
  - we order the non redundant subsets of the minimal sets
  - each chain of the order defines a scheduling of one plan
  - alternative plans may have intersections

A first set of risk mitigation plans

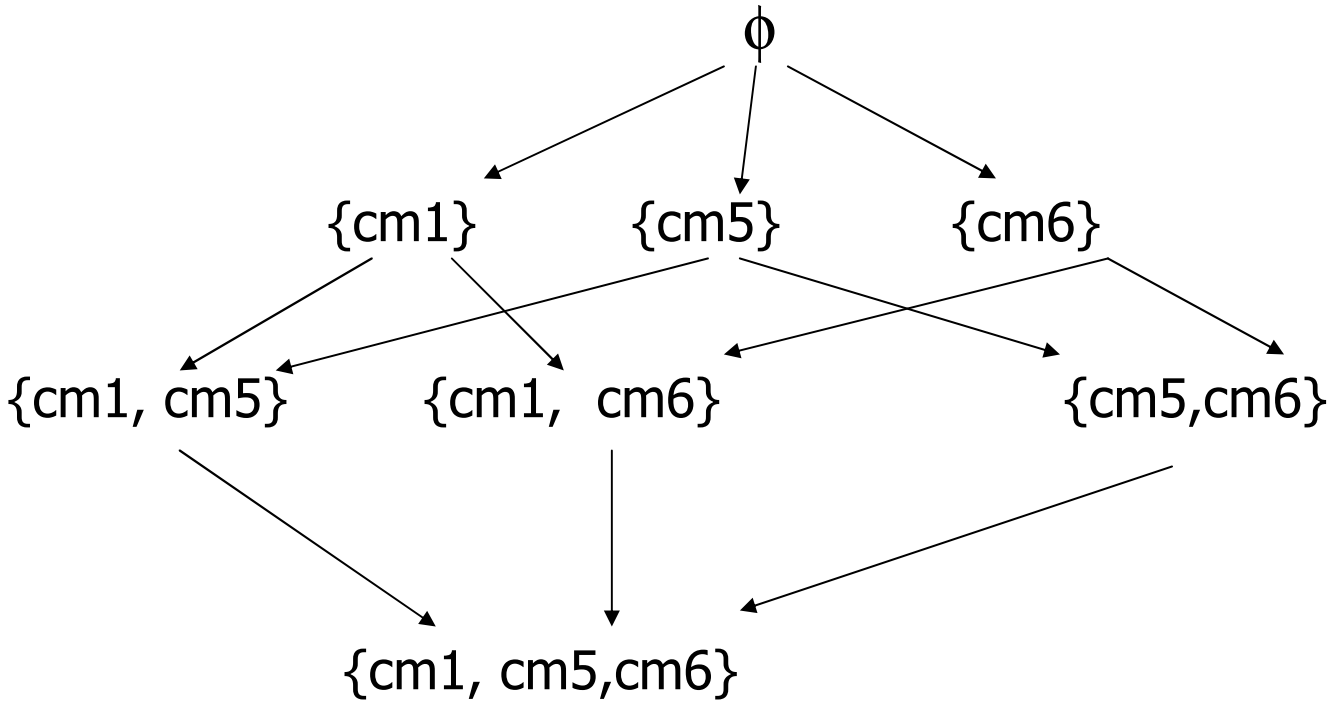


Implementation of the set {cm1, cm2, cm3,cm4} if the non redundant subsets are {cm1}, {cm4}, {cm2, cm3}
⇔ two equivalent evolutions are stopped by {cm2, cm3}

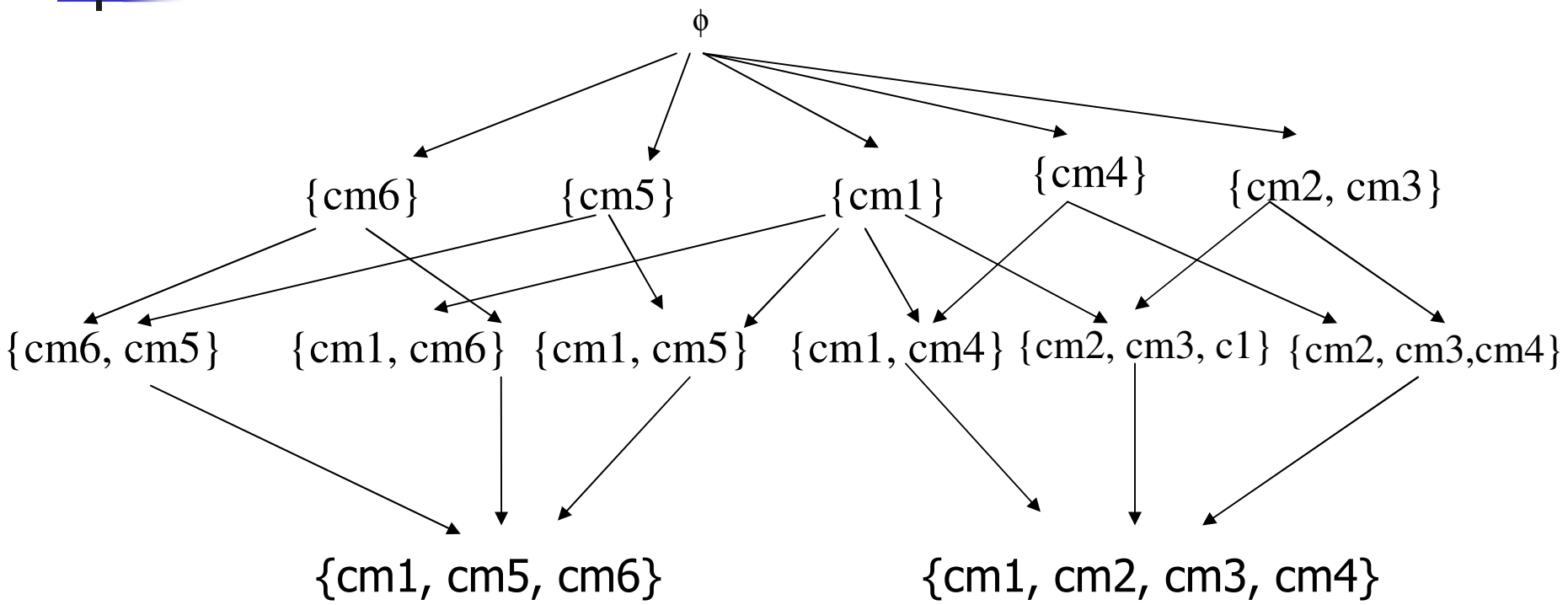
One Plan: Alternative schedulings

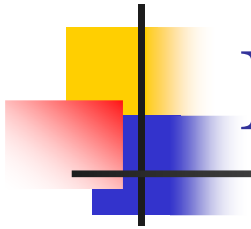


Another minimal set



Alternative plans = Global ordering

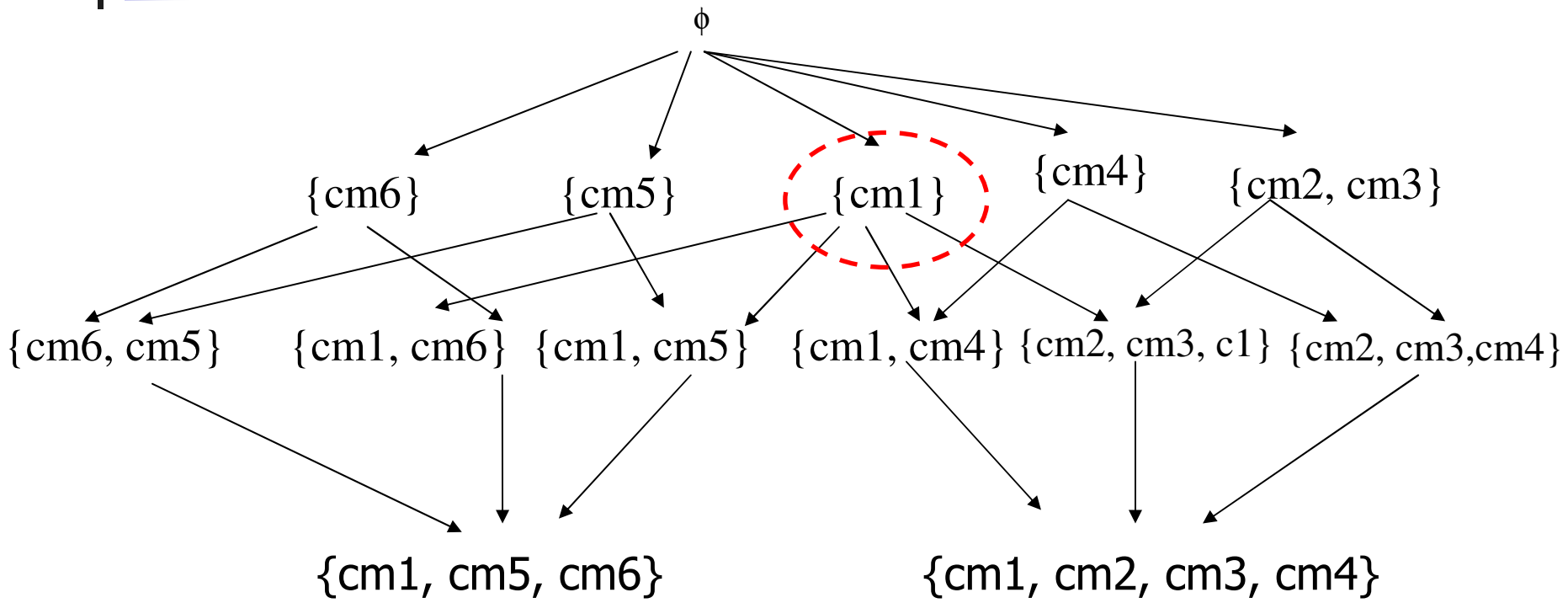




Least Commitment

- We can choose to implement a non redundant set belonging to distinct minimal sets and hence to distinct plans
- This delays the choice of the risk mitigation plan till information about investment on countermeasures is available

Global ordering = Alternative plans

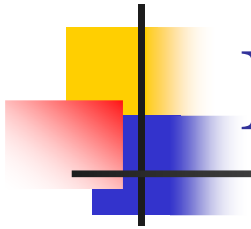


Taking risk (probability) into account

When computing minimal sets we neglect evolutions

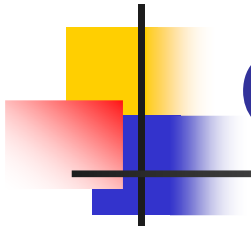
1. with an impact lower that *I*
2. requiring more than *N* attacks
3. with a complexity larger than *OC*
4. where an attack with a complexity larger than *C*

- Historical informaton about attacks is fundamental to determine thresholds such as *I*, *N*, *OC* and *C*
- Threat modelling to determine feasible attacks



Programming tools

- In the logic framework we have several sets of axioms , each set defines one of the following aspects
  - Vulnerability
  - Attacks
  - Dependencies
  - Initial user/threat rights
- Modular approach to the analyis
- The computation of the transitive closure is implemented as a deduction in the corresponding theory
- Backtracking is exploited to compute all evolutions



Current and Future works

- Extensive experimentations with
  - real world infrastructures
  - alternative implementations of programming tools
- Introduction of component types and the corresponding constrains on dependencies
- Non monotone evolutions