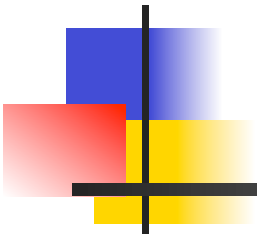


A Formal Approach to Risk Management



F. Baiardi

(1) Dip. di Informatica, Università di Pisa

(2) Polo G. Marconi, La Spezia

f.baiardi@unipi.it

www.di.unipi.it/~baiardi





Outline

- Long term goal: methodology and programming tools for a model based RA
- Basic Concepts
 - Dependencies and hypergraph
 - Security Status
 - Evolutions
 - Countermeasures
- The introduction of risk (probability +damage) reduces cases to be considered
- Alternative Risk Avoidance or Mitigation Plans
- Modular Approach to the Assessment = Hierarchical Decomposition of the model to reduce the complexity of the RA





Credits

- C.Telmon
- D.Sgandurra
- M.Pioli & F.Ceccarelli
- S.Suin
- Students in La Spezia & Pisa





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
 - a cost effective risk avoidance or mitigation plan
 - return of investment in countermeasures
- Focus on critical information infrastructures = ICT component of critical infrastructures
- Sometime not enough data is available (even Laplace had the same problem 😊)



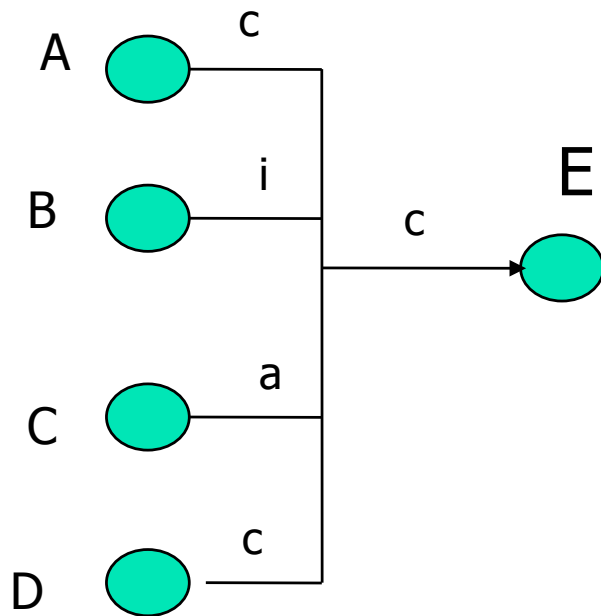


Key elements of the approach - I

1. A modular model of the information infrastructure
2. Security attributes of components (others are possible)
 - a) confidentiality
 - b) integrity
 - c) availability
3. Dependencies among attributes of distinct components
4. Each dependency is related an attribute of a component
5. Risk mitigation plans as sequences of sets of countermeasures
6. Rewriting systems or logic programming to define programming tools (backtracking to explore the solution space)



Dependency = Hyperarc



Anyone that controls

- A confidentiality
- B integrity
- C availability
- D confidentiality

also controls E confidentiality

A, B, C, D, E = infrastructure components (modules)

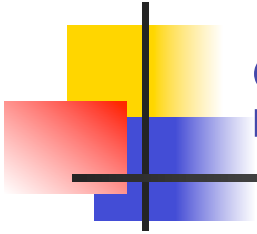




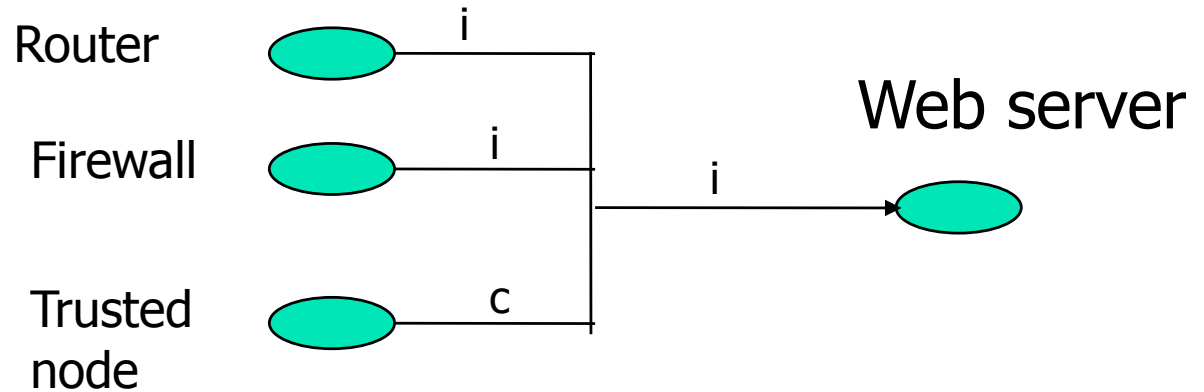
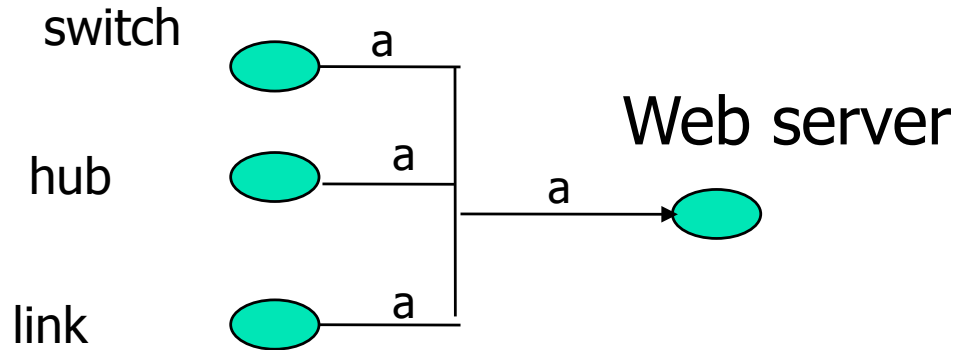
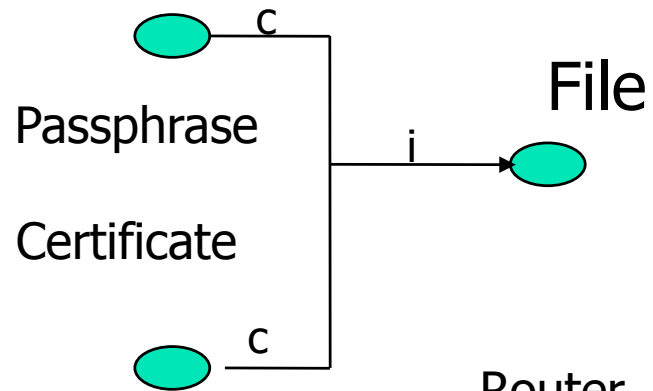
Attribute Control

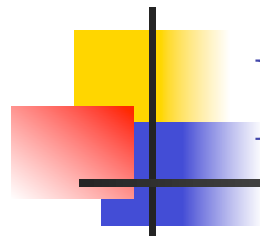
- Confidentiality control
 - read access to the inner state of the component
- Integrity control =
 - write access to the inner state of the component
- Availability control
 - who can invoke the component operations =
 - who can control the component confidentiality and availability (fundamental step to model DOS)





Some trivial examples

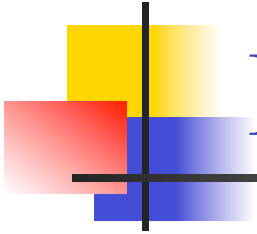




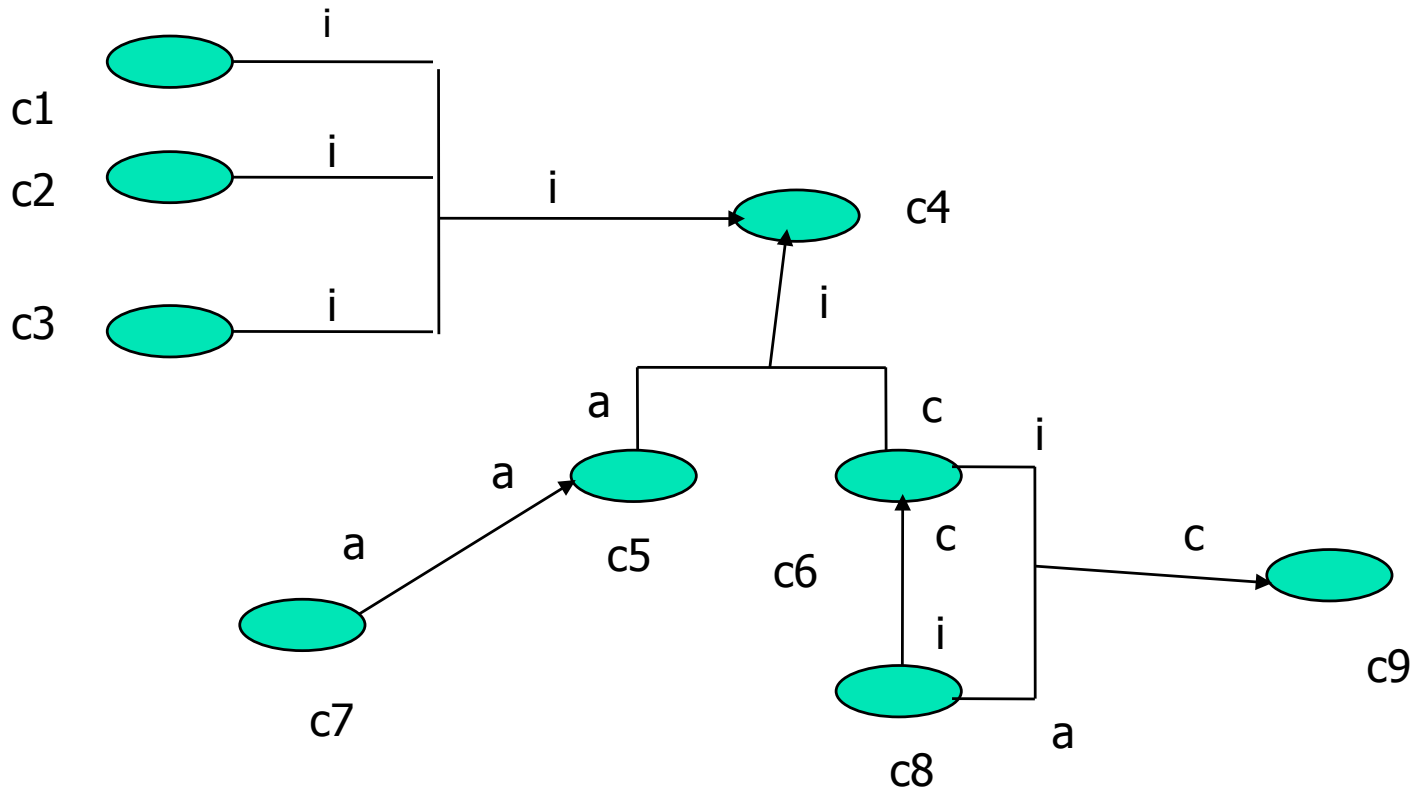
Model Based RA

- The model that RA considers is the hypergraph, infrastructure hypergraph, describing the dependencies among infrastructure components
- The hypergraph may describe also components that do not belong to the ICT infrastructure
- Several infrastructures can be represented by the same model



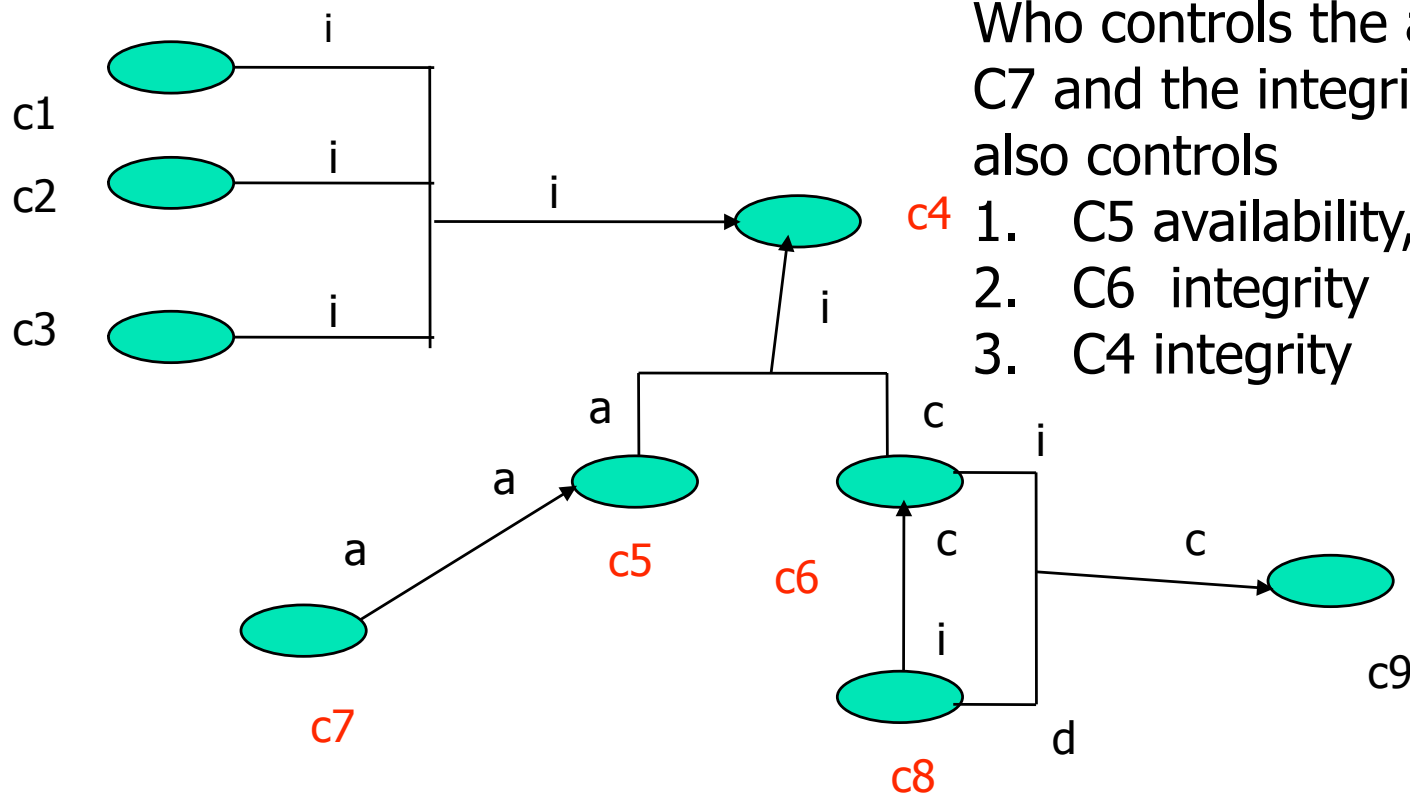


Infrastructure Hypergraph ☺





Transitive closure



Who controls the availability of C7 and the integrity of C8 also controls

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





Alternative equivalent representations

- A node for each attribute of each component
- Introduce a set of operations/methods rather than attributes and an hypergraph where
 - Each node represents a distinct method
 - Each node represents a component and hypergraph are labelled by methods involved in a dependency
- Use bigraphs to represent components and attributes or components and methods
- Rewrite rules (more later)





Key elements of the solution - II

- Security status =
 $\langle \langle \text{user}_1, \text{rights}_1 \rangle, \dots, \langle \text{user}_n, \text{rights}_n \rangle \rangle$
- Right = $\langle \text{component}, \text{attribute} \rangle$
- A status does not change if we consider
 - the transitive closure of each user rights
 - the closure is a function of the infrastructure hypergraph





Standard approaches to discover...

- Vulnerabilities
- Attacks
- Threats
- Goals (intelligent attacks)
- Impacts
- Countermeasures





Rights, Attacks and Goals

- For each attack we can determine
 - Precondition = rights required to implement the attack
 - Postcondition = rights gained if the attack is successful
 - Resources required to implement an attack
- Modelling a threat in terms of available resources and goal
- Each goal is expressed as a set of rights
- For each goal and a threat, an impact for the owner can be computed





Resource to implement an attack

Co-Sponsored by SAGE

LISA-NT
2nd Large Scale System Administration of
WINDOWS NT/2000 CONFERENCE

July 30 – August 2, 2000
Madison Renaissance Hotel
Seattle, Washington

Security and the System Administrator

Cost of Attack

- ◆ Work
- ◆ Access
- ◆ Indifference to detection
- ◆ Special Knowledge
- ◆ Time to corrective action

Any one can reduce the requirements for any of the others; there is enough of these in the world to break any system.

**Deloitte
Touche
Tohmatsu**





Transitive closure

- After any attack we compute the closure of the rights = compute reachable hypergraph nodes
- The closure propagates the rights the threat owns after the attacks to discover those that are acquired because of dependencies
- If the transitive closure does not include a goal of the threat T , then T will implement other attacks till the goal has been achieved





Dependencies vs Attack Tree

- An hypergraph describes both And and Or decompositions of complex attacks
- Automatic discovery of And-Or Attack trees by an analysis that considers
 - Structural dependencies among infrastructure components
 - Component vulnerabilities and the attacks they enable
 - Pre and Post conditions of each attack
 - Rewriting of user rights according to the attack
 - Goals as set of rights





Rewriting System

- $S = \langle \dots, \langle U_i, \{ \langle Ca, i \rangle, \langle Cb, c \rangle \dots, \} \rangle, \dots \rangle$
- $\text{Pre}(A_1) = \{ \langle Ca, i \rangle, \langle Cb, c \rangle \}$
- $\text{Post}(A_1) = \{ \langle Cd, c \rangle \}$
- $S' = \langle \dots, \langle U_i, \{ \langle Ca, i \rangle, \langle Cb, c \rangle, \langle Cd, c \rangle \dots, \} \rangle, \dots \rangle$
- $S'' = \text{transitive closure of } S'$

then

$$S \xrightarrow{U_i, A_1} S''$$





Infrastructure evolution

- An evolution due to a threat T is a sequence of states
 - in the initial one any threat owns only legal rights only
 - in the final one T has achieved one of its goals
- Any transition is due to a useful attack enabled by a component vulnerability + closure after the attack
- The final state is paired with an impact for the owner
- An evolution may be due to a set of cooperating threats
- Evolutions can be described through
 - Sequences of rewrite rules
 - Finite state automata (attack automata)
 - Paths of a labelled oriented graph = attack graph





Some interesting cases

- 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 rights to each other
- **Competitive** evolution where some users can revoke the rights of other users
- **Monotone** evolutions= Concurrent + Collusion





Evolution vs Planning

- In the case of intelligent goal oriented 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 differences are is
 - an assessment should discover **any** evolution (= all plannings) while in other fields the discovery of **one** optimal or nearly optimal plan suffices
 - a threat is interested in a set of rights that includes her goal rather than in one that is equal to her goal (a problem for backward reasoning)



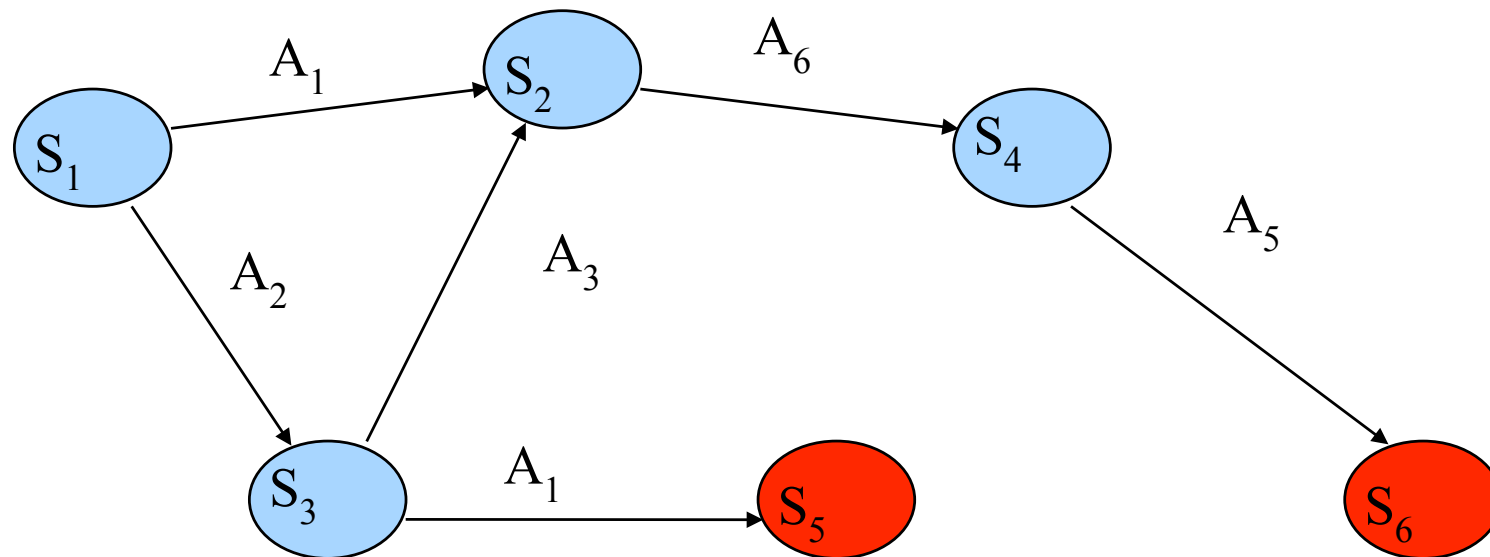


Countermeasures

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



Applying a countermeasure

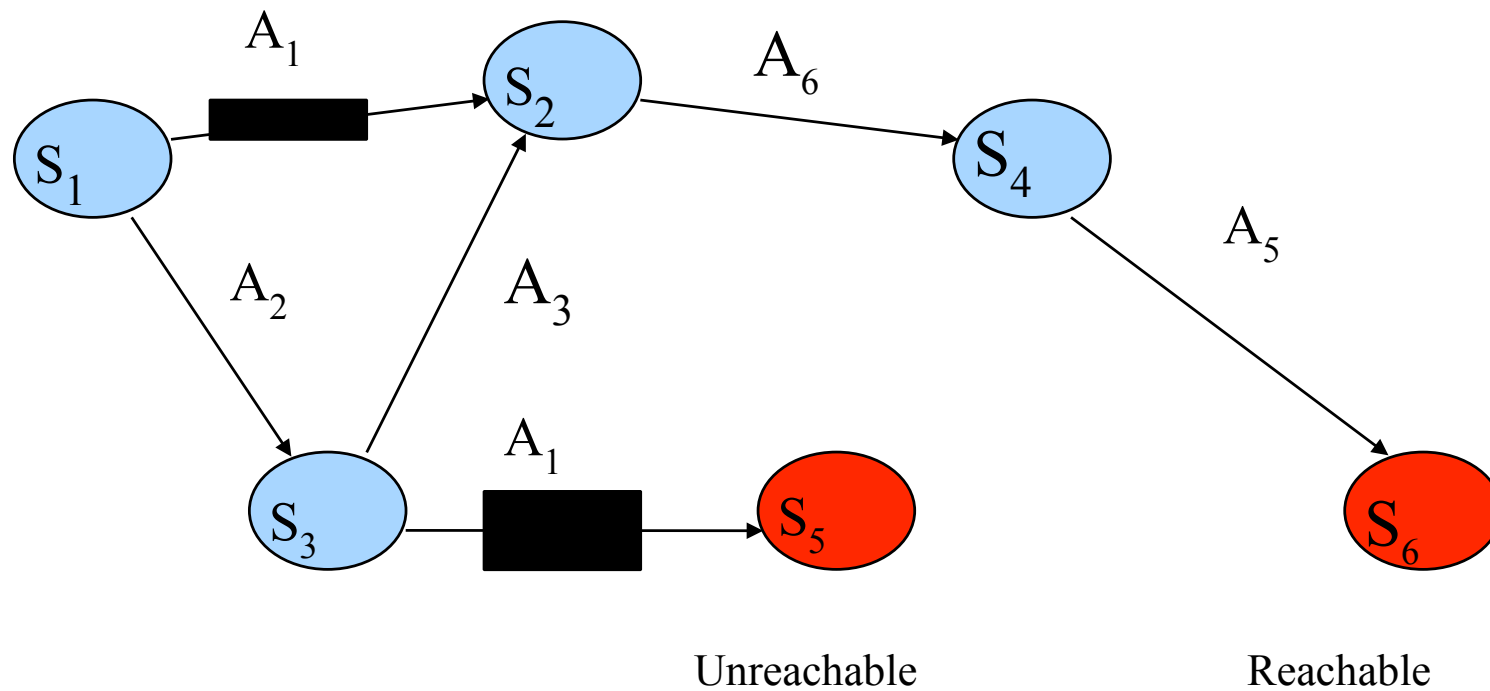


Goals of a threat

A countermeasure for A1 cuts two arcs



Applying a countermeasure



We also need a countermeasure for any of A_2 , A_3 , A_5 , A_5





Complete and minimal sets

- A set of countermeasures is complete if its countermeasures stop any evolution
- A complete set is minimal if none of its subset is complete
- Computation of minimal sets is an NP complete problem
- We can consider the cost of countermeasures in a set rather than the size of the set
- A minimal set of countermeasures does not correspond to a minimal cut set since all the arcs with a given label are removed





Ranking of vulnerabilities

- Any minimal set of countermeasure represent a plan = a distinct way to stop all the evolutions
- Some attacks may be successful but no threat will be able to achieve any of her goals
- A ranking of vulnerabilities can be defined by taking into account the percentage of minimal sets that remove a given vulnerability = the importance of removing a vulnerability to stop all the evolutions
- In biology, the average percentage is a measure of a fragility of an individual





Ranking

The previous ranking is useful

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

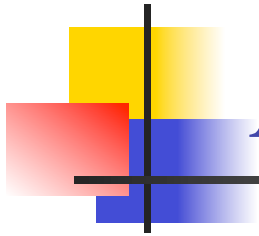




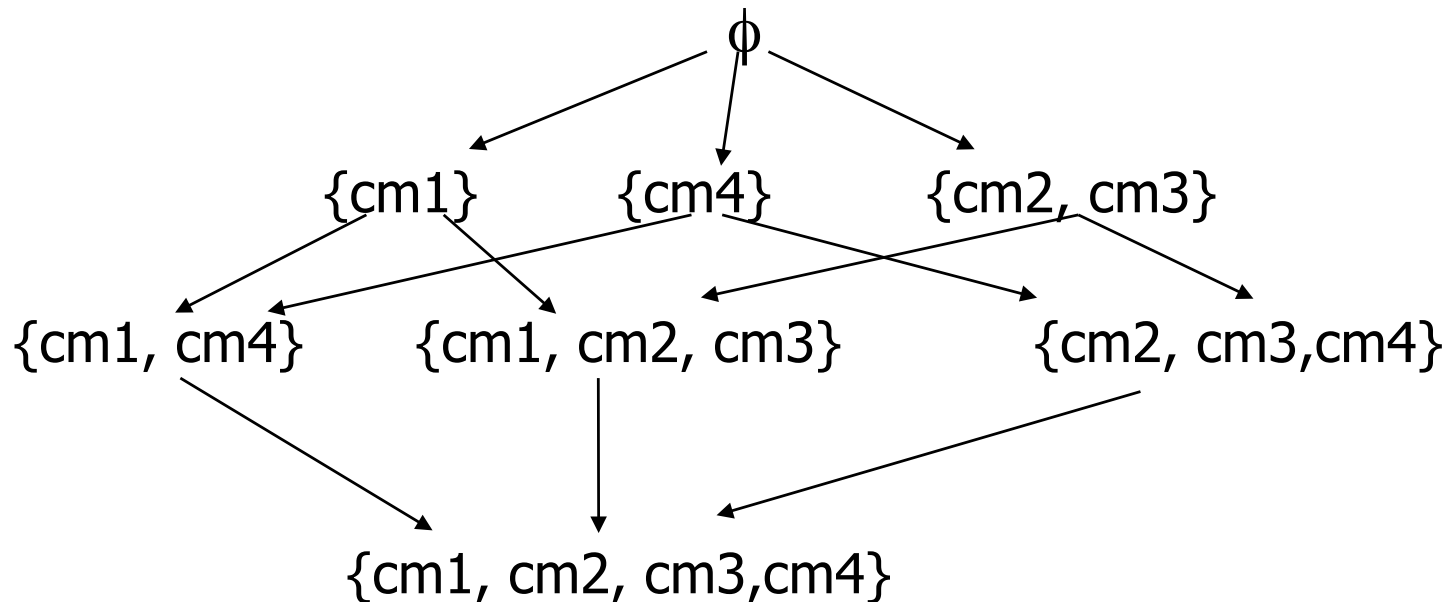
Risk avoidance plans

- Cost effective risk avoidance plan implements a minimal set of countermeasures
- A plan is defined by composing non redundant subsets of a minimal set
- A non redundant subset cuts all the path to a subset of final nodes
- Distinct compositions define alternative time schedulings of the same plan
- A risk avoidance plan is a chain in the ordering of non redundant subsets of minimal sets



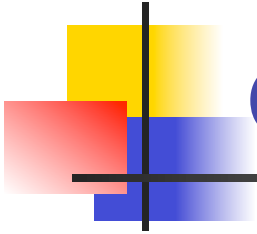


A first set of risk avoidance plans

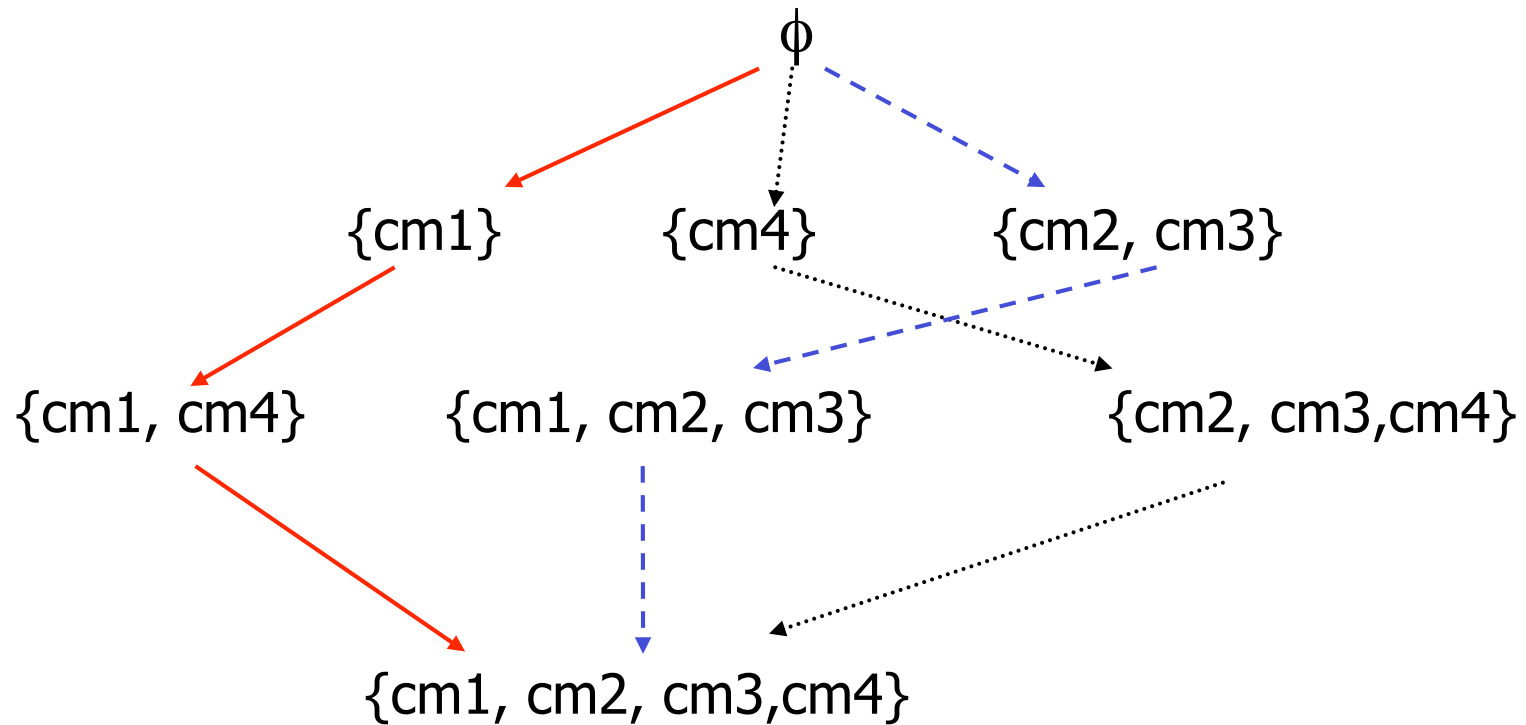


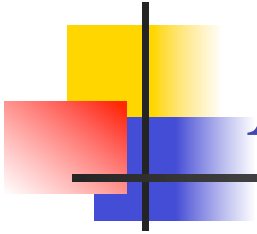
Implementation of the minimal set $\{cm1, cm2, cm3, cm4\}$
if the non redundant subsets are $\{cm1\}$, $\{cm4\}$, $\{cm2, cm3\}$



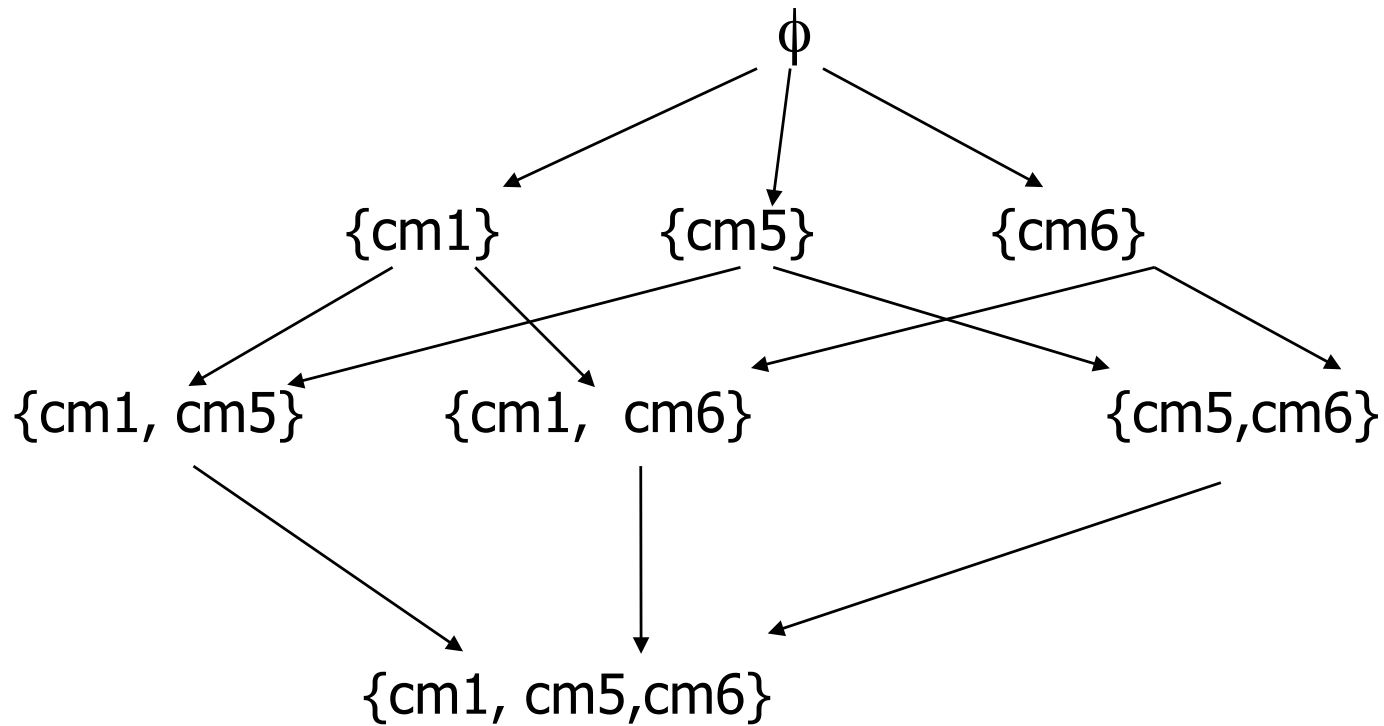


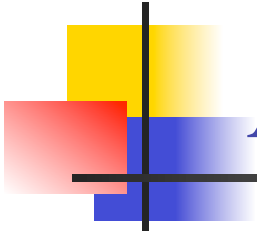
One Plan: Alternative schedulings



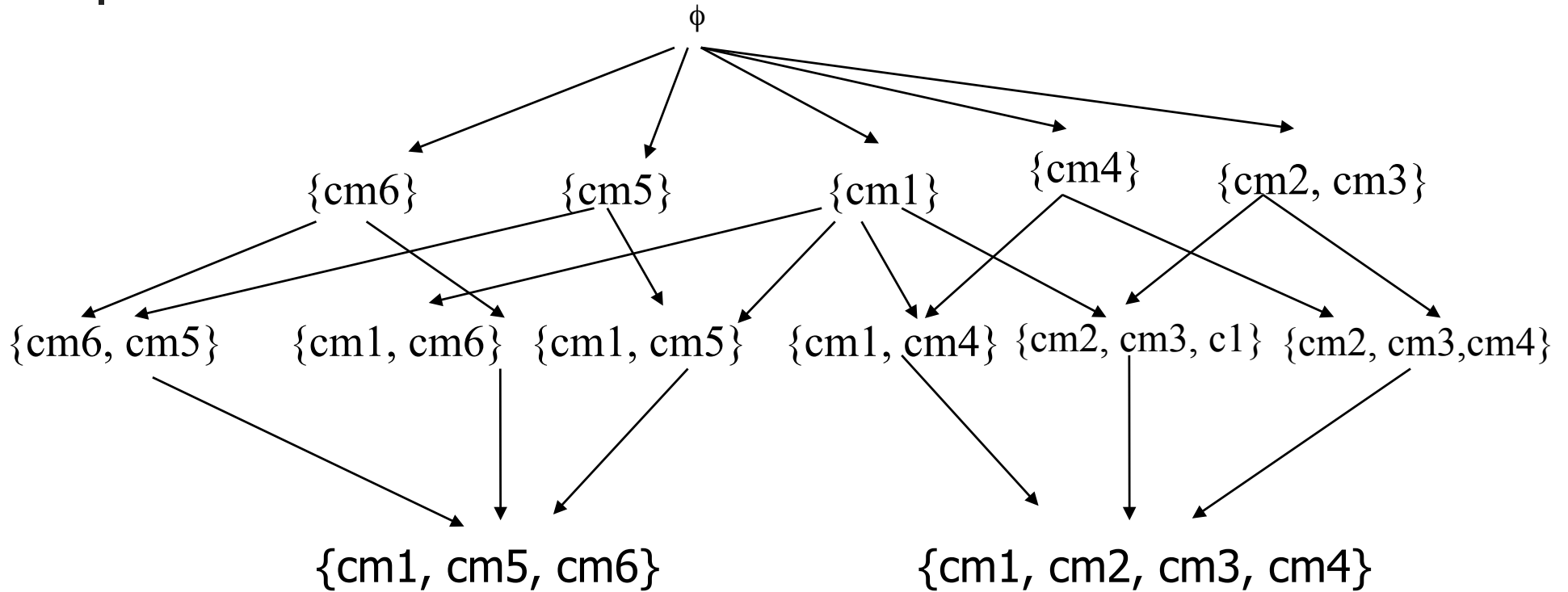


Another minimal set





Alternative plans = Global ordering

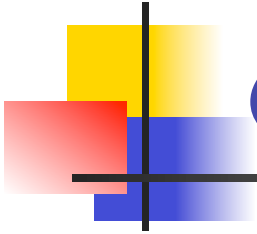




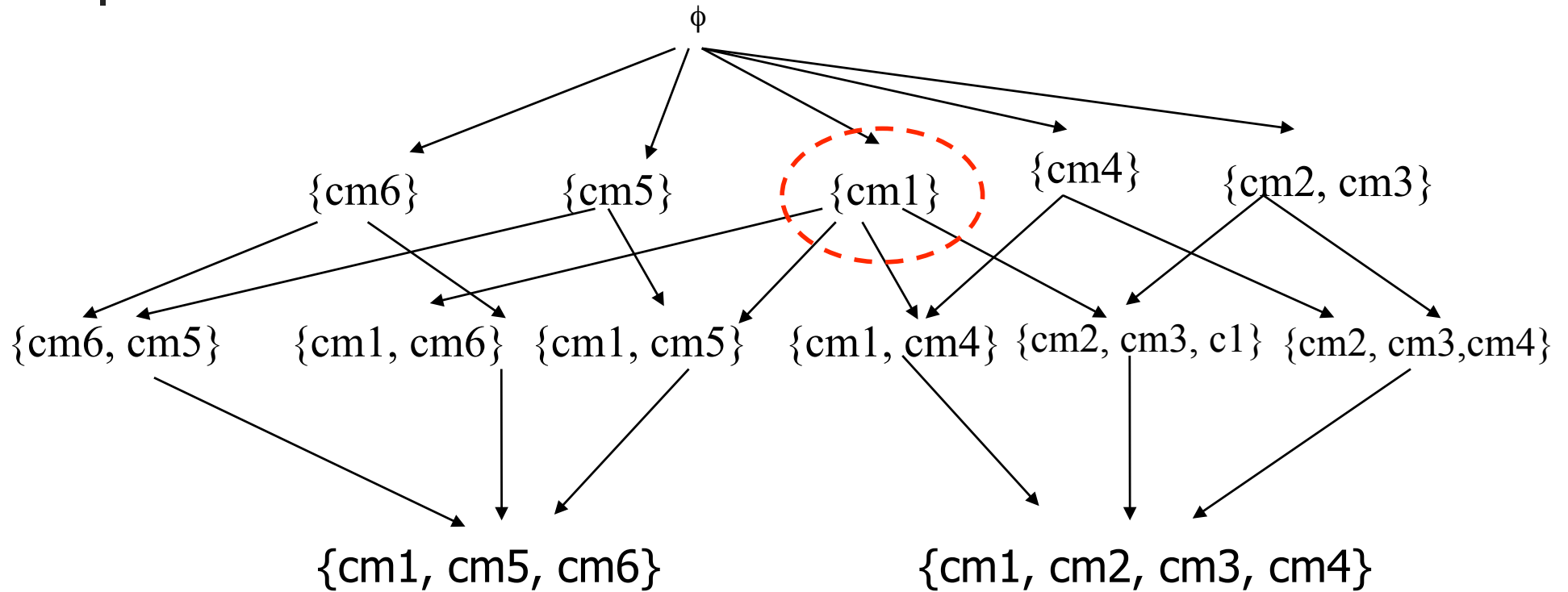
Least Commitment

- We can choose to implement at first those redundant sets that belong to distinct minimal sets and hence to distinct plans
- This delays the choice of a given risk mitigation plan till information about investment on countermeasures is available





Global ordering = Alternative plans





Taking risk into account

When computing minimal sets we neglect evolutions

1. with an impact lower than I
 2. requiring more than N attacks
 3. with an overall complexity larger than OC
 4. where an attack with a complexity larger than C
- 1 = neglectable for the owner
 - 2 – 4 too complex
 - Complexity of a sequence of attacks may be defined using the WAIST metric
 - Historical information about previous attacks simplifies the definition of the various thresholds





Programming tools

- Evolution Graph Builder
starting from a set of databases about dependencies, attacks and threat builds the evolution graph
- Graph Pruner
remove evolutions too complex or not convenient
- Countermeasure Selector
computes the optimal set of countermeasure by solving a 0/1 optimization problem





Increasing the detail level – I

- The model support the description of the infrastructure at distinct abstraction levels
- To achieve a more detailed description at a given abstraction level the component operations are considered
 - The rights are expressed in terms of operations
 - The control of the attribute of a component depends upon which operations a user can invoke



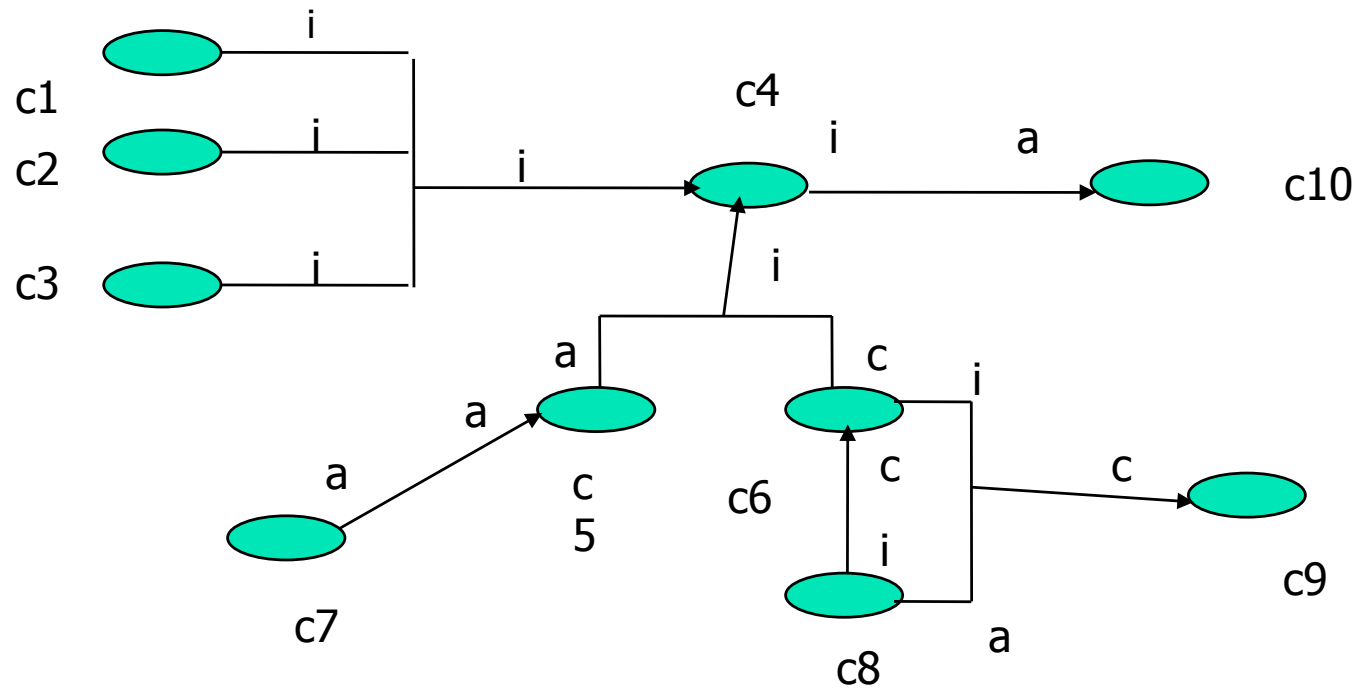


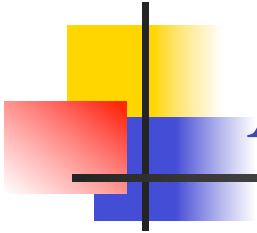
Increasing the detail level – II

- Zooming into the model = Hierarchical decomposition
- A node N of the infrastructure hypergraph is replaced by an hypergraph $\text{Dec}(N)$
- Hyperarcs to/from N are replaced by hyperarcs to/from nodes of $\text{Dec}(N)$

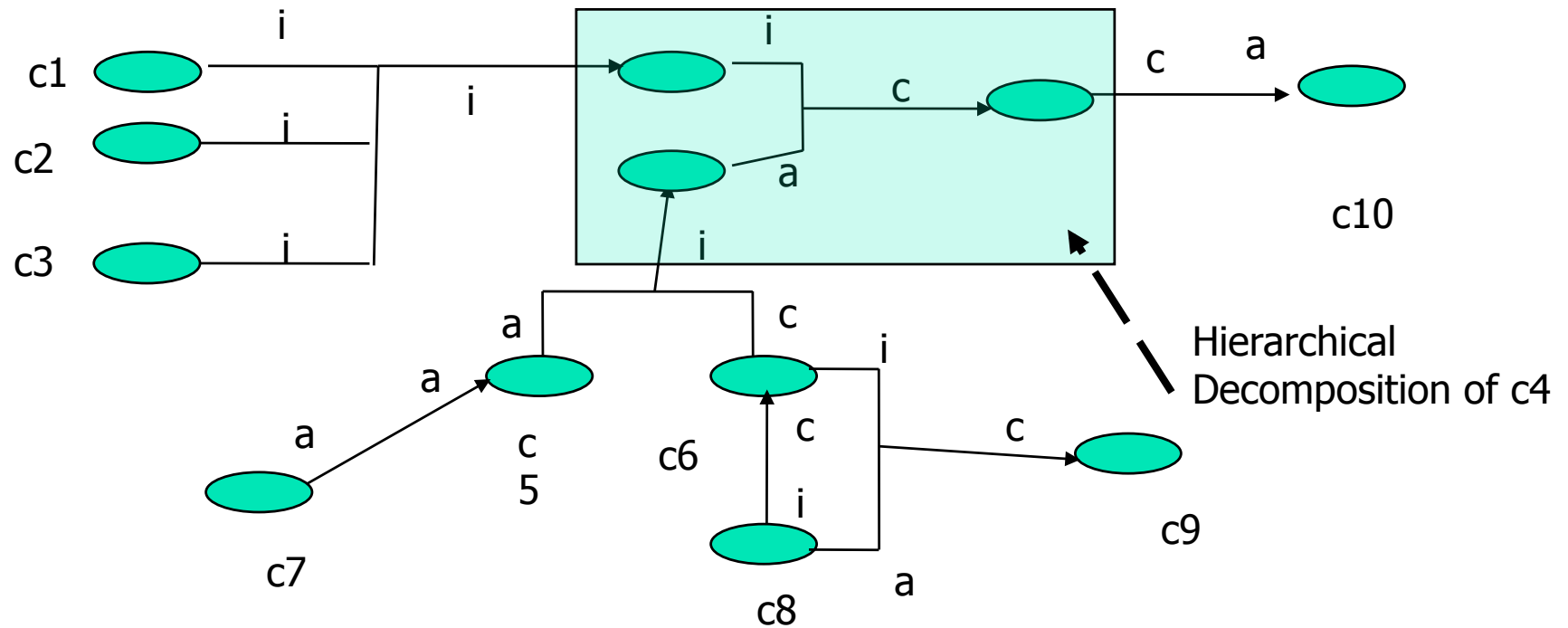


An Infrastructure Hypergraph





A more detailed hypergraph





Equivalence conditions

- We want to define conditions to guarantee that the assessment is not invalidated \Leftrightarrow we do not have to repeat the assessment
- These conditions are defined in terms of
 - Path in the hypergraph
 - Evolution of the overall infrastructure





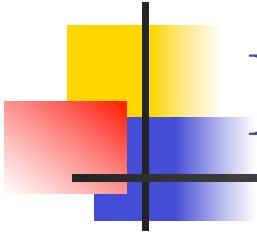
Path in the hypergraph

If there are two arcs

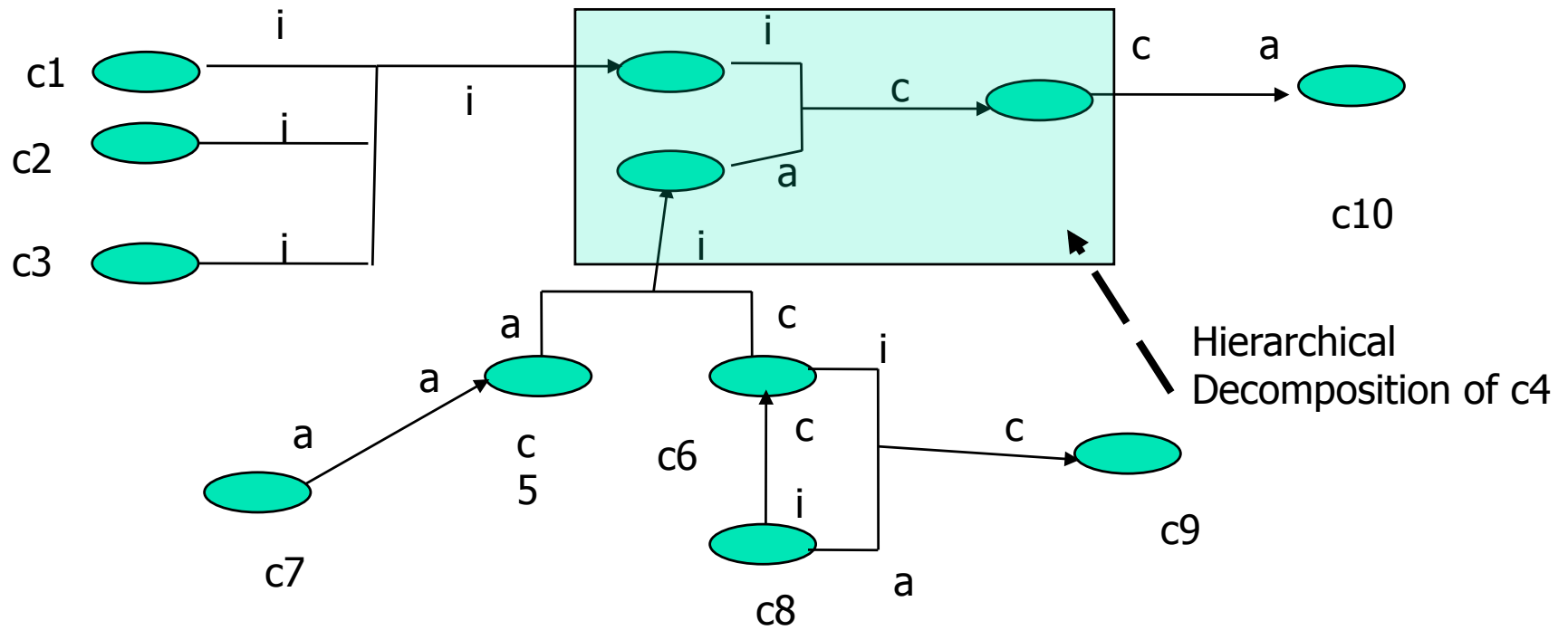
- one leading to N
- one with the same label (x) leaving N

then in the hypergraph that replaces N the transitive closure of the arc labelled x and leading to N includes the arcs leaving from N



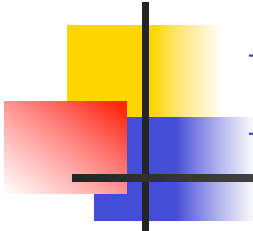


Example - 1

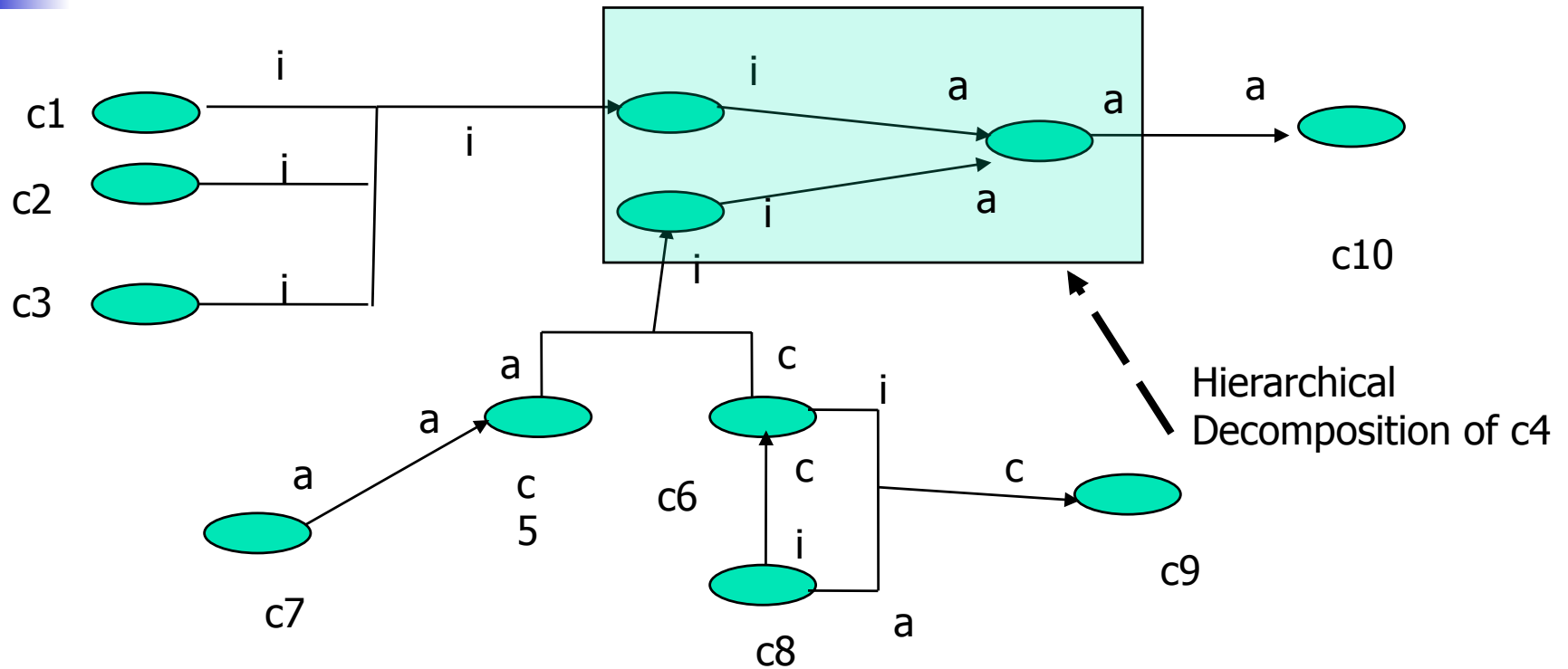


Not equivalent





Example - 2



Equivalent since the control of integrity of c4
Results in the control of availability of c10





More general cases

- How evolutions of the hypergraph that replaces N (**low level evolutions**) influence the evolutions in the original hypergraph, **high level evolutions**
- Low level evolutions describe in a more detailed way how rights on new and old components can be acquired
- No low level evolution should increase the rights of a threat on an old component (outside the decomposition of N) or these rights do not belong to an attack precondition (sufficient condition to avoid a new global assessment)





More general cases

An evolution increases the rights of a threat T on a component outside the decomposition of N but

- T cannot implement any further attack because of these rights or
- these rights cannot be acquired because of a countermeasure



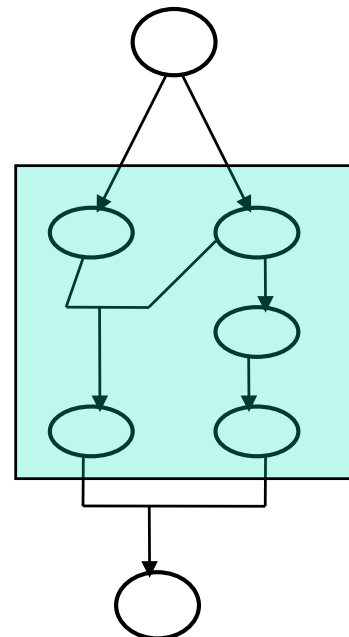
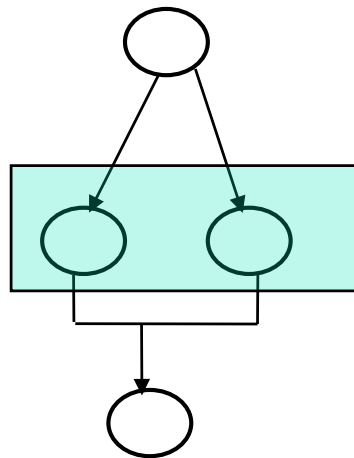
a countermeasure prevents the high level evolutions from achieving these rights

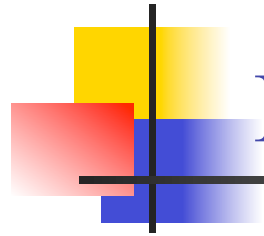




More general decompositions

- Several nodes can be decomposed simultaneously provided that no arcs to from these nodes are introduced





Decomposition and Cost Effectiveness

- The conditions that take the applications of countermeasure into account
 - guarantee that the risk mitigation plan is correct even after the decomposition
 - do not guarantee that the set of countemeasures is still a minimal one





Current and Future Work

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





References

1. *Constrained Finite State Automata for Risk Analysis and Assessment*. **NATO Adv. Res. Workshop on Information Security and Assurance**,
3. *A Theoretical Model for the Average Impact of Attacks on Billing Infrastructure*, **3rd Int. Workshop on Mathematical Methods, Models and Architectures for Network Security**
4. *Assessing the Risk of an Information Infrastructure through Security Dependencies*, **International Workshop on the Protection of Critical Infrastructure, CRITIS 2006**
5. *A Hierarchical, Model based Risk Management of Critical Infrastructures*, **European Symposium on Security and Reliability, ESREL 2007** and in review for publication on **Reliability Engineering and System Safety (RESS)**, Special Issue of Selected Papers from ESREL 2007.
6. *Risk Assessment of an Information Infrastructure: a Framework based upon Security Dependencies*, **International Journal of System of Systems Engineering**

