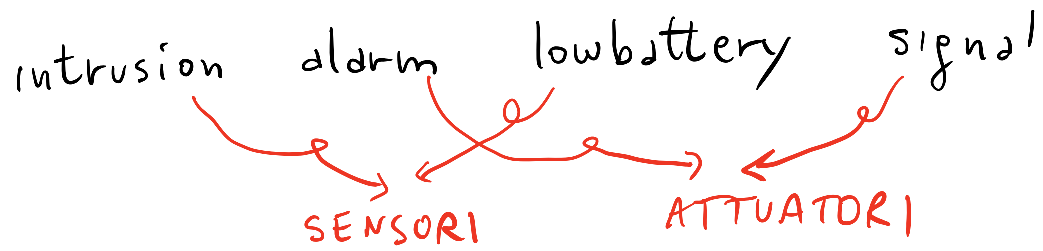
ESEMPI

3 · ABBIAMO A CHE FARE CON UN SISTEMA DI ALLARME CHE SEGNALA LA PRESENZA DI EVENTI ANOMALI CON CERTE GARANZIE TEMPORALI

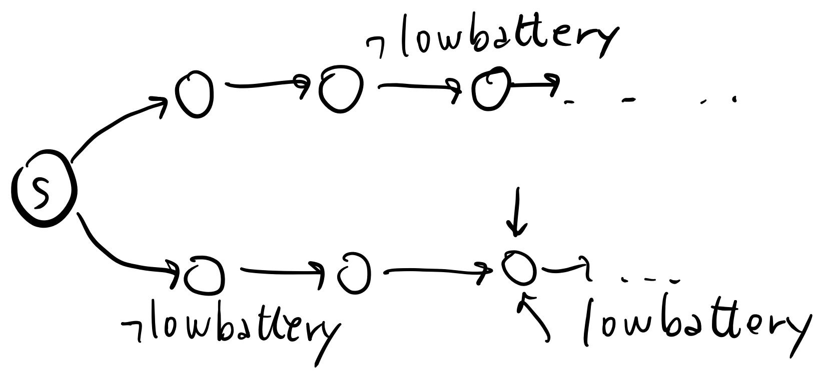
· ABBIAMO A DISPOSIZIONE QUATTRO ATOMI PROPOSIZIONALI



· VORREMMO, PER ESEMPIO, ESSERE SICURI CHE, SE VIENE RILEVATA UN'INTRUSIONE, SCATTI L'ALLARME ENTRO TRE ISTANTI

$$AG\left(intrusion \longrightarrow \left[AX(AX(AX(alarm))) \vee AX(AX(alarm)) \vee AX(alarm) \vee alarm\right]\right)$$

· SE C'È IL RISCHIO DI RILEVARE UN BASSO LIVELLO DI BATTERIA NEI PROSSIMI n ISTANTI, OCCORRE SEGNALARLO IMMEDIATAMENTE E PER ALMENO DUE ISTANTI.



DEFINIAMO LA FORMULA $EX^n F$ NEL MODO

SEGUENTE (PER INDUZIONE SU n):

$$EX^0 F \equiv F \qquad EX^{n+1} F \equiv F \vee EX(EX^n F)$$

IN QUESTO ABBIAMO CHE LA FORMULA EX^n lowbattery CATTURA PROPRIO IL RISCHIO CHE IN n PASSI IL SISTEMA SI POSSA TROVARE IN UNA SITUAZIONE DI BATTERIA SCARICA

- È CHIARO CHE LA SPECIFICA POTREBBE DIVENTARE

$$AG\left[EX^n\,\text{lowbattery} \rightarrow \text{signal} \wedge AX(\text{signal}) \wedge AX(AX(\text{signal}))\right]$$

<u>IL MODEL CHECKING IN CTL È UN PROBLEMA RISOLVIBILE IN TEMPO POLINOMIALE</u>

- EQUIVALENZA LOGICA TRA FORMULE CTL:

$$F \equiv G \quad \text{sse} \quad (\forall \mathcal{M}.\forall s.\ \mathcal{M},s \models F \Longleftrightarrow \mathcal{M},s \models G)$$

- <u>LEMMA</u>

$$AX\,F \equiv \neg(EX \neg F)$$
$$AG\,F \equiv \neg(EF \neg F)$$
$$A(F\ U\ G) \equiv \neg(E(\neg G\ U\ (\neg F \wedge \neg G))) \wedge \neg(EG \neg G)$$

$A(F \; R \; G) \equiv \neg E(\neg F \cup \neg G)$

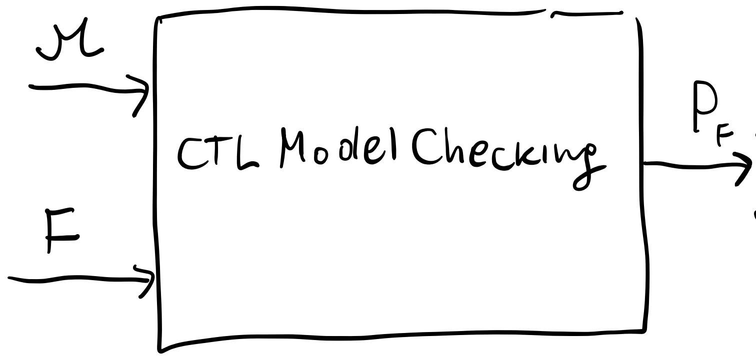
$AF \; F \equiv \neg EG(\neg F)$

$EF \; F \equiv E(\text{TRUE} \; U \; F)$

$E(F \; R \; G) \equiv \neg A(\neg F \; U \; G) \equiv \ldots$

LE EQUIVALENZE LOGICHE DI CUI SOPRA CI PERMETTONO DI ASSUMERE CHE LA FORMULA SU CUI SI VOGLIA FARE MODEL-CHECKING "CONTENGA" SOLO EX, EG, EU OLTRE AGLI OPERATORI BOOLEANI

- L'ALGORITMO CHE COSTRUIREMO HA LA FORMA SEGUENTE



. TALE CHE $\mathcal{M}, s \models F$ SSE SE P_F

- SE G È SOTTOFORMULA DI F, SCRIVEREMO $G \sqsubseteq F$. IL NUMERO DI SOTTOFORMULE DI UNA CERTA FORMULA F È NIENT'ALTRO CHE UN MODO PER MISURARE LA "DIMENSIONE" DI F, CHIAMIAMO $|F|$

$|EG(AF(alarm \vee signal))| = 5$

- DIAMO L'ALGORITMO

CTLModelChecking$((S, S_0, R, L), F)$
  for $G \sqsubseteq F$ do:
    Done[G] = False

while $\neg$Done[F] do:

pick G such that $\neg$Done[G] and Done[H] $\forall H \subsetneq G$

Done[G] = True

case G of:

$P \in AP \longmapsto$ States[P] = $\{ s \in S \mid P \in L(s) \}$

$\neg H \longmapsto$ States[$\neg$H] = $S \setminus$ States[H]

$H \vee K \longmapsto$ States[$H \vee K$] = States[H] $\cup$ States[K]

$H \wedge K \longmapsto$ States[$H \wedge K$] = States[H] $\cap$ States[K]

$EX\ H \longmapsto$ States[EX H] = $\{ s \in S \mid \exists q.\ (s,q) \in R \wedge q \in$ States[H]$\}$

$E(H\,U\,K) \longmapsto$ States[E(H U K)] = CheckEU(H, K, States)

$EG\ H \longmapsto$ States[EG H] = CheckEG(H, States)

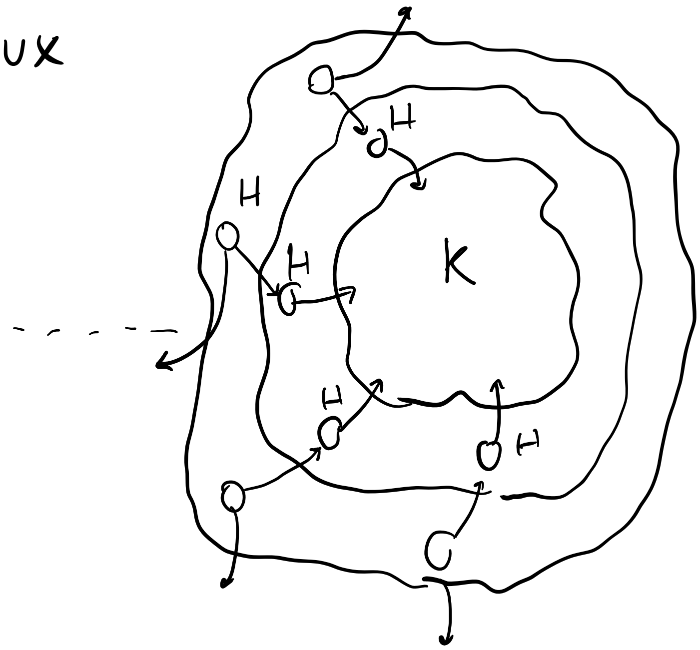
return States[F]

- QUESTA PROCEDURA HA COMPLESSITÀ POLINOMIALE IN TEMPO
- RESTANO PERÒ DA DEFINIRE GLI ALGORITMI AUSILIARI CheckEU e CheckEG
- COME FUNZIONERÀ $\boxed{\text{CheckEU}}$(H, K, States)?
  - L'ALGORITMO PROCEDERÀ COSTRUENDO L'INSIEME DEGLI STATI CHE SODDISFANO

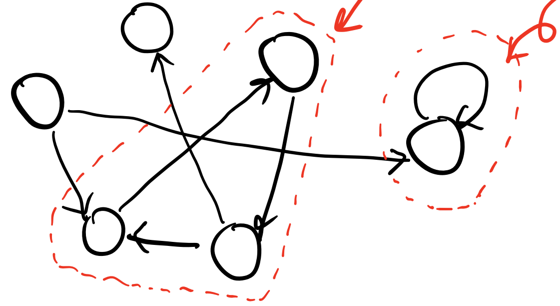
E(H U K) NEL MODO SEGUENTE

StatesInc = States[K]
repeat
  Aux = StatesInc
  StatesInc = StatesInc $\cup$ {s $\in$ States[H] | $\exists$q. (s,q) $\in$ R $\wedge$ q $\in$ Aux}

until Aux = StatesInc
return Aux



- COME POTREMO COSTRUIRE, INVECE, CheckEG(H, States)?
  - L'IDEA CRUCIALE È QUELLA DI UTILIZZARE IL CONCETTO COMPONENTE FORTEMENTE CONNESSA (SCC) DI UN GRAFO, OSSIA UN SOTTOINSIEME P DEI NODI DEL GRAFO TALE PER CUI OGNI ELEMENTO DI P SIA RAGGIUNGIBILE DA QUALUNQUE ALTRO ELEMENTO DI P
  - ESEMPIO

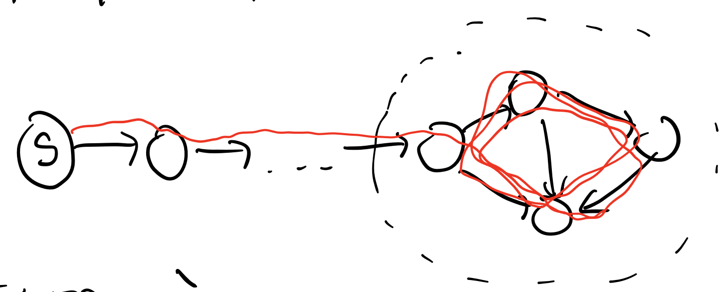


· LEMMA

$\mathcal{M}, s \models EG\ F$ SSE È POSSIBILE COSTRUIRE UN CAMMINO CHE DA s PORTI, TRAMITE R AD UNA SCC (MASSIMALE E NON TRIVIALE) CONTENENTE STATI CHE SODDISFINO F.

DIMOSTRIAMOLO

($\Leftarrow$) QUESTO PASSAGGIO È MOLTO SEMPLICE, PERCHÉ LE IPOTESI CI ASSICURANO DELL'ESISTENZA DI UN CAMMINO π CHE INIZI IN s E IN CUI F VALGA SEMPRE



($\Rightarrow$) QUESTO È UN PASSAGGIO UN PO' PIÙ COMPLESSO. SIA π UN CAMMINO CHE PARTE DA s E IN CUI F VALE SEMPRE (TALE CAMMINO ESISTE PER IPOTESI). ESTRAIAMO DA π UNA SCC. CERTAMENTE, π È UNA SEQUENZA <u>INFINITA</u>

I CUI ELEMENTI SONO PRESI DA UN INSIEME FINITO. CI DEVE ESSERE UN SUFFISSO ρ DI π TALE PER CUI <u>TUTTI</u> GLI STATI IN ρ OCCORRONO IN QUESTO UN NUMERO INFINITO DI VOLTE

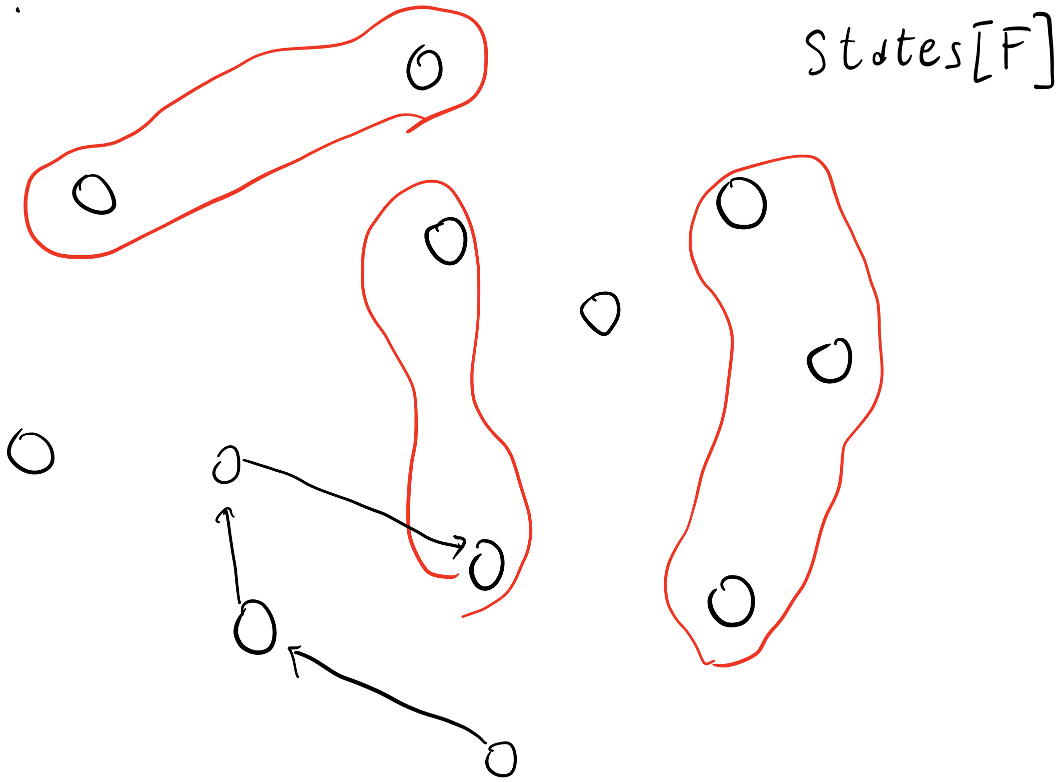
$$\pi = \underbrace{s_0 \; s_1 \; s_2 \; s_3 \; \ldots \ldots}_{\text{IL PREFISSO DI } \pi \text{ CONTENTE TUTTE LE OCCORRENZE DI TUTTI GLI STATI CHE OCCORRONO NI } \pi \text{ UN NUMERO FINITO DI VOLTE}} \; \rho$$

LA SCC CHE STIAMO CERCANDO NON SARÀ NIENT'ALTRO CHE L'INSIEME DEGLI STATI CHE OCCORRONO IN ρ. IL FATTO CHE ρ ESISTA È PROVA DEL FATTO CHE TALE INSIEME È UNA SCC: IL CAMMINO DA s A s' (ENTRAMBI IN ρ) SI PUÒ <u>SEMPRE</u> TROVARE PROPRIO PERCHÉ s E s' OCCORRONO IN ρ INFINITE VOLTE.

$\boxtimes$

· IL LEMMA CHE ABBIAMO APPENA DIMOSTRA:

TO È IN REALTÀ TUTTO CIÒ CHE SERVE A COSTRUIRE CheckEG, IL QUALE PROCEDERÀ DETERMINANDO LE SCC MASSIMALI E NONTRIVIALI DI $\mathcal{M}$ CHE CONTENGANO SOLO STATI IN States[F], PER POI CONTROLLARE DA QUALI STATI IN States[F] TALI SCC SIANO RAGGIUNGIBILI.



CONTROLLO DEGLI ACCESSI

- <u>BINDER</u>
- LE POLITICHE DI CONTROLLO DEGLI ACCESSI DIVENTANO PROGRAMMI DATALOG
- ESEMPI

ACL $\longmapsto$ can(john_smith, read, resource_r)
can(john_smith, write, resource_r)
can(fred_jones, read, resource_q)
⋮

POLICY $\longmapsto$ can(x, read, resource_r) :-
employee(x, bigco)
employee(john_smith, bigco)

POLICY BASATA SUI RAPPORTI GERARCHICI $\longmapsto$
can(x, ~~read~~ lod, ~~resource_r~~ res) :-
employee(x, bigco), res
boss(y, x),
approves(y, x, ~~read~~ lod, ~~resource_r~~ res)

LOGICHE DI HOARE

$(X = n \geq 0) \wedge (Y = m \geq 0)$

$\{P\}$ FORMULA AL PRIM'ORDINE CHE IDENTIFICA "GLI INPUT VALIDI"

```
while ¬(X=Y) do
    if X≤Y then
        Y := Y - X
    else
        X := X - Y
```

ALGORITMO DI EUCLIDE PER IL GCD

$\{R\}$ FORMULA AL PRIM'ORDINE CHE ESPRIME UNA CONDIZIONE CHE DEVE ESSERE VERIFICATA <u>DOPO</u> L'ESECUZIONE DEL PROGRAMMA

$$\left(X = Y = gcd(n, m)\right) \approx X = Y \wedge X \mid n \wedge X \mid m \wedge \forall z. \left[z \mid n \wedge z \mid m\right] \rightarrow z \mid X$$