Proprietà del framework *SOCS-SI* per la verifica di protocolli di interazione tra agenti

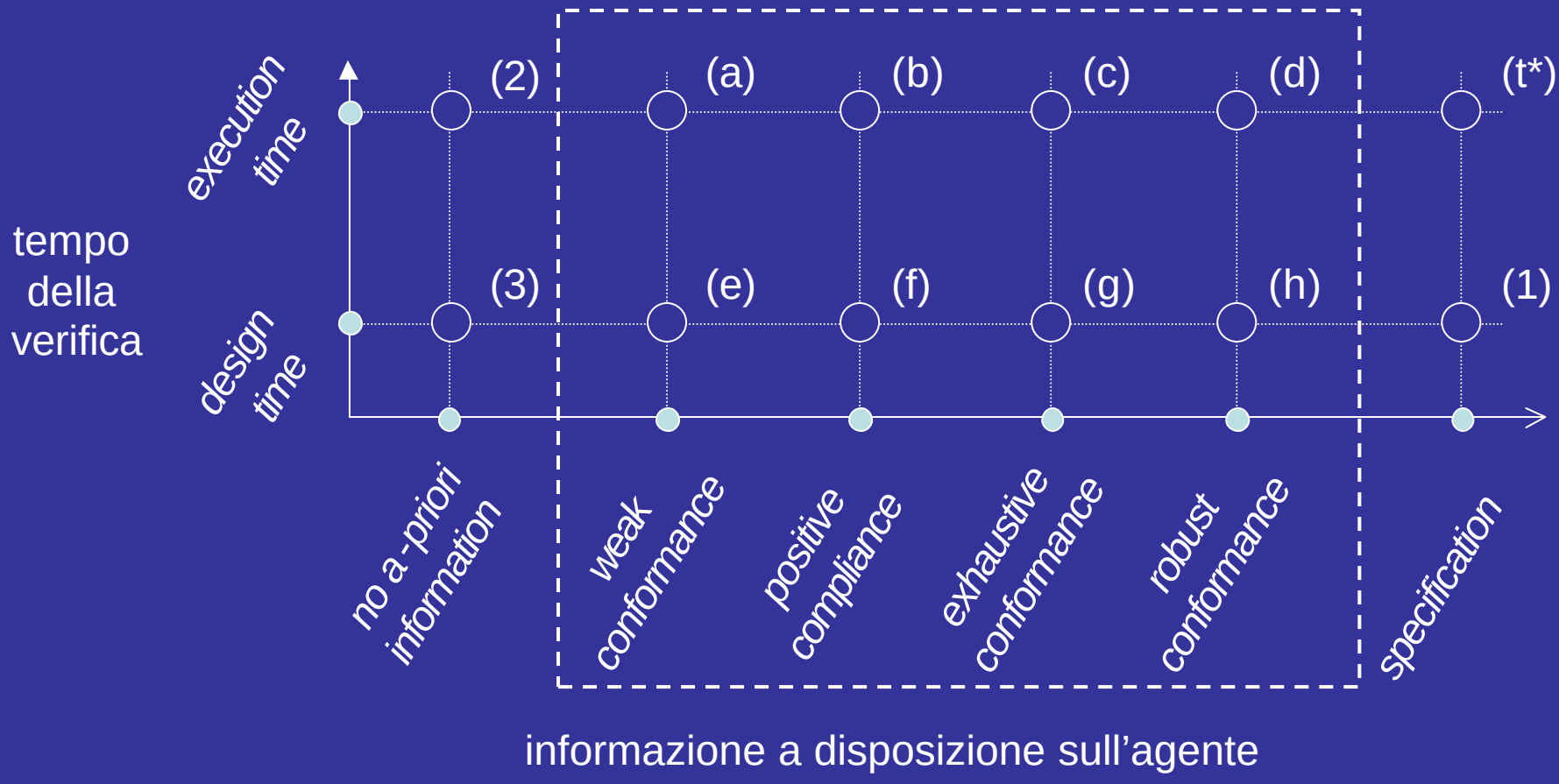
gruppo di Bologna

in collaborazione con Ferrara

Quali proprietà

- Proprietà generali (del framework)
  - well-definedness di programmi e vincoli ICS
  - terminazione della proof
- Proprietà dell'interazione
  - punto di vista del meccanismo:
    - proprietà "generali" (fairness, termination, …)
    - proprietà "specifiche" (espresse mediante formule)
  - punto di vista dell'agente: aderenza ai protocolli

Livelli di verifica



Proprietà generali

- Framework SOCS-SI:
  - espressività
  - corrispondenza con altri formalismi
- SCIFF ed estensioni:
  - correttezza
  - terminazione
  - completezza

Dimostrazione di proprietà specifiche

- Case study: Needham-Schroeder
  - verifica che il protocollo (non) è ‘sicuro’
  - generazione dell’attacco di Lowe
- Utilizzo del framework SOCS-SI:
  - come estendere le proof esistenti per fare una verifica statica?
  - è possibile verificare formule?
  - nell’esempio del Needham-Shroeder: è possibile generare l’attacco di Lowe?

Overview

- Background sul framework SOCS-SI
  - vincoli di integrità sociali e social KB
  - SCIFF
  - implementazione ed esperimenti fatti
- Descrizione del Needham-Shroeder
  - protocollo
  - attacco di Lowe
- SCIFF generativa: g-SCIFF
- Risultati nel caso del Needham-Shroeder
- Risultati generali (proprietà della g-SCIFF)

Il framework SOCS-SI

- Vincoli di integrità sociali
- Social KB
- SCIFF
- Implementazione e integrazione
- Sperimentazione e testing

Vincoli di integrità sociali (*ics*)

- concetto di *expectations*:
  - positive expectations, **E**(tell(…), T)
  - negative expectations, **EN**(tell(…), T)

(¬) H(Event, Time) ? …? (¬) H(Event, Time)
⇒ E(Event, Time) ? …? EN (Event, Time)
? E(Event, Time) ? …? EN (Event, Time)

Esempio di *ics*

H(tell(B, A, bid(Item, Q)), T_{Bid})
? **H**(tell(A, B, answ(win, Item, Q)), T_{Win})
? $T_{bid} < T_{Win}$
? deadline(delivery, $T_{Deliver\_deadline}$),
$\Rightarrow$
E(tell(A, B, deliver(Item)), T_{Del}) ?
$T_{Del} < T_{win} + T_{Deliver\_deadline}$

Social KB

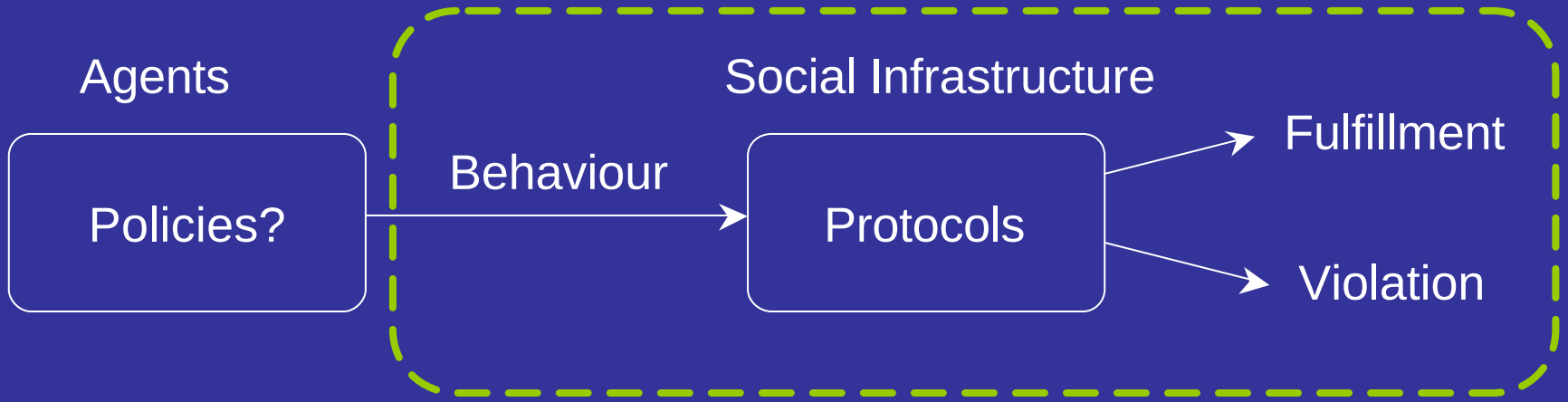
- Parte statica:
  - vincoli sociali, per esprimere:
    - protocolli
    - semantica degli atti (comunicativi)
  - 'SOKB' (programma logico che definisce clausole del tipo: `deadline( delivery, 10 ))`
- Parte dinamica:
  - eventi che sono accaduti
  - expectations generate
  - violazioni / fulfillment

SCIFF

- Dati:
  - conoscenza statica (ICS) e dinamica (HAP)
- Fornisce:
  - EXP:

    (E(…) ? EN(…) ? …) ? (E(…) ? EN(…) ? …) ? …
    - pend
    - fulf
    - viol

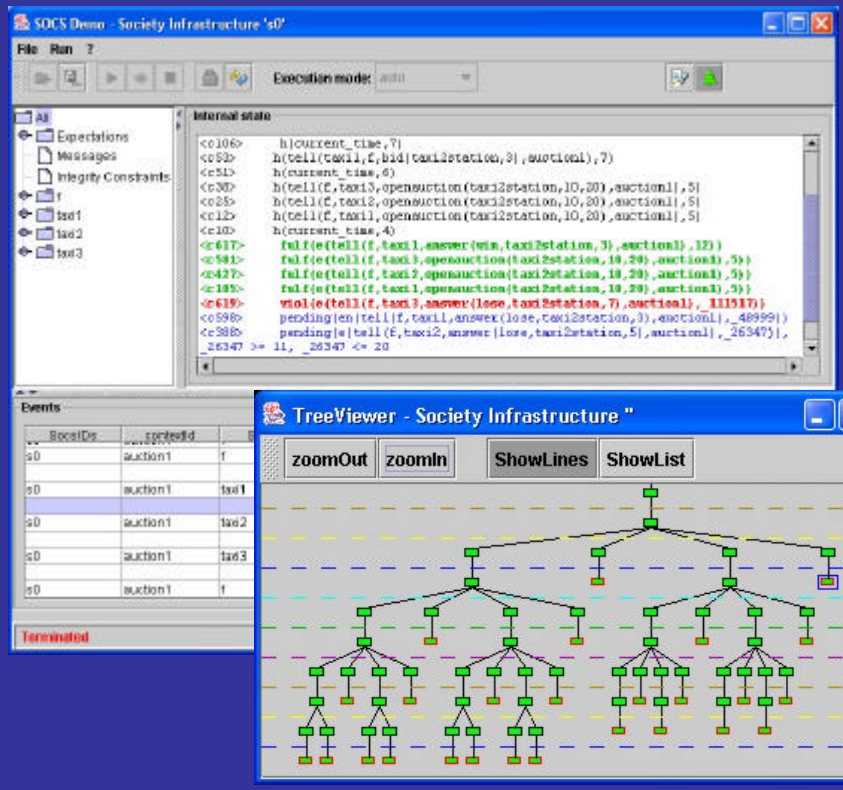
SOCS - Social Infrastructure



- Verifica a run-time di compliance
- Implementazione con SISCtus (CHR) + java

Snapshot

- Elenco degli agenti
- Elenco delle expectations
- Colori che evidenziano lo stato delle expectations
- Tree-view (SCIFF)
- Tasti di controllo

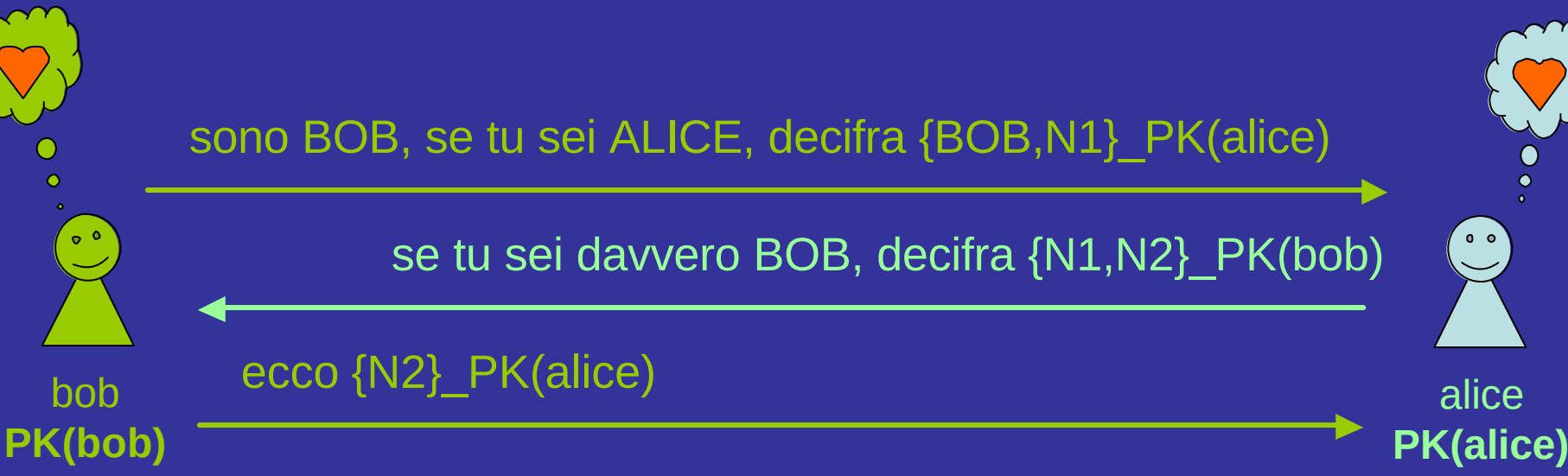


Integrazione e testing

- Integrazione con altre piattaforme:
  - PROSOCS
  - JADE
  - *tucson*
  - e-mail
- Testing su vari scenari, tra cui:
  - negoziazione a due e aste (combinatorie)
  - acquisto di merci elettroniche: NetBill
  - sicurezza: Needham-Schroeder

Needham-Shroeder

Protocollo di autenticazione con chiave pubblica

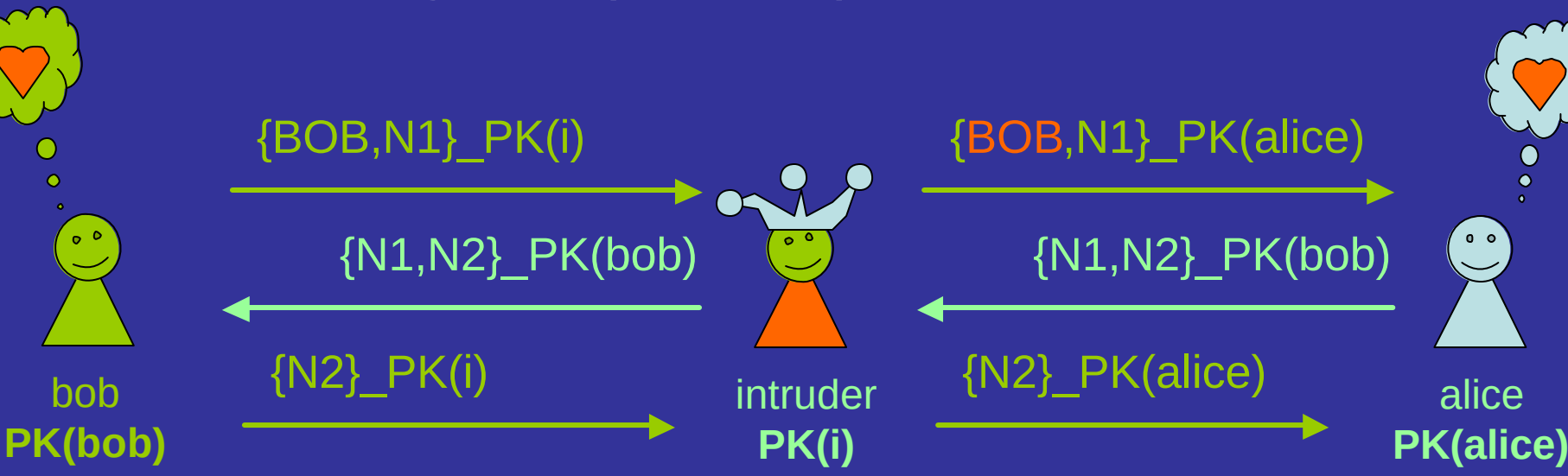


Ipotesi

- Perfetta crittografia:
  - Bob (Alice) non può *indovinare* il nonce di Alice (Bob)
- Canale insicuro:
  - chi spedisce un messaggio non può sapere se è arrivato
  - chi riceve un messaggio non può sapere chi è stato il mittente (a meno che non sia scritto nel messaggio)
  - il contenuto di un messaggio può venire manipolato

Attacco di Lowe

- Un terzo agente (intruder) fa finta di essere BOB



Codifica tramite *ics*

- Codifica del protocollo
- Codifica dell'ipotesi di perfetta crittografia
  - > È necessaria una rappresentazione esplicita di PK e contenuto (in chiaro) dei messaggi:

```
send( Intruder, Alice,
   content( key(PK_Alice), agent(Bob), nonce(N1)), 2 )
```

Verifica di compliance

- Nel caso dei protocolli di sicurezza, così come per gli altri protocolli, si può verificare che gli agenti seguano il protocollo
- Non basta a dimostrare che un protocollo è sicuro…

Verifica di proprietà del NS (1)

- Proprietà: se X si è autenticato come Bob nei confronti di Alice, allora X è Bob.
- L'attacco di Lowe è la prova che questa proprietà non vale (X=Intruder)
- Definizione di 'X si è autenticato come Bob' in termini di messaggi scambiati
- Definizione di 'X non può *indovinare* il nonce di Y' in termini di vincoli di integrità sociali

Verifica di proprietà del NS (2)

- Tecnica usata:
  - specifica del protocollo (NS)
  - specifica della ipotesi di perfetta crittografia
  - specifica della proprietà in forma negata (X si è autenticato come Bob e X non è Bob)
  - generazione di una sequenza di messaggi compliant con il protocollo che soddisfano la proprietà negata (attacco di Lowe)

Verifica statica di proprietà

- Le proprietà sono goal (formule)
- Di solito esprimono la negazione di ciò che si vuole confutare
- Se esiste una history compliant che soddisfa tale goal, la confutazione ha avuto successo (ipotesi di soundness)
- Invece, la impossibilità di generare una history può essere una prova di inconutabilità, nell'ipotesi di completezza (work in progress)

SCIFF generativa: g-SCIFF

- input:
  - una sequenza di eventi
  - KB sociale (statica e dinamica)
- output:
  - history compliant, oppure
  - fallimento
- usa un modulo aggiuntivo: *fulfiller*

Fulfiller

- Dato un nodo $\mathbf{N}_k$ in cui le exp. pend. sono:

$$\mathbf{PEXP}_k = \mathbf{PEXP}' \ ? \ \{\mathbf{E}(E,T)\}$$

e non sono applicabili transizioni di fulfillment, genera un nodo $\mathbf{N}_{k+1}$ identico a $\mathbf{N}_k$ eccetto che:

$$\mathbf{PEXP}_{k+1} = \mathbf{PEXP}'$$

$$\mathbf{FULF}_{k+1} = \mathbf{FULF}_k \ ? \ \{\mathbf{E}(E,T)\}$$

$$\mathbf{HAP}_{k+1} = \mathbf{HAP}_k \ ? \ \{\mathbf{H}(E,T)\}$$

Risultati specifici: Needham-Shroeder

È possibile generare una history compliant, a partire dal goal:

g ←

isNonce(NA)

? NA ? nb

? **E**(send(b, i, cont(key(ka), n(NA), n(nb)), 3)))

? **E**(send(i, b, cont(key(kb), n(nb)), 6)))

Risultati generali: (1) proprietà della SCIFF

- Soundness:
  - data una istanza di società S_{HAP}^{f},
  - dato un goal G

$$S_{HAP}^{i} \overset{HAP^{f}}{?}_{EXP} G \quad \text{con expect. answ. } (EXP \overset{?}{} FULF, \sigma)$$

$$\Rightarrow \quad S_{HAP}^{f} \overset{?}{}_{EXP\sigma} G\sigma$$

Risultati generali: (1) proprietà della SCIFF

- Termination
  - la SCIFF termina se:
    - i vincoli di integrità sociali + social KB costituiscono nell'insieme un programma aciclico rispetto a un level mapping, e
    - il goal G e tutte le implicazioni negli ICS sono bounded rispetto a tale level mapping

Risultati generali: (1) proprietà della SCIFF

- Completeness:
  - manca ancora una dimostrazione formale
  - *in via preliminare*: la SCIFF è completa in tutti i casi in cui è garantita la terminazione

Risultati generali: (2) proprietà della g-SCIFF

- Soundness:
  - data una istanza di società S_{HAP}^{f},
  - dato un goal G

$$S_{HAP}^{i} \overset{g}{?}_{EXP} G \quad \text{con expectation answer } (EXP,\sigma)$$

$$\Rightarrow \quad S_{HAP}^{f} ?_{EXP\sigma} G\sigma$$

Risultati generali: (2) proprietà della g-SCIFF

- Termination
  - la g-SCIFF termina se i vincoli di integrità sociali + social KB + il vincolo:

    E(X, T) ? **H**(X, T)

    costituiscono nell'insieme un programma aciclico.

Conclusioni

- Proprietà di soundness, termination, (completeness) per SCIFF e g-SCIFF
  - da verificare: completezza di SCIFF e g-SCIFF
- Applicazione a protocolli di sicurezza
  - mediante la generazione di un controesempio
  - laddove non sia garantita la completezza non è possibile trarre conclusioni dal fatto che la proof non riesce a generare un controesempio
- Implementazione e sperimentazione
  - migliorare efficienza e proseguire con testing