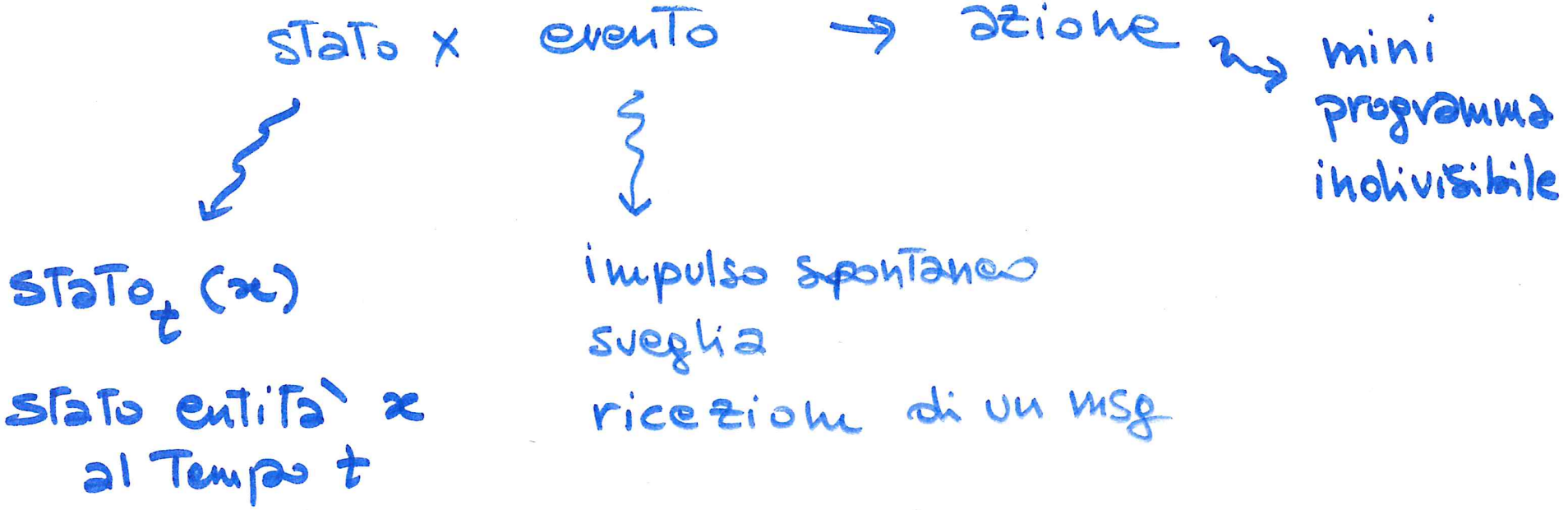
Algoritmo distribuito
PROTOCOLLO

"Insieme" di regole della forma:



Esecuzione di un protocollo:

Sequenza di configurazioni successive del sistema

Def. di configurazione

- $\Sigma(t)$ = il contenuto dei registri delle entità al tempo t

- Futuro (t) = eventi già generati al tempo t ma non ancora processati

$$C(t) = (\Sigma(t), \text{Futuro}(t))$$

↓

config. al tempo T del sistema

Esempio

$$C(0) = (\ \Sigma(0),\ \text{Futuro}(0)\ )$$

$\Sigma(0)$: registri inizializzati

Futuro(0): impulso spontaneo

L'esecuzione del protocollo descritta da seq. di conf. succ. è Tale che:

$$C(0) \xrightarrow{\text{protocollo}} C(f)$$

$C(0)$: iniziale

$C(f)$: finale

Notazione:

Quando una configurazione C soddisfa un predicato P

Scriveremo: $C \in P$

Definire:
- un problema (già visto)
- come un protocollo risolve un problema

$Problema = < P_{init}, P_{final}, R >$

Problema Broadcasting.

P_{init} : $\exists x \in \mathcal{E}$ t.c. valore$(x) = I$

$\wedge \ \forall y \neq x \in \mathcal{E}$ valore$(y) = \phi$

P_{final} : $\forall x \in \mathcal{E}$ valore$(x) = I$

$R = RI = (BL, TR, CN, UI)$

$S = \{$ iniziatore, inattivo $\}$

S_{init} = stati delle entità in $C(0)$

S_{term} = stati delle entità in $C(f)$

I versione PROTOCOLLO per BCAST

$S = \{$ iniziatore, inattivo $\}$

$S_{INIT} = \{$ iniziatore, inattivo $\}$

$S_{FINAL} = \{$ inattivo $\}$

Iniziatore:

impulso spontaneo

{ send (M) To N(x);

become inattivo;

}

Inattivo:

ricezione (M)

{ processa (M);

send (M) To N(x);

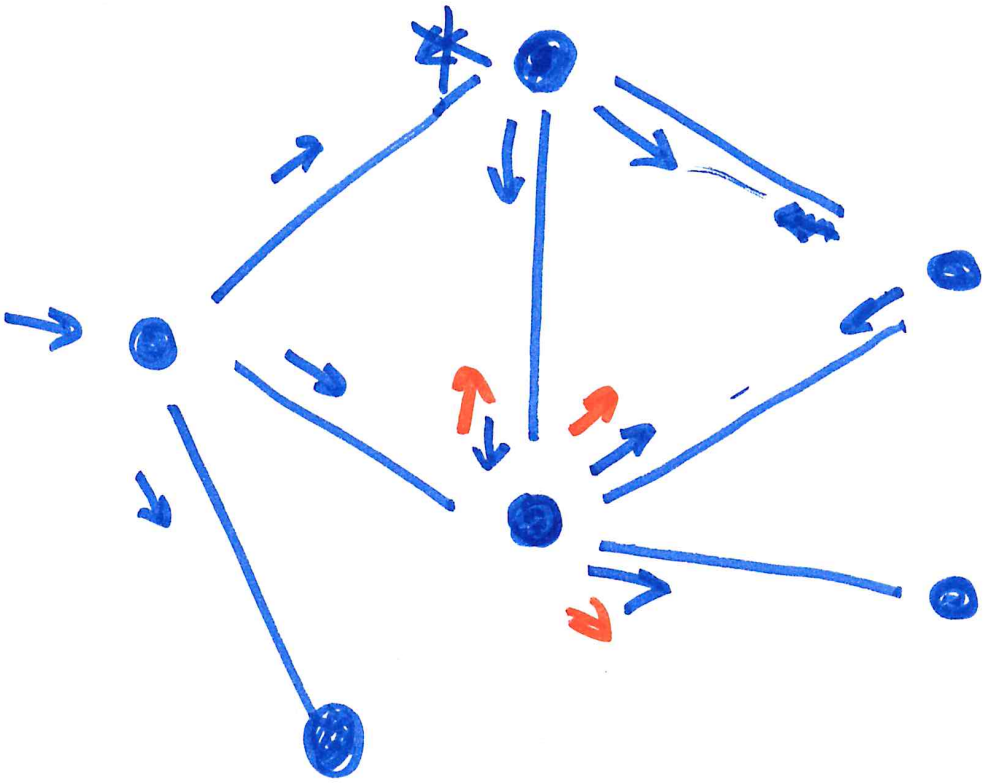
become inattivo;

}

Messaggio M:

$M = (t, o, d, I)$

c'è un problema in questa versione!
Qual è?



Il protocollo è corretto ma non termina:

Futuro(t) $\neq \emptyset$ e la computaz. non termina

Risposta al problema è il raffinamento degli stati S_{TERM}.

$S_{START} \subseteq S_{INIT}$: stati che fanno iniziare il protocollo

$S_{FINAL} \subseteq S_{TERM}$: stati per cui la sola azione è quella nulla

Esempio BCAST

$S_{INIT} = \{$ iniziatore, inattivo $\}$

$S_{START} = \{$ iniziatore $\}$

$S_{TERM} = S_{FINAL} = \{$ finito $\}$

$|S| = 3$

Osservazione:

Grazie all'uso dello stato "finito"
il protocollo riesce a terminare

Formalmente: la soluzione per $\mathcal{P}$ è:

CORRETTEZZA

$$\forall\ C(0) \in P_{init} \quad \exists\ t' \text{ t.c. } \forall\ t > t' \quad C(t) \in P_{final}$$

$$\wedge$$

TERMINAZIONE

$$\forall\ x \in \mathcal{E} \qquad stato_t(x) \in S_{FINAL}$$

II VERSIONE PROTOCOLLO PER BCAST

Flooding

$S = \{iniziatore, inattivo, finito\}$

$S_{START} = \{iniziatore\}$

$S_{FINAL} = \{finito\}$

Iniziatore:

impulso spontaneo

{ send (M) To N(x);
  become finito;
}

Inattivo:

ricezione (M)

{ processa (M);
  send (M) To N(x) - sender;
  become finito;
}

Le coppie (stato, evento) non indicate eseguono l'azione nulla

Complessità di Flooding:

- $M[\text{Flooding}] = \sum_{x \in E} (N(x) - 1) + 1 =$

$$= 2m - n + 1$$

$2m$: n° di archi; n: n° di nodi

- $T[\text{Flooding}] \leq d$ ← diametro della rete

Per i lower bound si ha:

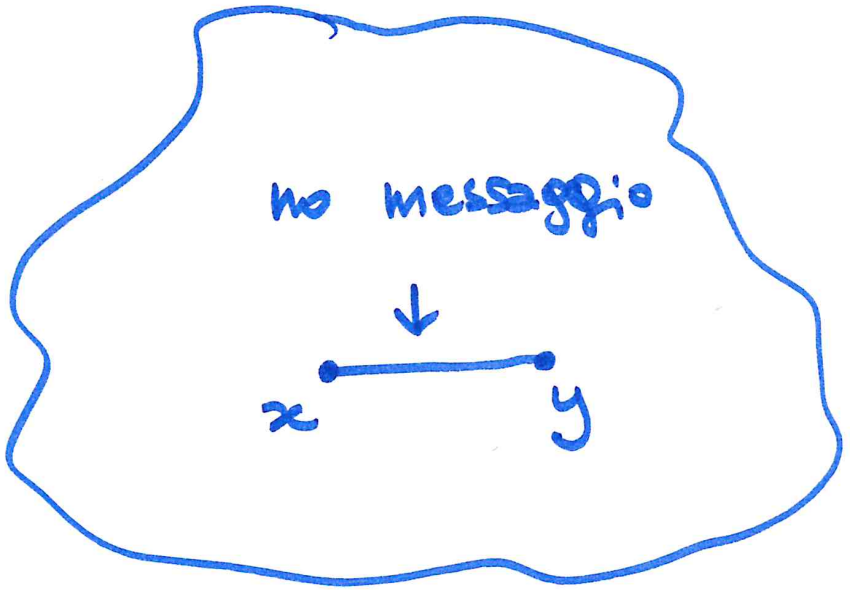
- $T[\text{BcastT} / \text{RI}] \geq d$
- $M[\text{BcastT} / \text{RI}] \geq m$ ← Teorema 2

⇓

il protocollo flooding
è OTTIMALE

Teo: $M[\text{Bcast}/\text{RI}] \geq m$

dim. per assurdo. risolvo il problema con meno di m messaggi

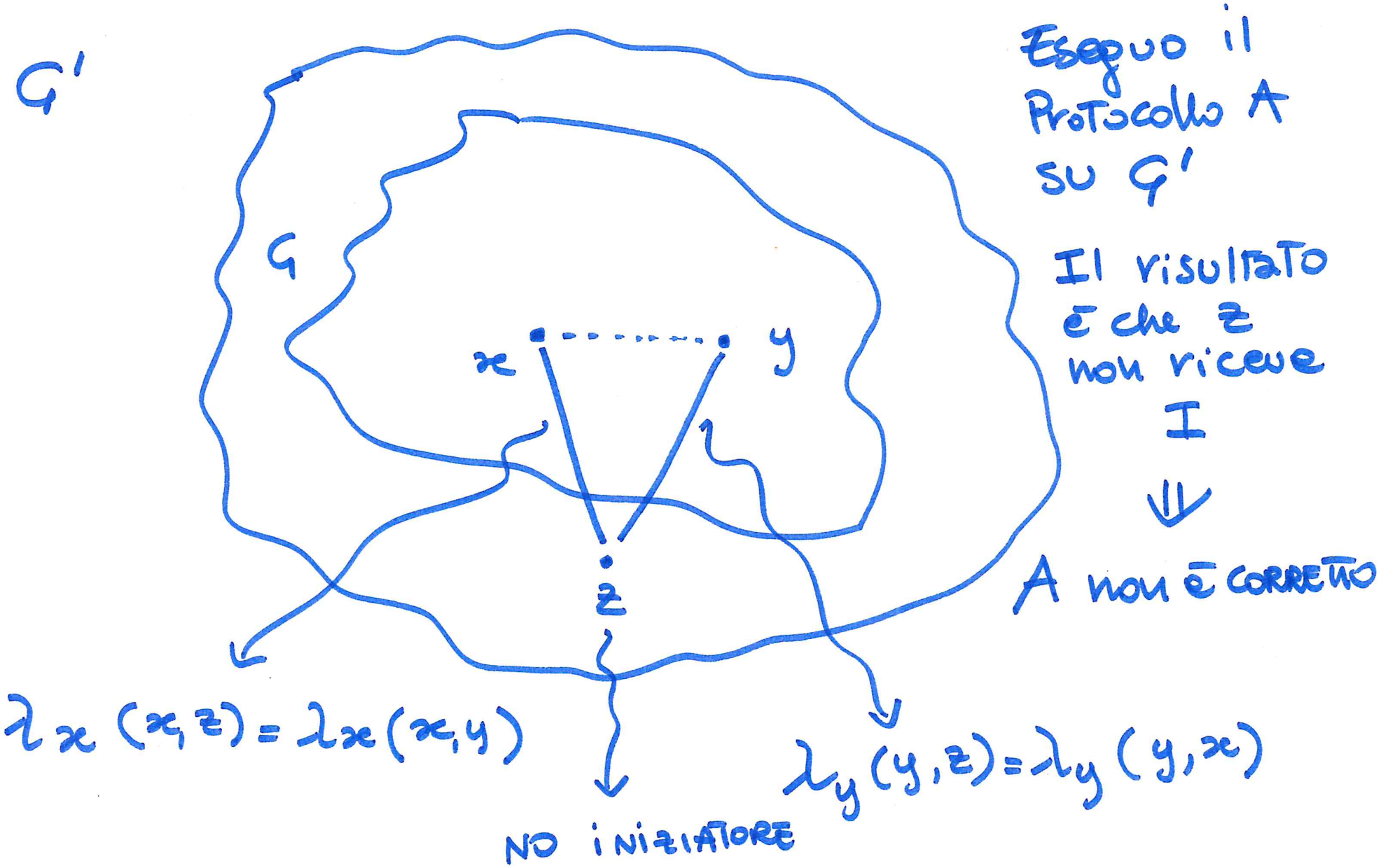


Sia A il protocollo che non manda msg su (x,y)

A è corretto e deve lavorare bene su ogni G

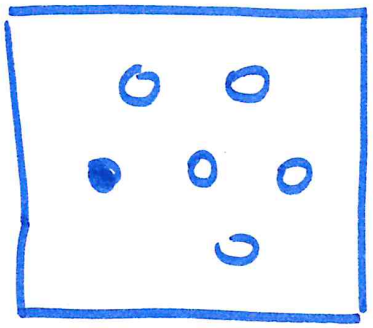
$\exists$

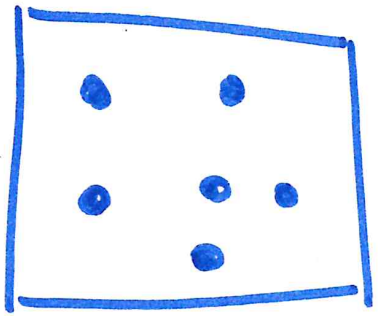
G'


G'
G
x
y
z
Eseguo il Protocollo A su G'
Il risultato è che z non riceve I
A non è corretto
$\lambda_x(x,z) = \lambda_x(x,y)$
$\lambda_y(y,z) = \lambda_y(y,x)$
NO INIZIATORE

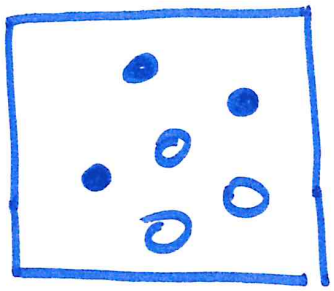
Problema Wake-up

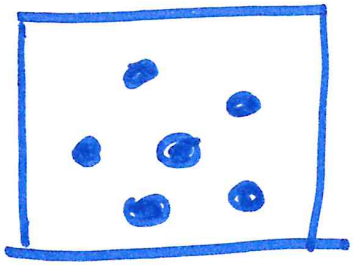
Bcast





Wake-up





Protocollo WFlood

$S = \{$ dormiente, attivo $\}$

$S_{INIT} = \{$ dormiente $\} = S_{START}$

$S_{TERM} = \{$ attivo $\} = S_{FINAL}$

DORMIENTE

IMPULSO SPONTANEO

{ SEND (W) TO N(x)

BECOME ATTIVO

}

```
RICEZIONE (W)
{
    SEND (W) TO N(x) - SENDER
    BECOME ATTIVO
}
```

COSTO:

- $T[W\,Flood] \leq d$
- $2m-n+1 \leq M[W\,Flood] \leq 2m$

↑ 1 solo entità si attiva

↑ tutte le entità si attivano