Model checking come supporto per le scelte di sistemi adattivi

Marco Tinacci
marco.tinacci@gmail.com

Università degli Studi di Firenze
www.unifi.it

15 Ottobre 2012

Definizione

"In che caso un sistema si dice adattivo?"

Un sistema è *adattivo* quando il suo comportamento dipende da un insieme di dati di controllo che possono variare durante l'esecuzione

"Cosa si vuole ottenere?"

Si vuole elaborare una strategia che permetta al sistema di raggiungere il suo obiettivo

"Perché usare un approccio adattivo?"

Questo approccio si adatta a situazioni dove il sistema ha una conoscenza parziale o nulla dell'ambiente

Esempio - Agenti mobili

Sensori

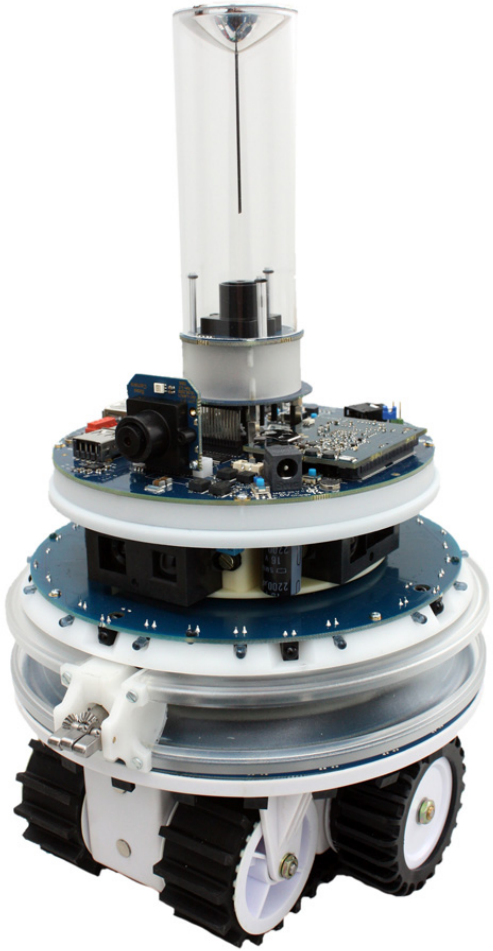
- sensori di prossimità
- sensori di luce
- videocamere

Attuatori

- ruote
- led luminosi
- dispositivi di connessione

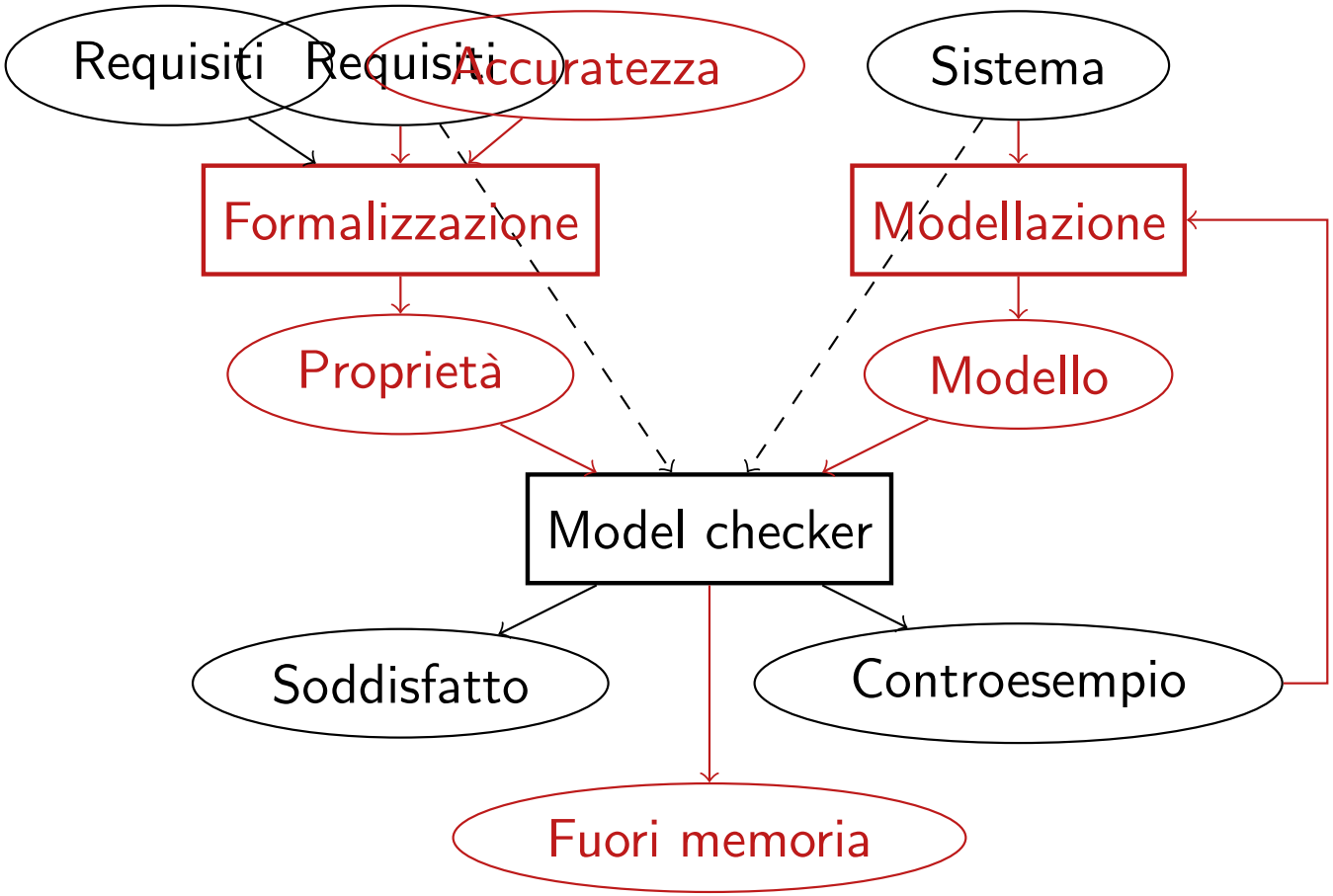
Ciclo di feedback

- controllo - dati di controllo
- elaborazione - strategia
- attuazione - comportamento



Definizione

Il *model checking* è un metodo di verifica formale di un modello



PRISM

PRISM è un model checker che presenta le seguenti caratteristiche:

- permette di definire modelli con aspetti *nondeterministici* e *probabilistici* tramite un linguaggio specifico
- permette di formalizzare le proprietà tramite formule *PCTL*

Esempio di proprietà qualitativa

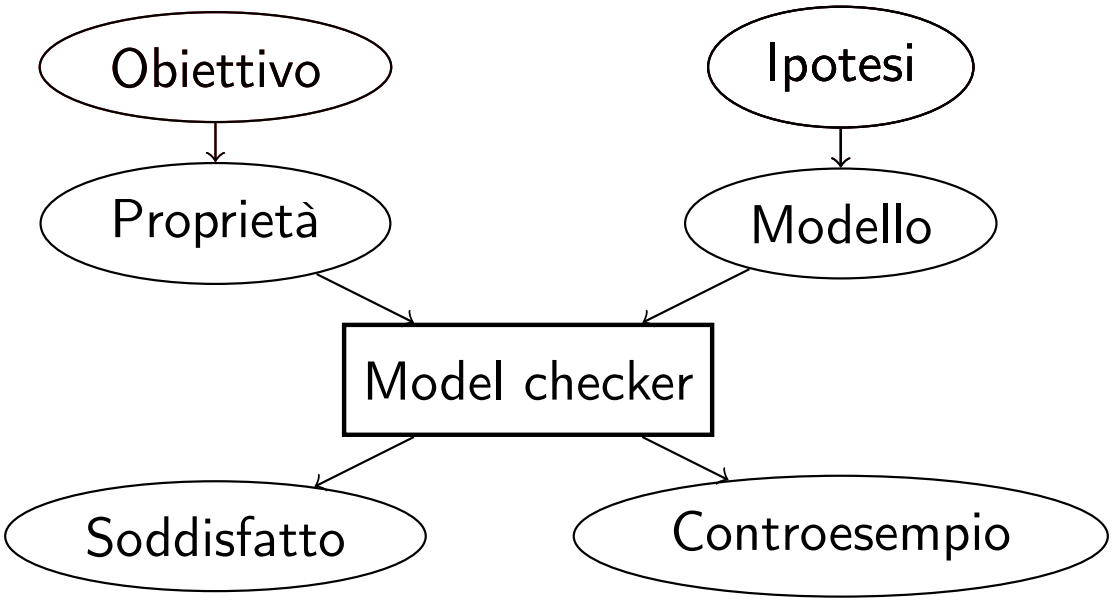
I messaggi scambiati tra due terminali vengono trasmessi correttamente

Esempio di proprietà quantitativa

Il 99% dei messaggi scambiati tra due terminali viene trasmesso correttamente

Risolvere le scelte

Utilizziamo il model checking all'interno del criterio di risoluzione delle scelte



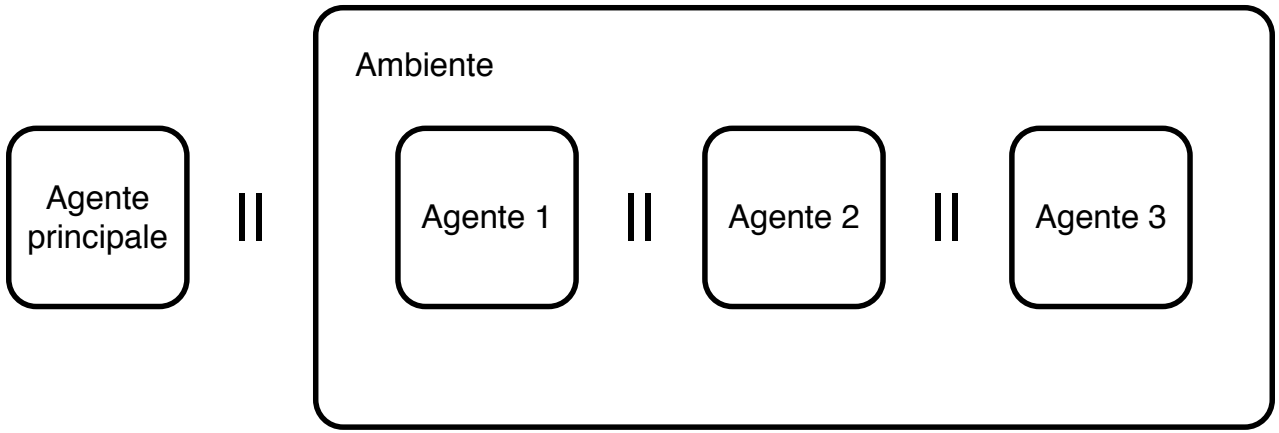
- le lacune sulla conoscenza dell'ambiente vengono colmate da ipotesi
- come proprietà da verificare viene data una formula che descriva l'obiettivo del sistema che deve prendere le decisioni
- il model checker viene usato per fare previsioni invece che verifiche
- viene valutato l'esito delle previsioni per ogni scelta eseguibile dal sistema, la decisione da prendere è quella con la più alta probabilità di successo

LAPSA: Language for Population of Self-adaptive Agents - 1

Impone la visione dell'agente principale consentendo di descrivere le ipotesi che si fanno sull'ambiente

Programma LAPSA

$$\begin{array}{lll} \textit{program} & ::= & \textbf{subject}\ \textit{module} \\ & & \textit{modules} \\ & & \textit{environment} \end{array}$$



LAPSA: Language for Population of Self-adaptive Agents - 2

Modulo

$$\textit{module} ::= \textbf{module}\ \text{module-id}\ \{\textit{variables}\ \textit{rules}\ \textit{targets}\}$$

Transizione

$$\textit{rules} ::= \textit{condition}[\text{action-id}] \Rightarrow \textit{distribution};\quad |\quad \textit{rules}\ \textit{rules}$$

Obiettivo

$$\textit{targets} ::= \textbf{target never}\ \textit{condition}\quad |\quad \textit{targets}\ \textit{targets}$$

LAPSA: Language for Population of Self-adaptive Agents - 3

Condizione

$$\begin{aligned} \textit{condition} \quad ::= \quad & \textbf{exists}\ \text{variable-id} : \text{module-id}\ \textbf{such that}\ \textit{condition} \\ | \quad & \textit{expression} \bowtie \textit{expression} \quad | \quad \textit{condition}\ \textbf{or}\ \textit{condition} \\ | \quad & \textbf{not}\ \textit{condition} \quad | \quad (\textit{condition}) \quad | \quad \textbf{true} \end{aligned}$$

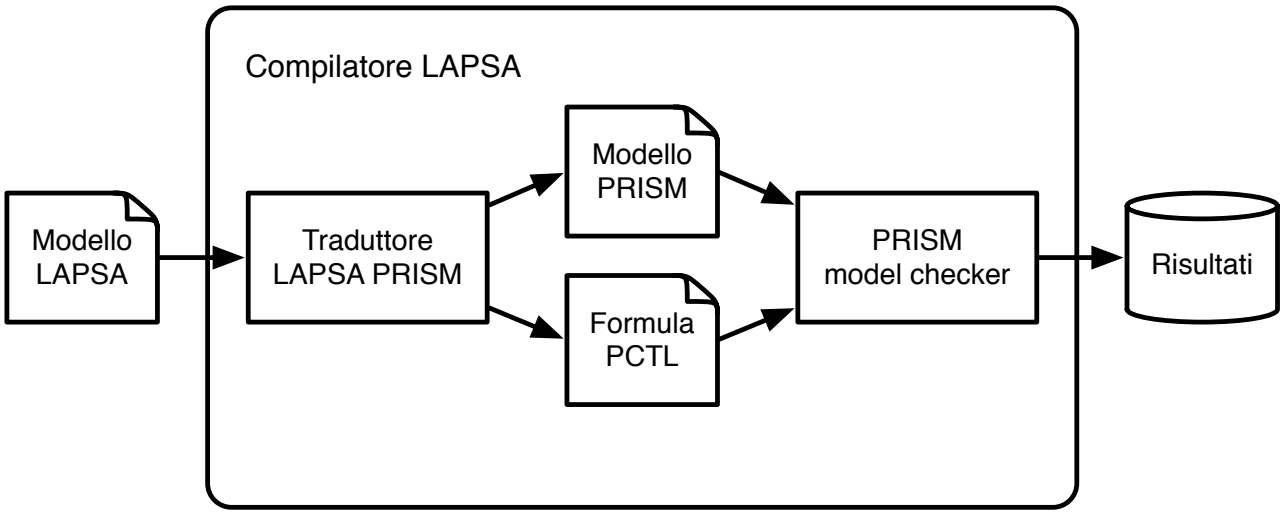
Distribuzione

$$\textit{distribution} ::= < \textit{expression} > \textit{update} \quad | \quad \textit{distribution} \ \# \ \textit{distribution}$$

Compilatore LAPSA

Il compilatore del linguaggio LAPSA è stato implementato in Java con l'ausilio del plugin Xtext ed è composto di due parti:

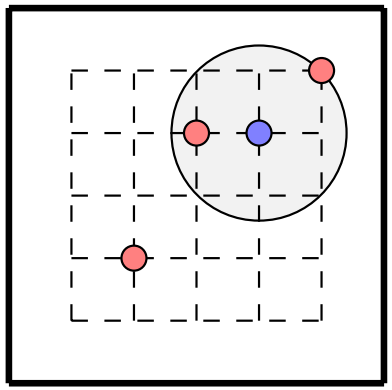
- un modulo che traduce il programma LAPSA in un modello PRISM e una formula PCTL
- l'esecuzione del model checker di PRISM sui parametri generati



Il risultato della compilazione è un insieme di probabilità associate a stati che quantificano la possibilità di successo

Scenario

Lo scenario analizzato è un'arena quadrata contenente una popolazione di agenti mobili



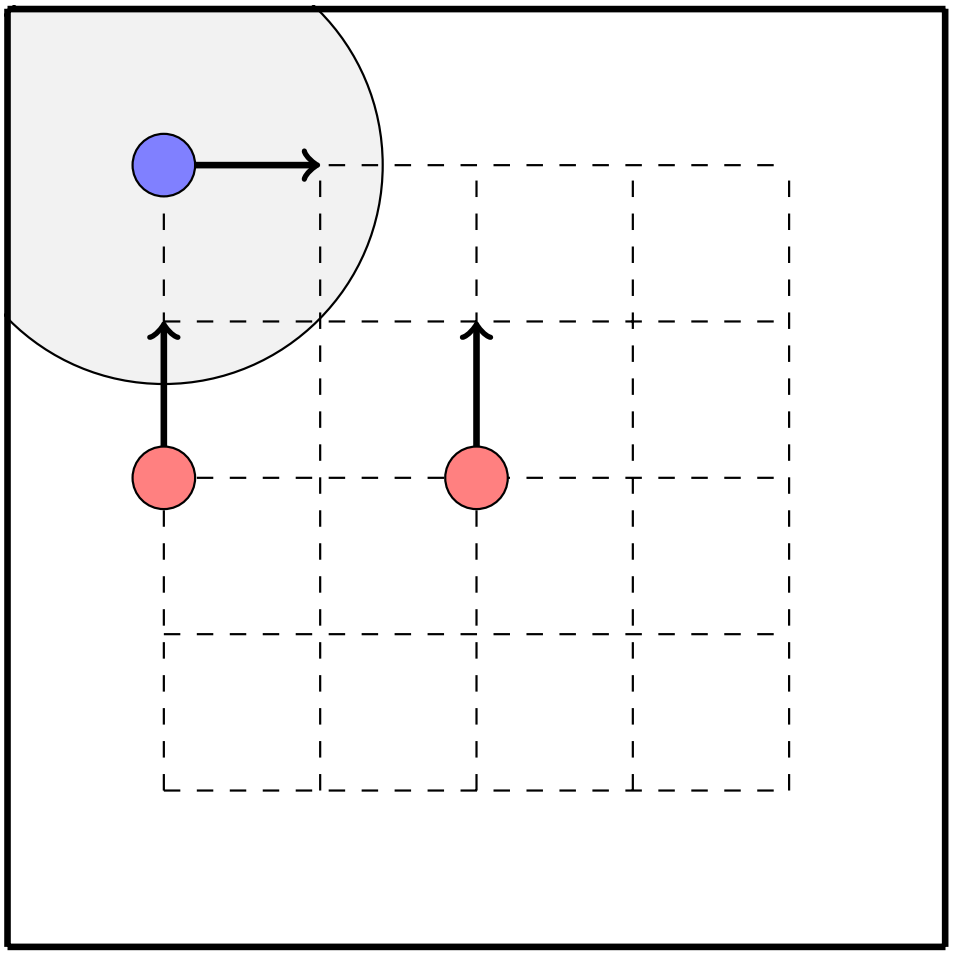
- il cerchio blu è l'agente principale, quelli rossi sono secondari
- l'obiettivo dell'agente blu è di minimizzare gli scontri con altri agenti
- tutti gli agenti sono sincronizzati nello spazio e nel tempo e possono scegliere ad ogni passo se stare fermi o muoversi a nord, sud, ovest o est
- il cerchio grigio rappresenta l'area visibile dall'agente blu
- gli agenti rossi si muovono casualmente

Descrizione approccio

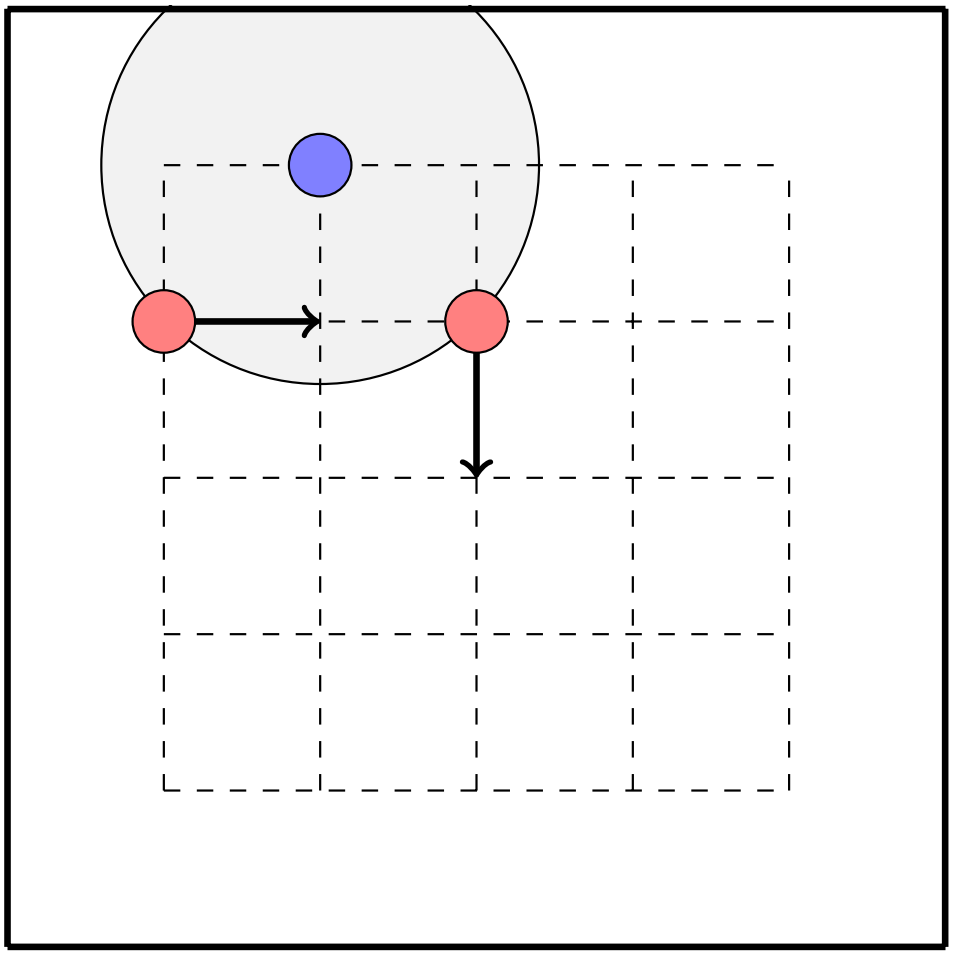
L'approccio utilizzato può essere riassunto nei seguenti passi

1. scrittura del programma LAPSA
   - si ipotizza la presenza di agenti che si muovono casualmente
   - si astrae da tutti i dati esterni all'area visibile dall'agente principale
2. compilazione, si ottiene una tabella hash che associa ad ogni stato la rispettiva probabilità di successo
3. scrittura dello scheduler dell'agente principale che utilizza la tabella hash per ricavare la decisione migliore ad ogni passo

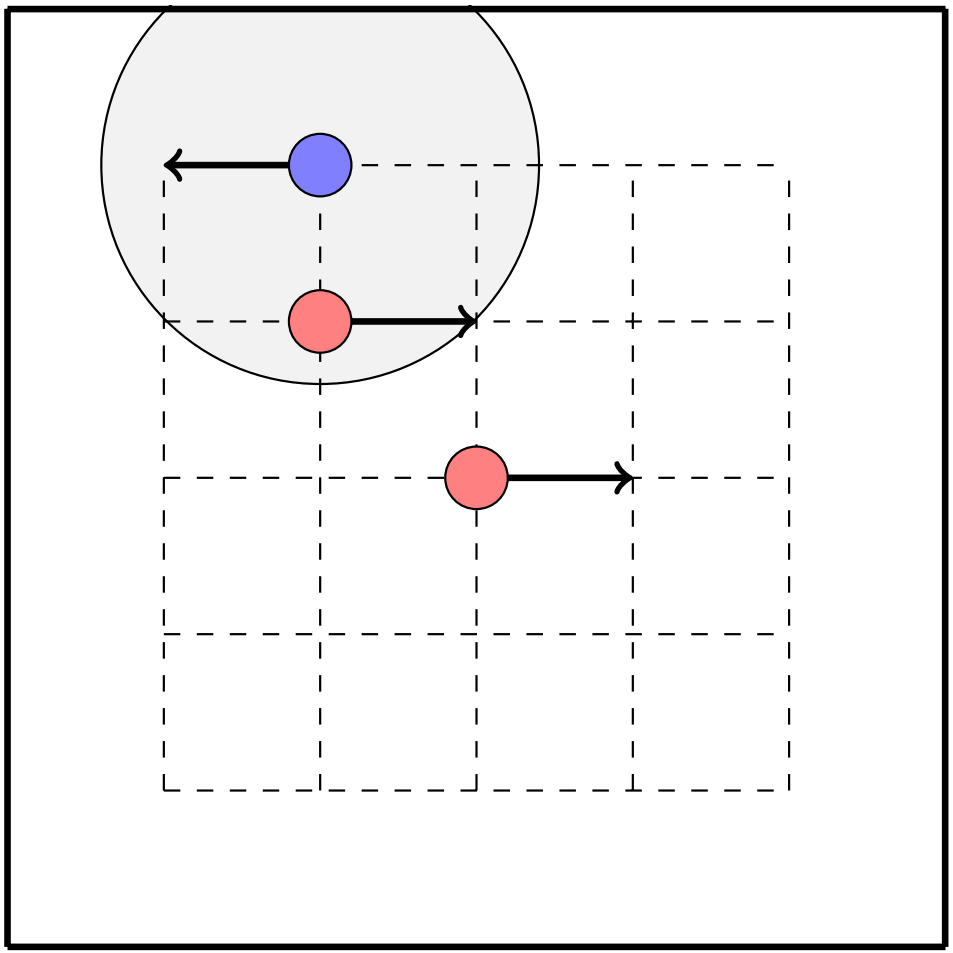
Simulazione - 1



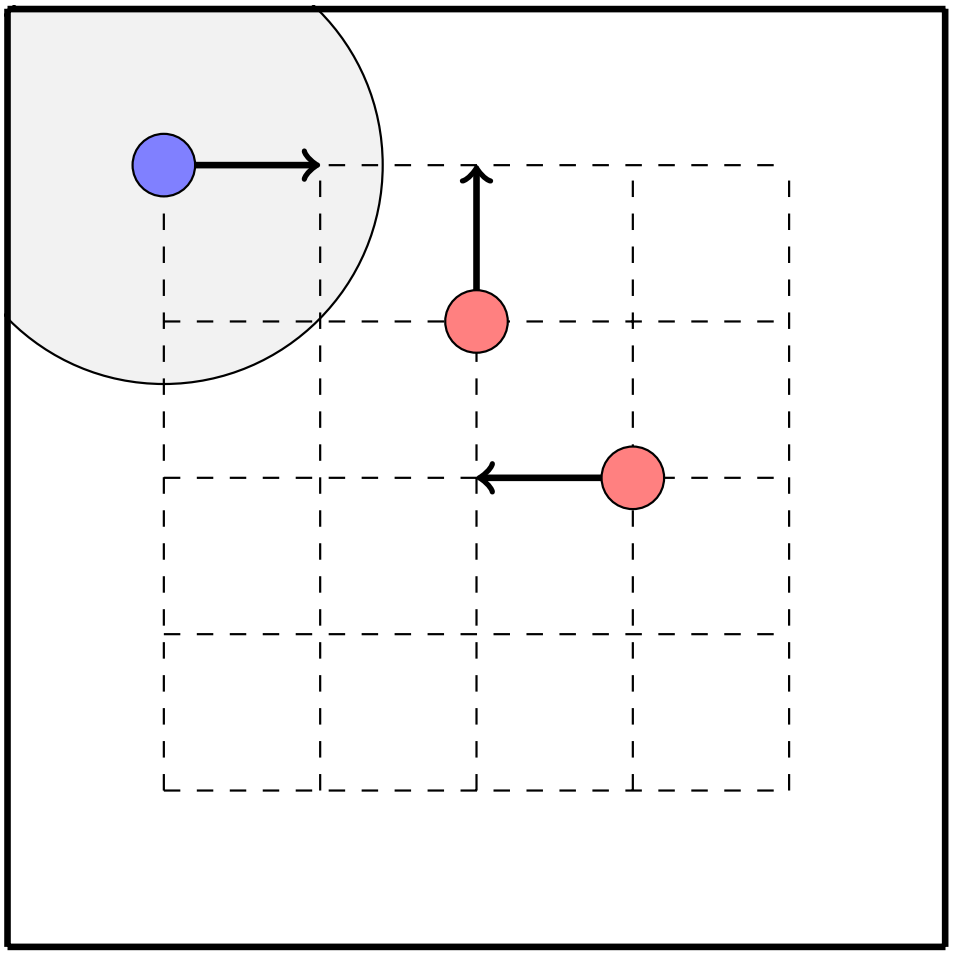
Simulazione - 2



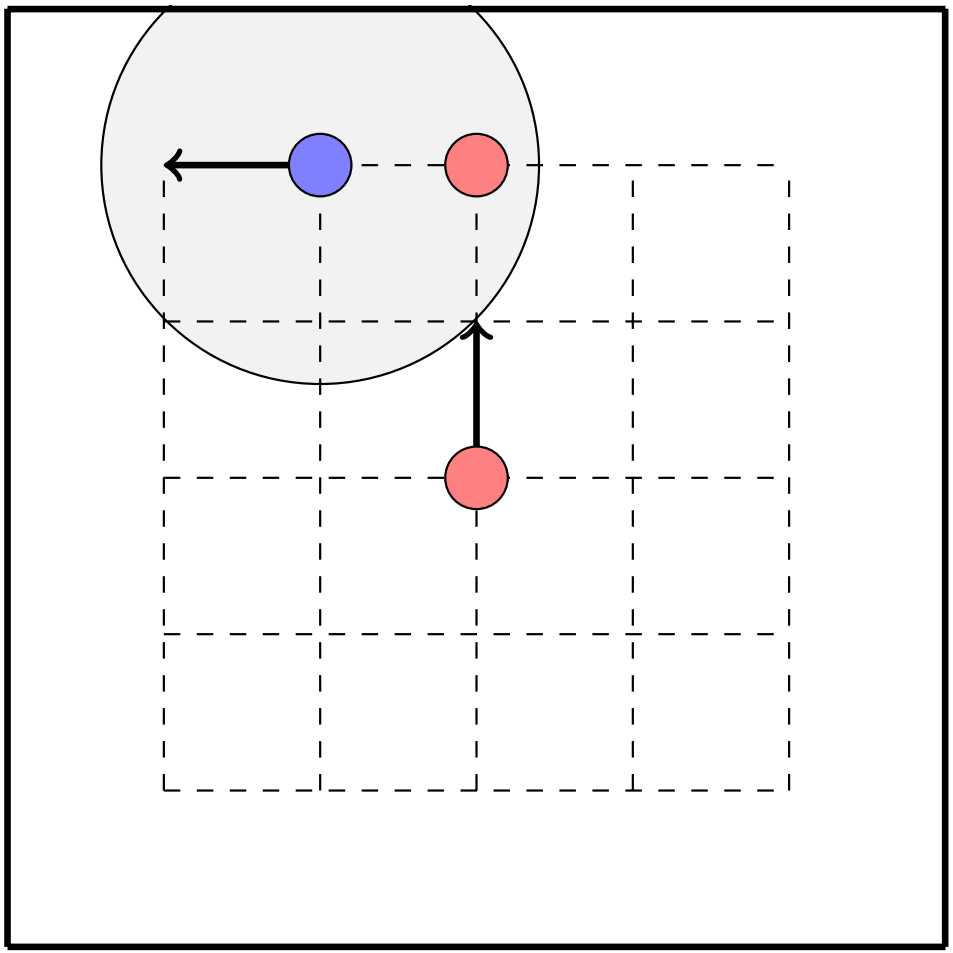
Simulazione - 3



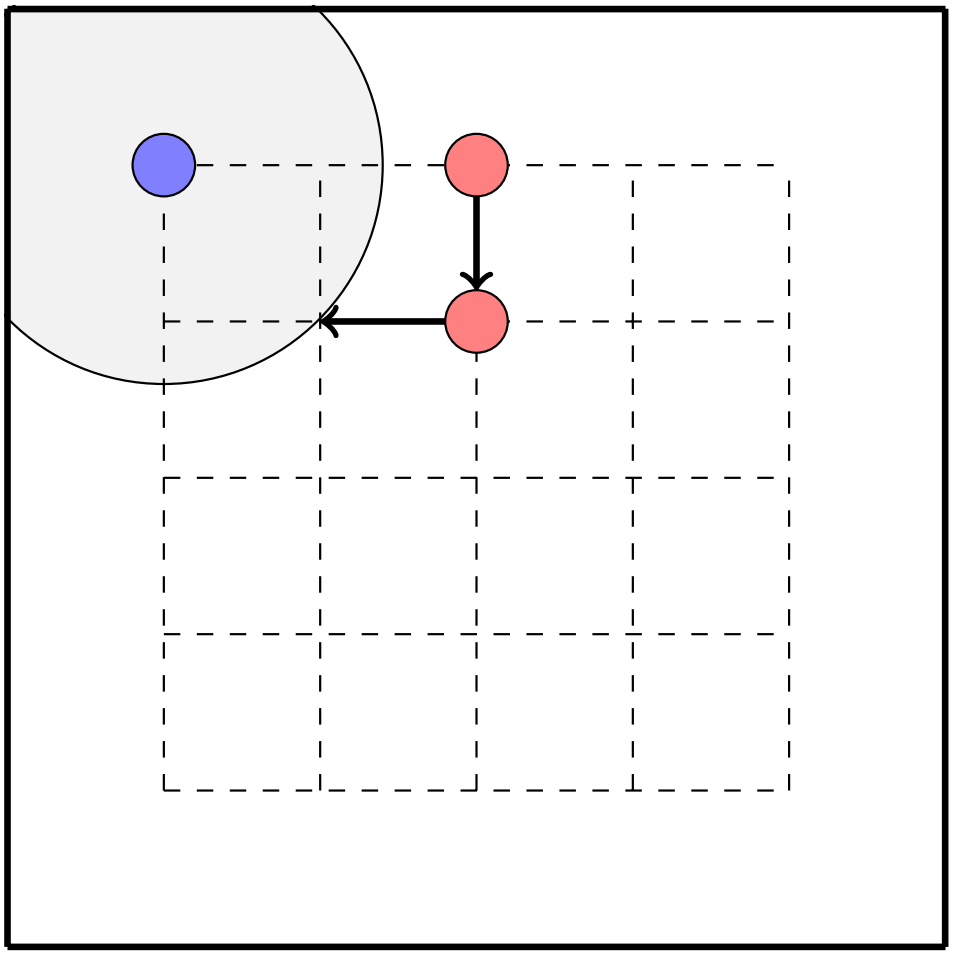
Simulazione - 4



Simulazione - 5



Simulazione - 6



TODO conclusioni

TODO