Model checking come supporto per le scelte di sistemi adattivi

Marco Tinacci

Relatore
Rocco De Nicola

Correlatore
Michele Loreti

Dipartimento di Sistemi e Informatica
Università degli Studi di Firenze

15 Ottobre 2012

Definizione di sistema adattivo

"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 tramite l'adattività?"

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

"Perché usare un approccio adattivo?"

Questo approccio risulta utile in situazioni dove il sistema ha una conoscenza parziale o nulla dell'ambiente

Esempio - marXbot

Sensori

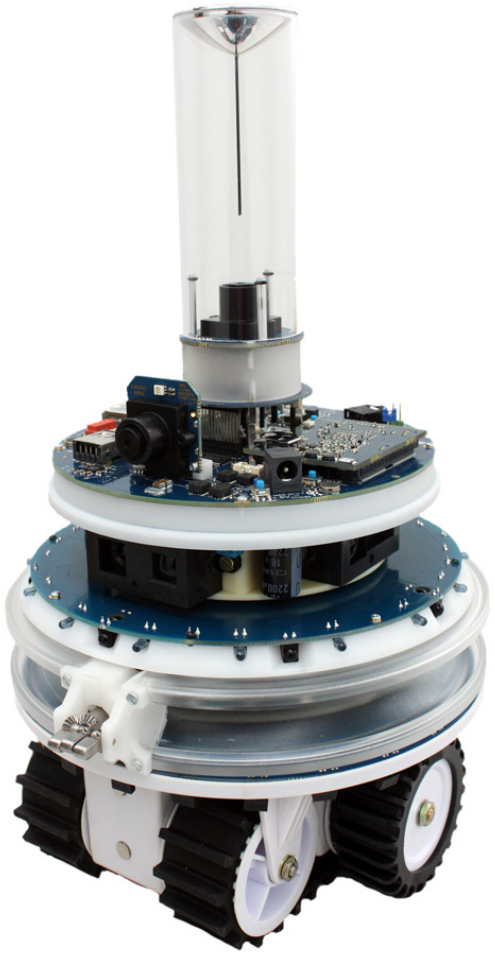
- sensori di prossimità
- sensori di luce
- videocamere

Attuatori

- ruote
- led luminosi
- dispositivi di connessione

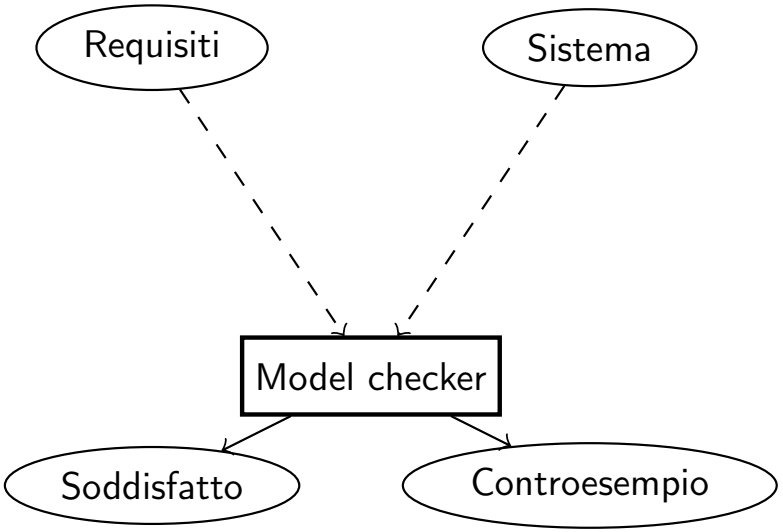
Ciclo di feedback

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



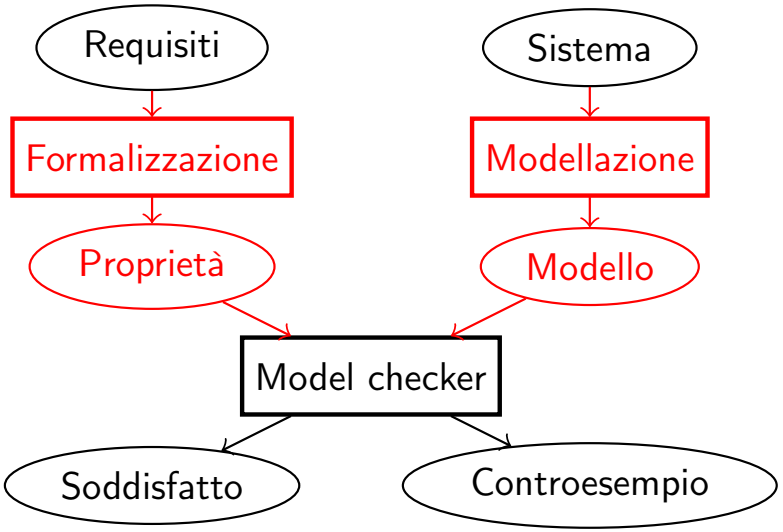
Model checking

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



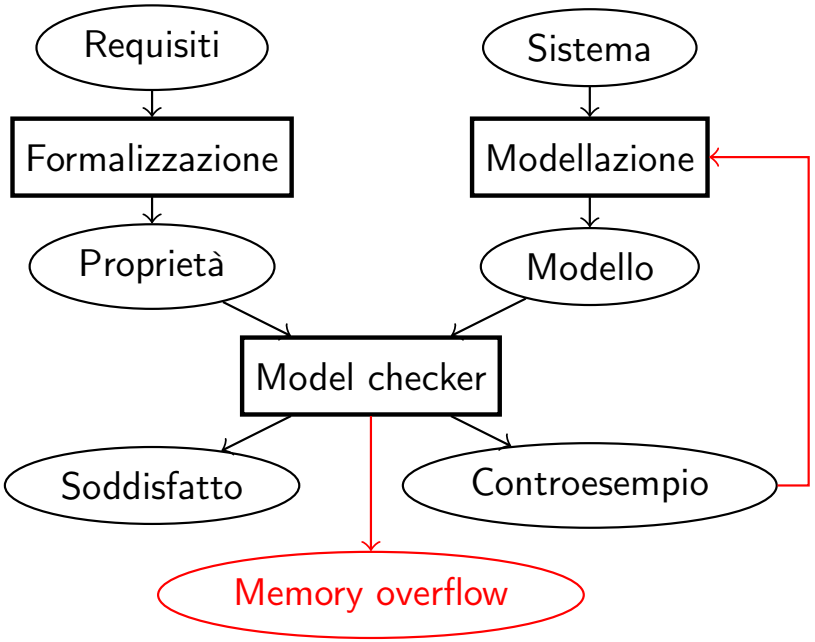
Model checking

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



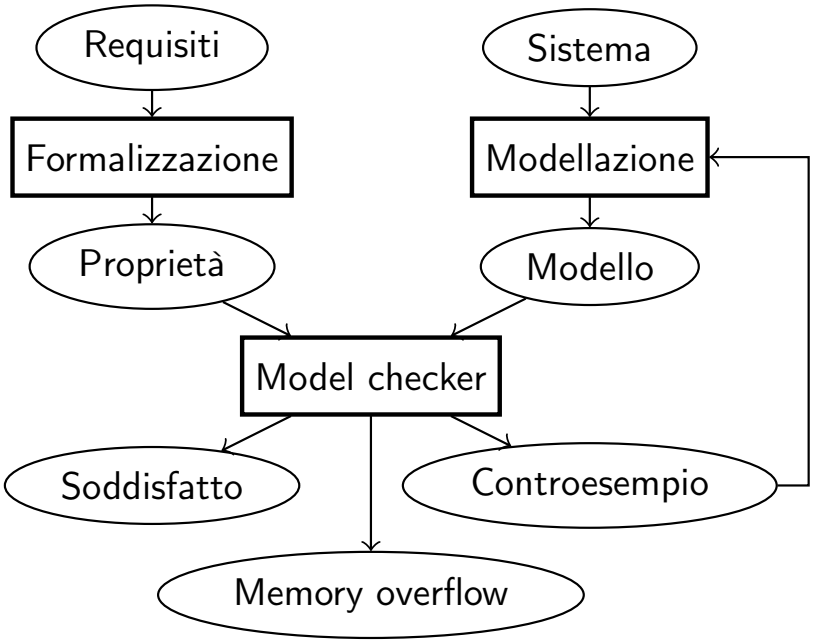
Model checking

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



Model checking

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

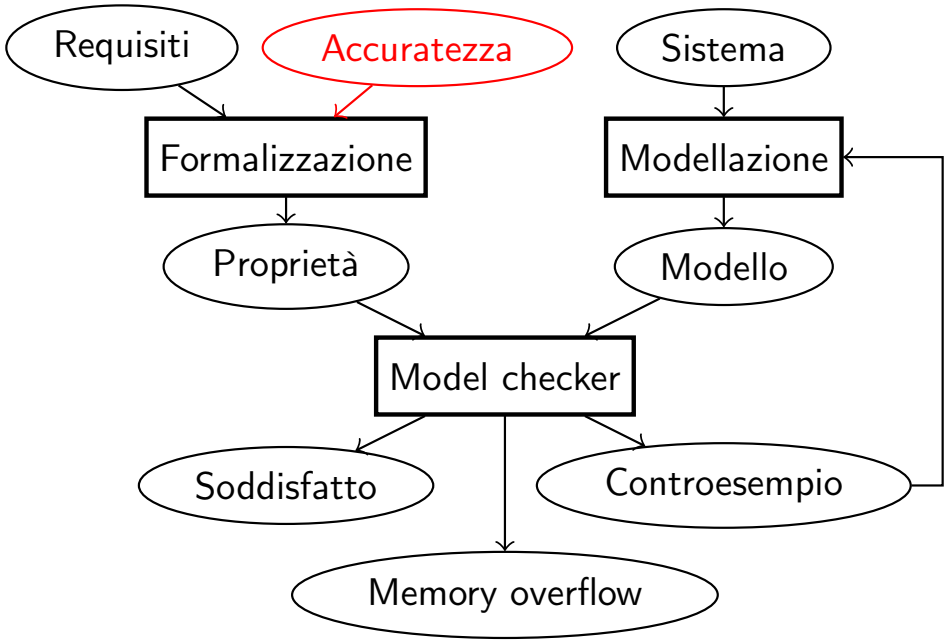


Esempio di proprietà

I messaggi scambiati tra due terminali vengono trasmessi correttamente

Model checking

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



Esempio di proprietà

I messaggi scambiati tra due terminali vengono trasmessi correttamente

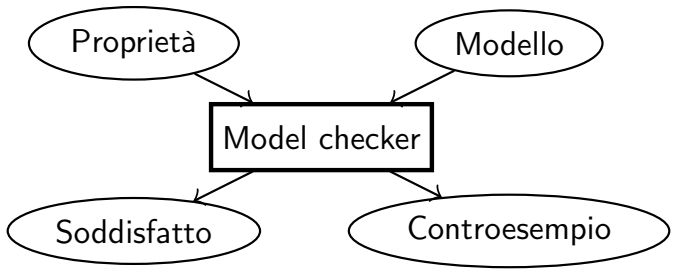
Esempio di proprietà quantitativa

I messaggi scambiati tra due terminali vengono trasmessi correttamente con una probabilità maggiore di 0.99

Problema: come risolvere le scelte?

Idea

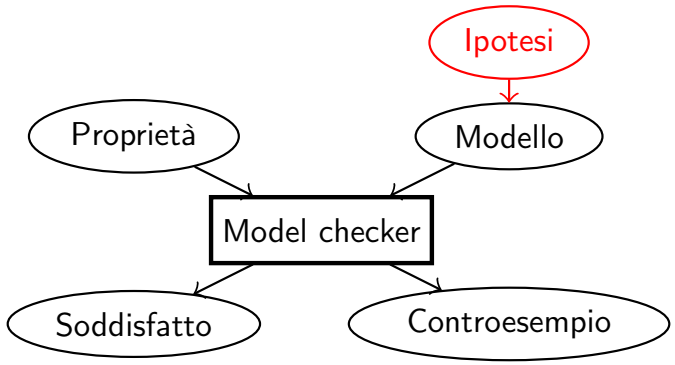
Utilizziamo il model checking all'interno del criterio di risoluzione delle scelte



Problema: come risolvere le scelte?

Idea

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

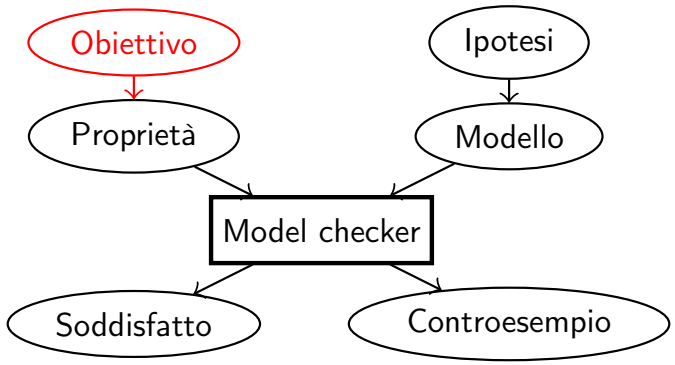


- le lacune sulla conoscenza dell'ambiente vengono colmate da ipotesi

Problema: come risolvere le scelte?

Idea

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

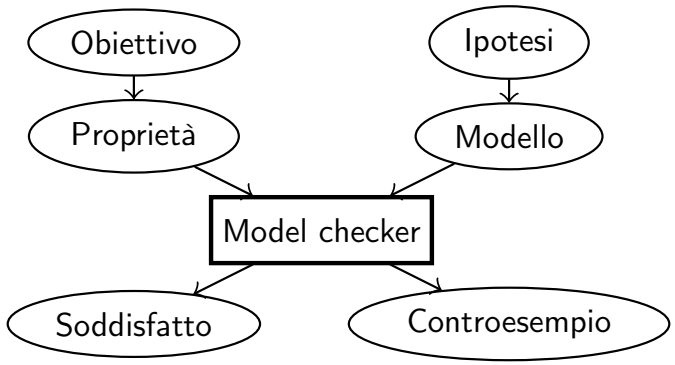


- le lacune sulla conoscenza dell'ambiente vengono colmate da ipotesi
- come proprietà da verificare viene data una formula che descrive l'obiettivo del sistema che deve prendere le decisioni

Problema: come risolvere le scelte?

Idea

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

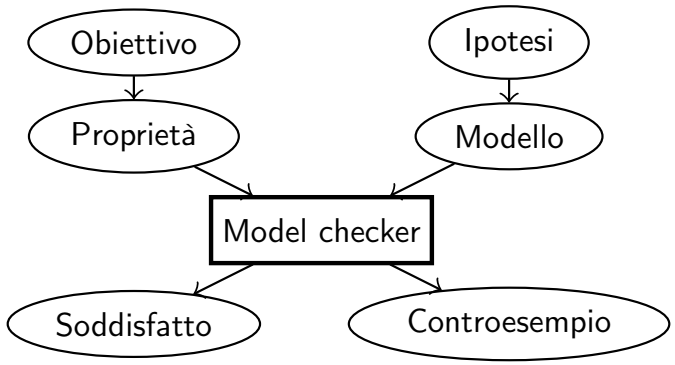


- le lacune sulla conoscenza dell'ambiente vengono colmate da ipotesi
- come proprietà da verificare viene data una formula che descrive l'obiettivo del sistema che deve prendere le decisioni
- il model checker viene usato in un modo non convenzionale per fare previsioni invece che verifiche

Problema: come risolvere le scelte?

Idea

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



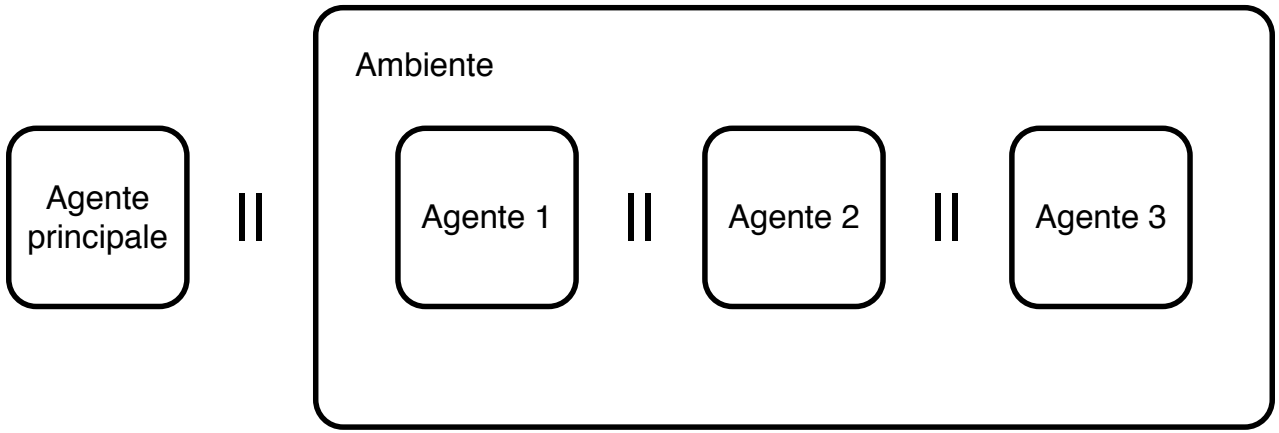
- le lacune sulla conoscenza dell'ambiente vengono colmate da ipotesi
- come proprietà da verificare viene data una formula che descrive l'obiettivo del sistema che deve prendere le decisioni
- il model checker viene usato in un modo non convenzionale per fare previsioni invece che verifiche
- viene valutato l'esito delle previsioni per ogni stato del modello raggiungibile da una scelta dell'agente principale, la decisione da prendere è quella che porta alla 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

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



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

Modulo

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

Transizione

$$\textit{rules} ::= \textit{condition} \Rightarrow \textit{distribution};\quad |\quad \textit{rules}\ \textit{rules}$$

Obiettivo

$$\textit{targets} ::= \textbf{target}\ \textit{condition}\quad |\quad \textit{targets}\ \textit{targets}$$

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

Condizione

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

Esempio di quantificatore esistenziale

$$\textbf{exists}\ r \ :\ \textit{robot}\ \textbf{such that}\ r.x > 3$$

Distribuzione

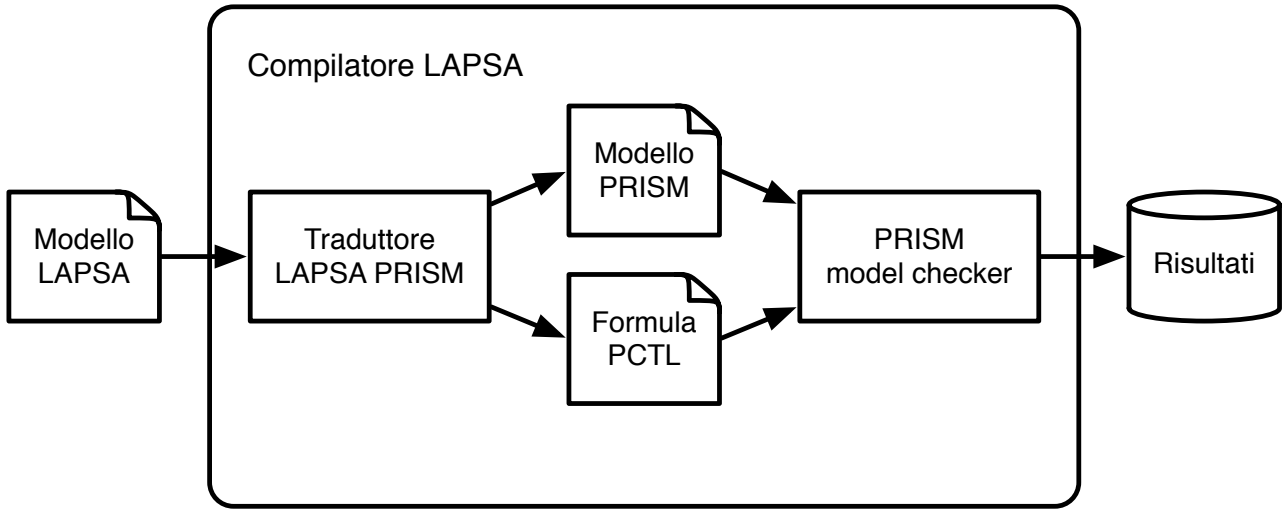
$$\textit{distribution} ::= < \textit{expression} > \textit{update} \quad | \quad \textit{distribution} \ \# \ \textit{distribution}$$

Esempio di distribuzione

$$< 1 > x = x + 1 \ \# \ < 2 > x = x - 1$$

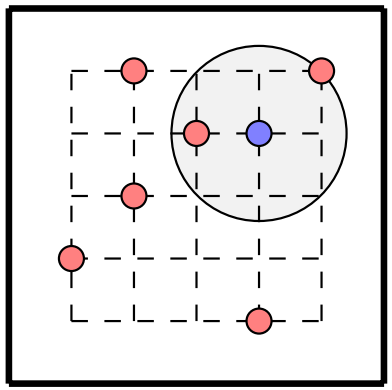
Compilatore LAPSA

- implementazione in Java
- plugin di Eclipse tramite Xtext
- utilizza PRISM, un model checker probabilistico
- usa una logica temporale per definire le formule di proprietà
- il risultato 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 marXbot



- il cerchio blu è il robot principale, quelli rossi sono secondari
- l'obiettivo del robot blu è di minimizzare gli scontri con altri robot
- tutti i robot possono scegliere periodicamente di muoversi di un passo a nord, sud, ovest o est o di stare fermi
- il cerchio grigio rappresenta l'area visibile dal robot blu
- i robot rossi si muovono casualmente

Descrizione approccio

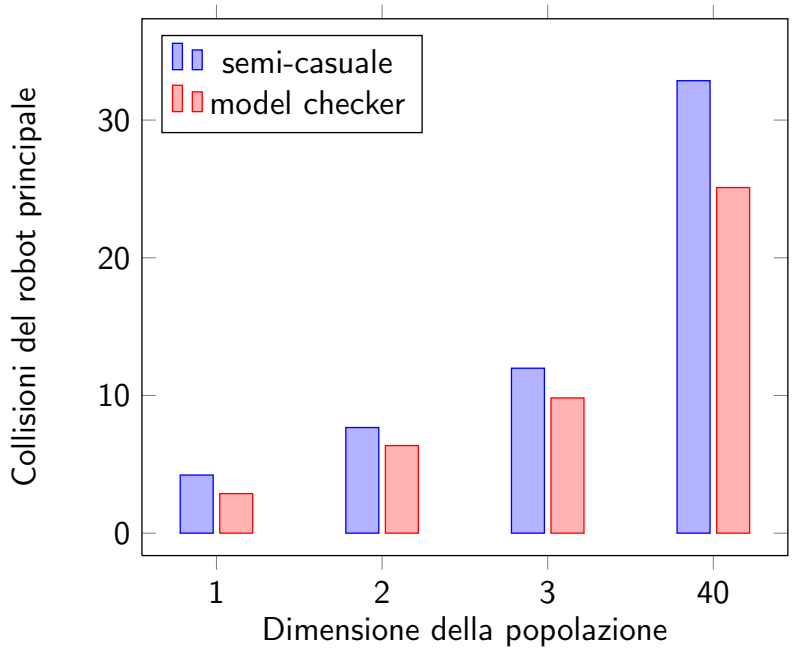
L'approccio utilizzato può essere riassunto nei seguenti passi

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

Risultati

I risultati sono stati mediati su 300 simulazioni di 100 passi e confrontati con quelli ottenuti da uno scheduler semi-casuale

- per le popolazioni composte da un massimo di 3 robot secondari in un'arena 5×5 il numero di scontri con il robot principale viene diminuito almeno del 20% rispetto a uno scheduler semi-casuale
- per una popolazione di 40 robot secondari in un'arena 10×10 il numero di scontri con il robot principale viene diminuito del 31% rispetto a uno scheduler semi-casuale



Conclusioni e lavori futuri

Conclusioni

- Abbiamo presentato un nuovo approccio per risolvere le scelte di un sistema adattivo
- dai risultati abbiamo verificato sperimentalmente la possibilità di un approccio di scelta basato sul model checking
- i risultati ottenuti dipendono molto dalle astrazioni e dalle ipotesi che si fanno nel modello LAPSA

Possibili sviluppi futuri

- estendere LAPSA
  - dare memoria agli agenti per permettergli di fare scelte basandosi sulle informazioni ricavate dagli stati precedenti
  - aggiungere primitive che consentano all'agente di modificare l'ipotesi fatta sull'ambiente
- la teoria dei giochi

Grazie per l'attenzione